Data-Based Control

Egemen Kolemen

Associate Professor

Princeton University / PPPL

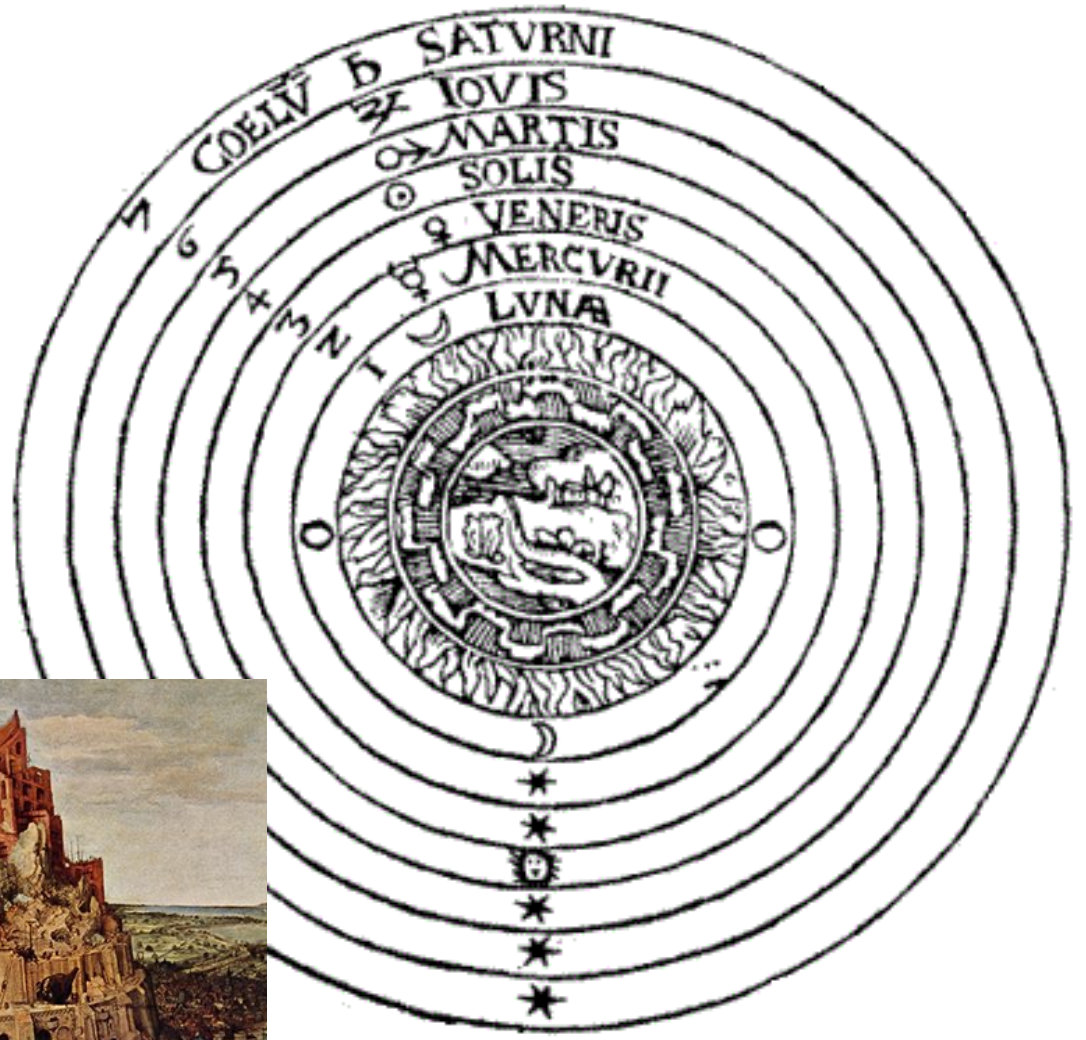
And ITER Fellow in Control

Princeton Plasma Control

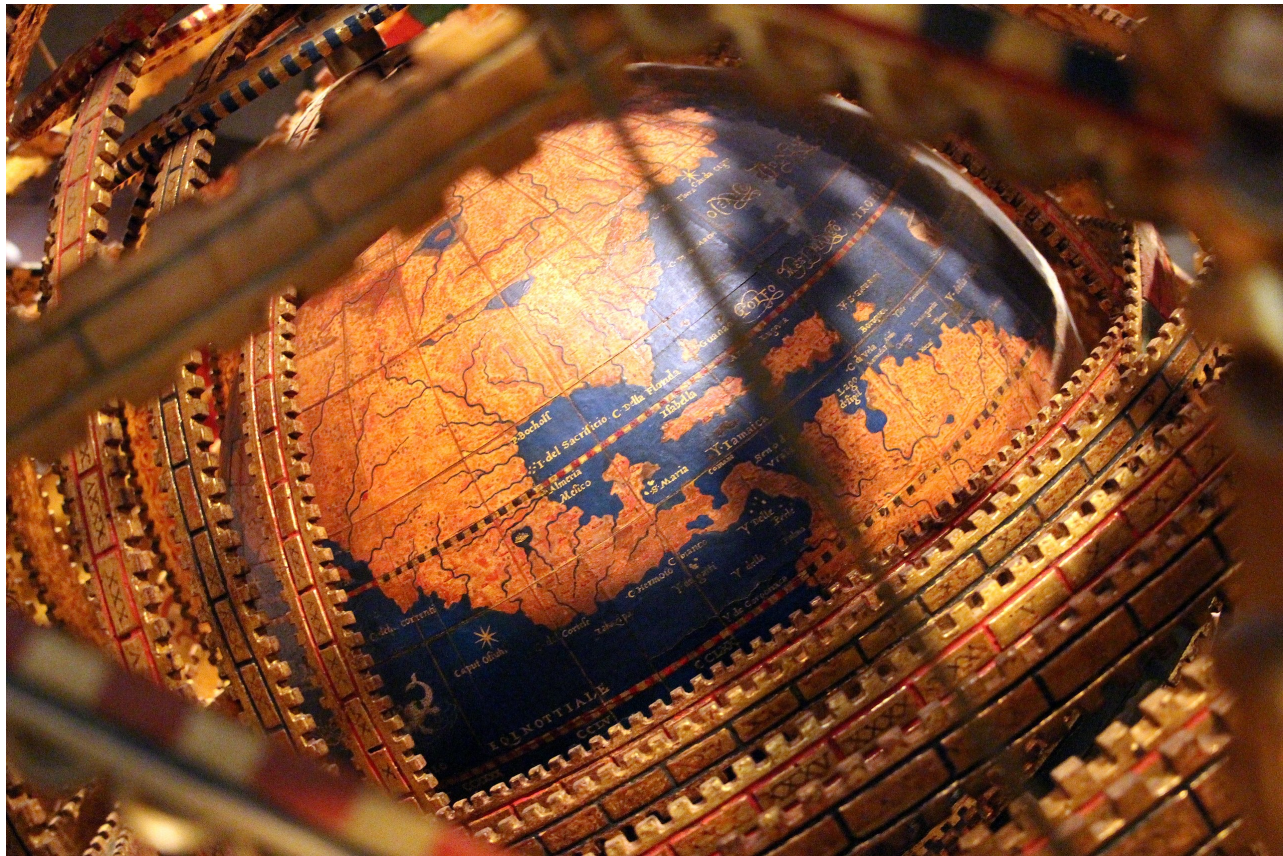
control.princeton.edu


@art_only
©Aryeh Nirenberg

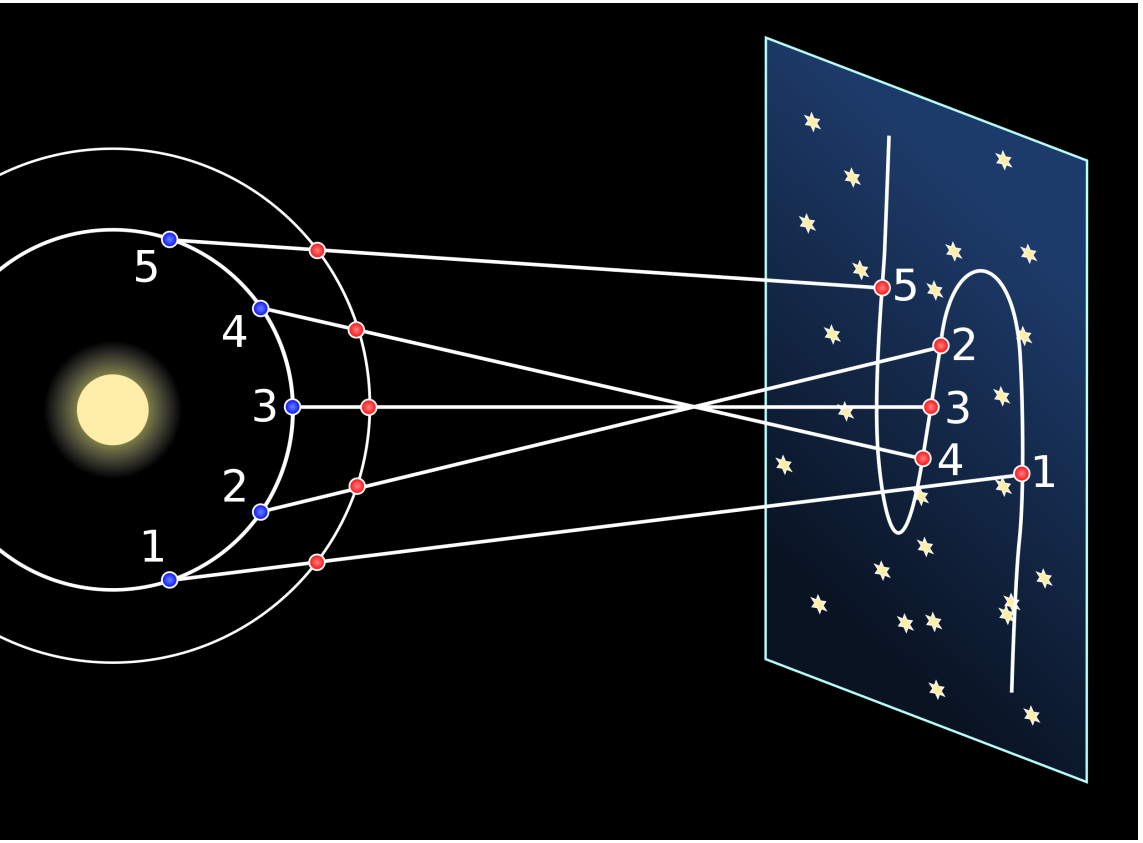
7 Heavenly Bodies



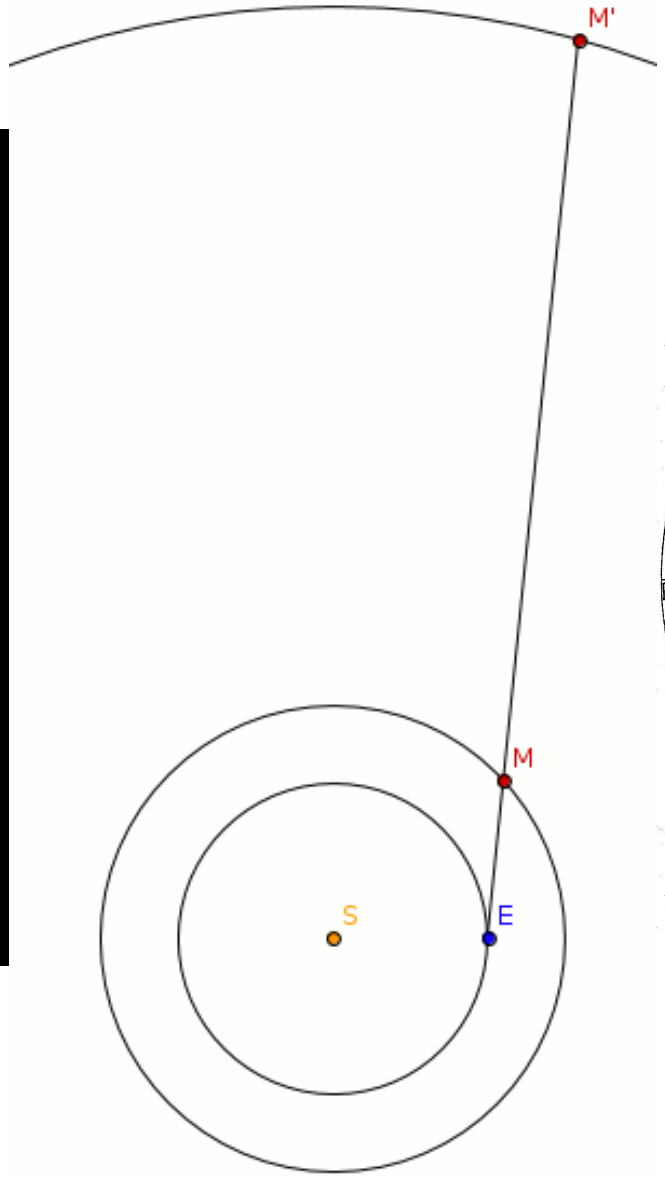
Armillary Sphere (Geocentric)

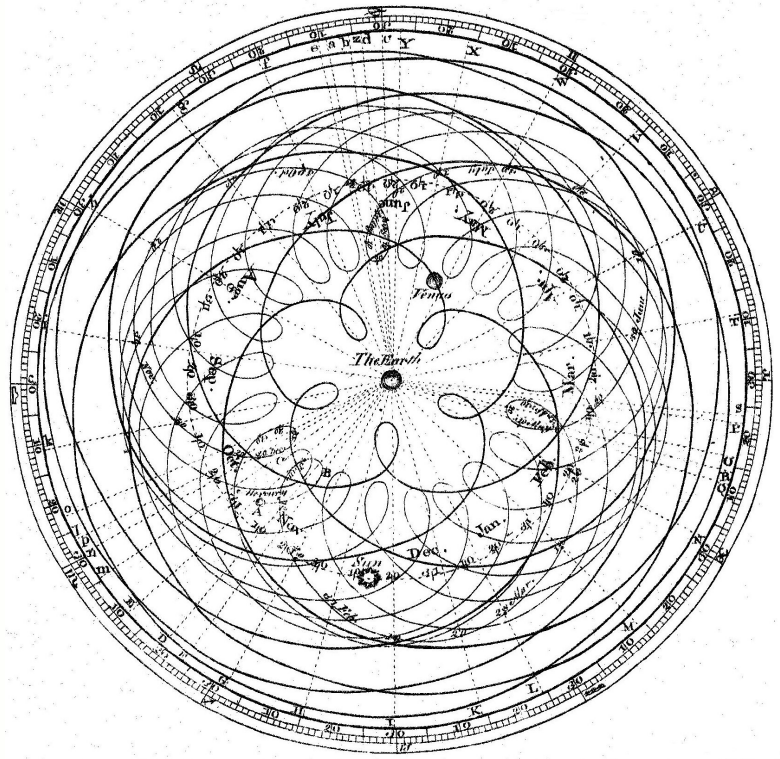


Better System Identification: Geocentric ➔ Heliocentric



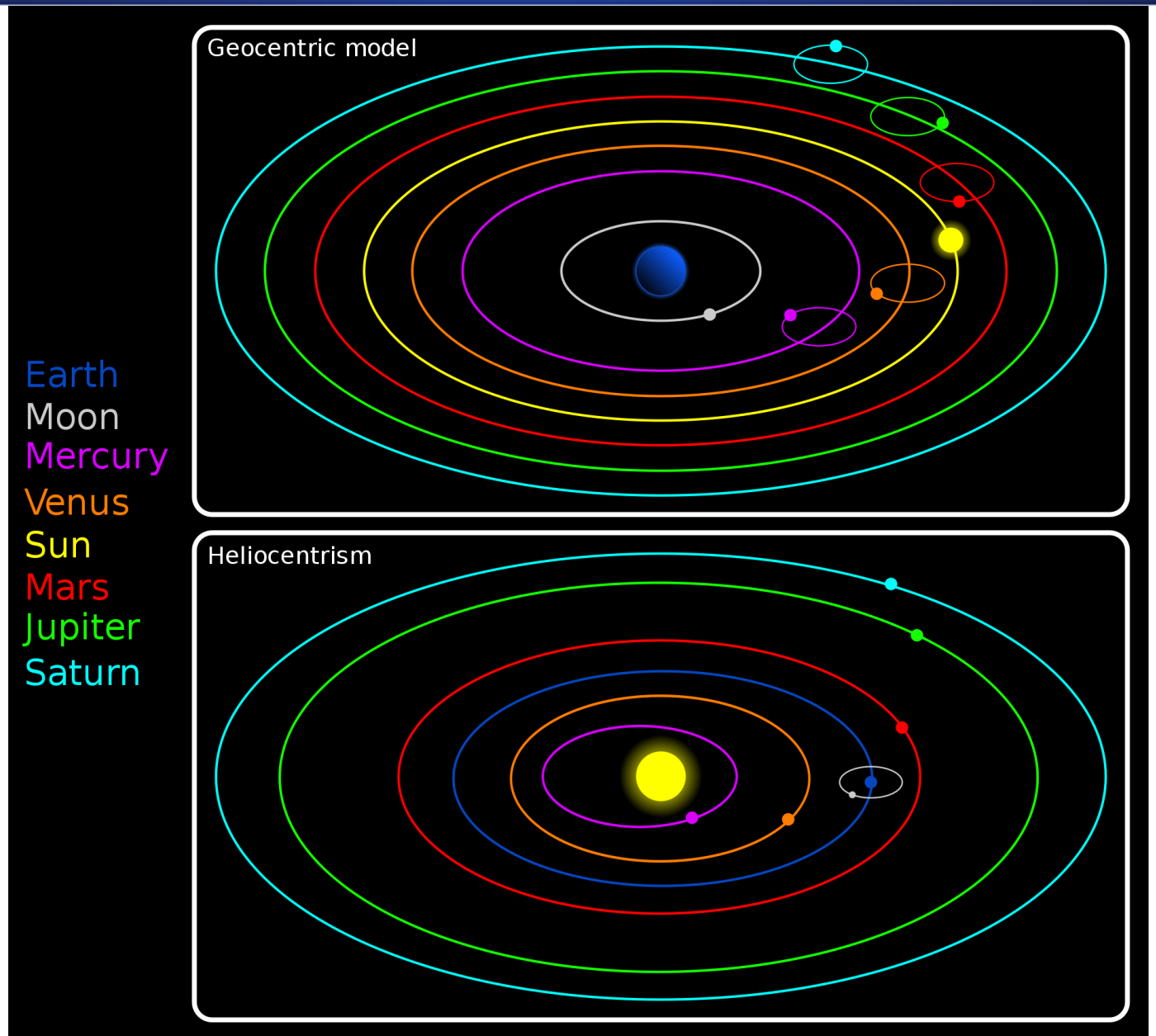
Retrograde motion of Mars





Epicycles (~Fourier)

Better System Identification: Geocentric ➔ Heliocentric



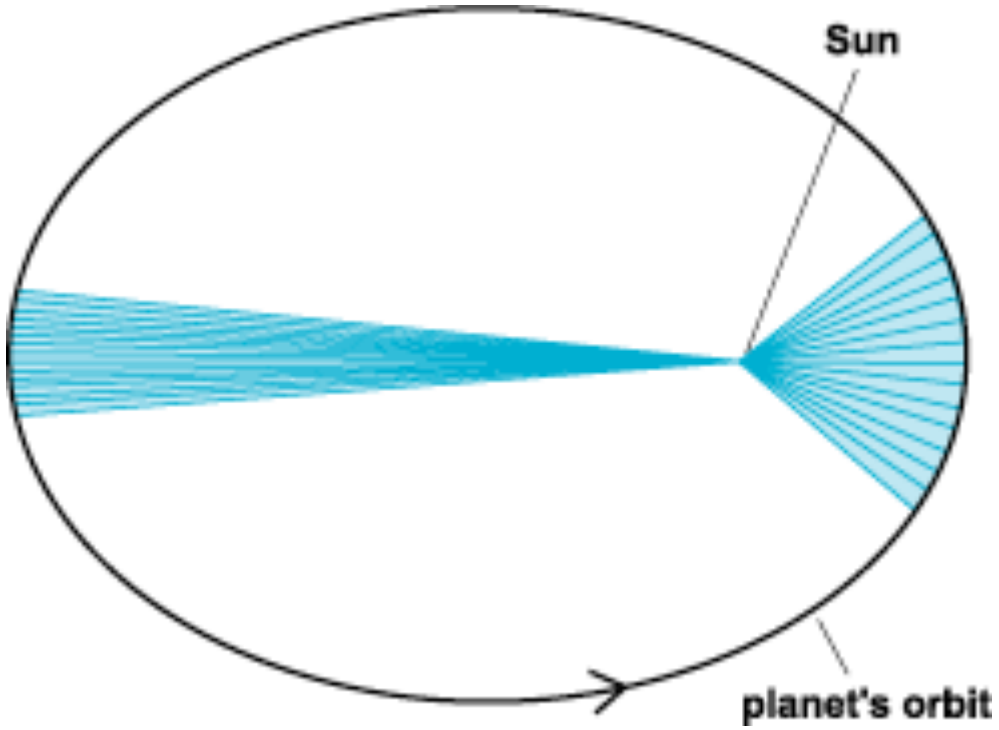
Even Better System Identification

- **System Id: Identify the internal dynamics and effect of the actuator on the state of a system.**

y= f(x,u) or

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$$

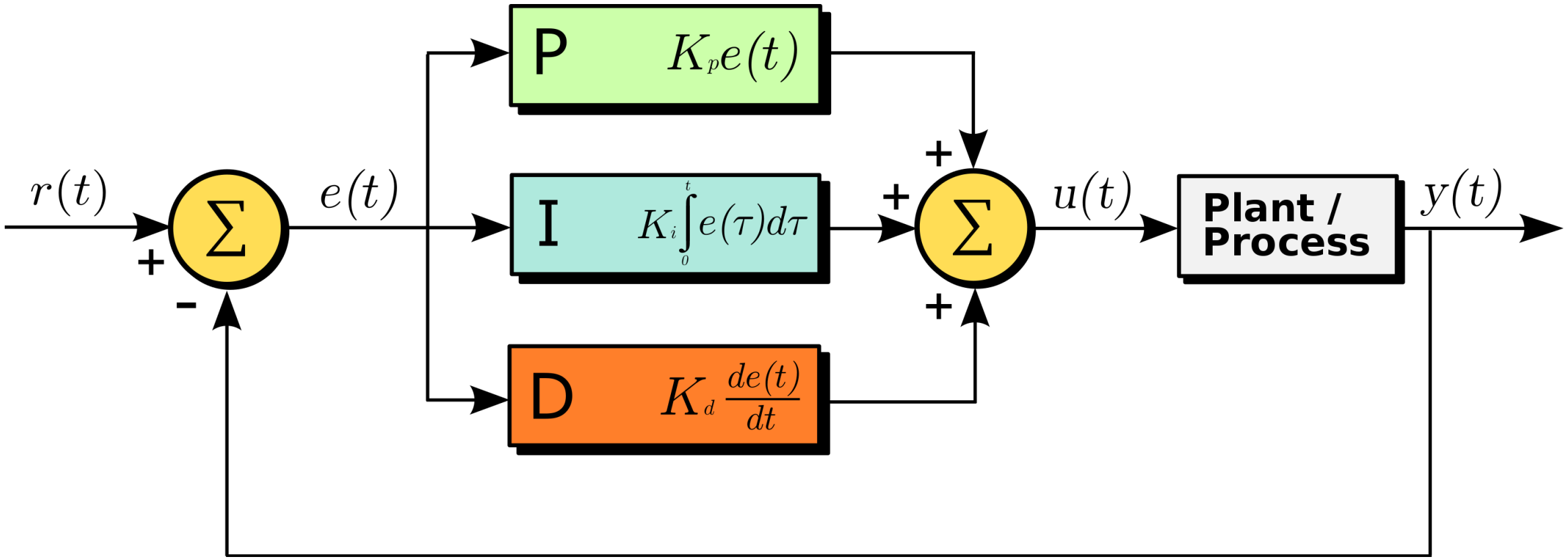
$$\mathbf{y} = g(\mathbf{x}, \mathbf{u})$$



$$\Rightarrow \ddot{\mathbf{r}} = \frac{GM}{\mathbf{r}^2}$$

If the Aim is Control: >99% use PID! ➔ Need Data to Tune P, I and D only.

- For >99% industrial control PID is enough!
- For most control, you need enough info (System Identification) to tune a PID.
- System Id to get enough data/info to tune P, I and D.

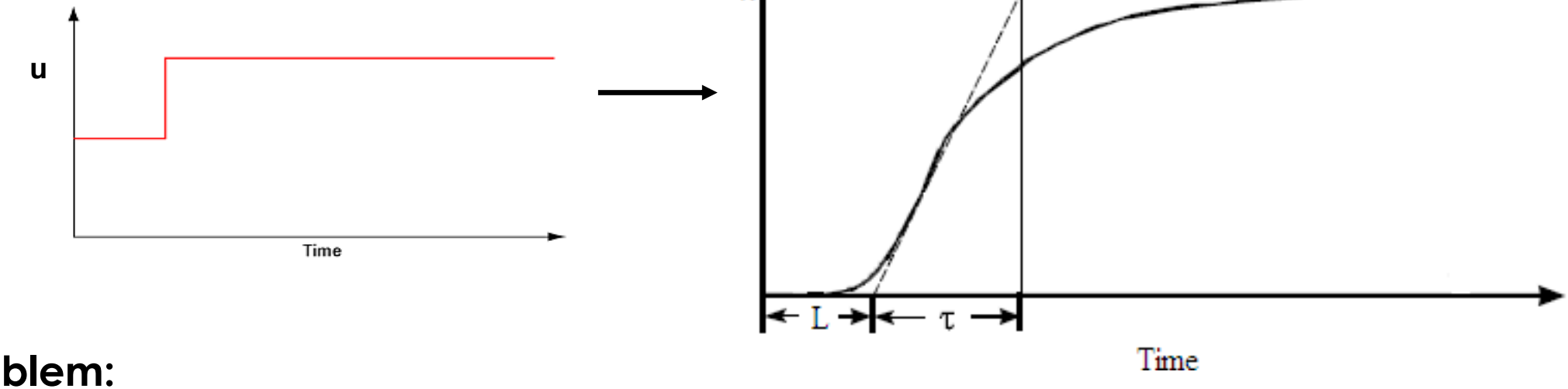


How to get data? Experimental System ID (Open Loop)

- **For >99% industrial control PID is enough! For most control, you need enough info (System Identification) to tune a PID.**
- **System Id: First order ODE with time delay**

$$\dot{y}(t)T + y(t) = Ku(t - L) \quad \rightarrow \quad y(t)=K\cdot(1-e^{-(t-L)/T})$$

- **Classic Reaction Curve Method**



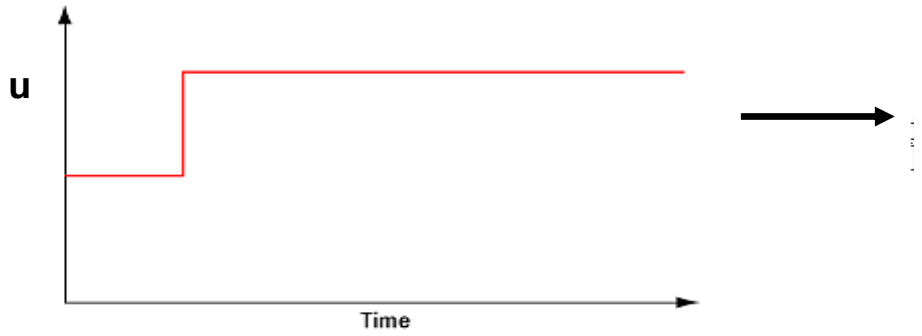
- **Problem:**
  - Many shots needed
  - Need the actuator in open loop.

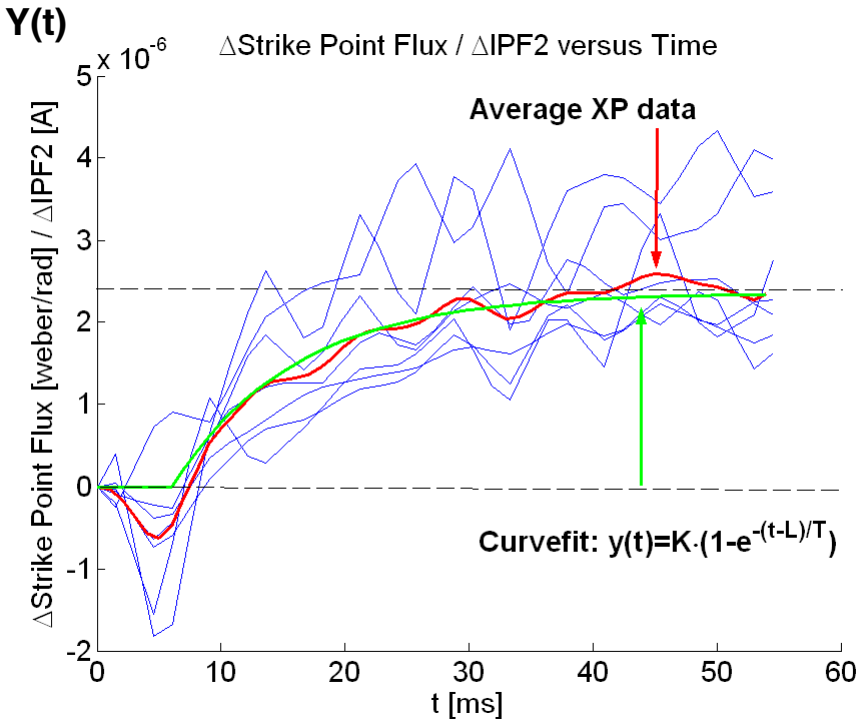
How to get data? Experimental System ID (Open Loop)

- **For >99% industrial control PID is enough! For most control, you need enough info (System Identification) to tune a PID.**
- **System Id: First order ODE with time delay**

$$\dot{y}(t)T + y(t) = Ku(t - L) \quad \rightarrow \quad y(t)=K\cdot(1-e^{-(t-L)/T})$$

- **Classic Reaction Curve Method**





System Id results NSTX (Kolemen 2009)

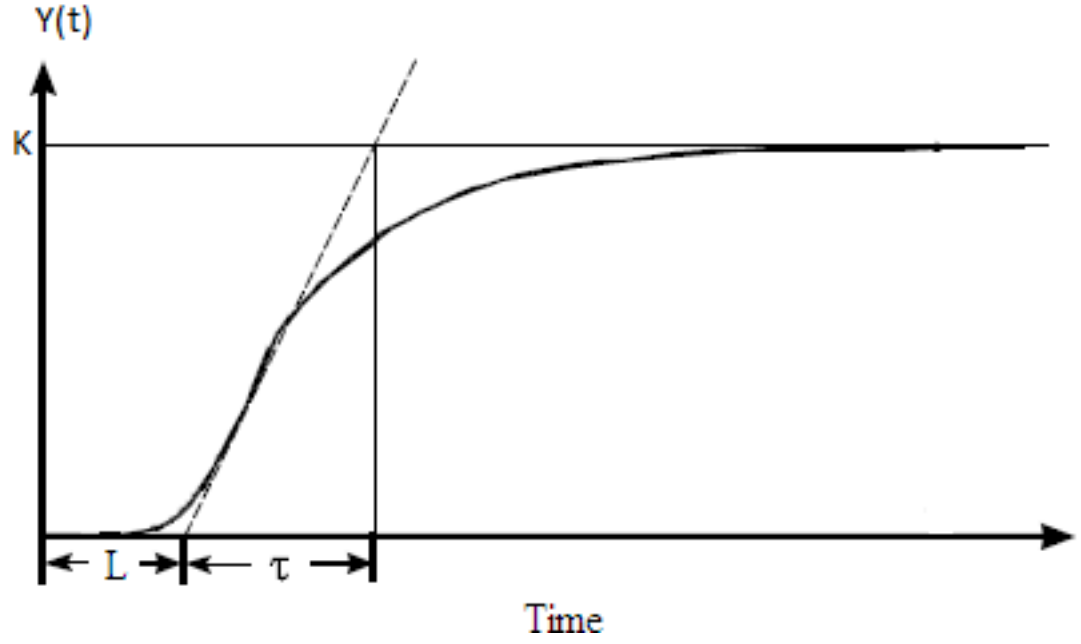
- **Problem:**
  - Many shots needed
  - Need the actuator in open loop.

How to get data? Experimental System ID (Open Loop)

- **[P,I,D] control can be designed are functions of K, L, T.**

$$\dot{y}(t)T + y(t) = Ku(t - L)$$

$$y(t)=K\cdot(1-e^{-(t-L)/T})$$

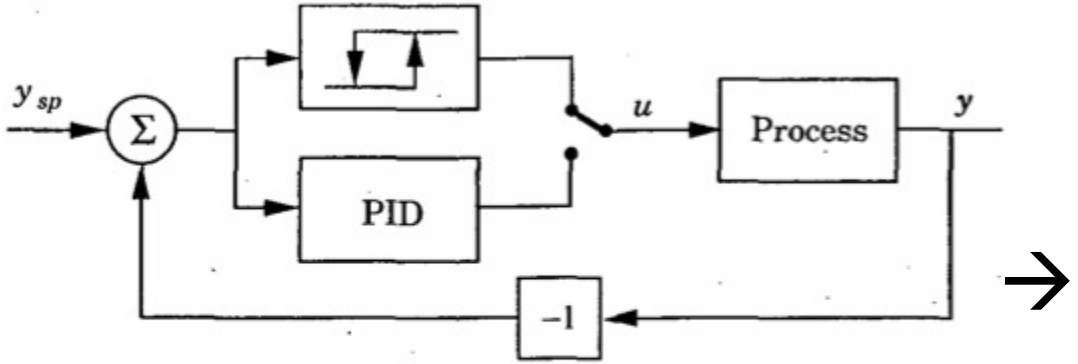


Example Tuning Chart

Cohen Coon	K_c	T_I	T_D
P	$\frac{1}{a}\left(1+\frac{0.35\tau}{1-\tau}\right)$		
PI	$\frac{0.9}{a}\left(1+\frac{0.92\tau}{1-\tau}\right)$	$\frac{3.3-3.0\tau}{1+1.2\tau}L$	
PD	$\frac{1.24}{a}\left(1+\frac{0.13\tau}{1-\tau}\right)$		$\frac{0.27-0.36\tau}{1-0.87\tau}L$
PID	$\frac{1.35}{a}\left(1+\frac{0.18\tau}{1-\tau}\right)$	$\frac{2.5-2.0\tau}{1-0.39\tau}L$	$\frac{0.37-0.37\tau}{1-0.81\tau}L$

$\tau = L/(L+T)$; $a = K_{process}L/T$

Closed Loop System ID:
Closed Loop Auto-tune PID with Relay Feedback

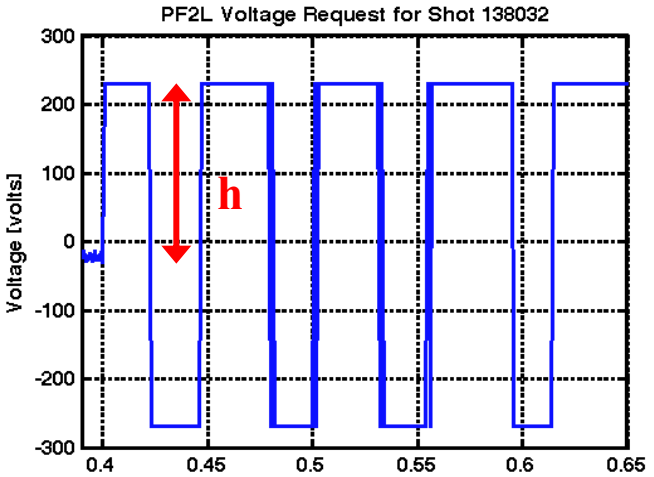


- The closed-loop plant response period (P_u) & amplitude (A) give (for example):

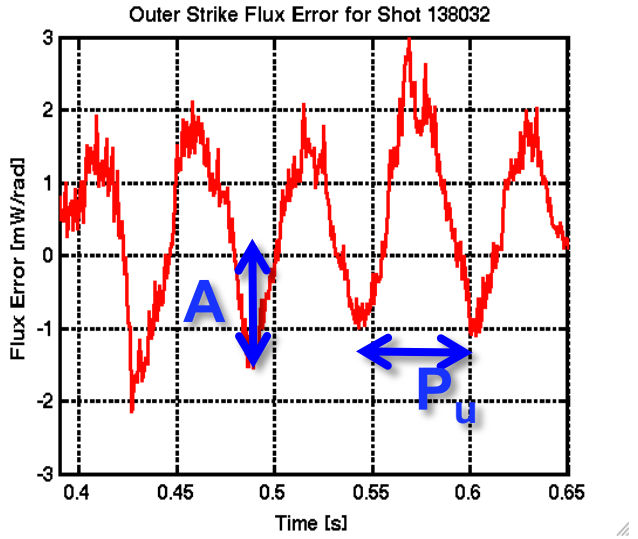
$[P,I,D]=4h/(\pi A)*[0.6, 2/P_u, P_u/8]$

- Advantages:
  - Only a single experiment is needed to tune many different regimes.
  - Closed loop:
    1. More stable
    2. Enable tuning for actuator that can't be open loop (e.g.: Vertical Ctrl, EFC). Methods exist to join with the existing control

Control Output



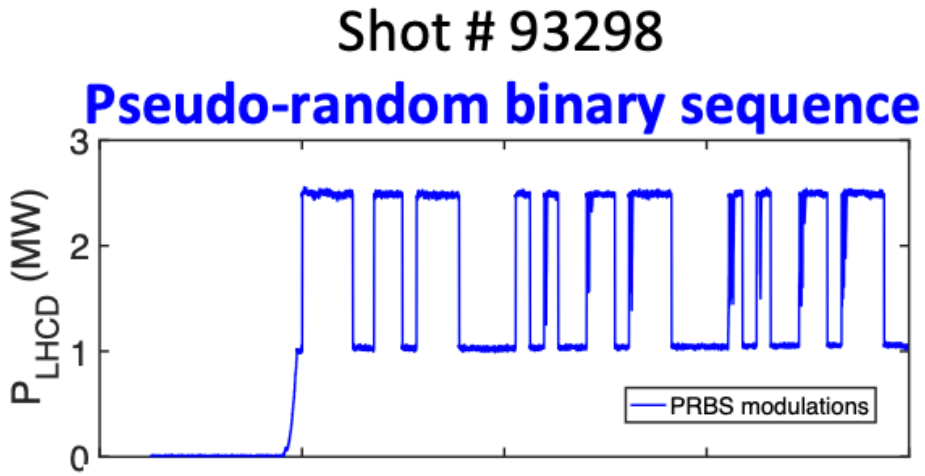
Process Output



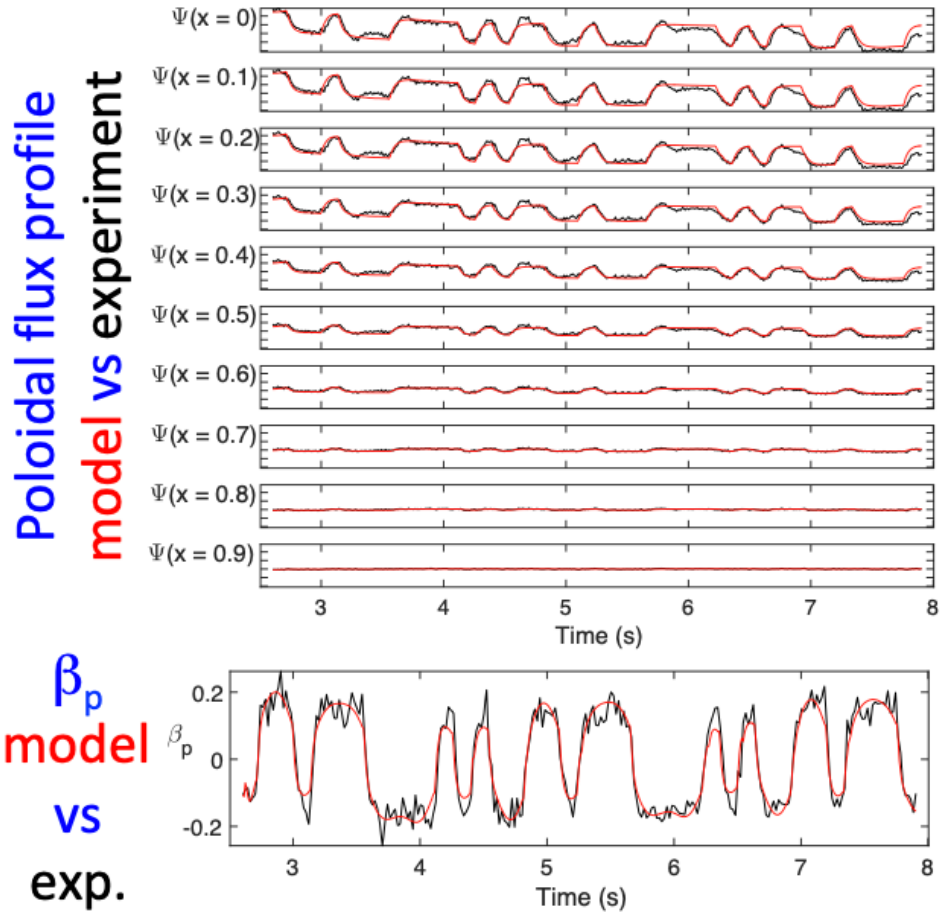
E. Kolemen *et al* 2011
***Nucl. Fusion* 51 113024**

MIMO Linear System Experimental System ID

- Pseudo-random binary sequence
- In essence multiple reaction curves
- SISO ➔ MIMO
- Less noise, but plasma might be evolving



➔

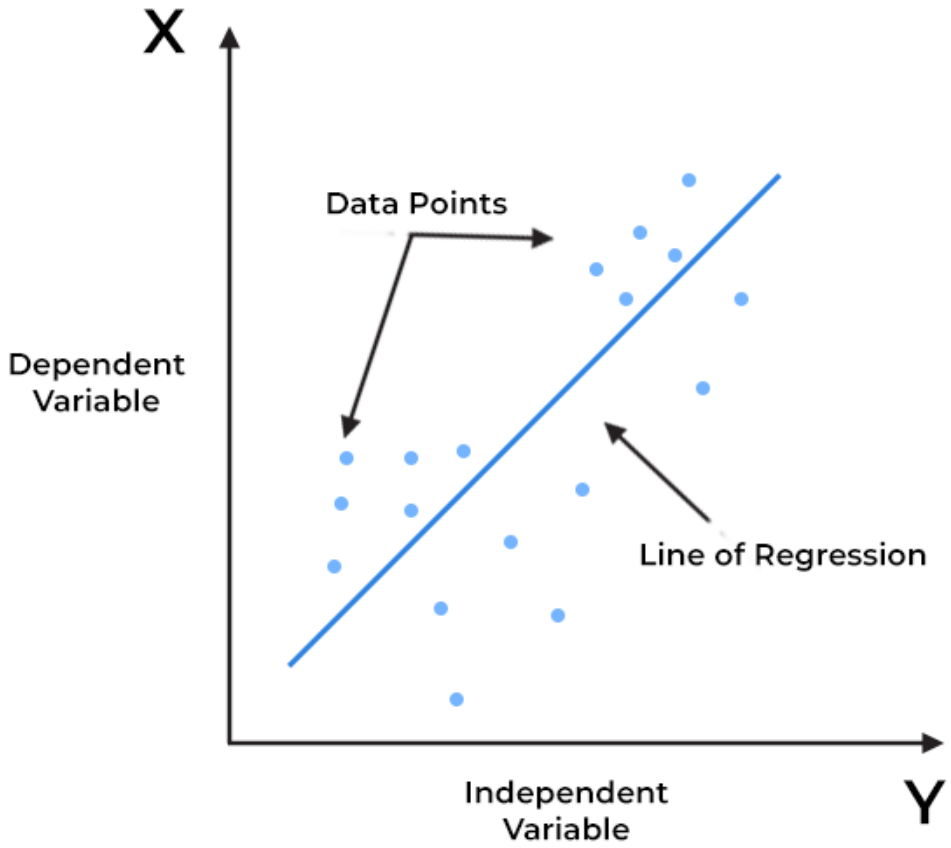


D. Moreau, "Model-Predictive Kinetic Control Experiments on EAST", IAEA, 2021

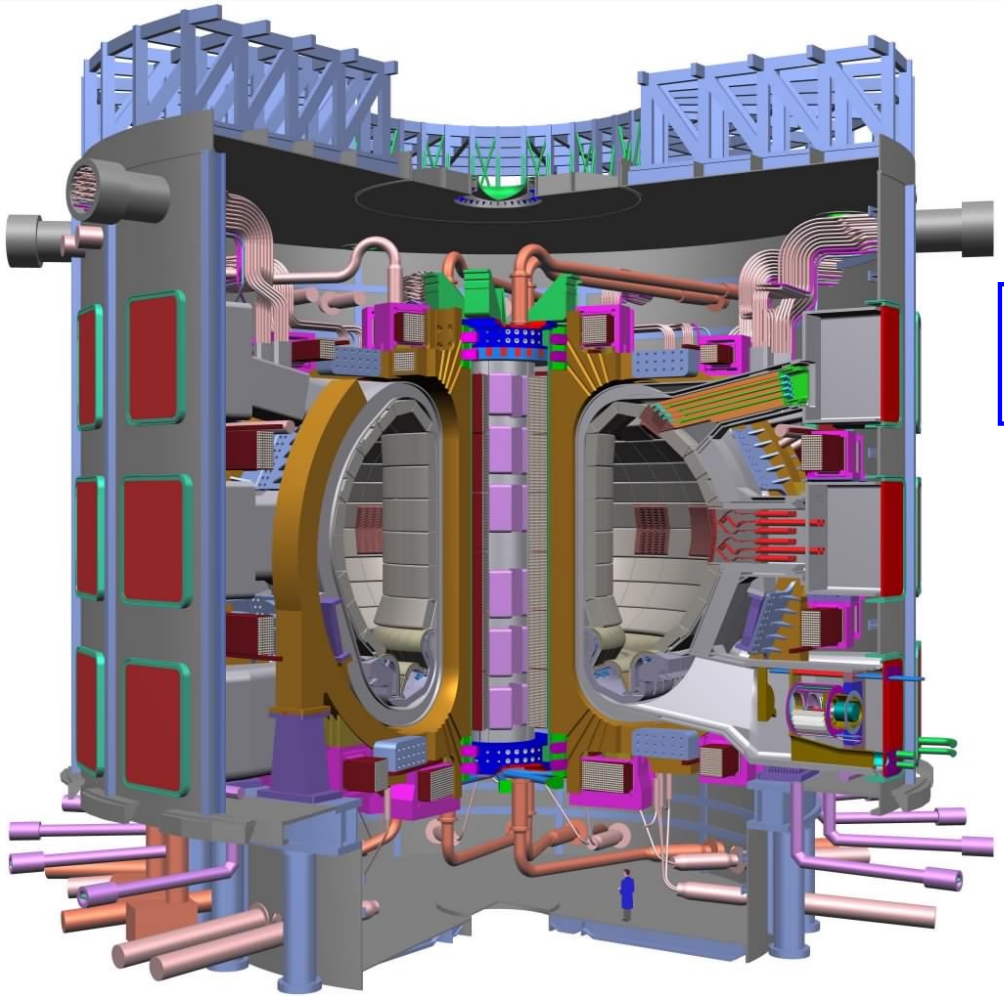
MIMO Linear System Experimental System ID

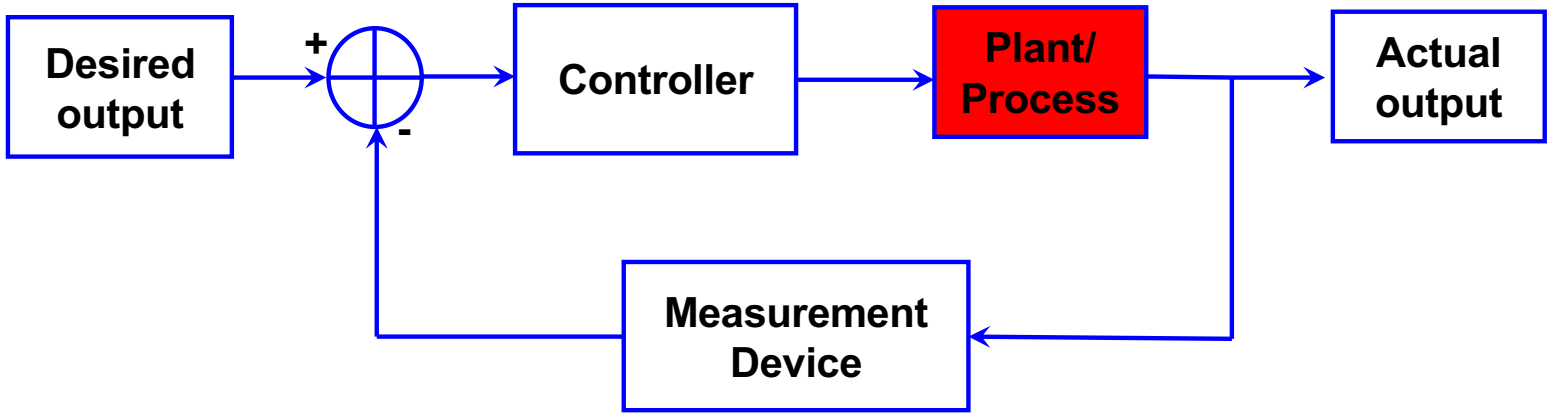
- Obtain u and y measurements
- MIMO Linear State Space Model can then be obtained by least squares methods
- model = ssest(data,n); %Matlab command
- Then use a linear control (e.g. LQR)

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k + Du_k$$



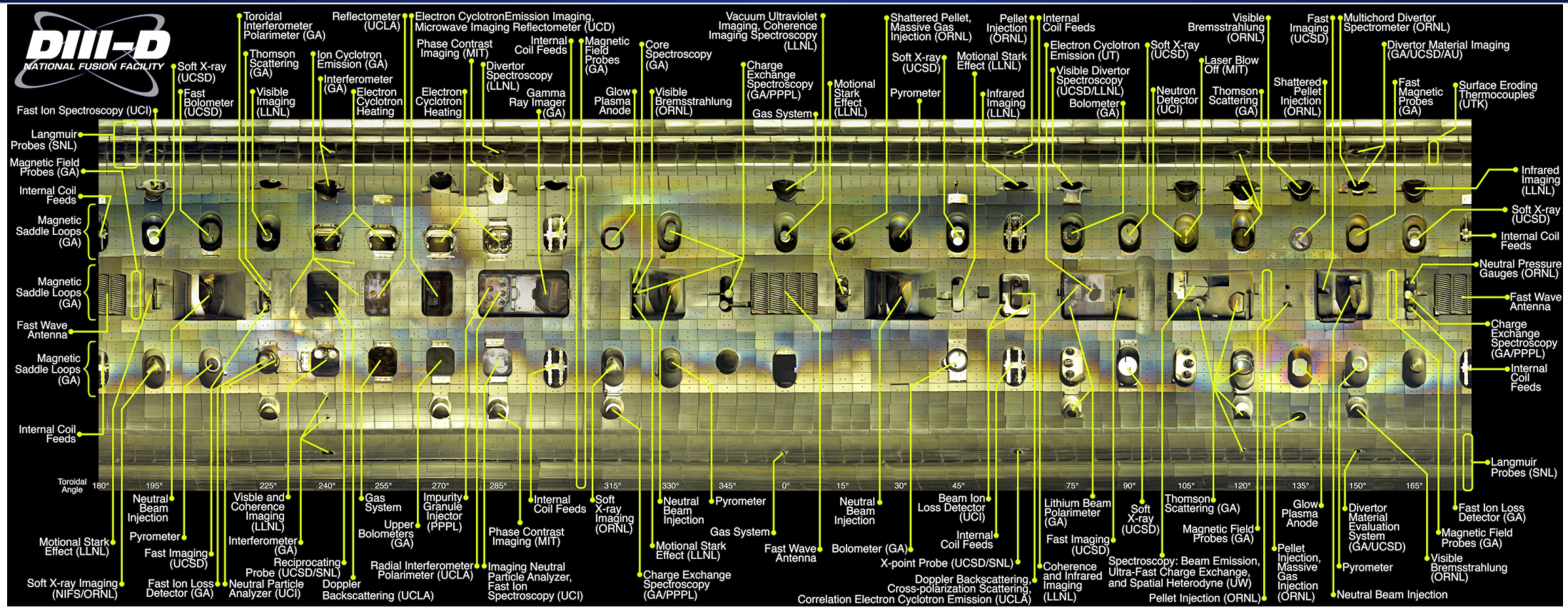
Data-Based Control for Fusion





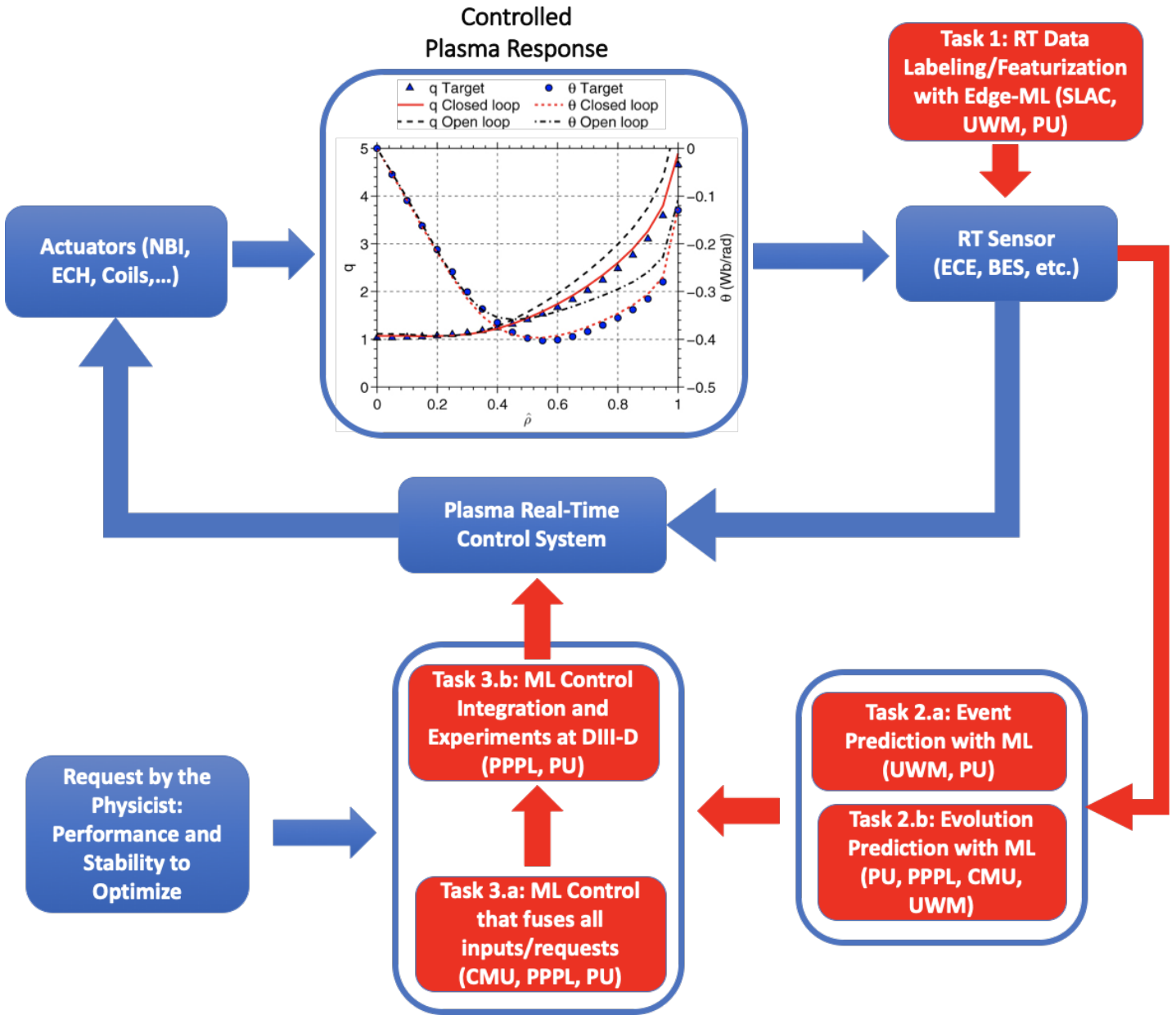
- Fusion plasmas/reactors has very complicated nonlinear physics
- There is a lot of diagnostic measurement
- Prime target for data-based control design!

Fusion Has Huge Amounts of Nonlinear Data: How to Utilize This for Control?



- **How can we bring this immense information into control?**
  - ***Many not available or usable in real-time (RT)***
  - ***Too much data to pass to a central CPU***
  - ***Mostly not automated: Post-discharge analysis by physicists***
- **Machine Learning ➔ RT data featurization + automated analysis + control design**

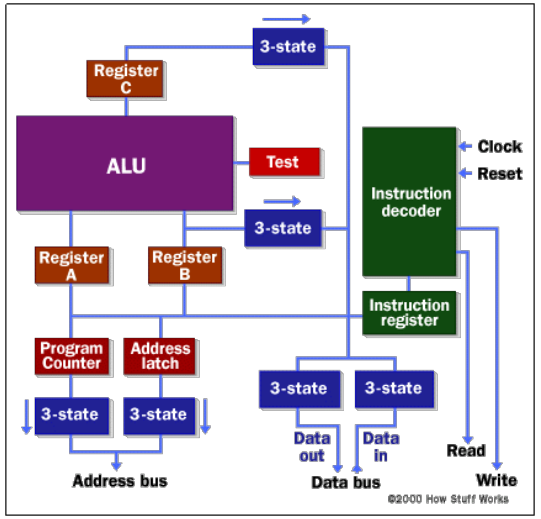
Machine Learning for Real-time Fusion Plasma Behavior Prediction and Manipulation



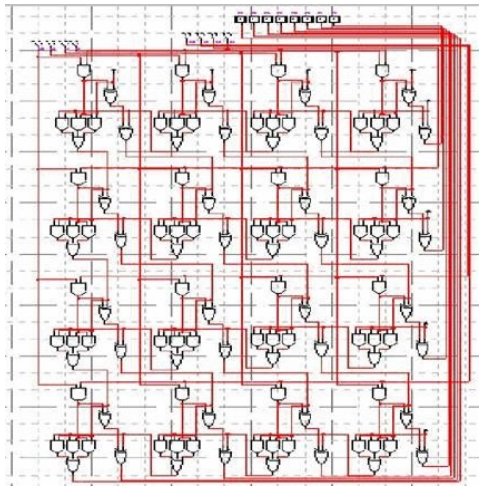
- **DOE ML for Fusion report identified ML for Fusion Control as priority**
- **A multi-institutional collaboration of *a) ML/AI Scientists, b) Diagnosticians and c) Fusion Control***
- ***Experts:***

CMU, PPPL, PU, SLAC and UW-M

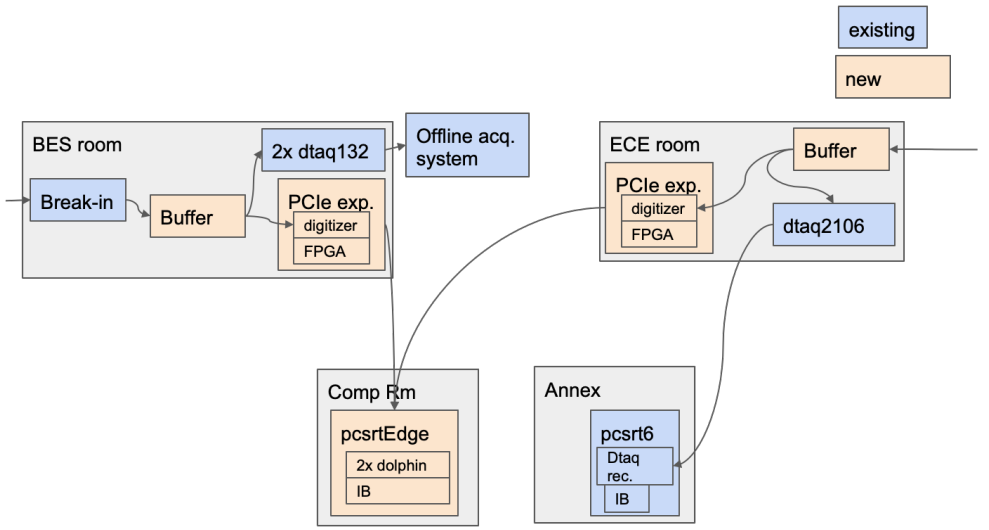
Real-time measurements/calculations



CPU/GPU: instruction set, registers, buses, addresses



FPGA: bits flowing through gates



- **Extend RT control to fast plasma dynamics and fluctuation diagnostics**
  - Fluctuation diagnostics capture fast plasma dynamics with MHz sampling
  - RT calculations on ~10-100 signals at ~1 MHz is not feasible on CPU/GPU → requires FPGA at sensors – **"Edge ML"**
    - Highest throughput on FPGA imposes restrictions on algorithm/model architecture
    - Calculation output captures information about fast plasma dynamics and output is available to downstream plasma control system
  - Diagnostics: BES, interferometers, ECE
  - Physics applications: ELM onset prediction, characterization of turbulence and confinement mode, Alfven eigenmodes

What is machine learning?

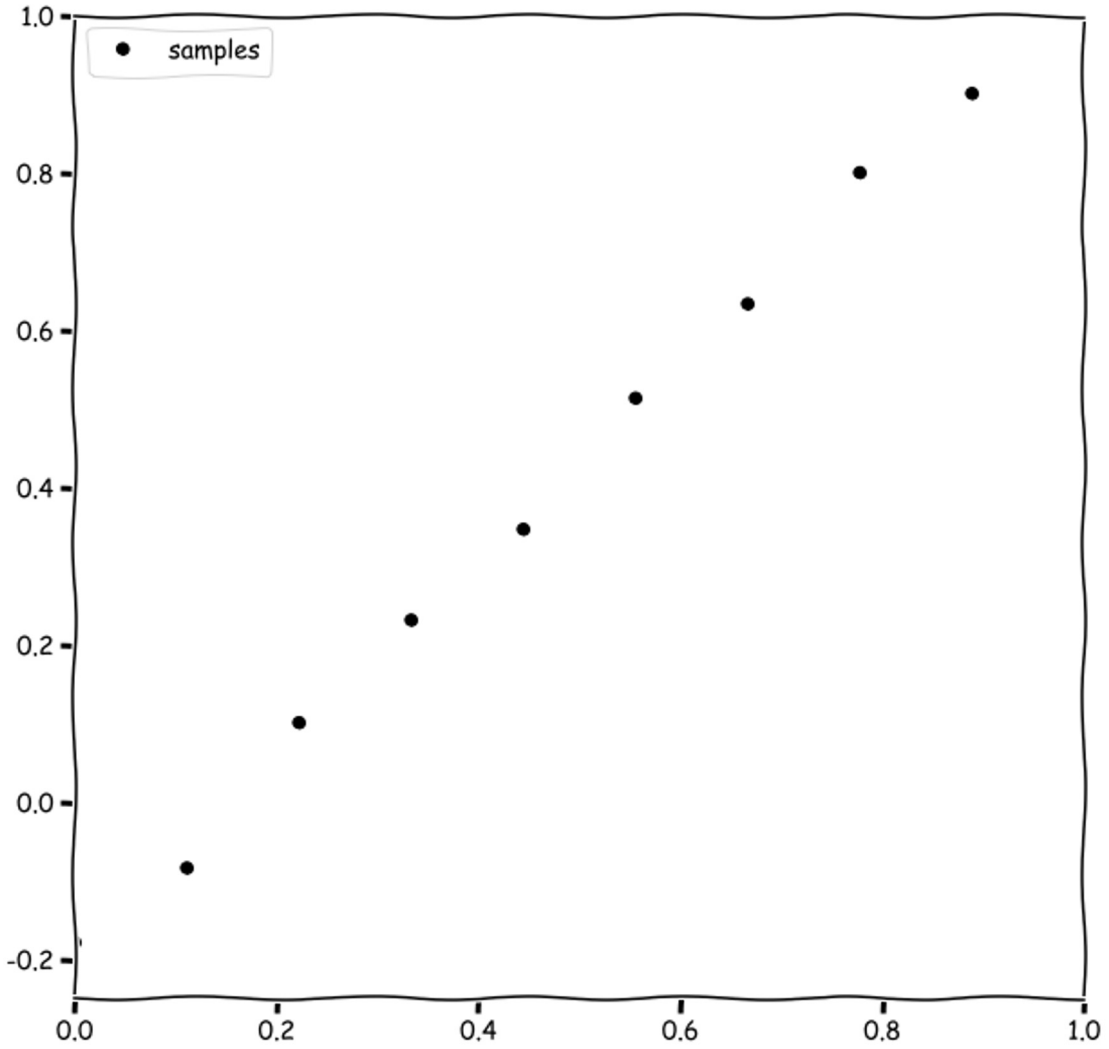
Consider y = f(x)

	Known	Known	Want to Find
"Standard" problem	f(x) (function)	x (input)	y (output)
"Inverse" problem	f(x) (function)	y (output)	x (input)
"Learning" problem	x (input)	y (output)	f(x) (function)

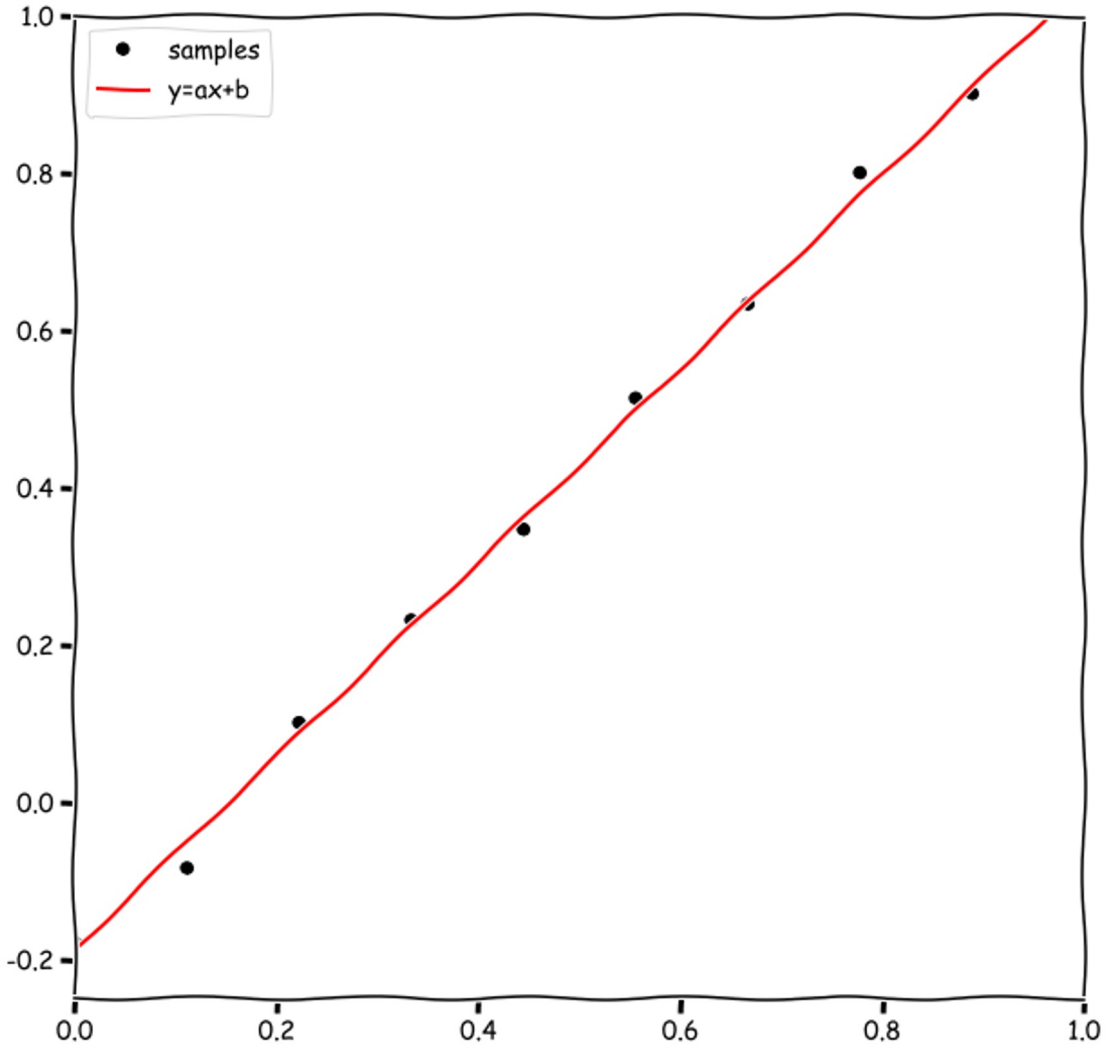
Know data + function, want to find more data

Only know data, need to find the right function

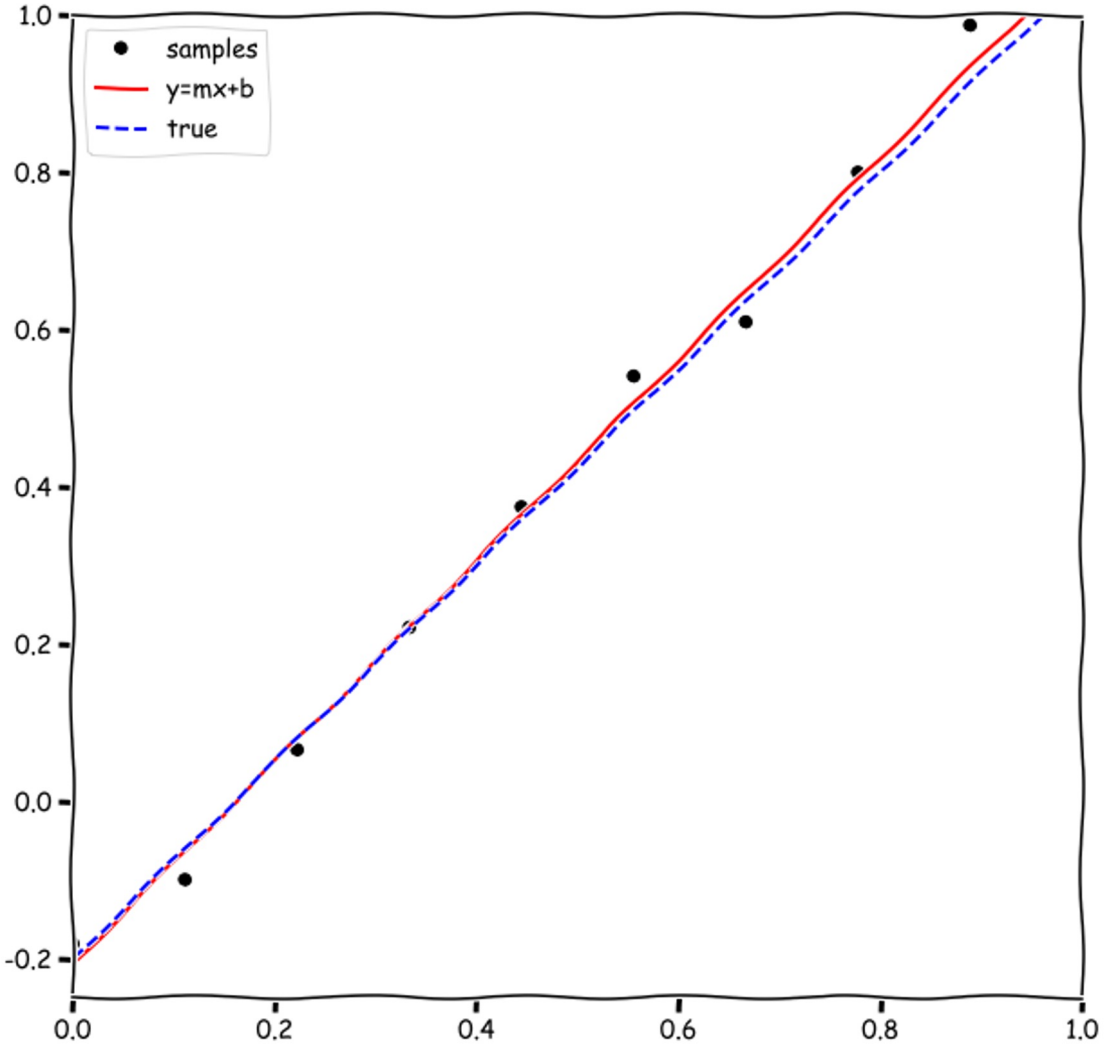
So you have some data



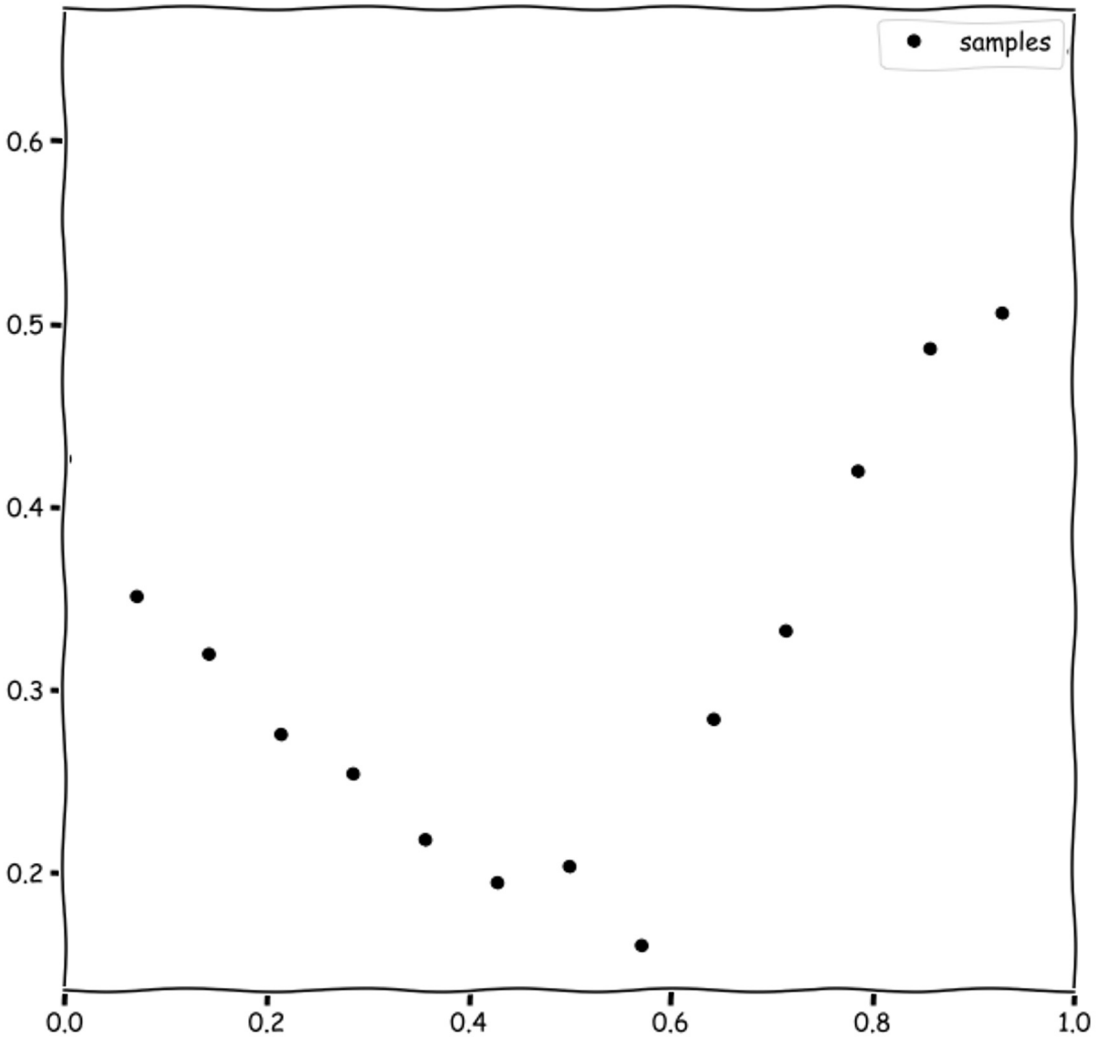
Linear modeling



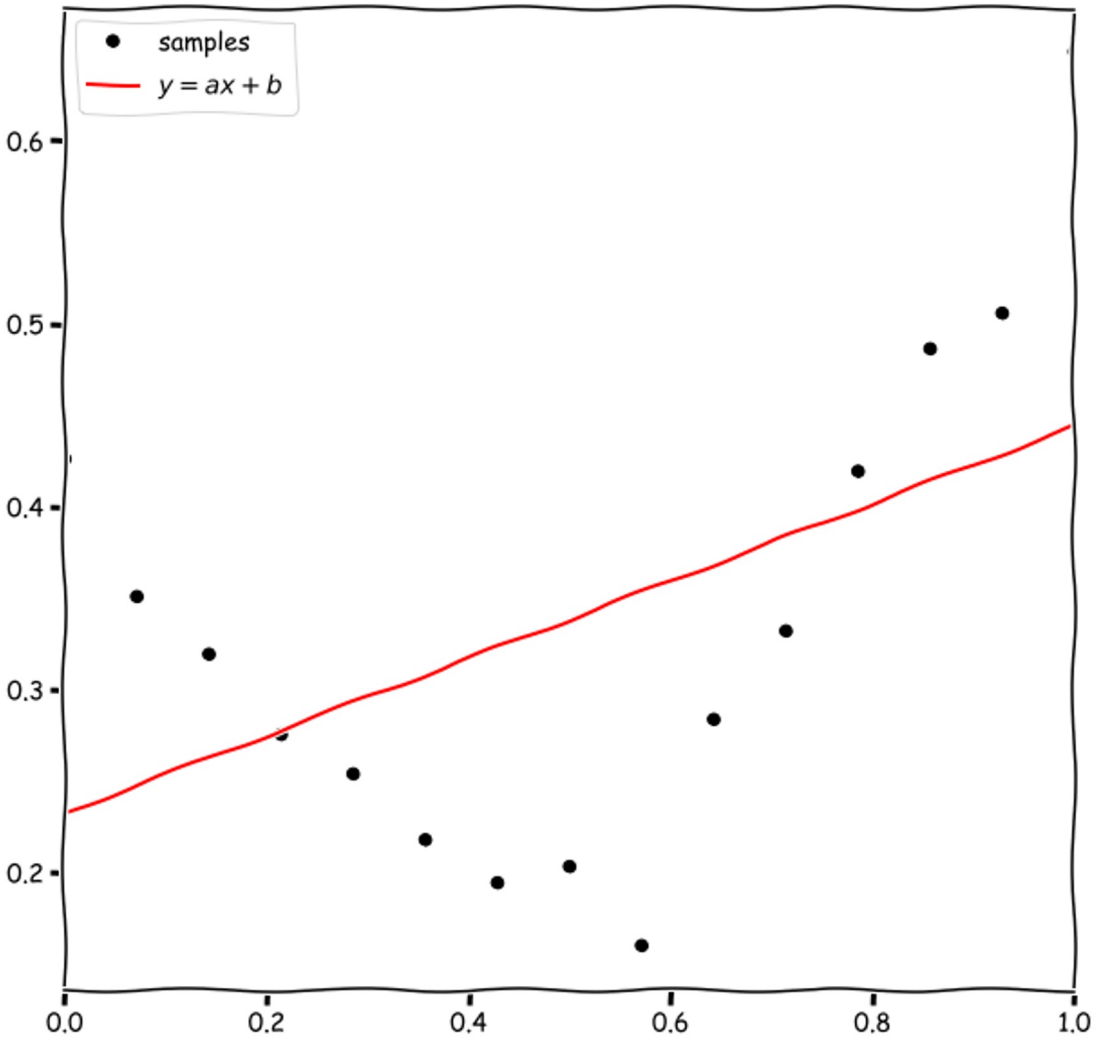
And it mostly worked!



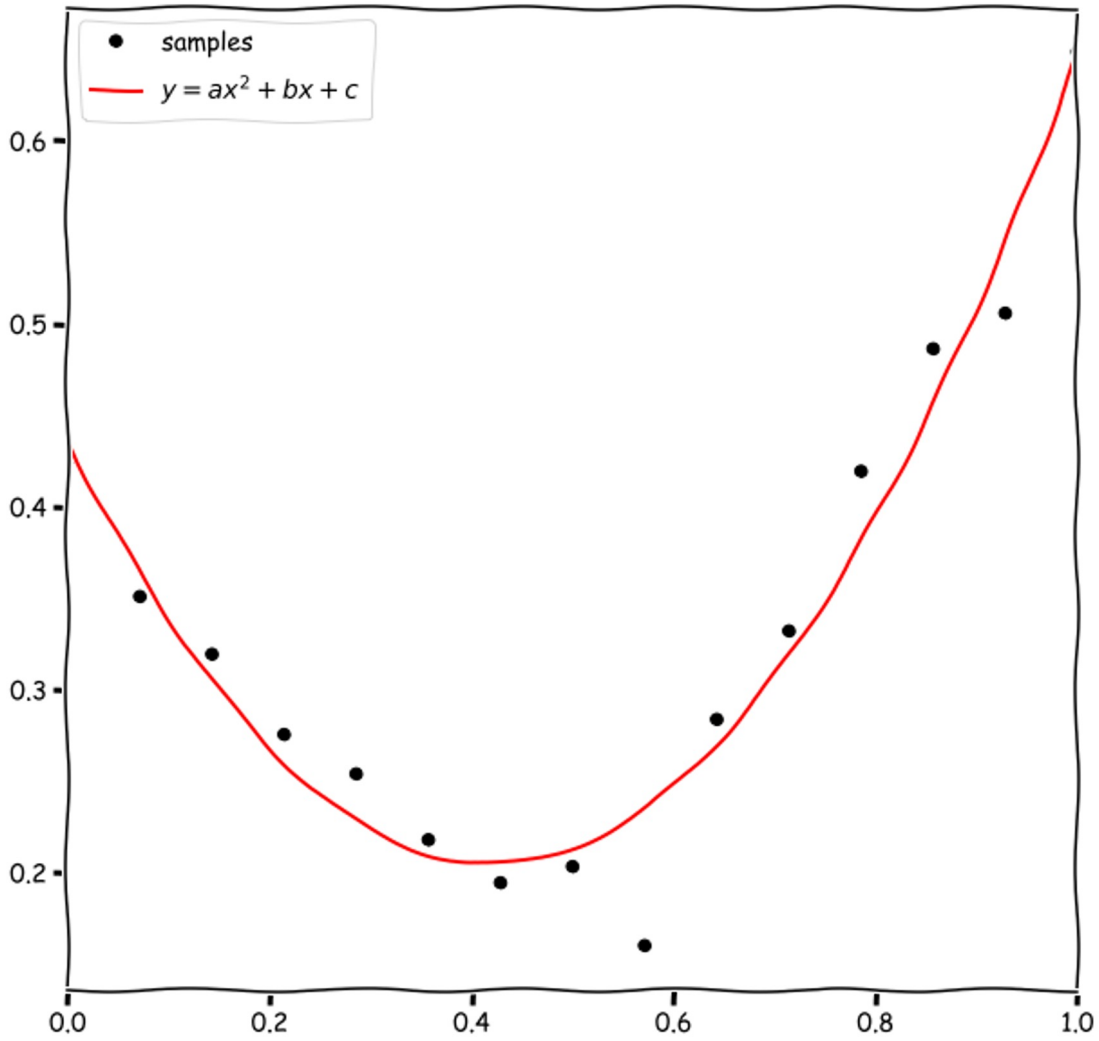
A bit harder



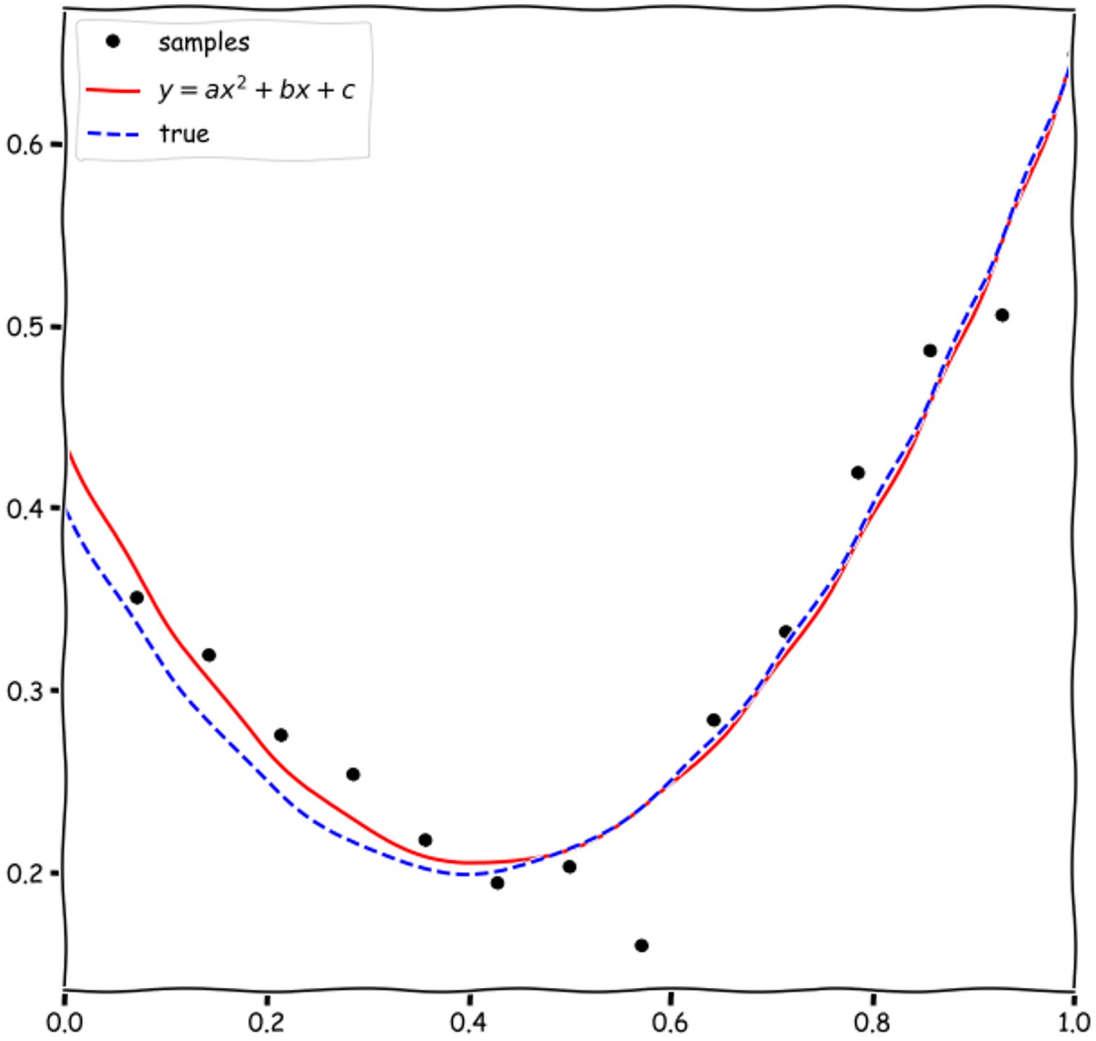
Linear fit: not so great...



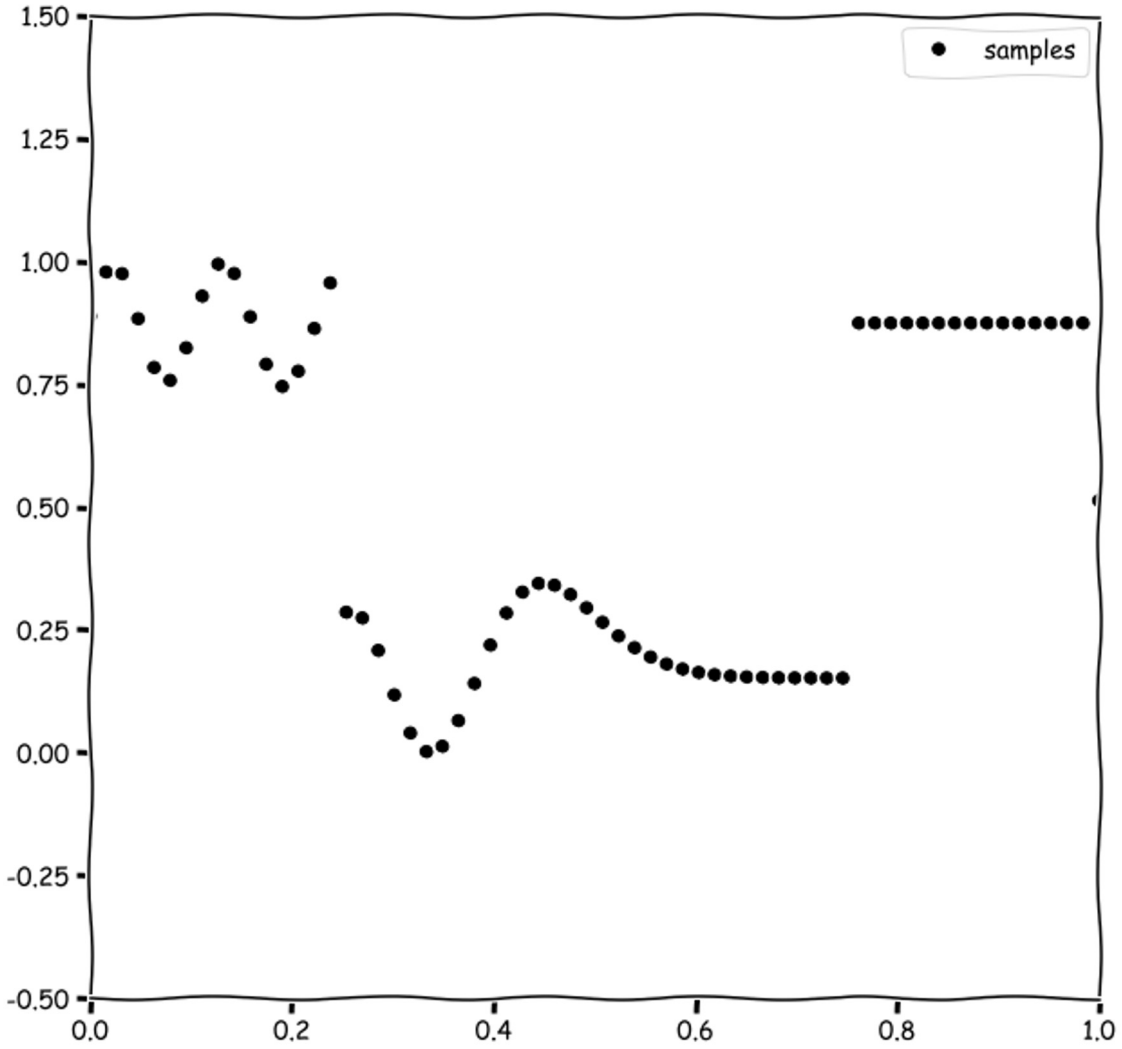
Oh hey it looks quadratic:
Gray-box system identification, give the structure



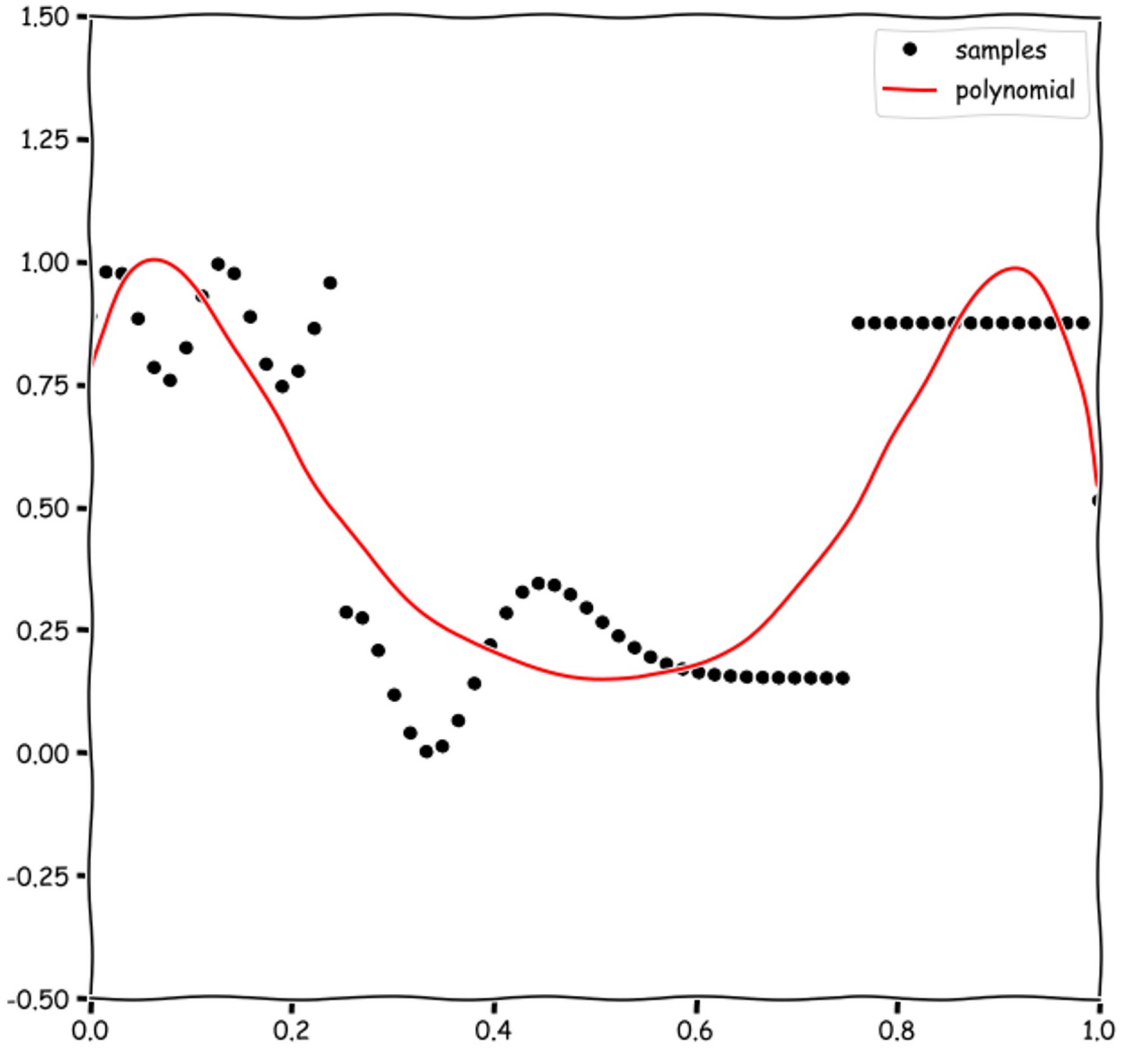
It really is!



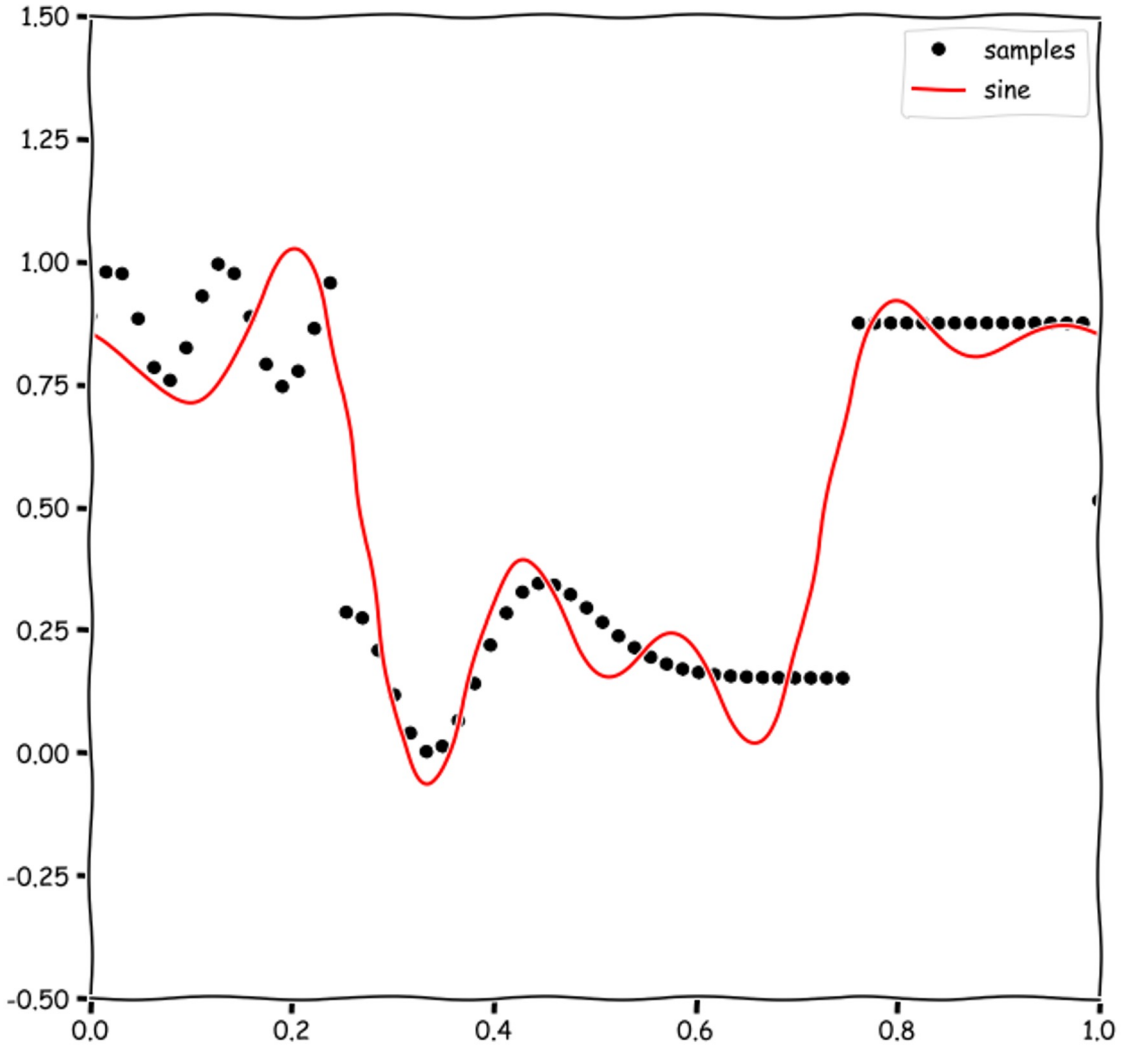
Let’s go really nonlinear



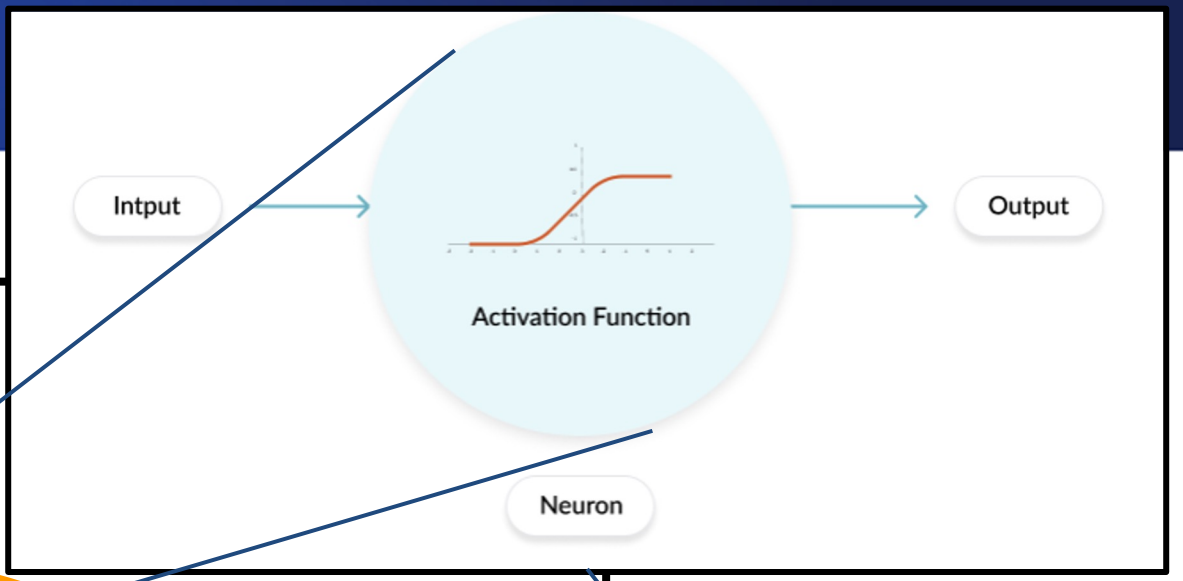
Polynomial fit: nope



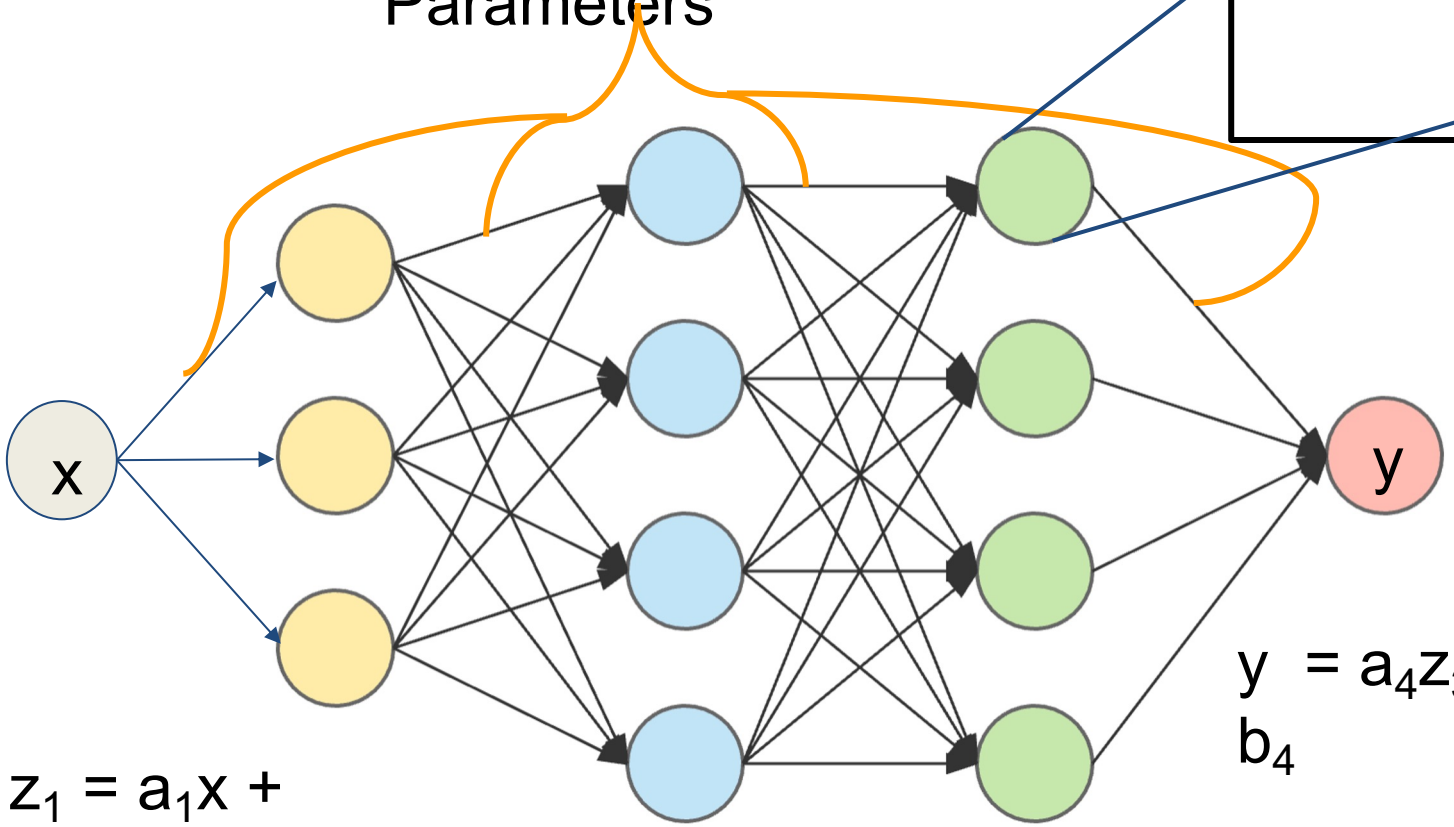
Sine wave: better, but still nope



Maybe a Neural Network?



Weights / Parameters



$z_1 = a_1x + b_1$

$z_2 = a_2z_1 + b_2$

$z_3 = a_3z_2 + b_3$

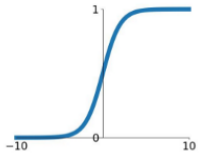
$y = a_4z_3 + b_4$

$y = f(x)$

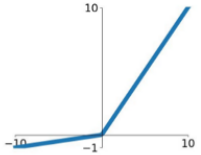
Maybe a Neural Network?

Activation Functions

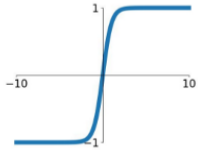
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



Leaky ReLU
$\max(0.1x, x)$

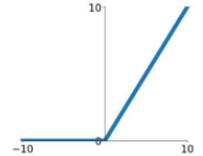


tanh
$\tanh(x)$

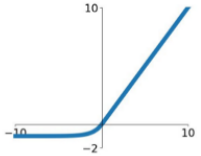


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

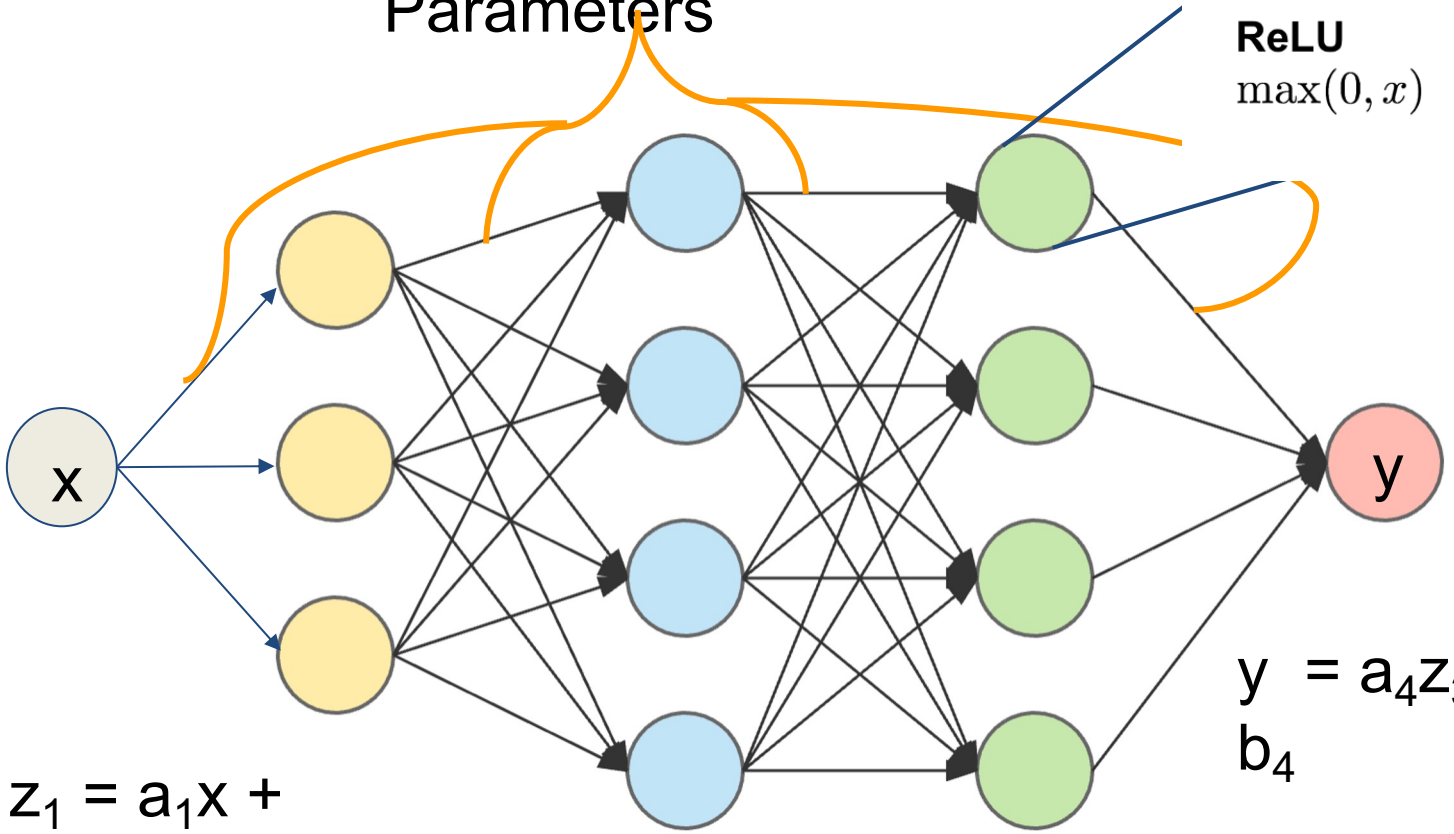
ReLU
$\max(0, x)$



ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Weights / Parameters



$z_1 = a_1x + b_1$

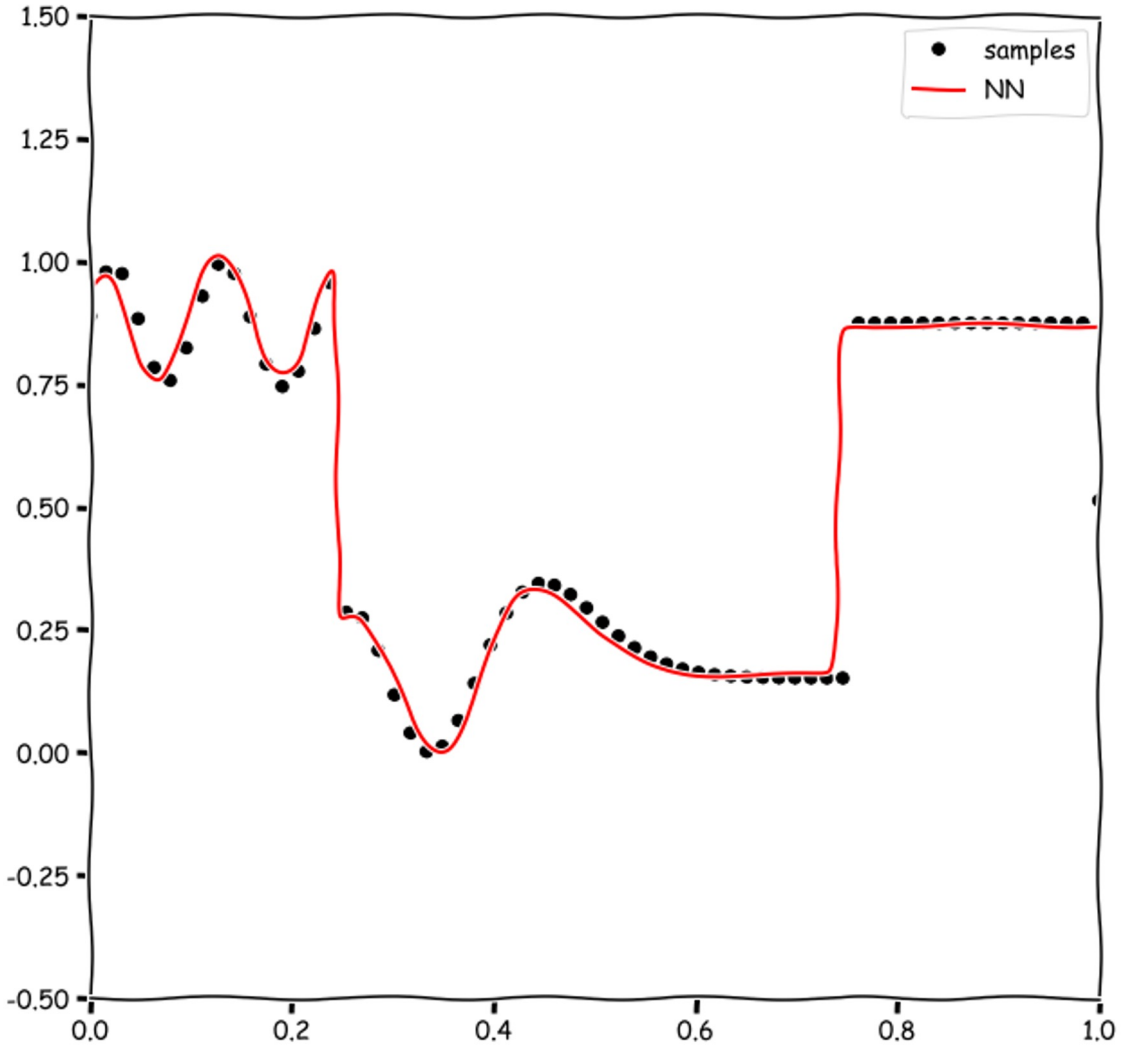
$z_2 = a_2z_1 + b_2$

$z_3 = a_3z_2 + b_3$

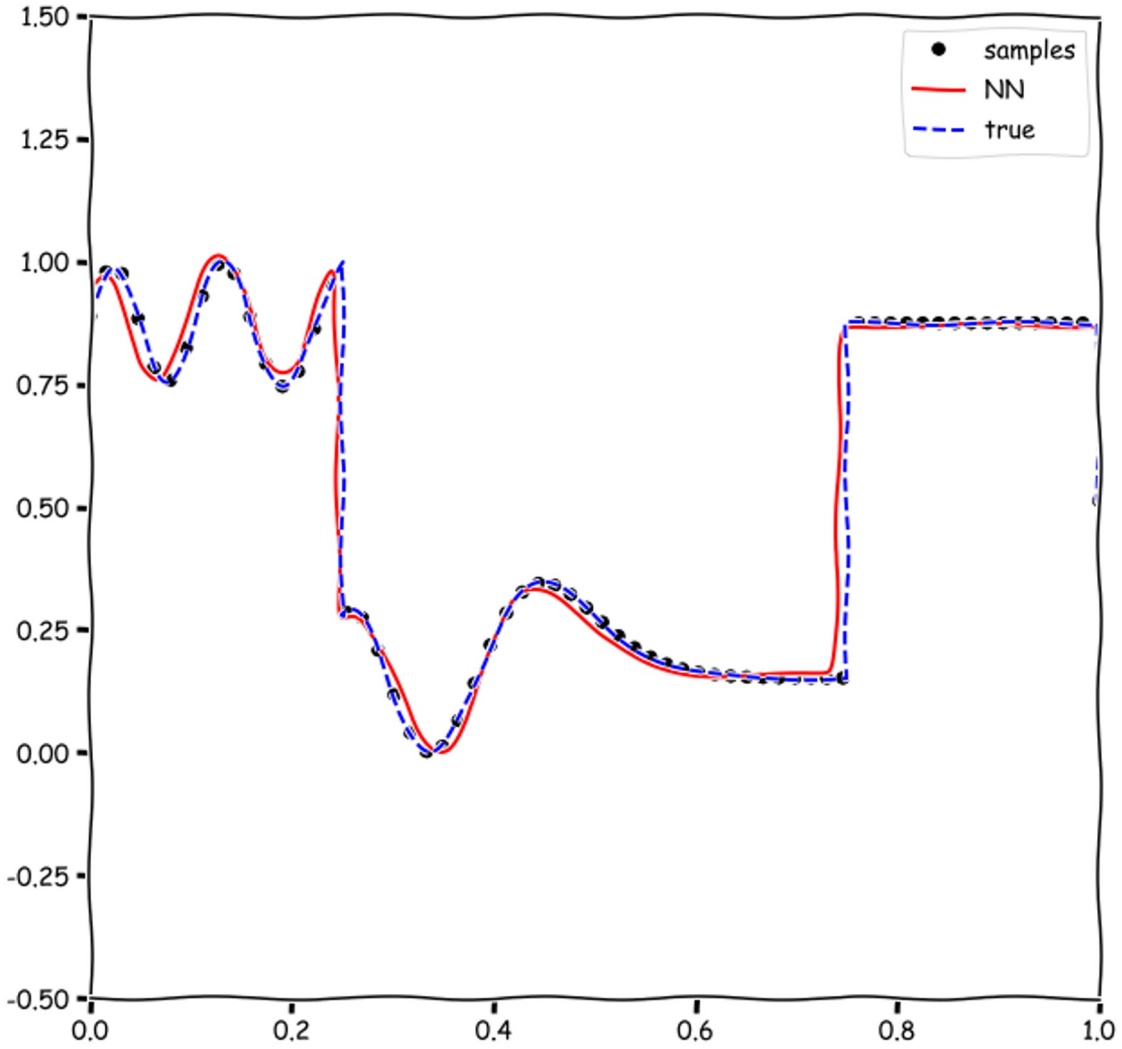
$y = a_4z_3 + b_4$

y = f(x)

Works well!

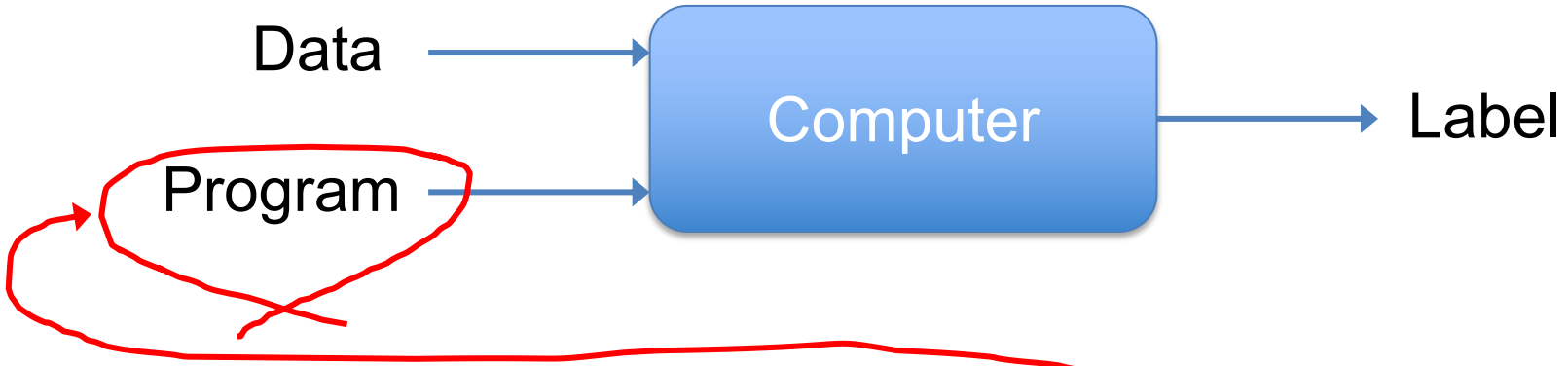


Works well!


samples
NN
true
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
0.0
0.2
0.4
0.6
0.8
1.0

What is Machine Learning?

- **Traditional programming**

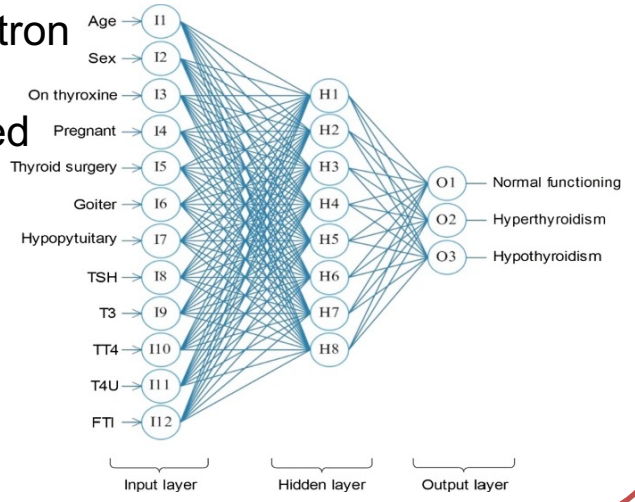


- **Machine Learning programming**

Neural network architectures (famous ones)

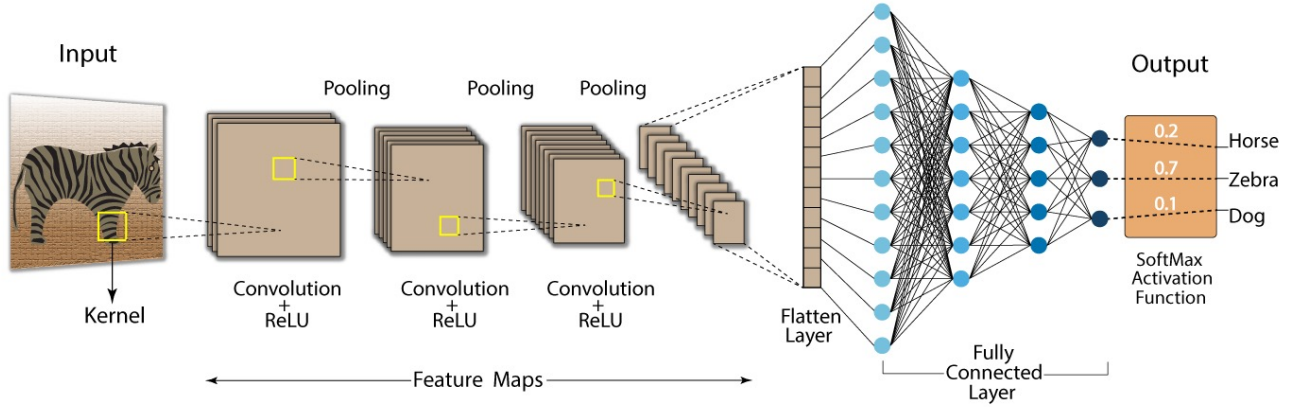
Multilayer perceptron (MLP)

- Fully connected layers
- One or more hidden layers



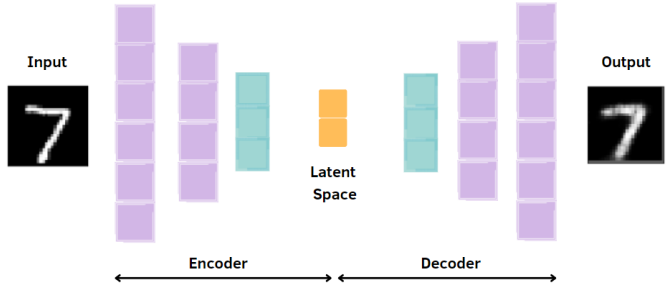
Convolutional neural network (CNN)

- Convolve multiple kernels over image to build feature space
- Use feature space for classification with MLP (a common recipe in DL)



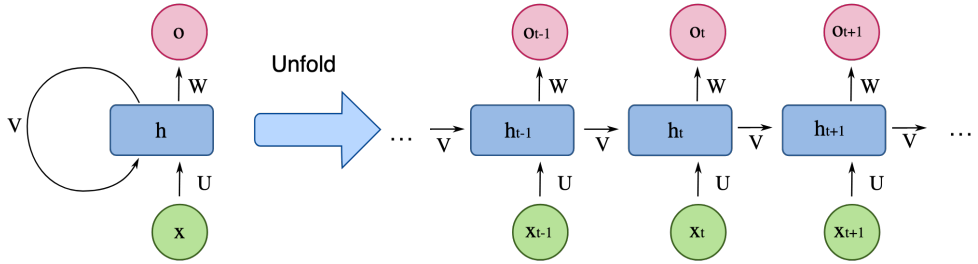
Autoencoder (AE)

- Output is a reconstruction of input
- Latent space provides low-dimensional representation



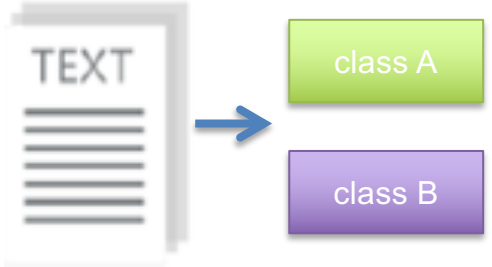
Recurrent neural network (RNN)

- History-dependent internal state provides a mechanism for memory
- Long short-term memory (LSTM), gated recurrent unit (GRU)

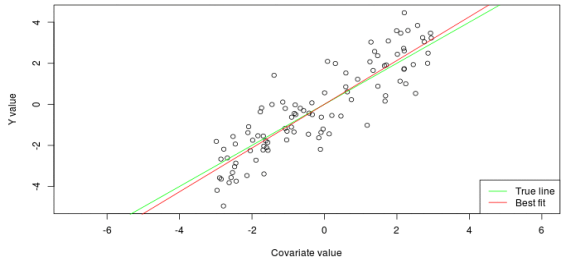


Machine Learning Types

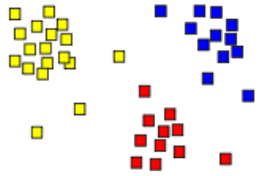
- ***Supervised*: learning with labeled data**
  - Example: image classification
  - Example: regression for predicting real-valued outputs
- ***Unsupervised*: discover patterns in unlabeled data**
  - Example: cluster similar data points
- ***Reinforcement learning*: learn to act based on feedback/reward**
  - Example: learn to play Go



Classification



Regression

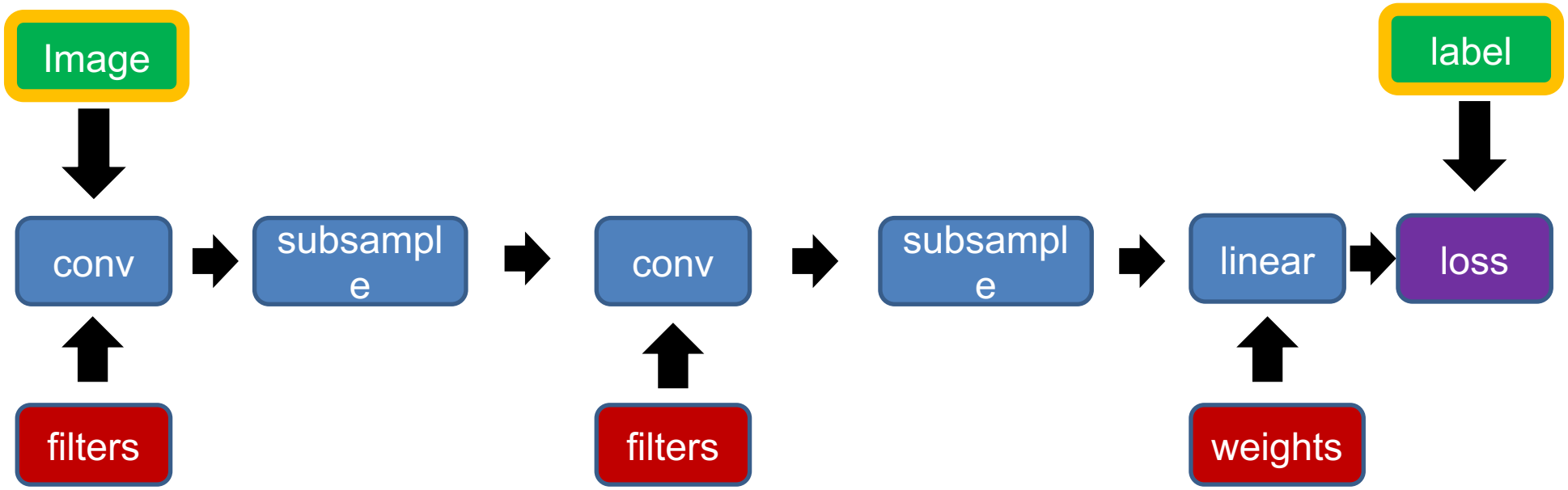


Clustering

Slide credit: Ismini Lourentzou – Introduction to Deep Learning

Training the NN

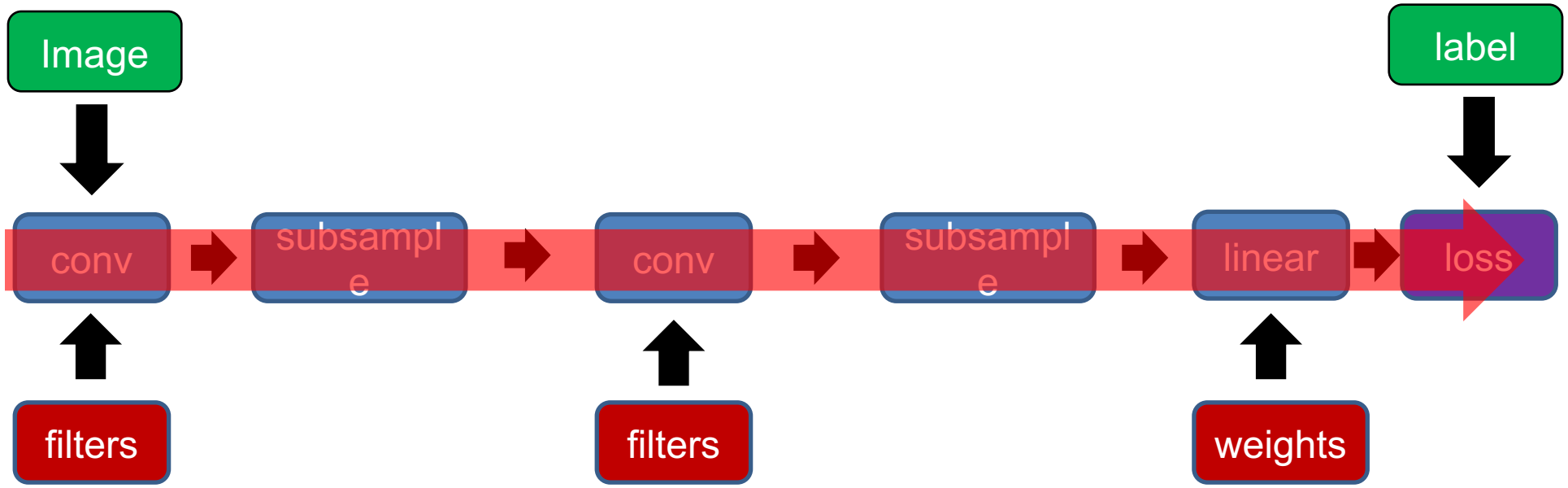
1. **Sample image and label**



https://www.cs.cornell.edu/courses/cs4670/2018sp

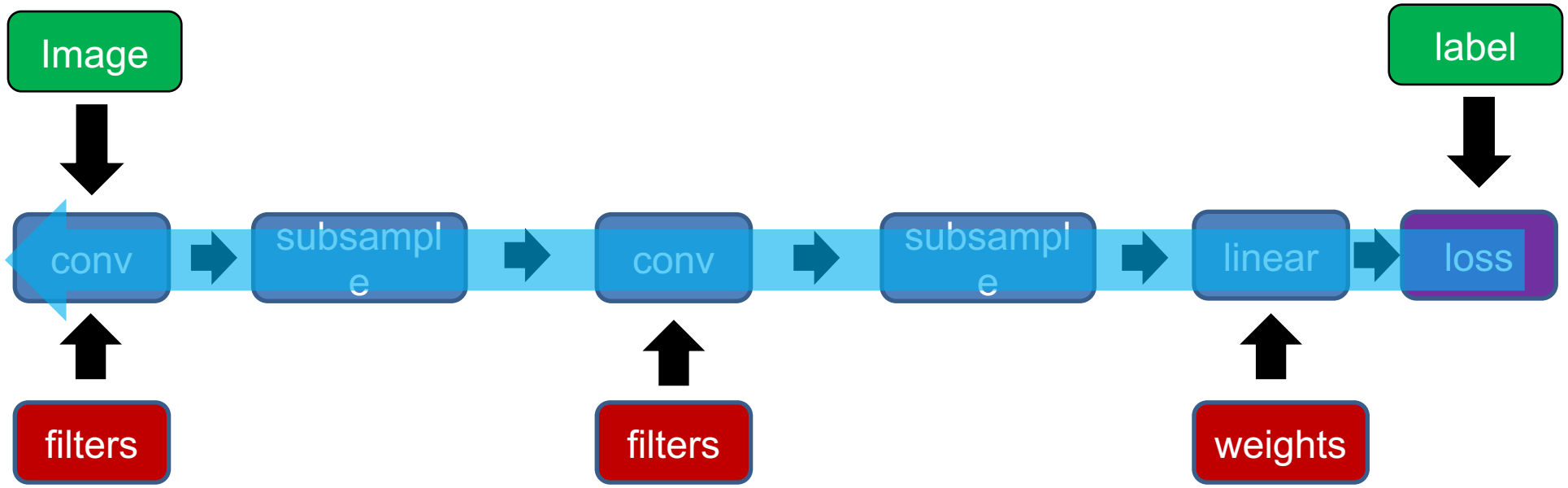
Training the NN

1. Sample image and label
2. Pass image through network to get loss (forward)



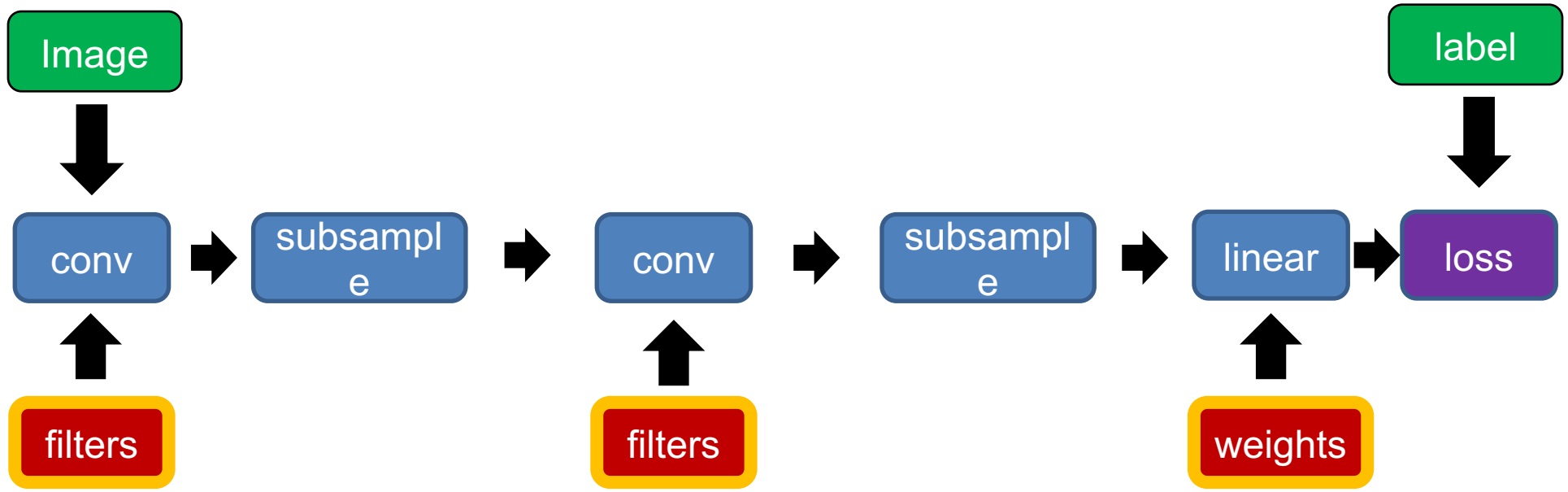
Training the NN

1. Sample image and label
2. Pass image through network to get loss (forward)
3. Backpropagate to get gradients (backward)



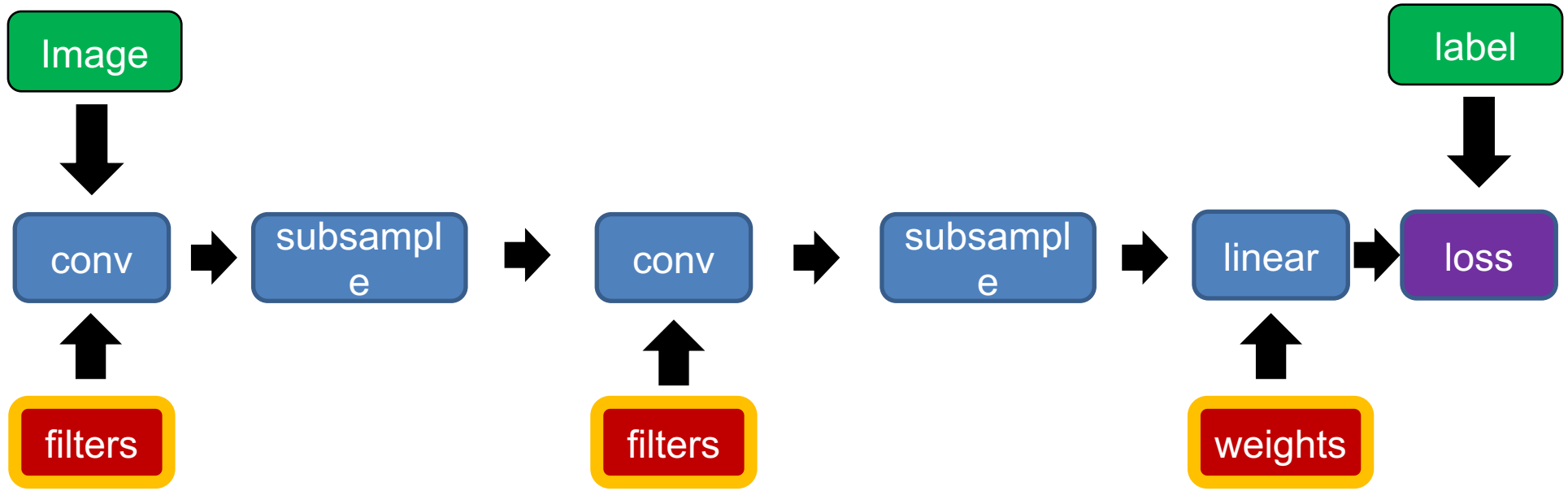
Training the NN

1. Sample image and label
2. Pass image through network to get loss (forward)
3. Backpropagate to get gradients (backward)
4. Take step along negative gradients to update weights



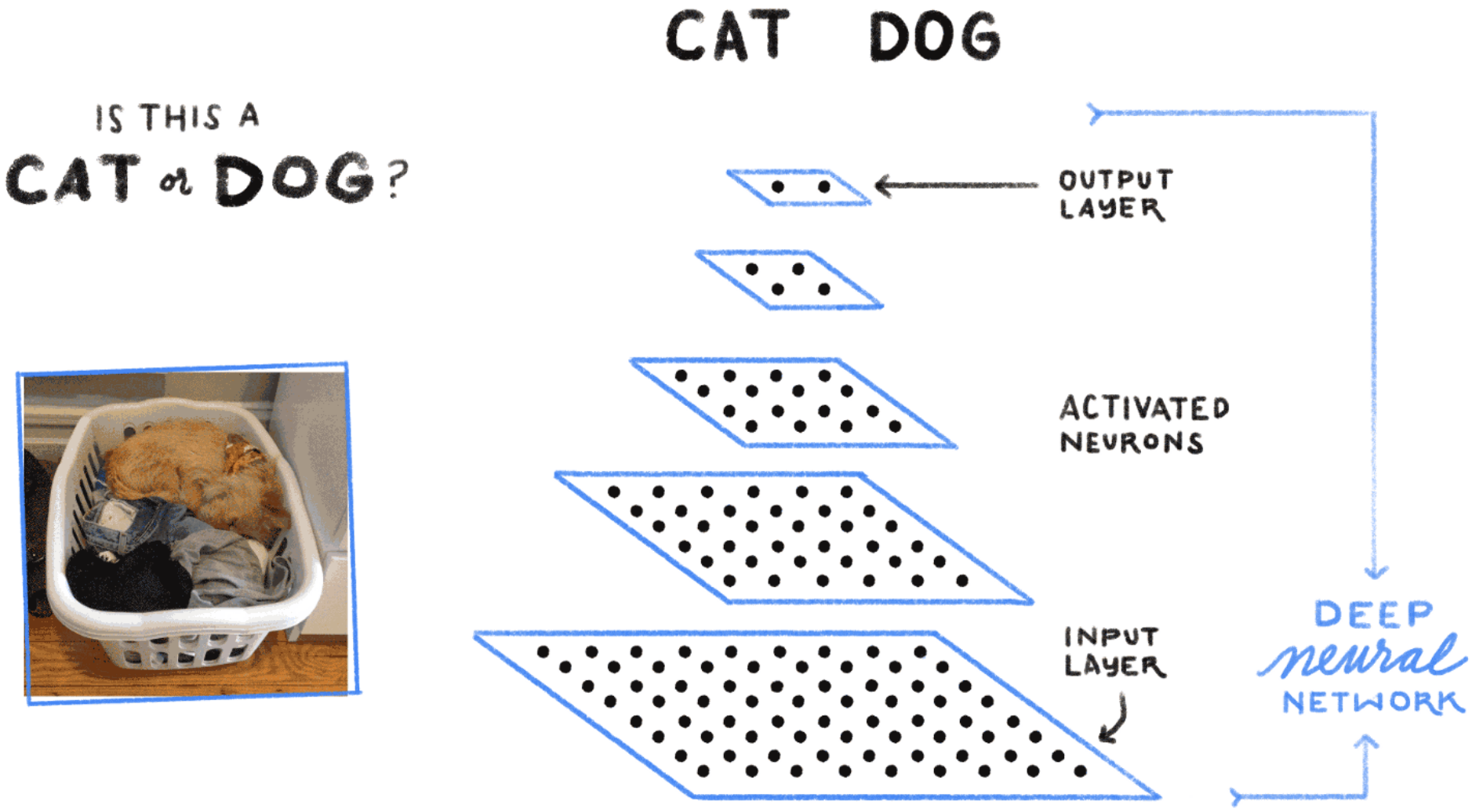
Training the NN

1. Sample image and label
2. Pass image through network to get loss (forward)
3. Backpropagate to get gradients (backward)
4. Take step along negative gradients to update weights
5. Repeat!



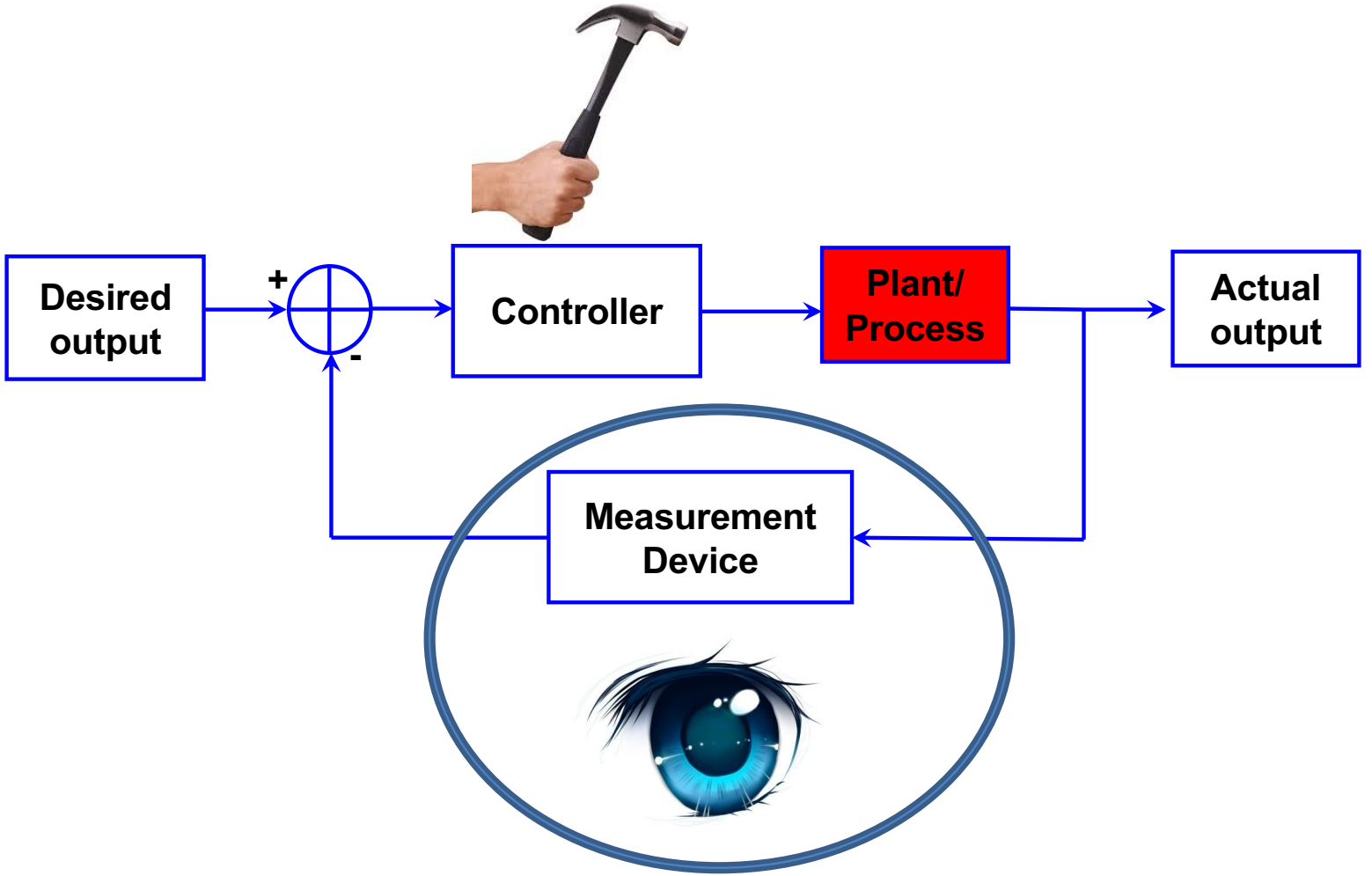
https://www.cs.cornell.edu/courses/cs4670/2018sp

Example Classifier NN



https://www.cs.umd.edu/class/fall2018/cmsc320

ML Control for Fusion:
ML as a measurement going into human designed control

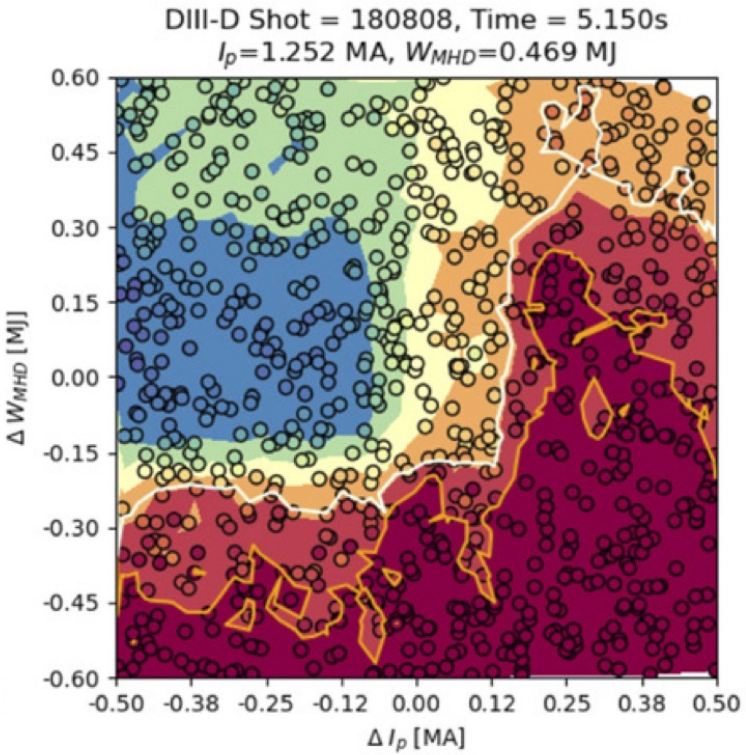


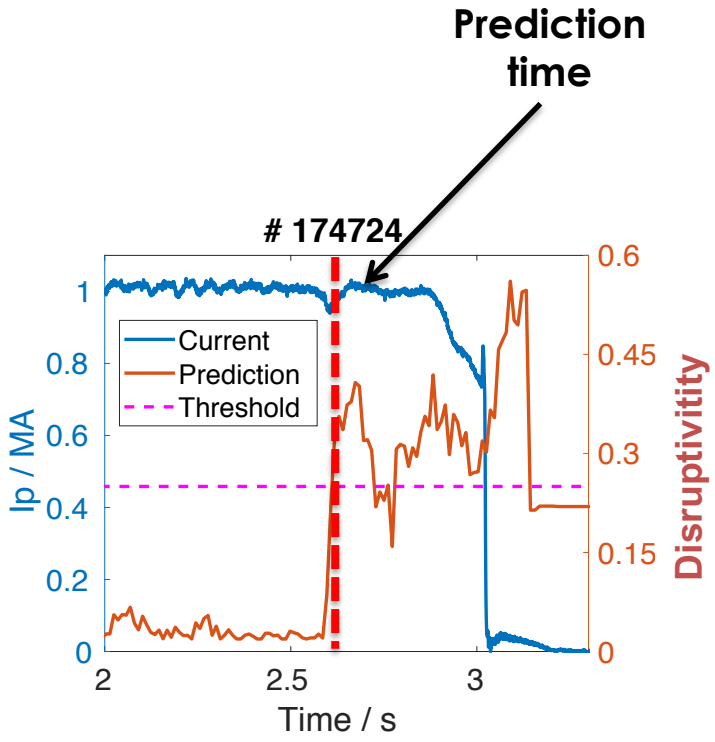
ML as eyes

ML use in Fusion: Event Prediction

- **Simplest and first ML use were Event Prediction**
- **x= 0D signals, y= Disruption [0,1]**
- **Convert to probability**

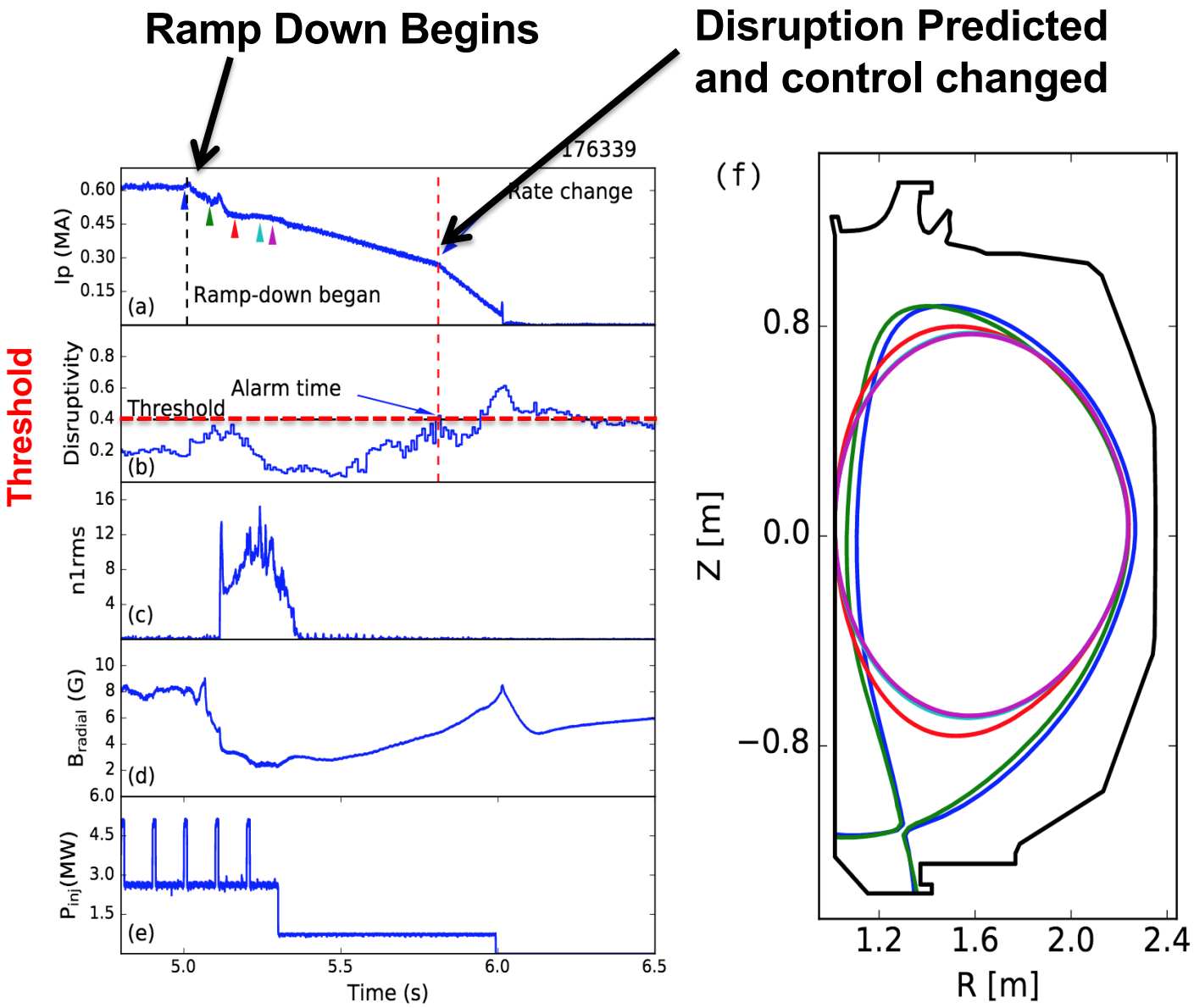
Signal description
Electron density, n_e (m^{-3})
Plasma current, I_p (A)
Squareness, ζ
Plasma minor radius, a
Normalised internal inductance, ℓ_i
Stored plasma energy, W_{MHD} (J)
Elongation, κ
Triangularity, δ





Boyer, NF 2022 + Fu PoP 2020

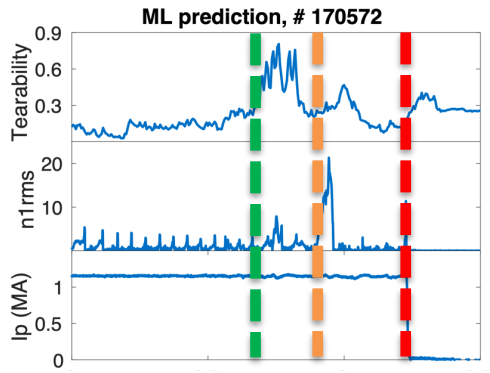
ML to Predict + Feedforward Control (Rampdown Scenario Change)

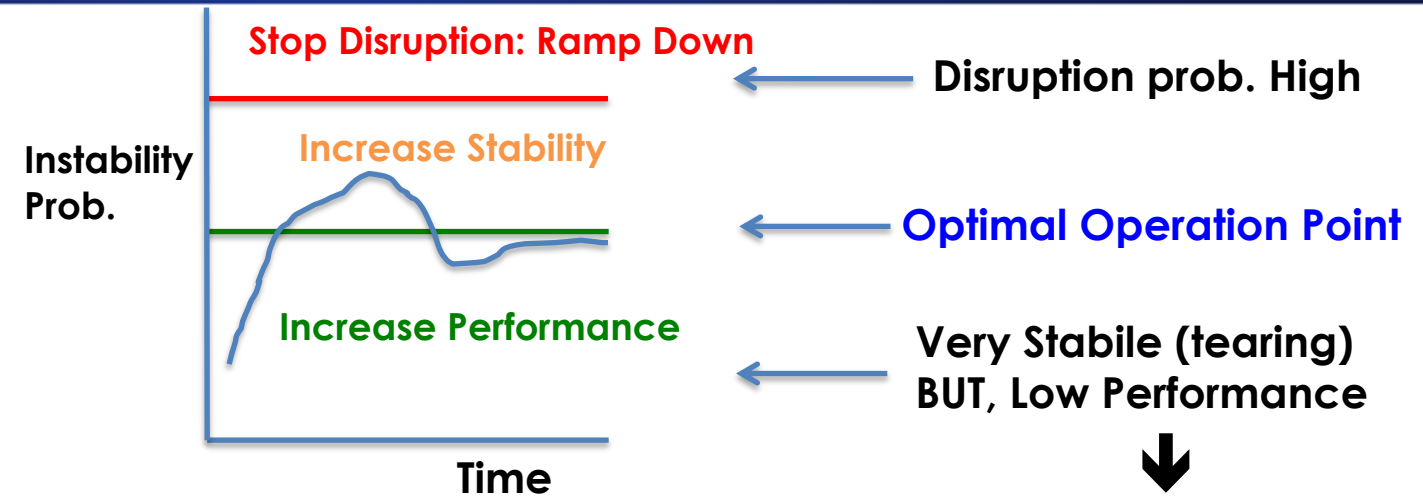


- ML algorithm to find the disruption time.
- **Use it to change the off-normal response:**
  - ➔ New ramp down sequence of the plasma (feedforward) when disruption is expected
  - ➔ Reduce disruption impact (Ip at disruption)

Fu et al. PoP 2020

ML gives a Feedback Control Target:
ML Predict Stability (Tearing) ➔ Optimize Performance

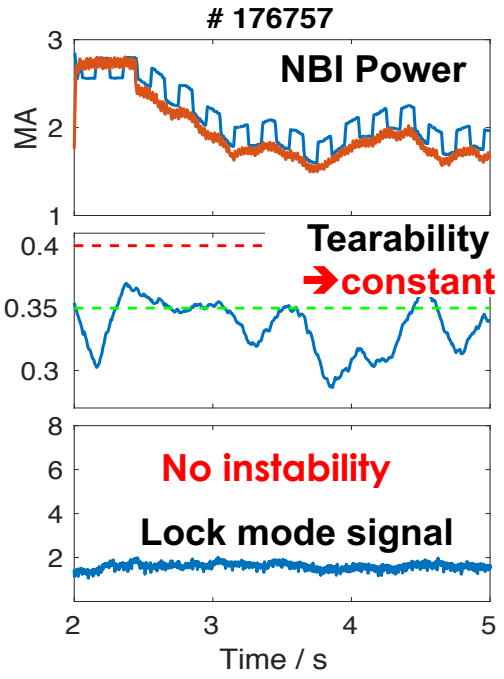




1. Predict Instability using ML
2. Instability Starts
3. Disruption Occurs

- Instability prediction gives lots more time than disruption prediction
- Enough time to control the plasma and avoid shutdown
- Choose a stability level to operate at (say %1)
- Then Machine Learning Controls the NBI/ECH for highest performance at that stability level
- PCS Algorithm (Kolemen, Yu, Boyer, Erickson)
- *Fu et al. PoP 2020*

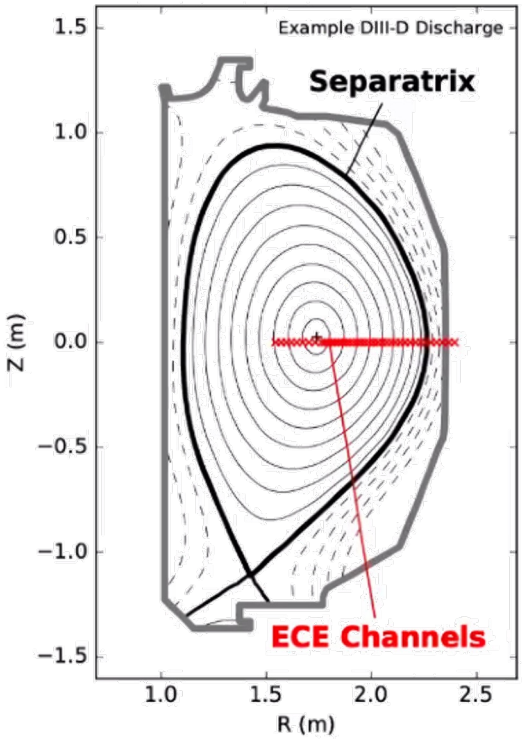
Beam Power Controlled with ML

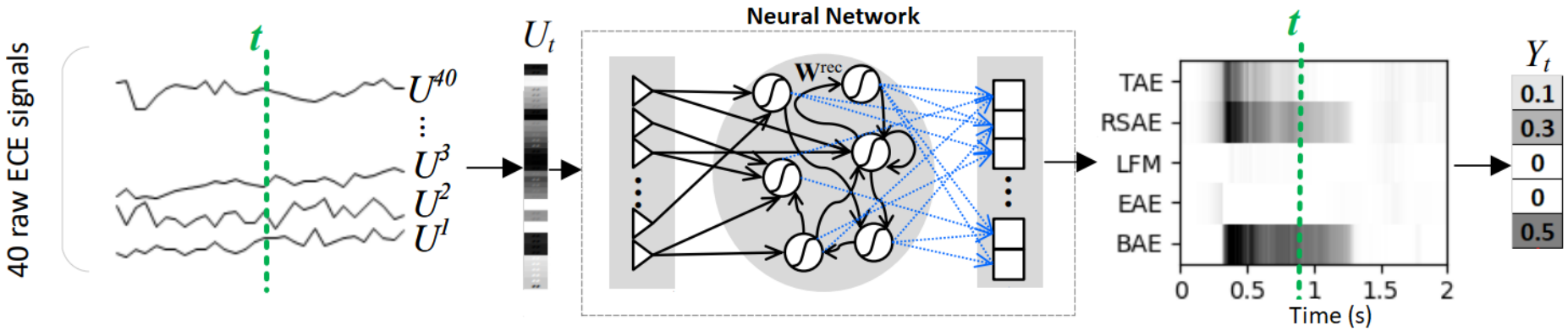


NN-based AE detector: Detecting Alfvén Eigenmode Using ECE

Jalalvand *et al* 2022 *Nucl. Fusion* **62** 026007

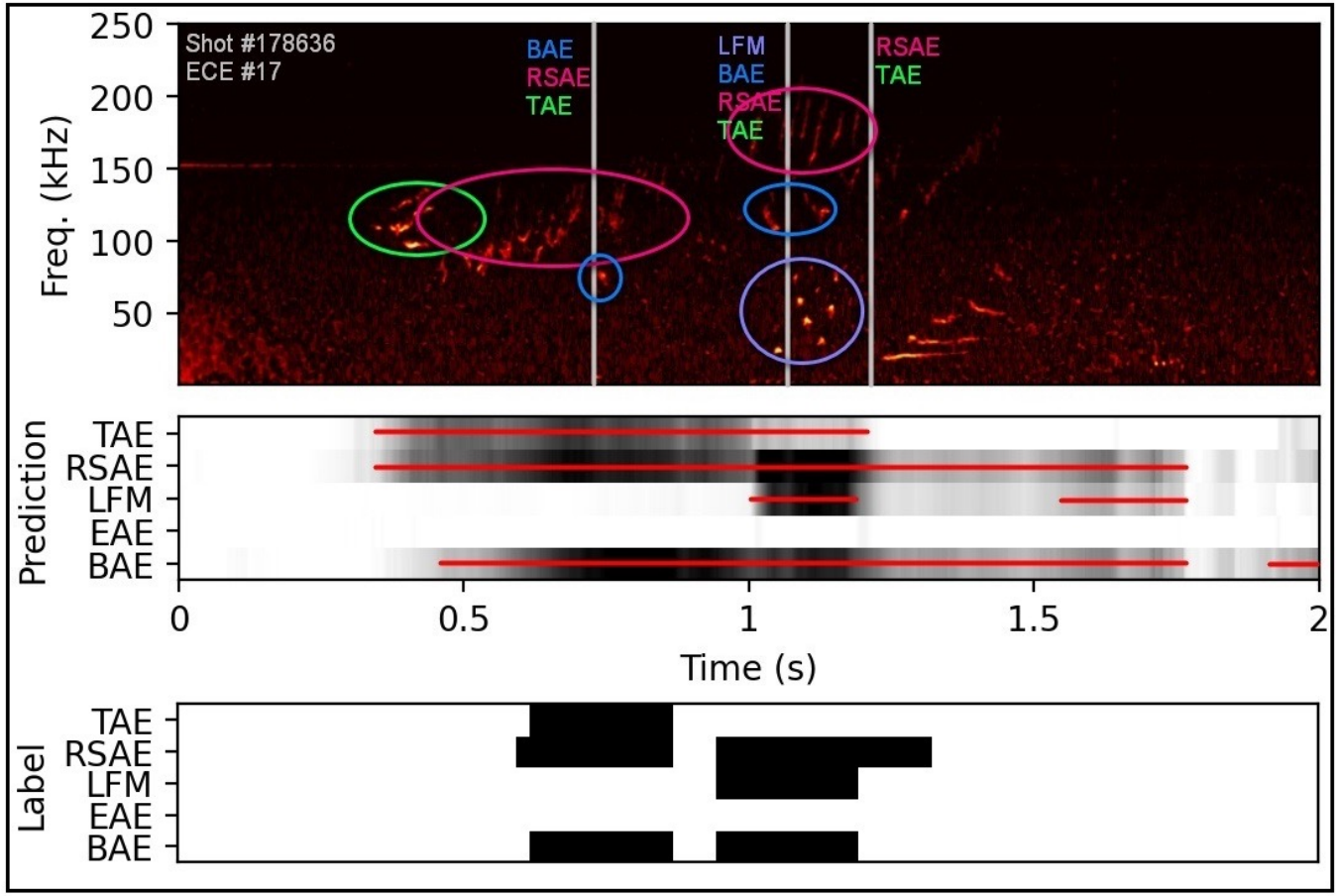
- Input: **40 raw ECE signals** (No preprocessing!)
- Output: Score of five AE modes (LFM, BAE, EAE, RSAE, TAE) per time step
- ML model: Reservoir Computing Network (RCN)





NN-based AE detector: Detecting Alfvén Eigenmode Using ECE

Jalalvand *et al* 2022 *Nucl. Fusion* **62** 026007



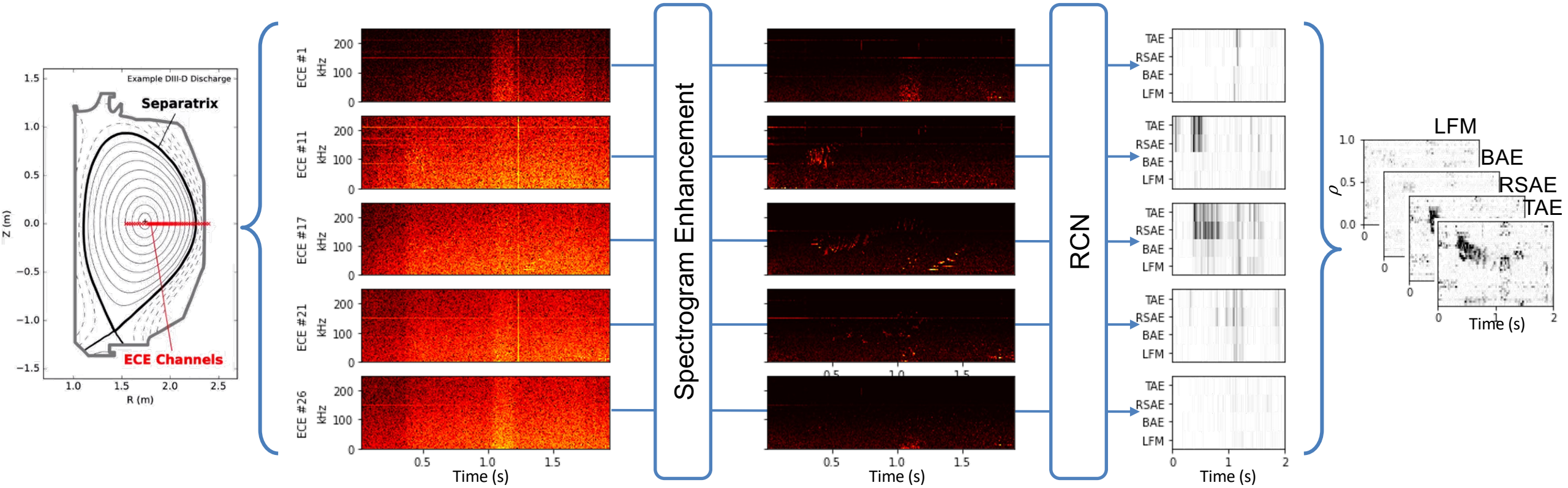
True Positive Rate: **%91**

False Positive Rate: **%7**

NN-based AE localizer: Detecting and Locating Alfvén Eigenmode Using ECE

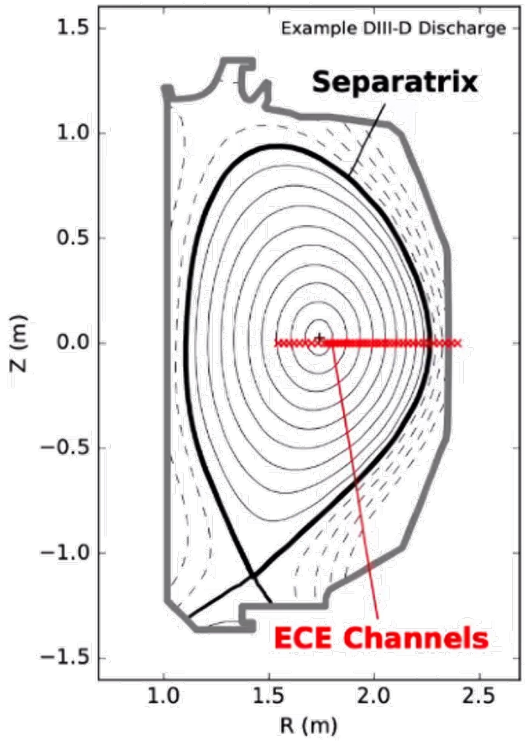
A. Kaptanoglu et. al., 2022 Nucl. Fusion (Accepted)

- Input: Spectrogram of each ECE channel
- Output: Probability of AE modes <u>per ECE per time step</u>



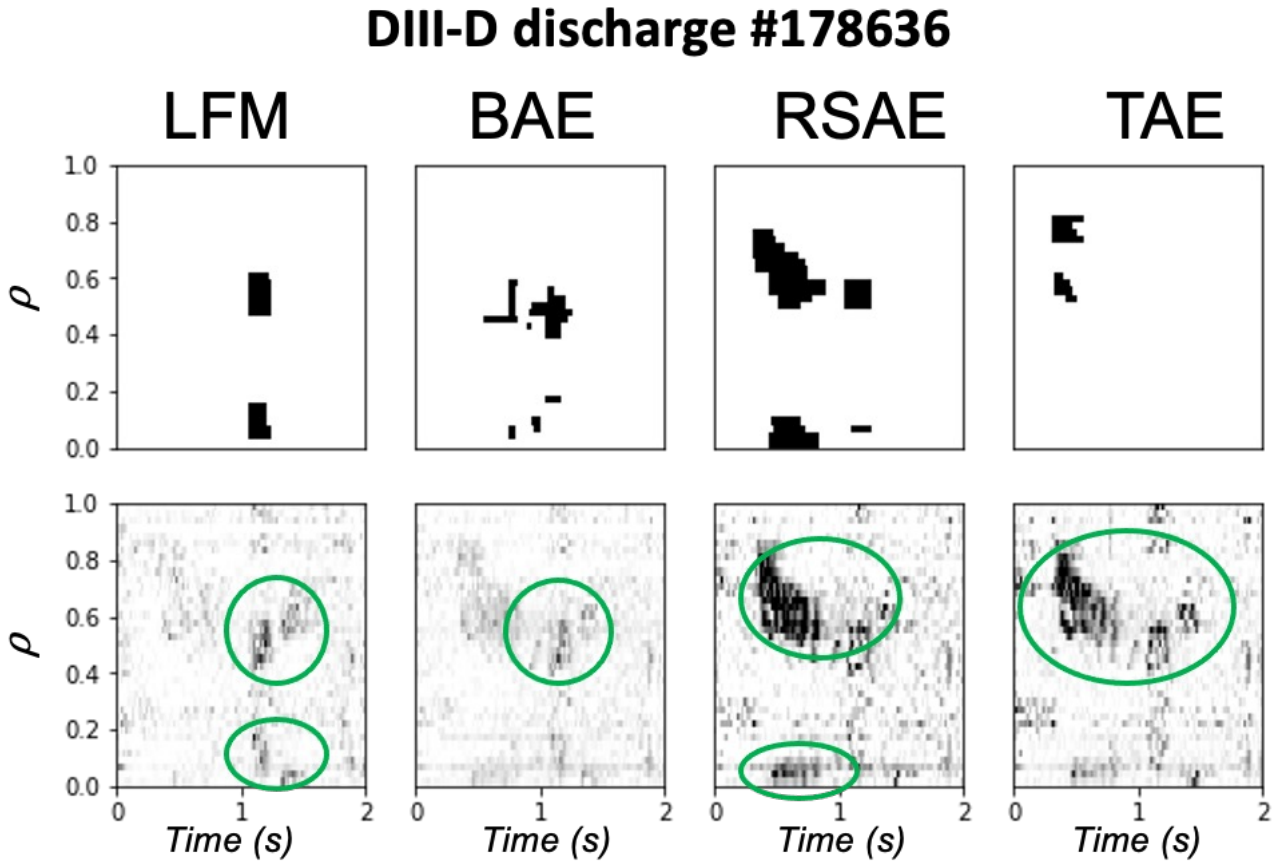
NN-based AE localizer:
Detecting and Locating Alfvén Eigenmode Using ECE

A. Kaptanoglu et. al., 2022
Nucl. Fusion (Accepted)



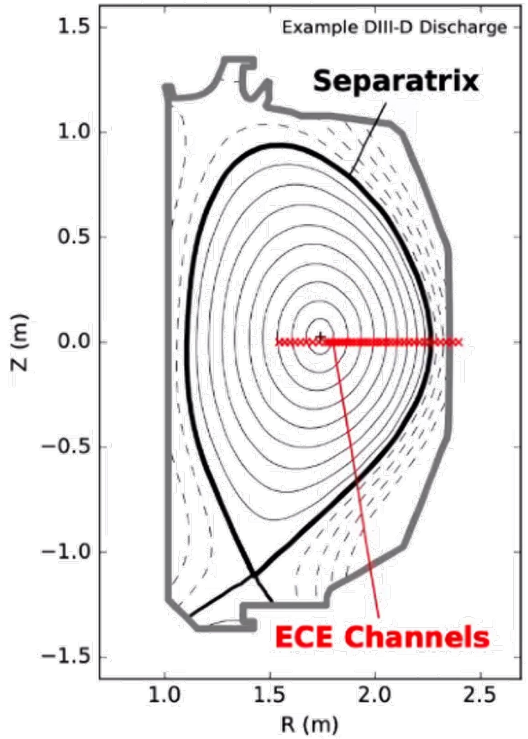
Label:

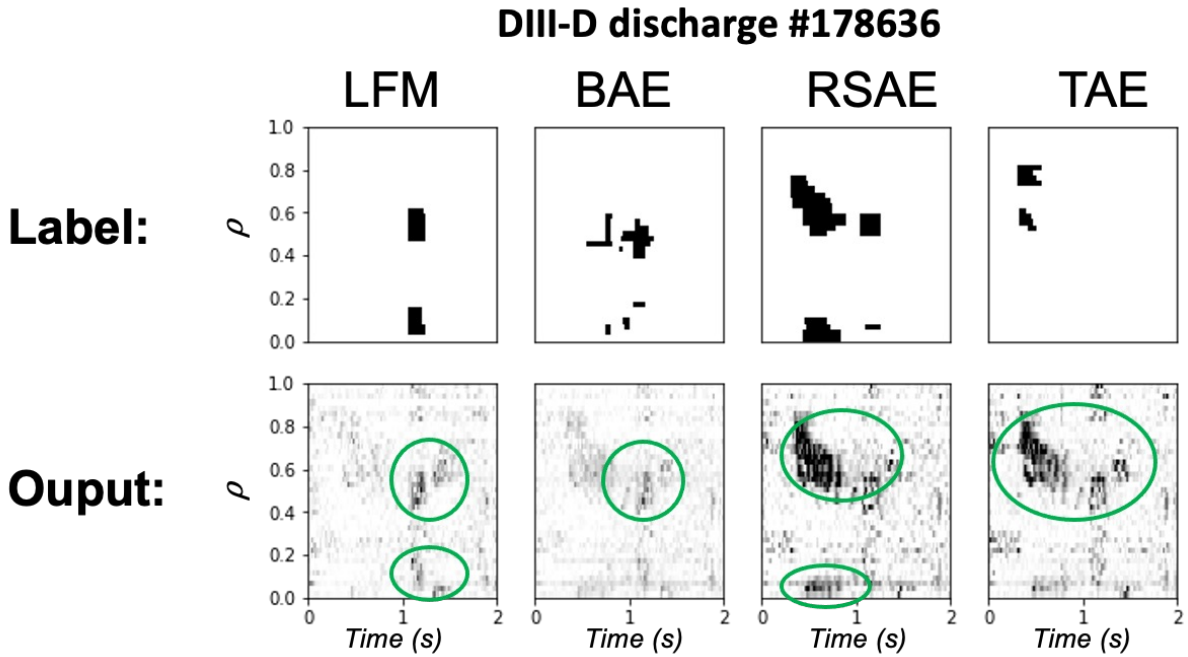
Ouput:

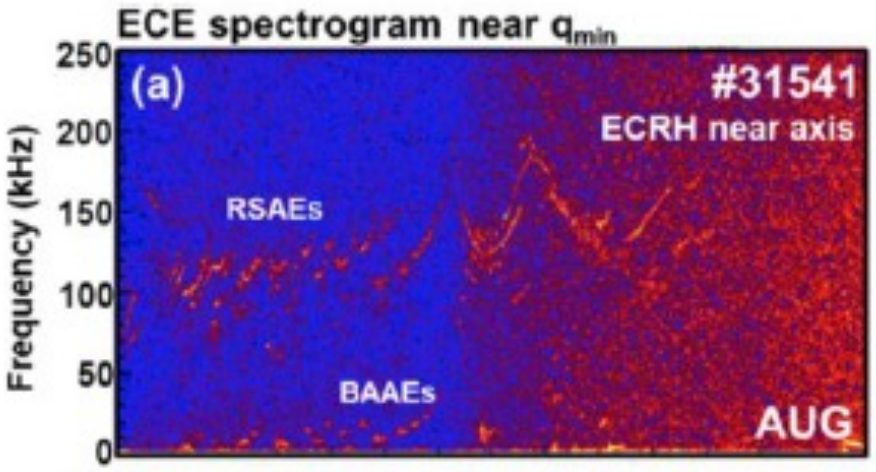


NN-based AE localizer: Detecting and Locating Alfvén Eigenmode Using ECE

A. Kaptanoglu et. al., 2022 Nucl. Fusion (Accepted)

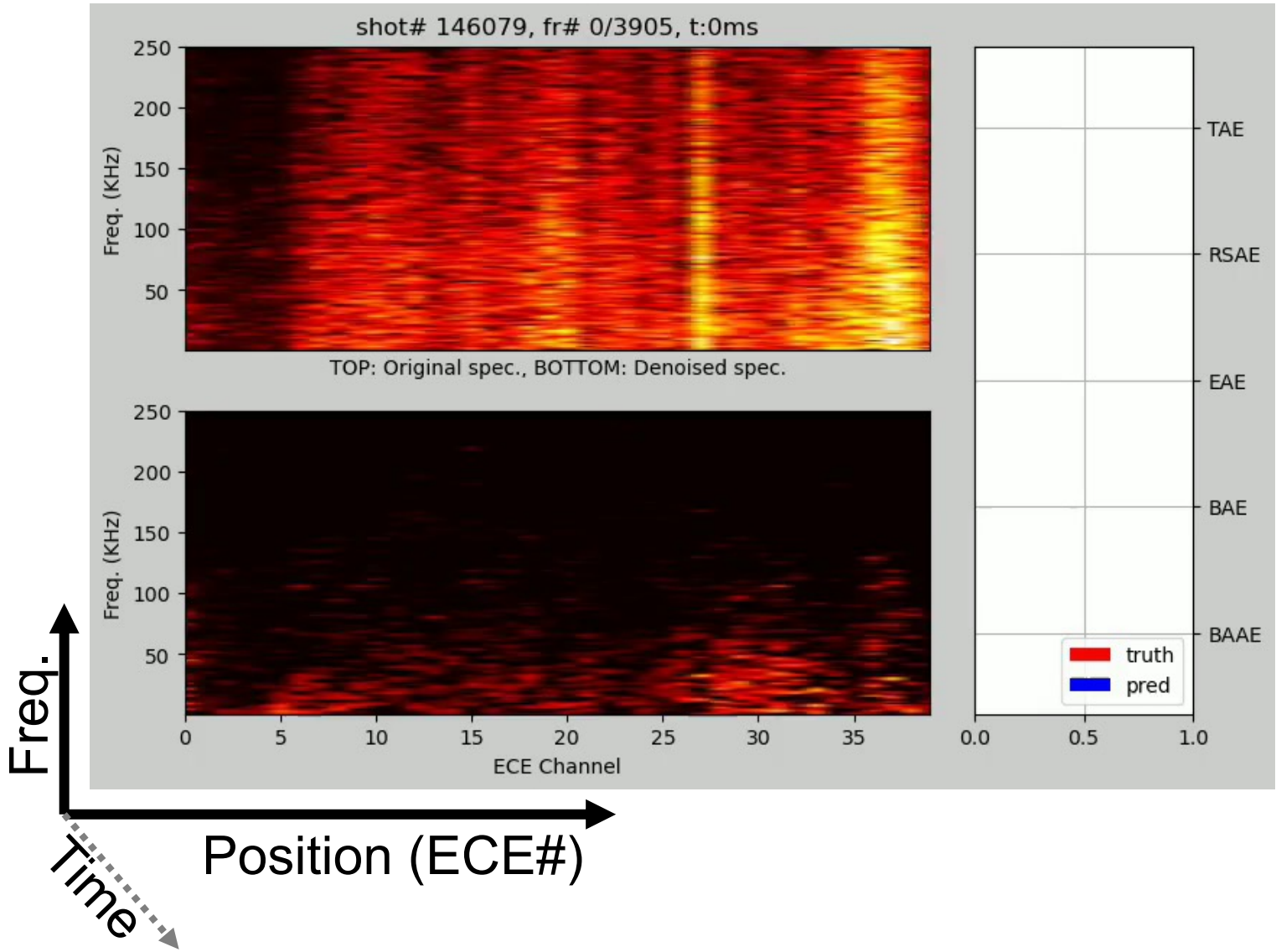




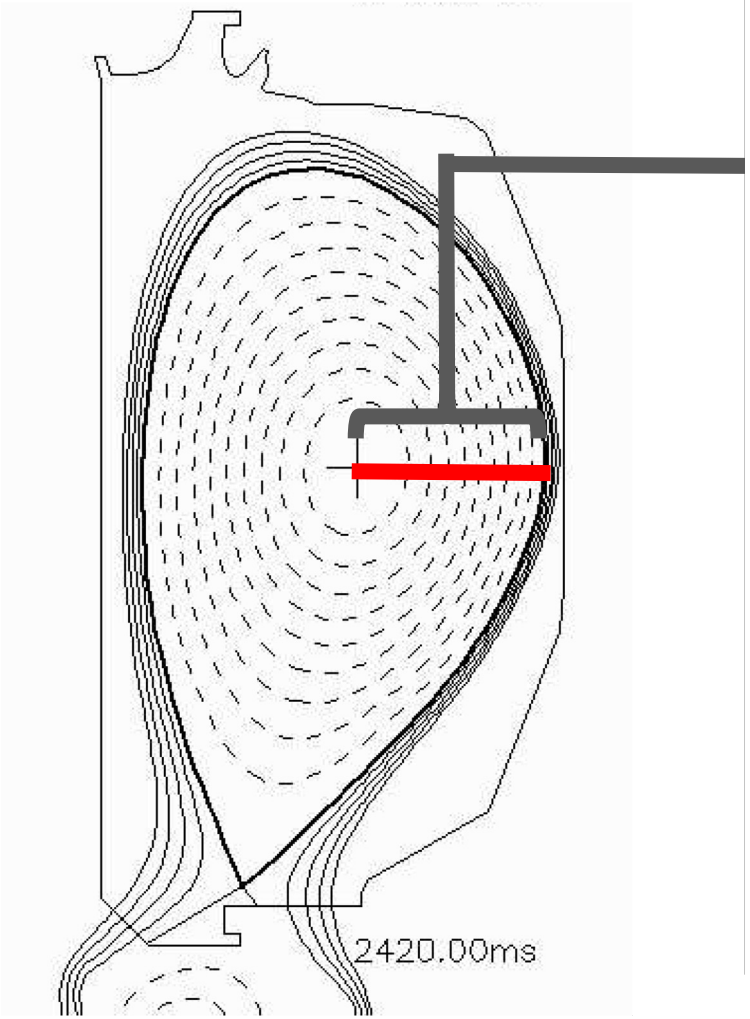


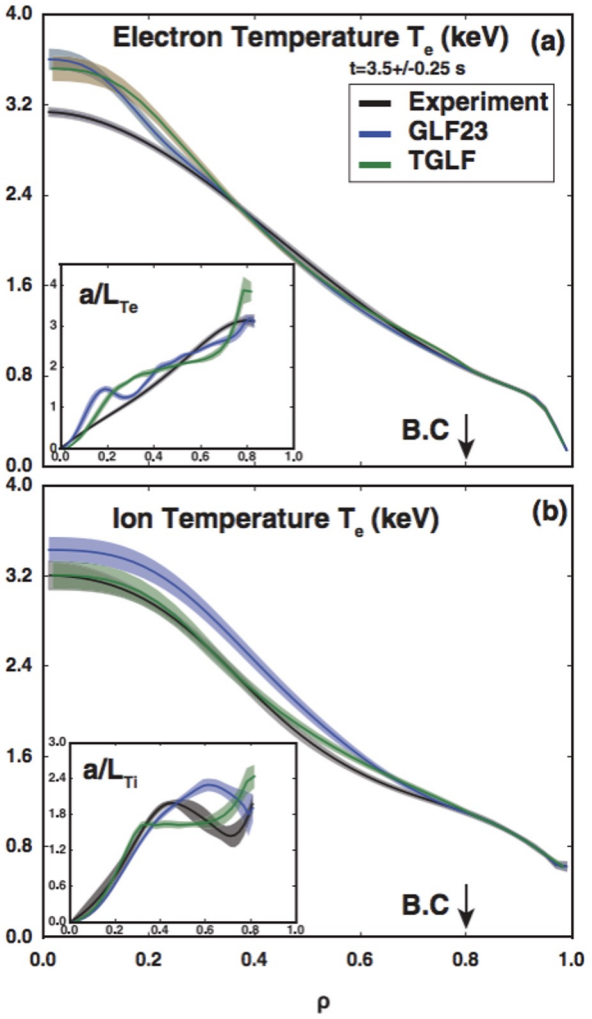
- **Control aim ECH to the location to suppress AE**

NN-based AE localizer:
Detecting and Locating Alfvén Eigenmode Using ECE



Plasma Evolution Modeling: Plasma Profile



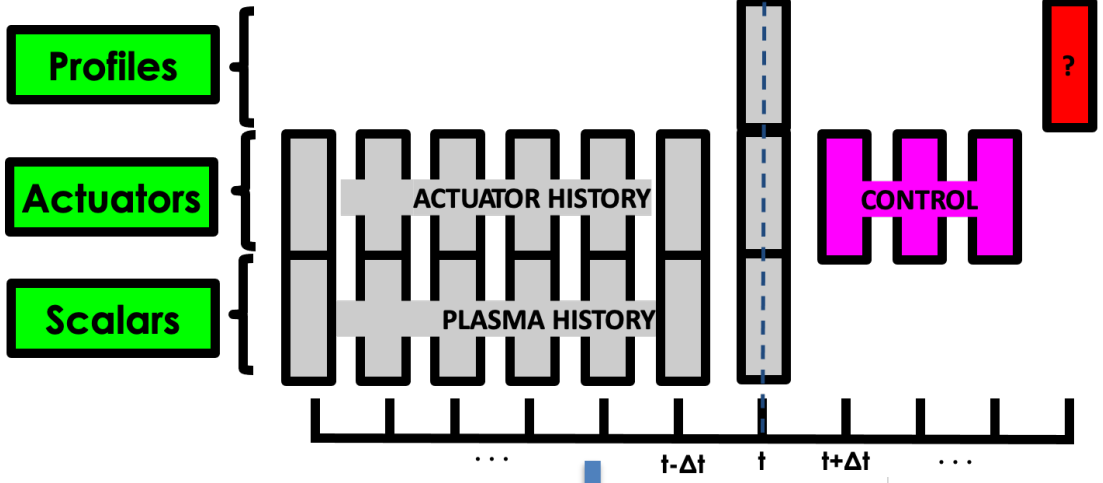


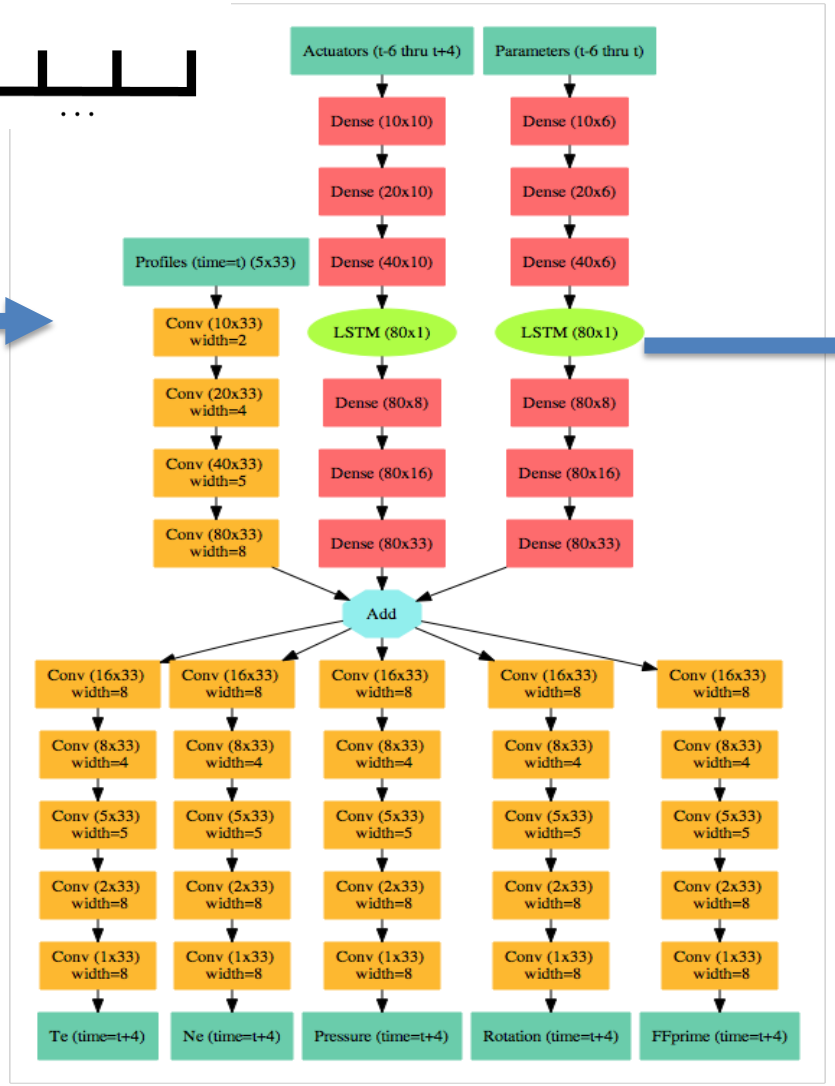
- Define the state of the plasma by 1D profiles.
- Due to symmetry and high transport along flux sections

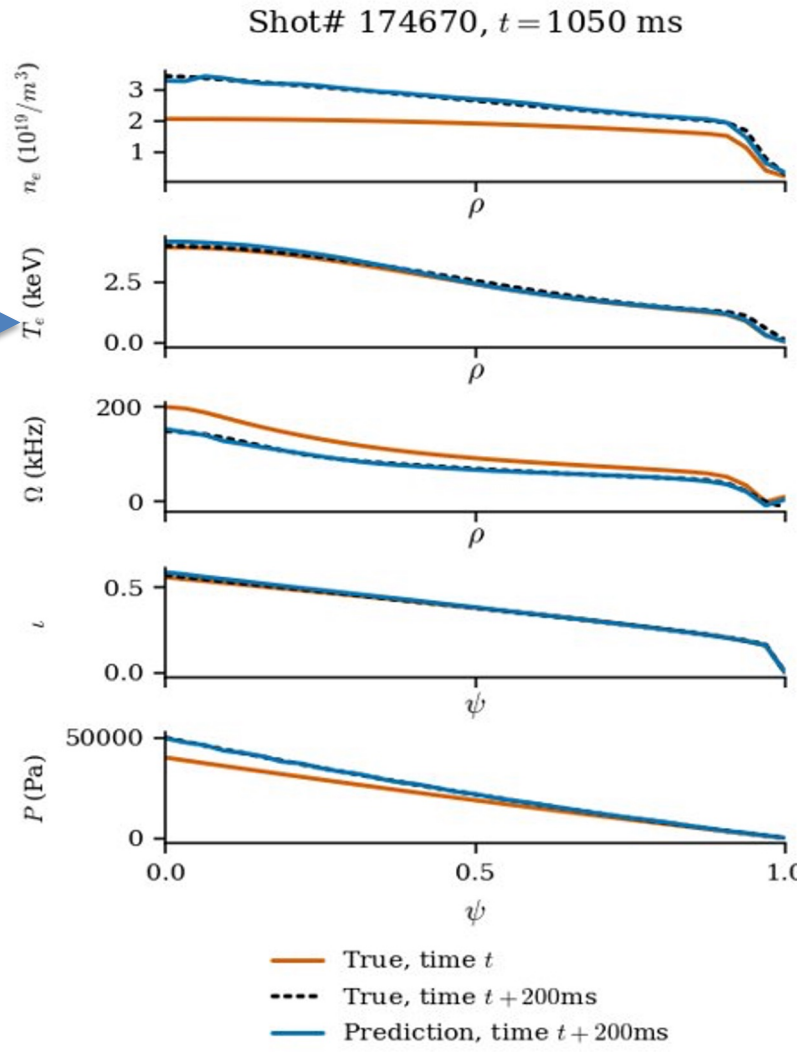
Full state of plasma determined by 1D profiles:

- Pressure (P)
- Current (J)
- Electron temperature and density (T_e, n_e)
- Ion temperature and density (T_i, n_i)
- Rotation (Ω)

ML as Plasma Evolution Model: ML Profile Prediction







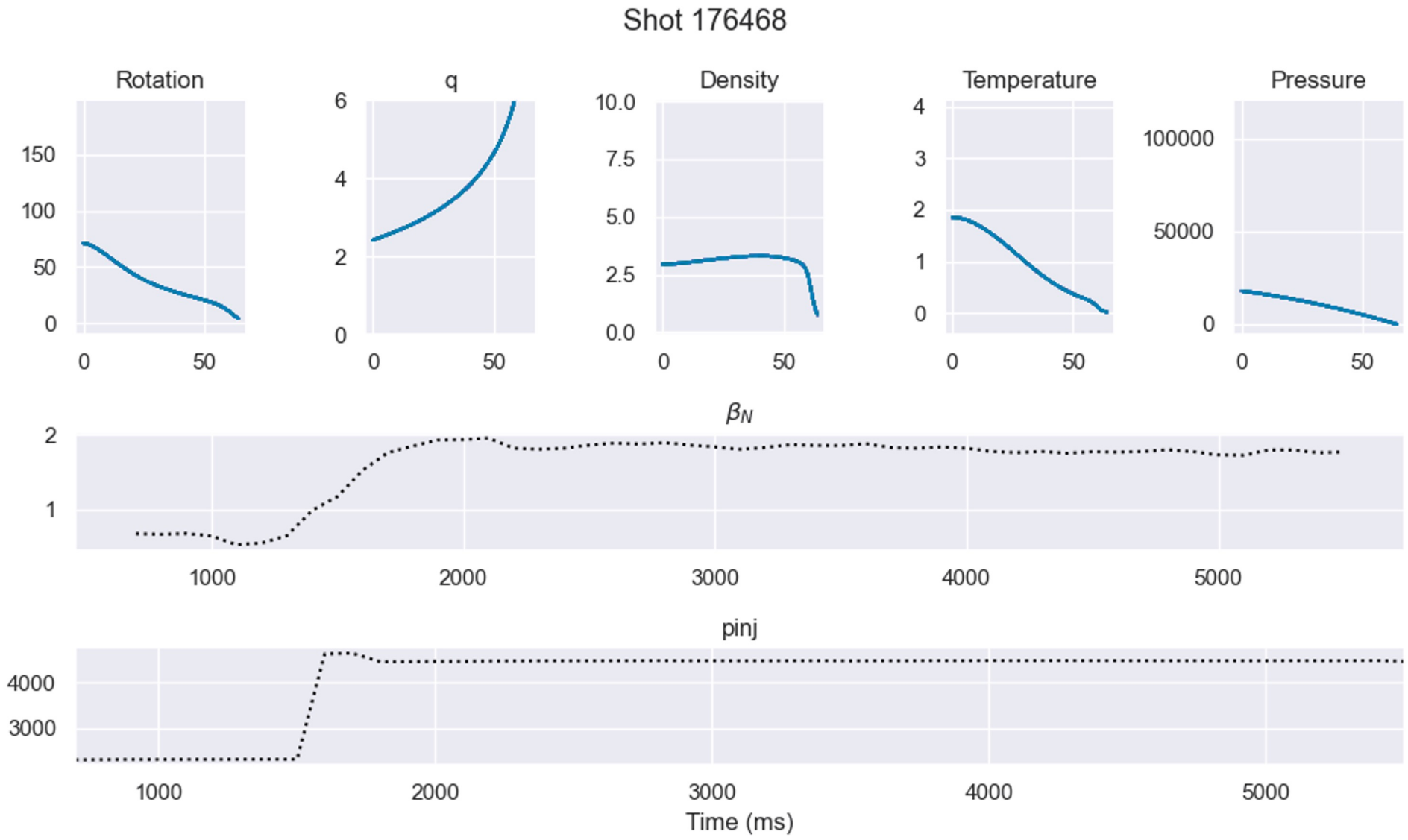
- **ML-Based Prediction of Profile Evolution.**
- **Input:** DIII-D Historic Data -- 2013-2019 (5 Profiles, Shape, NBI, Density,…)
- **Output:** Profile Evolution predicting NN

Abbate, Conlin NF 2021

Replaying a test set shot for BetaN, Li prediction (Ian Char, CMU)

Ground Truth

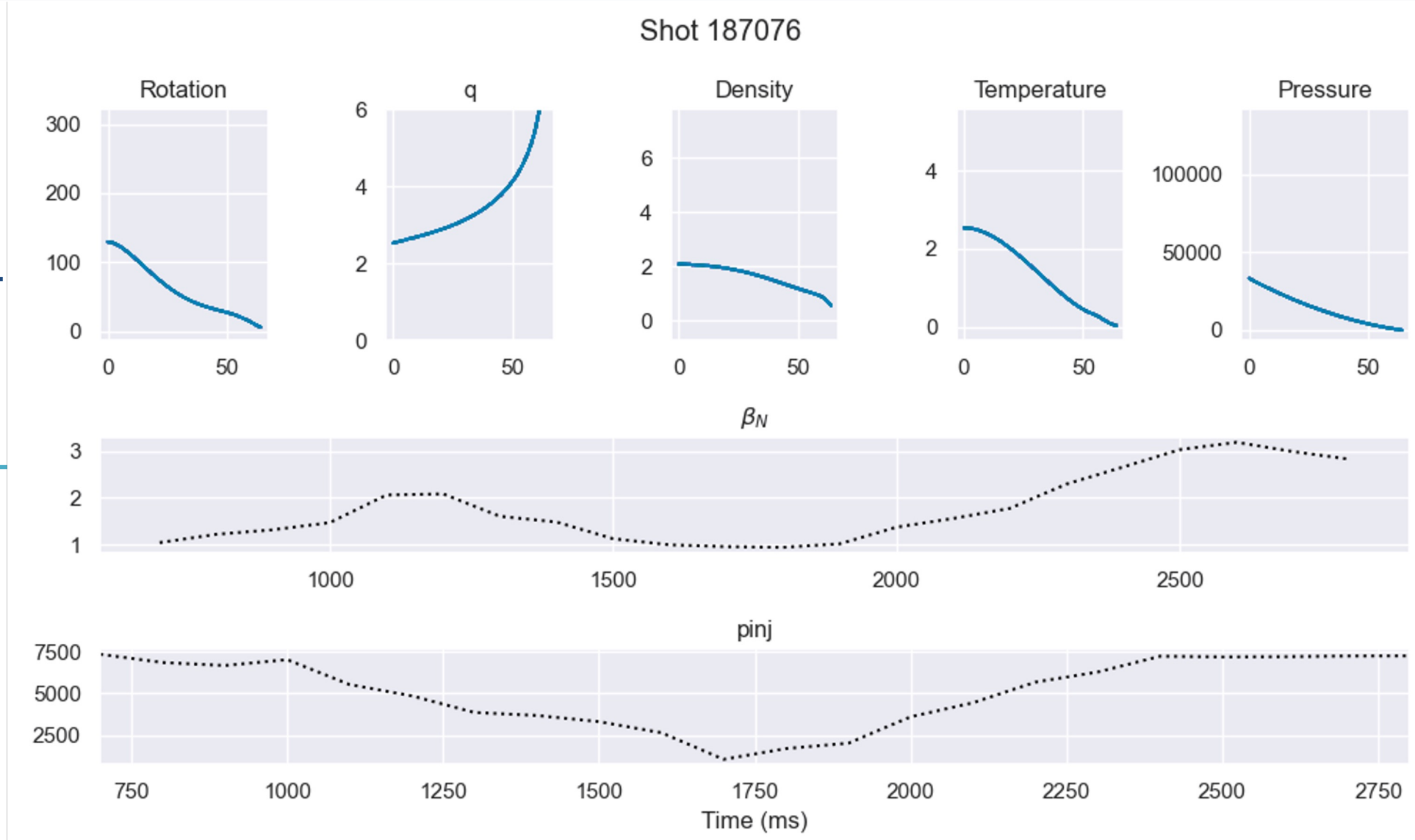
Predictions



Replaying a more interesting test shot (Ian Char, CMU)

Ground Truth

Predictions



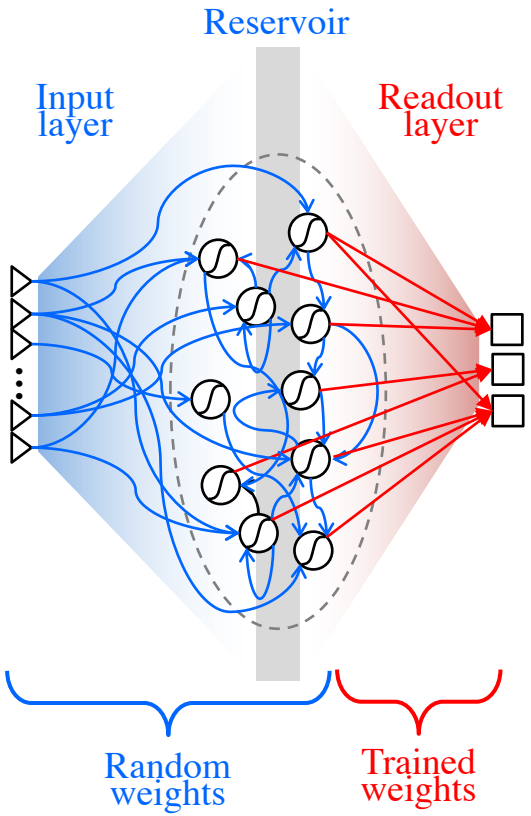
Realtime Adaptive ML Plasma Model: Reservoir Computing Network (RCN)

Azarakhsh (Aza) Jalalvand
Ghent University - Belgium

A **recurrent** neural network with **random** and **sparsely connected** early layers.
Only the last layer is trained using **linear regression**.

Specifications of RCN:

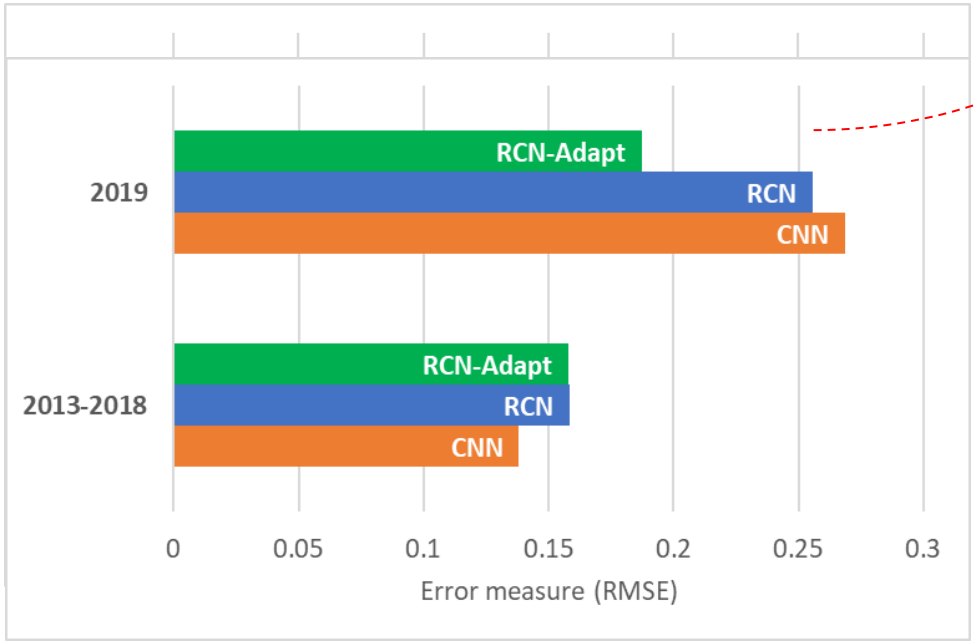
- Projects the inputs to a random very high-dimensional space.
- Ability to process **temporal information** (time-series data analysis)
- Much **faster** and **easier training** procedure compared to DNNs.
  - LSTM: **5 hours** on GPU
  - RCN (with similar performance to LSTM): **4 Minutes** on CPU
  - Easy & fast training makes "in-situ" **model adaptation** possible



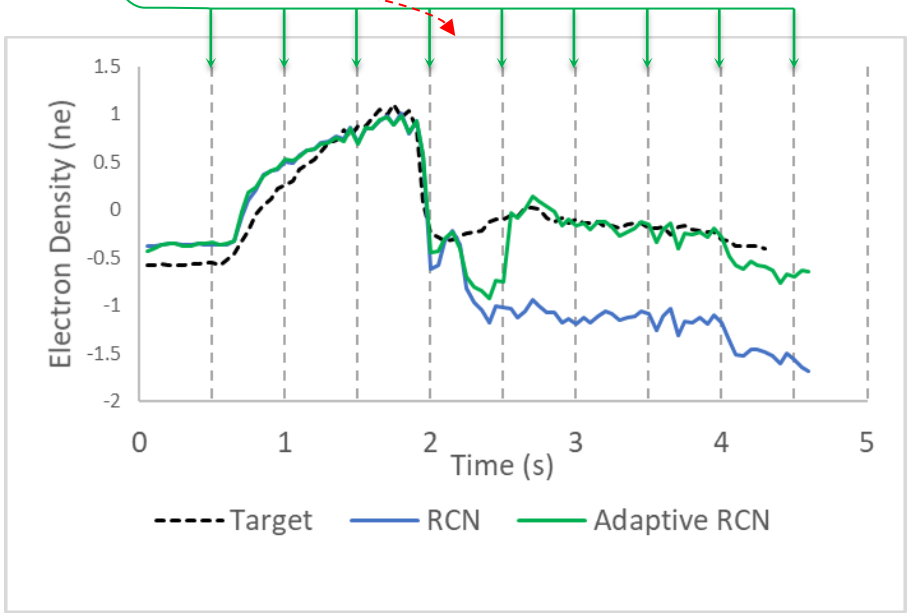
Adaptive Data-driven Profile Prediction Model

* A. Jalalvand, J. Abbate, R. Conlin, G. Verdoolaege, E. Kolemen,
"**Real-Time and Adaptive Reservoir Computing with an Application to Profile Prediction in Fusion Plasma**",
IEEE Trans. on Neural Net. & Learning Systems, 2021.

	CNN/LSTM	RCN
Training	5h on GPU	45s on CPU
Performance	SOTA	Close to DNN
Adaptation	Difficult	Easy (100ms)

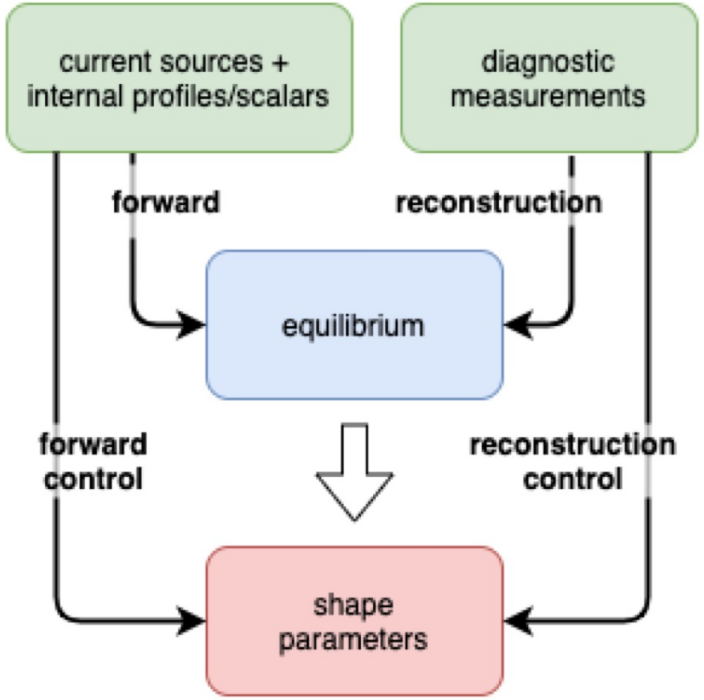


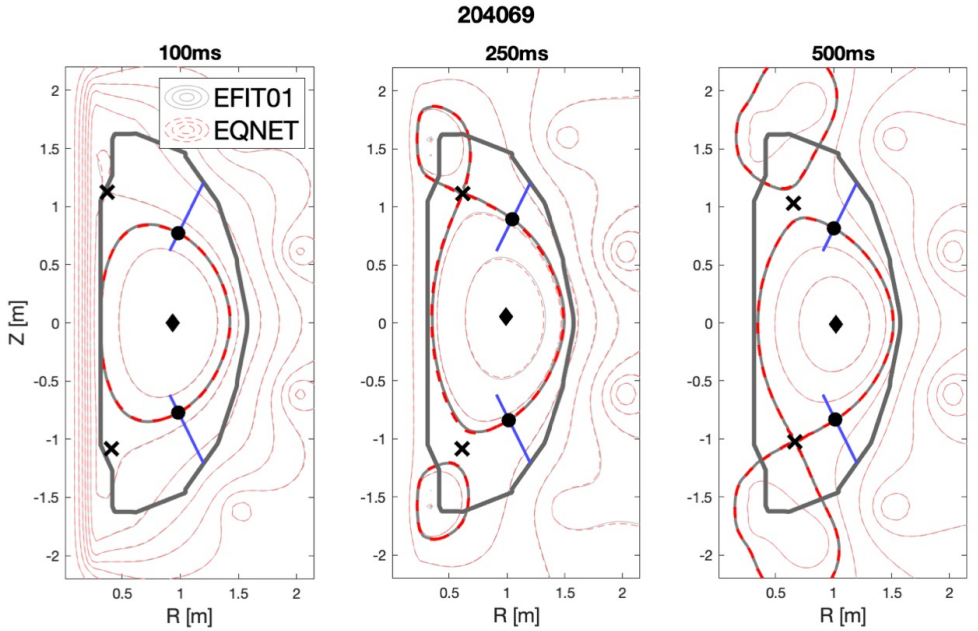
Real-time Adapting every 500ms



ML as a Surrogate for Physics code: EQNet, RT Equilibrium Reconstruction with ML

- **Data does not have to be experimental! It can simulation as well.**
- **We developed a NN that represents G-S Reconstruction for NSTX-U**
- ***More accurate* than rt-EFIT (<1ms)**



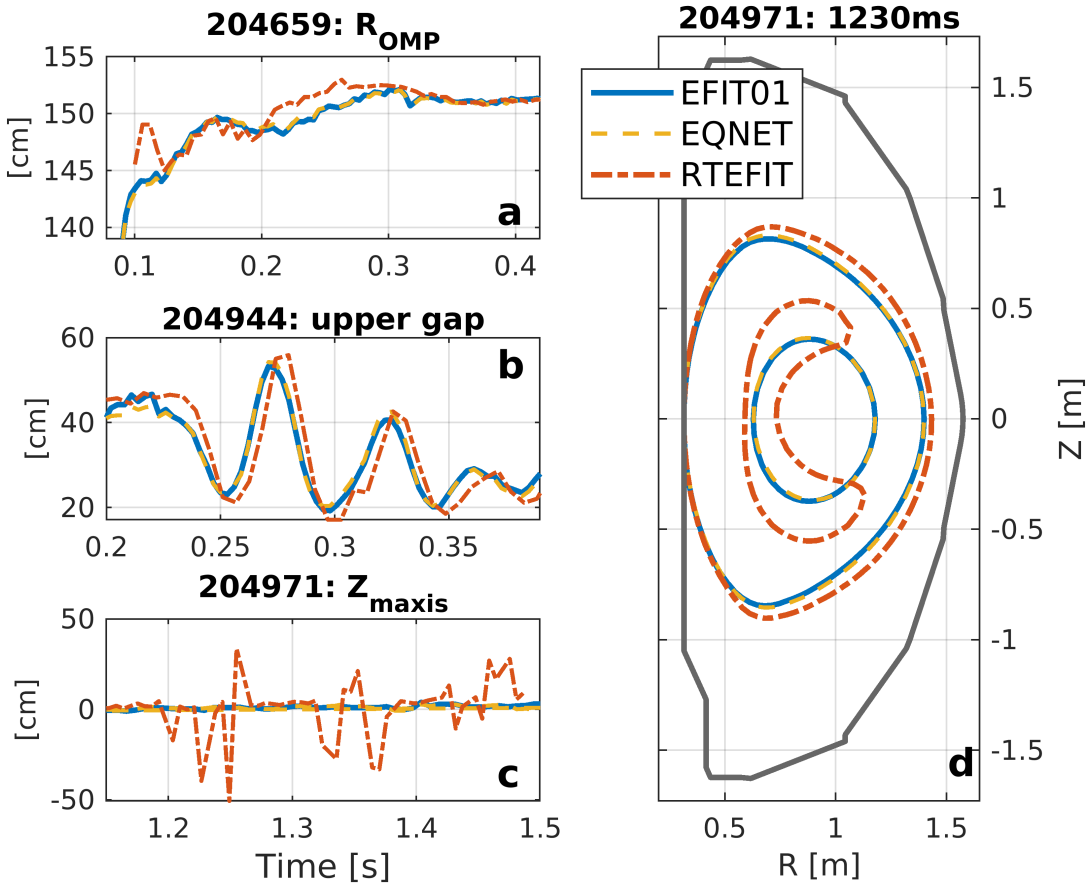


J.T. Wai, M.D. Boyer, E. Kolemen, "Neural net modeling of equilibria in NSTX-U", *NF*, 2022.

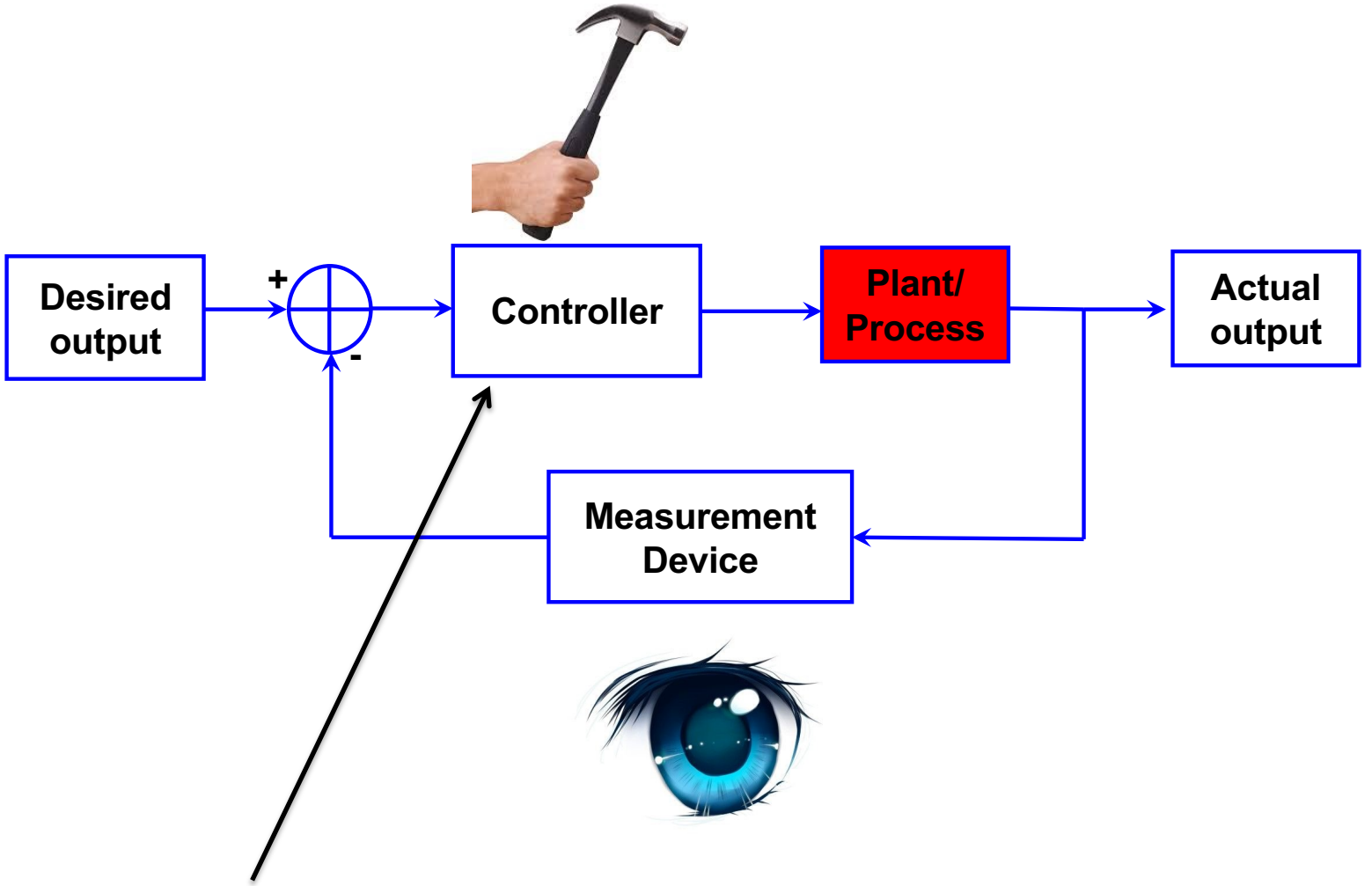
ML as a Surrogate for Physics code: Faster, More Robust

- Neural net responds better to dynamic changes and induced vessel currents than online method
- Faster: Removes 5ms phase delay during oscillations – improves controller chance for recovery
- Robust: Trained with all good and bad sensors. ➔ Losing a sensor degrades prediction BUT does not fail

- ML can output Linear State Space System which can then be used in control

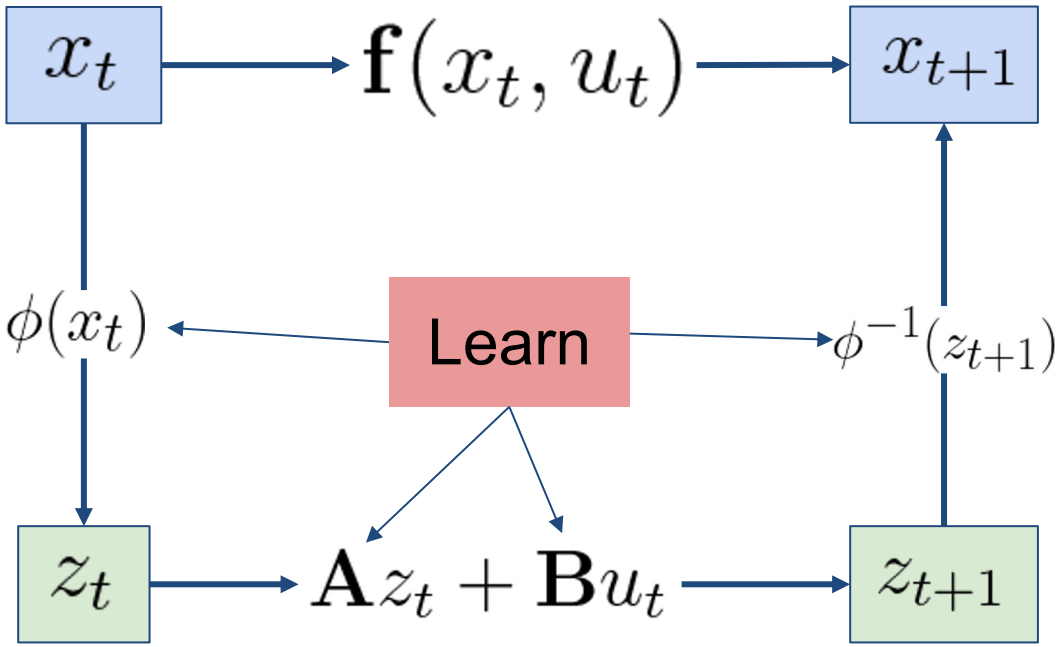


ML Control for Fusion:
ML to produce model for control



ML as model to use in Control

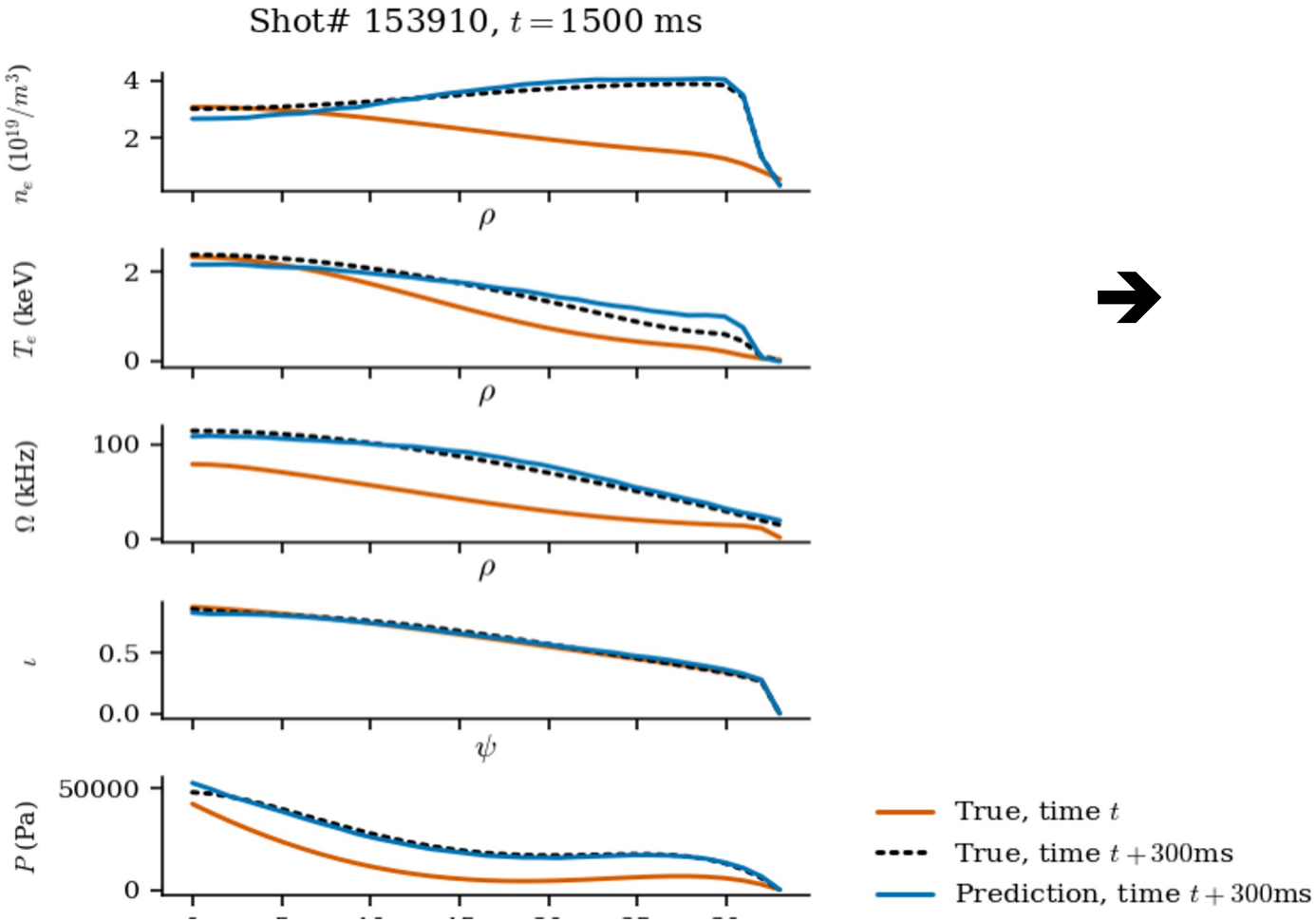
Plasma dynamics are nonlinear - use ML to get linear model

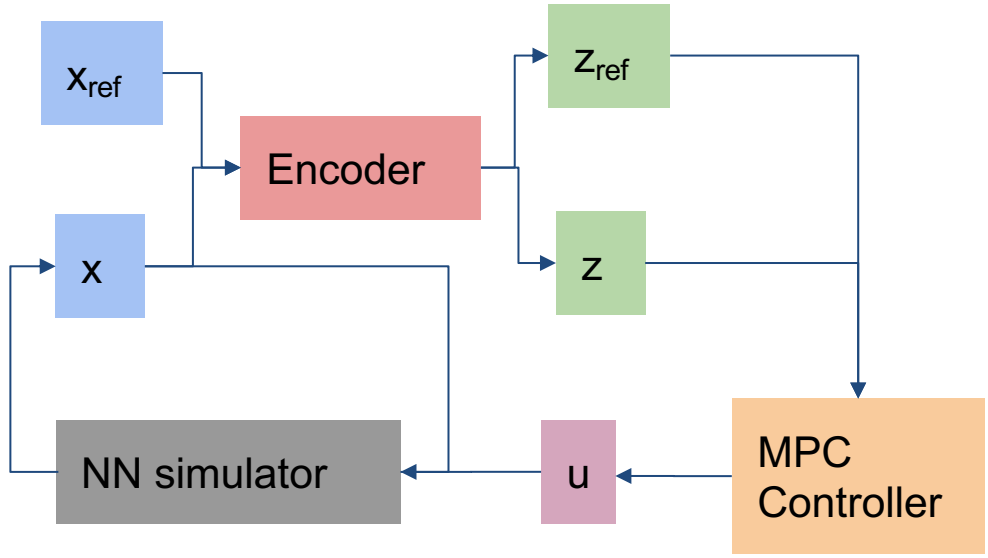


- **Traditional linearization uses Jacobian of f**
  - Only valid locally (perhaps very locally)
- **Alternative - Learn nonlinear embedding to linear space**
  - Linearly Recurrent Autoencoder Network (LRAN)
- **Can be valid *globally***
  - S. E. Otto and C. W. Rowley, 2019
  - Mezic, 2019.

LRAN applied to profile prediction

- **Model trained on experimental data from DIII-D 2013-2018 give similar prediction as before**
- **Linear model can be used in control**





Model Predictive Control:
Convert cost function to the "standard form" of a quadratic program

Control system formulation

$$J_k = \sum_{i=1}^{N} x_{k+i}^T Q x_{k+i} + u_{k+i-1}^T R u_{k+i-1}$$

Subject to:

- Dynamics constraint

$$x_{i+1} = Ax_i + Bu_i$$

- Input constraints
- Output constraints

LQR is a special case of this with N = ∞
and no input or output constraints

Standard form for quadratic programs

$$J_k(z) = z^T H z + f^T z$$

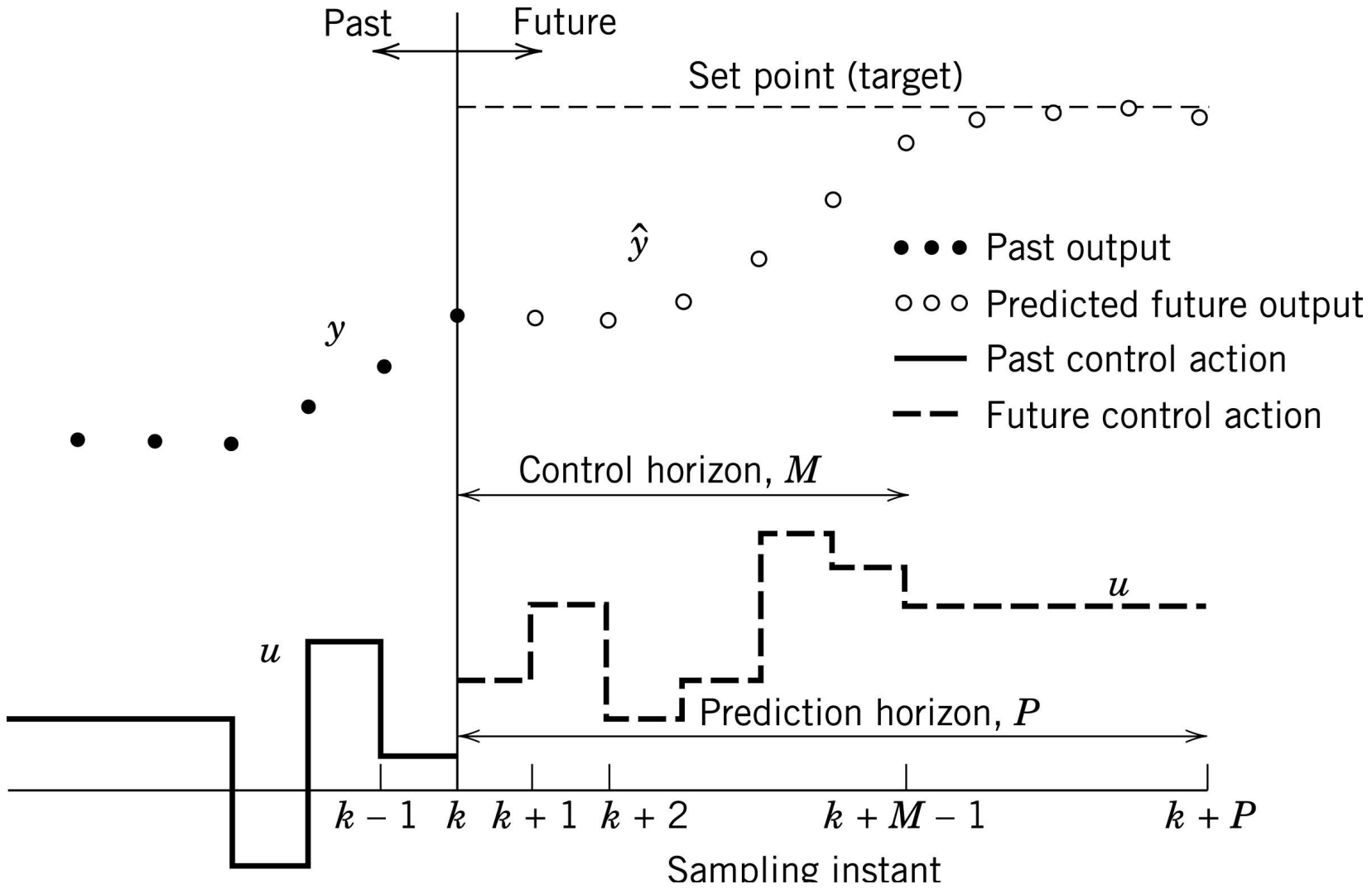
Subject to:

$$A_{eq} z = b_{eq}$$

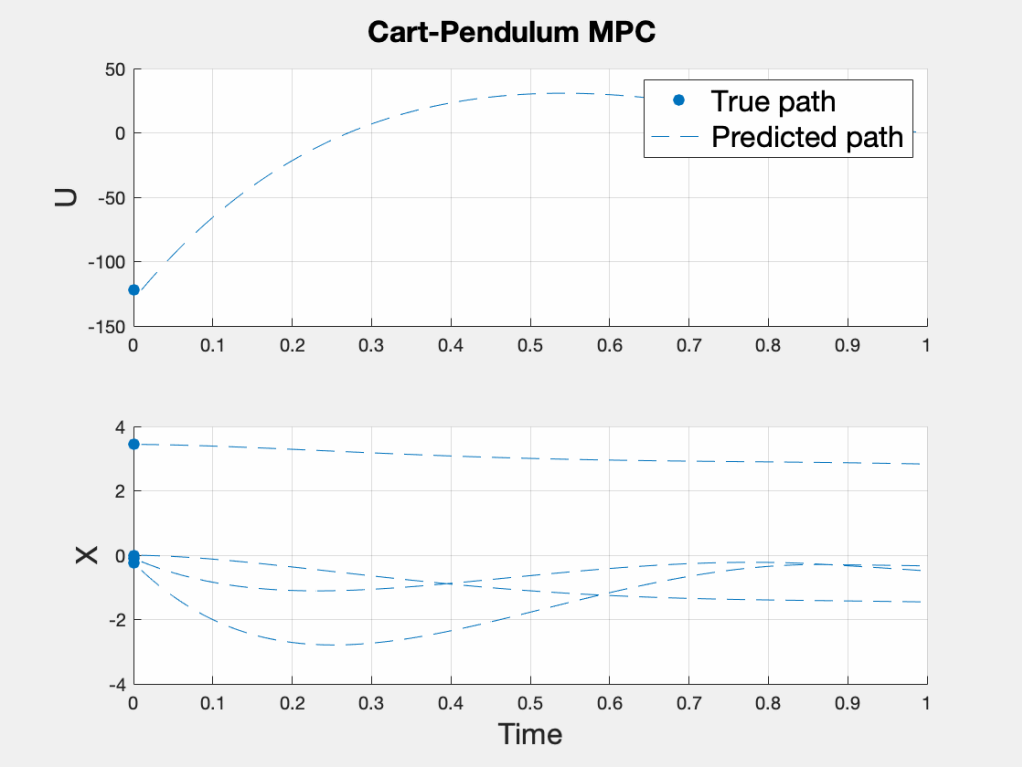
$$A_{ineq} z \leq b_{ineq}$$

Model Predictive Control:
Convert cost function to the "standard form" of a quadratic program

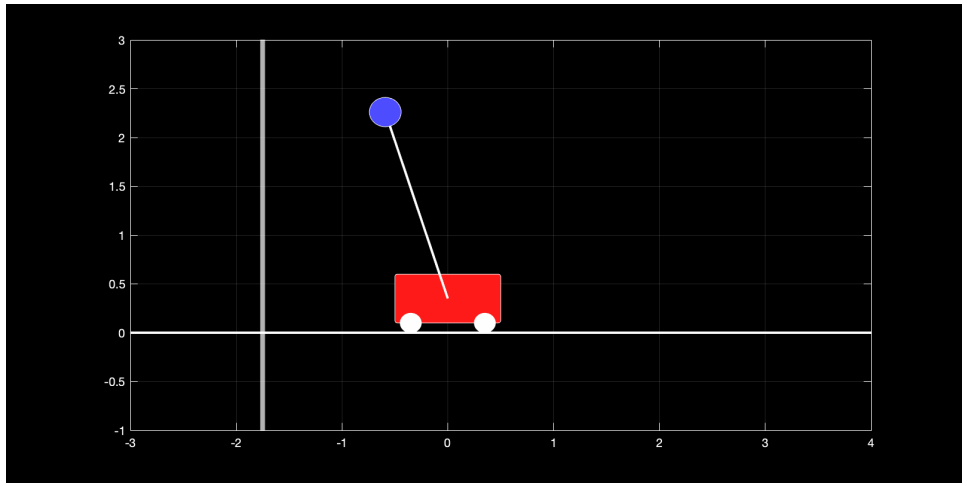
- Predict state
- Find the new constraints
- Solve Quadratic Program
- Find the discrete control action
- Calculations can be done fast enough in tokamaks



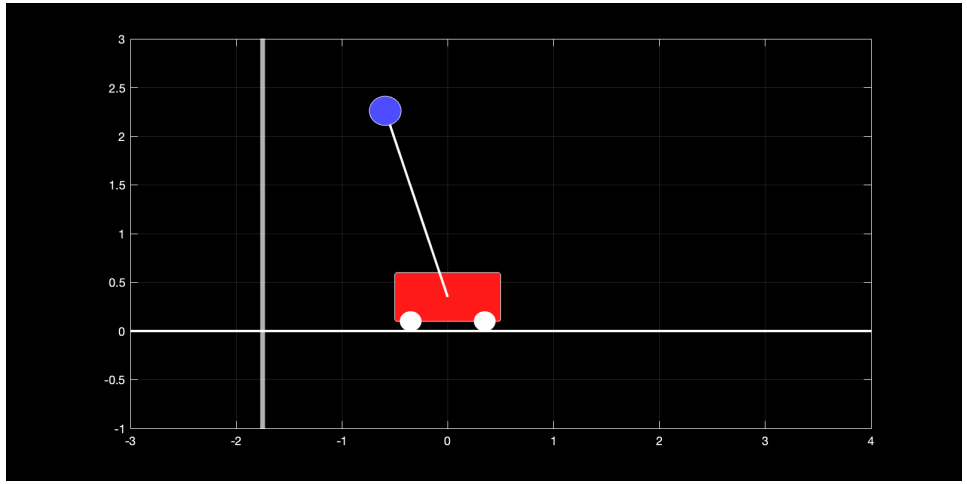
Example: inverted pendulum on a cart with wall constraint



LQR



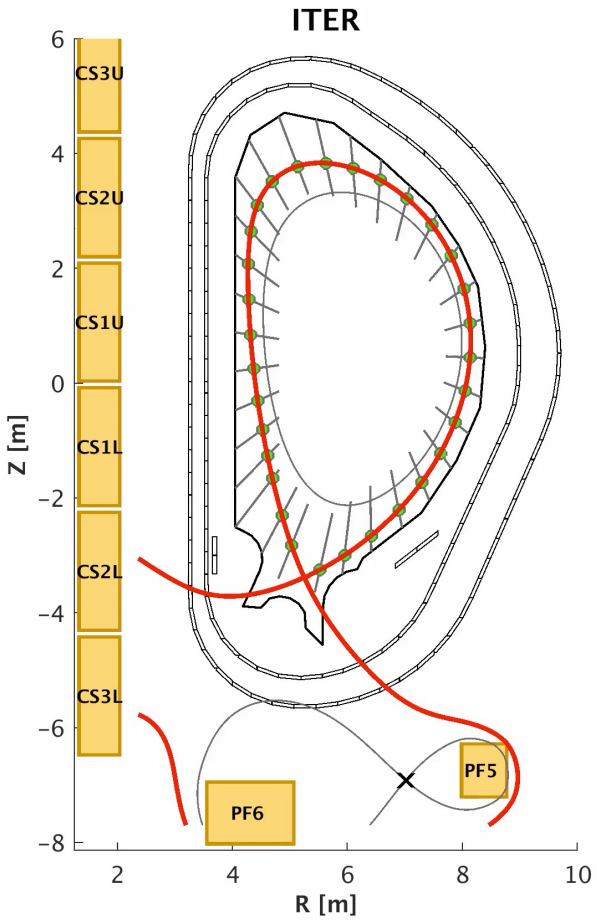
MPC



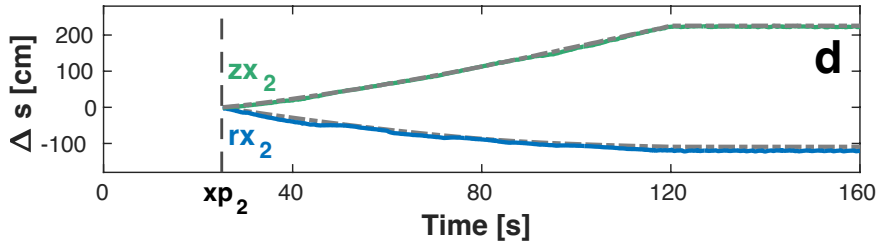
Motivated by Steve Brunton Control Bootcamp (youtube)

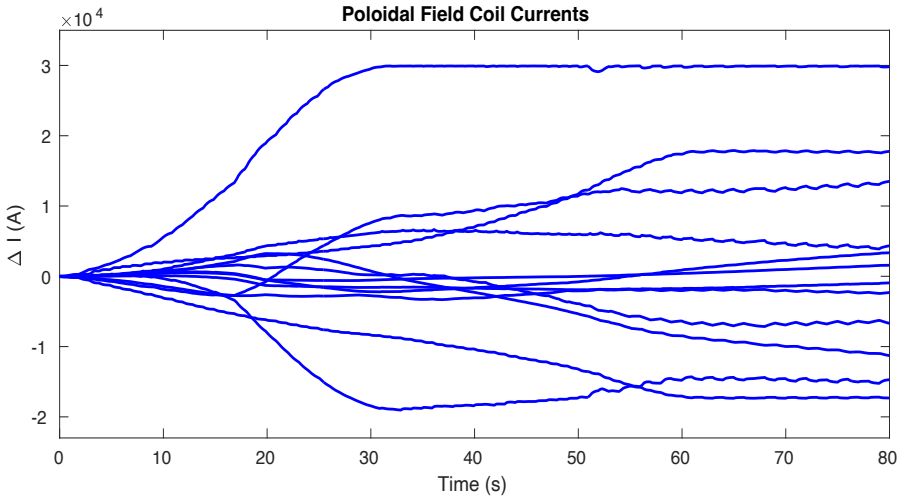
Example: MPC for shape and secondary x-point control

MPC controller guides x-point while maintaining power supply and boundary constraints



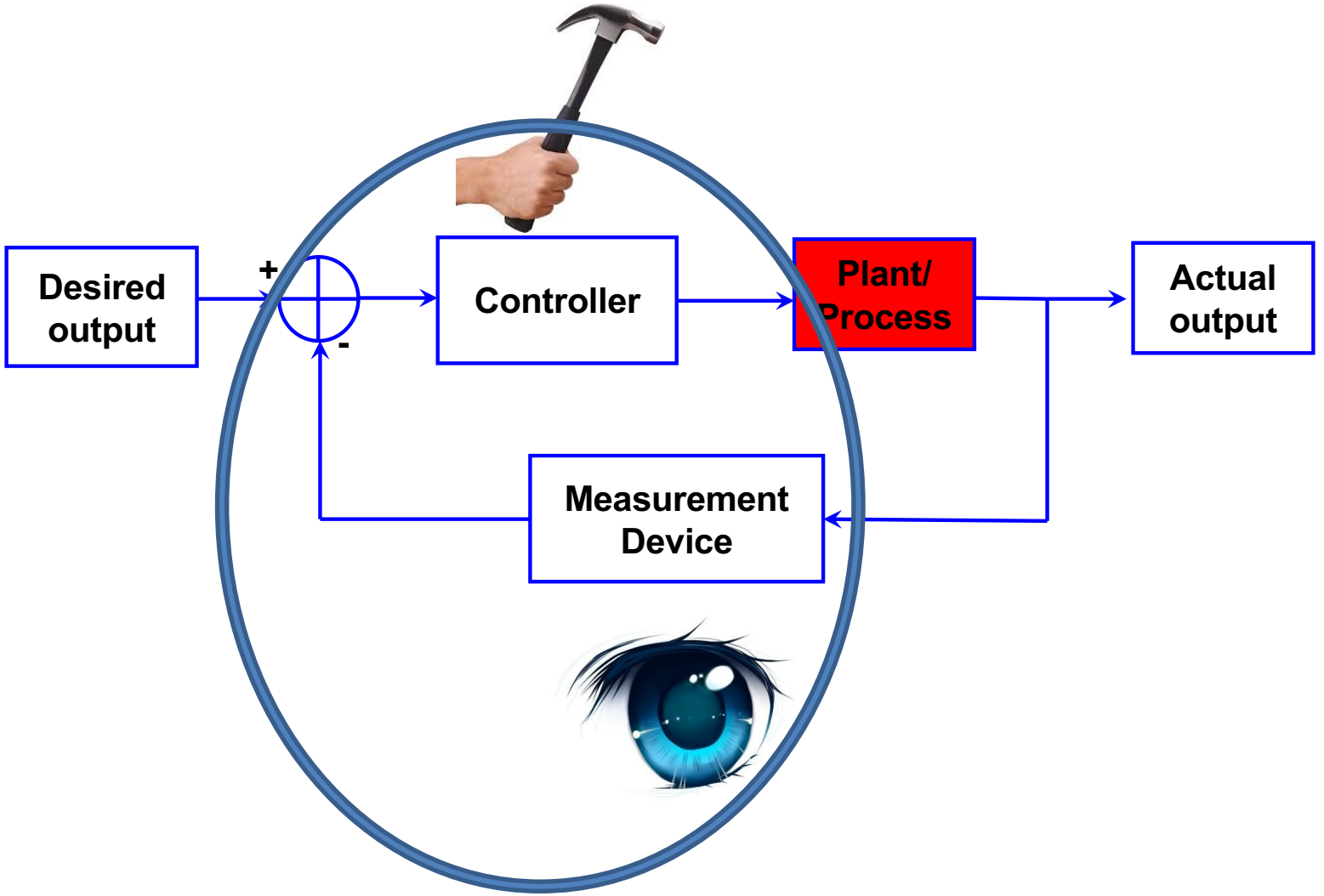
Activated Constraints			
PF Coils			Outputs
Coil #	I	V	--
PF1	< 48 kA	< 1.5 kV	$\dot{P}$ < 200 MW/s
PF2	< 55 kA	< 1.5 kV	P < 250 MW
PF3	< 55 kA	< 1.5 kV	r_{strike} on plate
PF4	< 55 kA	< 1.5 kV	z_{strike} on plate
PF5	< 52 kA	< 1.5 kV	cp_{1-9} gaps > 10 cm
PF6	< 52 kA	< 1.5 kV	cp_{10} gap > 10 cm
CS1U	< 45 kA	< 1.5 kV	cp_{11-15} gaps > 10 cm
CS1L	< 45 kA	< 3.0 kV	cp_{16} gap > 10 cm
CS2U			cp_{17} gap > 10 cm
CS2L	< 45 kA	< 1.5 kV	cp_{18-31} gaps > 10 cm
CS3U	< 45 kA	< 1.5 kV	
CS3L	< 45 kA	< 1.5 kV	





J.T. Wai, *Fusion Eng. Des.*, 2020

ML Control for Fusion:
ML as end-to-end control

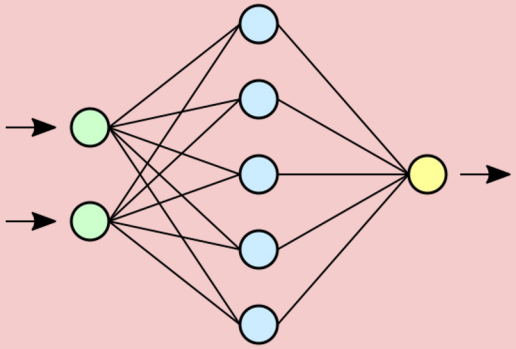


Reinforcement Learning

ML does everything: measurement + prediction + control

Learning to Control From Data: Model Predictive Control (MPC) vs Reinforcement Learning (RL)

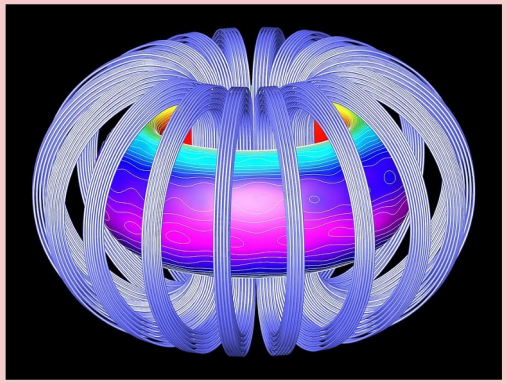
Model Predictive Control (MPC)



Model

Observe current state of the tokamak

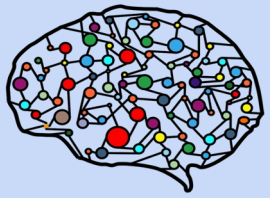
Search for action that has most benefit for the next few timesteps



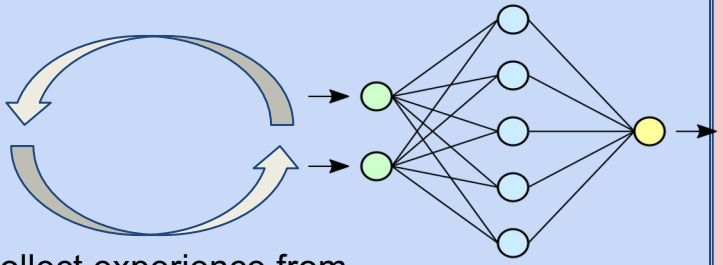
Tokamak

Online

Reinforcement Learning (RL)



Neural Net Controller



Collect experience from the model and find the best actions.

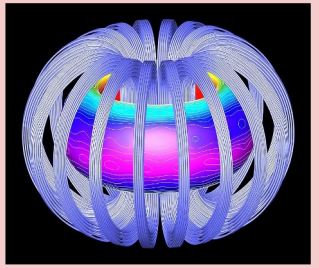
Model

Beforehand

Neural Net Controller

Observe state

Neural net forward pass to get action

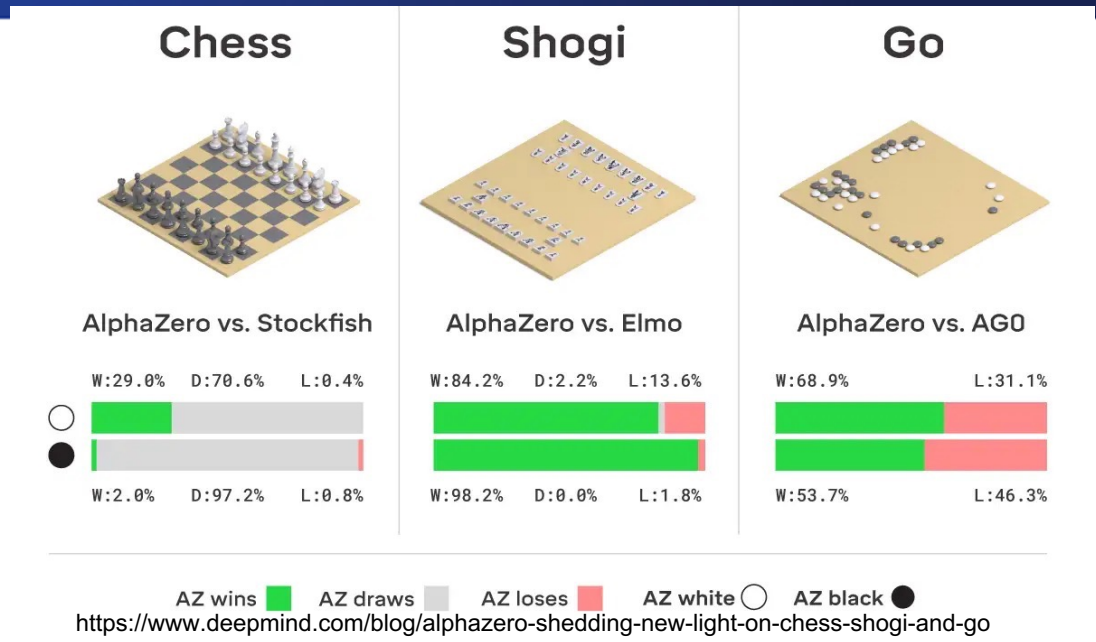


Tokamak

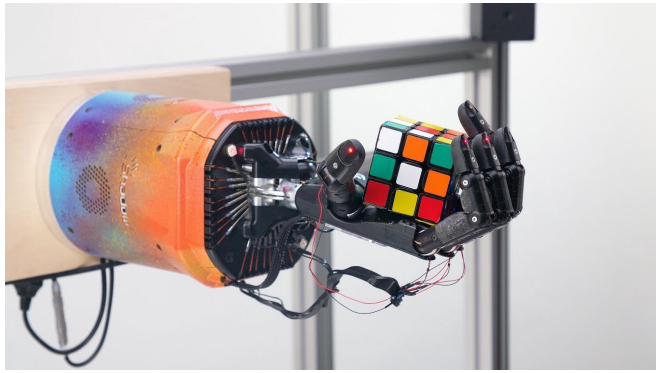
Online

Reinforcement learning

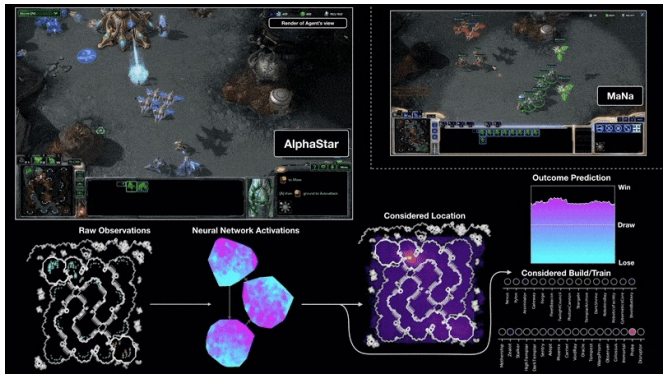
- **How do we (humans) learn to solve problems?**
  - Trial and error interaction with the environment
- **Reinforcement learning (RL) is a general framework to express how this process is performed.**
- **There are two important aspects to the paradigm**
  - It allows us to specify the goal (Reward function)
  - It can deal with long-term dependencies



https://www.deepmind.com/blog/alphazero-shedding-new-light-on-chess-shogi-and-go


https://openai.com/blog/solving-rubiks-cube/

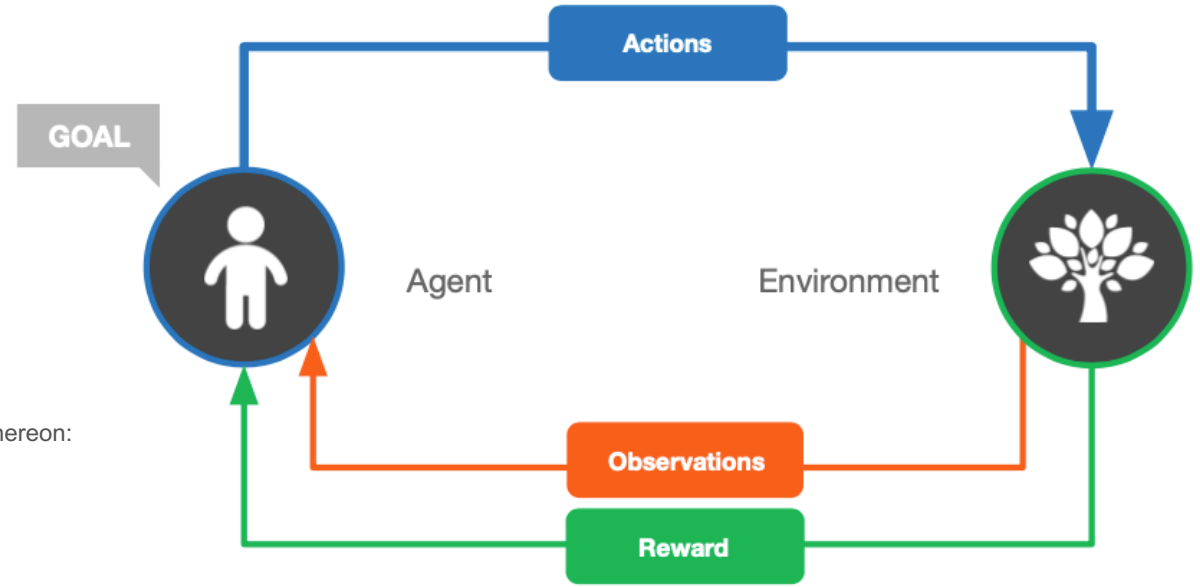
StarCraft II



https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii

Reinforcement learning versus other learning

- **Reinforcement learning: Learn by trial-and-error how to act on an environment to achieve high reward**
  - Exploration to gather experience + learning from the experience



[Figure and RL slide material from hereon: courtesy A. Abdolmaleki]

Recent applications of RL to tokamak control:

- 0D parameter control in KSTAR [Seo et al, NF 2021]
- Safety factor control [Wakatsuki et al, NF 2019]
- Ion profile control [Wakatsuki et al, NF 2021]
- Beta, Profile control [Char APS'21, Mehta APS'21]
- Shape Control
- [J. Degrave, F. Felici et al. Nature 2022]

See also: [Sutton and Barto, *Reinforcement Learning, an Introduction*. MIT Press]

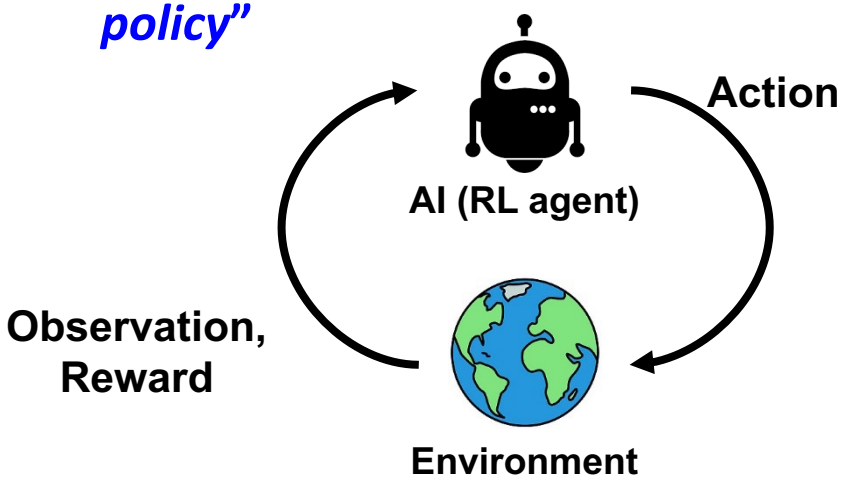
EPFL

DeepMind

Reinforcement Learning

[1] V Mnih et al, Nature 518 (2015) 519

- **Deep reinforcement learning (RL) [1]** ***"Finding the best decision-making policy"***



Time

Action	Bad	Good	Bad	Good	Good
Reward	-1	+1	-1	+1	+1

- The AI takes the action that would yield a higher reward.
- It gradually learns the **optimal decision-making policy** via experiences.

- **Difficulties of RL application in fusion research**

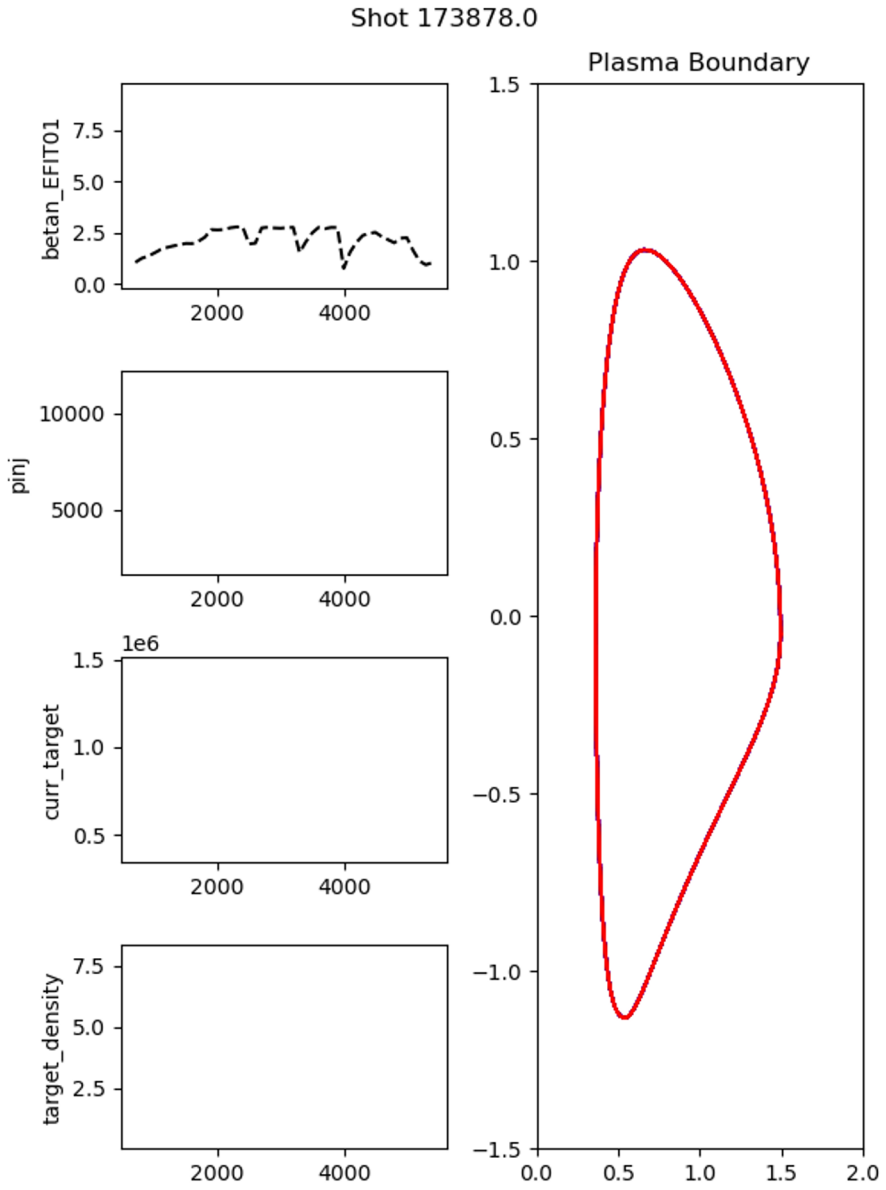
- RL requires a reliable training environment (simulation).

But we don't have a perfect all-in-one simulator including gyro-kinetic, MHD stability, H&CD, …

- RL typically requires $> 10^5$ simulation iterations to train.

But reliable theory-based simulation (TGLF, EPED, …) takes minutes to hours for a single step calculation.

Use ML Plasma Evolution Model: DIII-D training a policy for high β_N



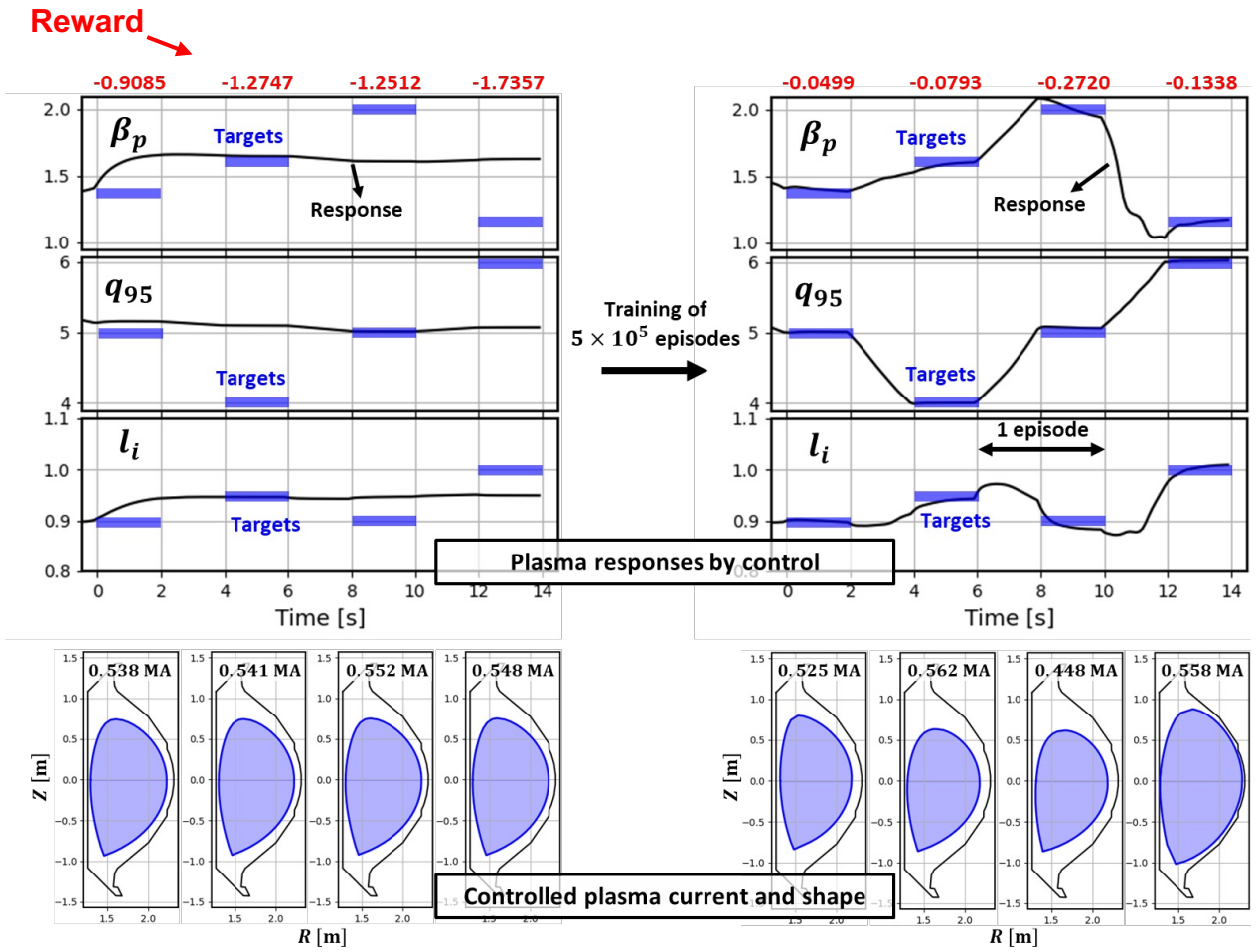
- How does this work? Simple example
- Train a policy using to maximize β_N
- This controller observes all scalar information about the plasma and is allowed to control the beams, current, density, and shape parameters.
- Rotation profile control is tested on DIII-D.
- J. Schneider, I. Char, V. Mehta et al.

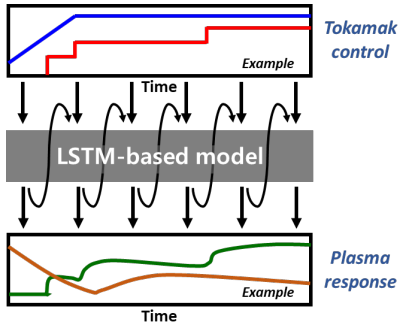
Red lines are RL Policy

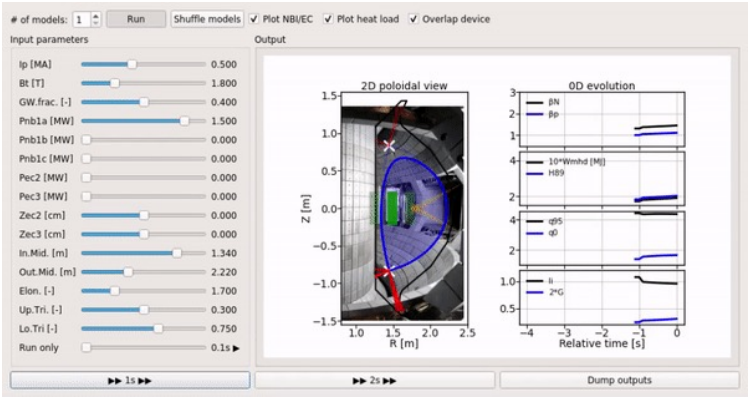
Blue lines are historical replay

Use the ML Plasma Model for RL at KSTAR

- **RL validation in the exp. databased simulation**







- After enough training, the AI determines reasonable solution of I_p and the boundary shape to reach the target of multiple parameters.

- You can try it out at *https://github.com/jaem-seo/AI_tokamak_control* if interested.

KSTAR Operation Design with RL

[1] J Seo et al, Nucl. Fusion 61 (2021) 106010
[2] J Seo et al, Nucl. Fusion 62 (2022) 086049

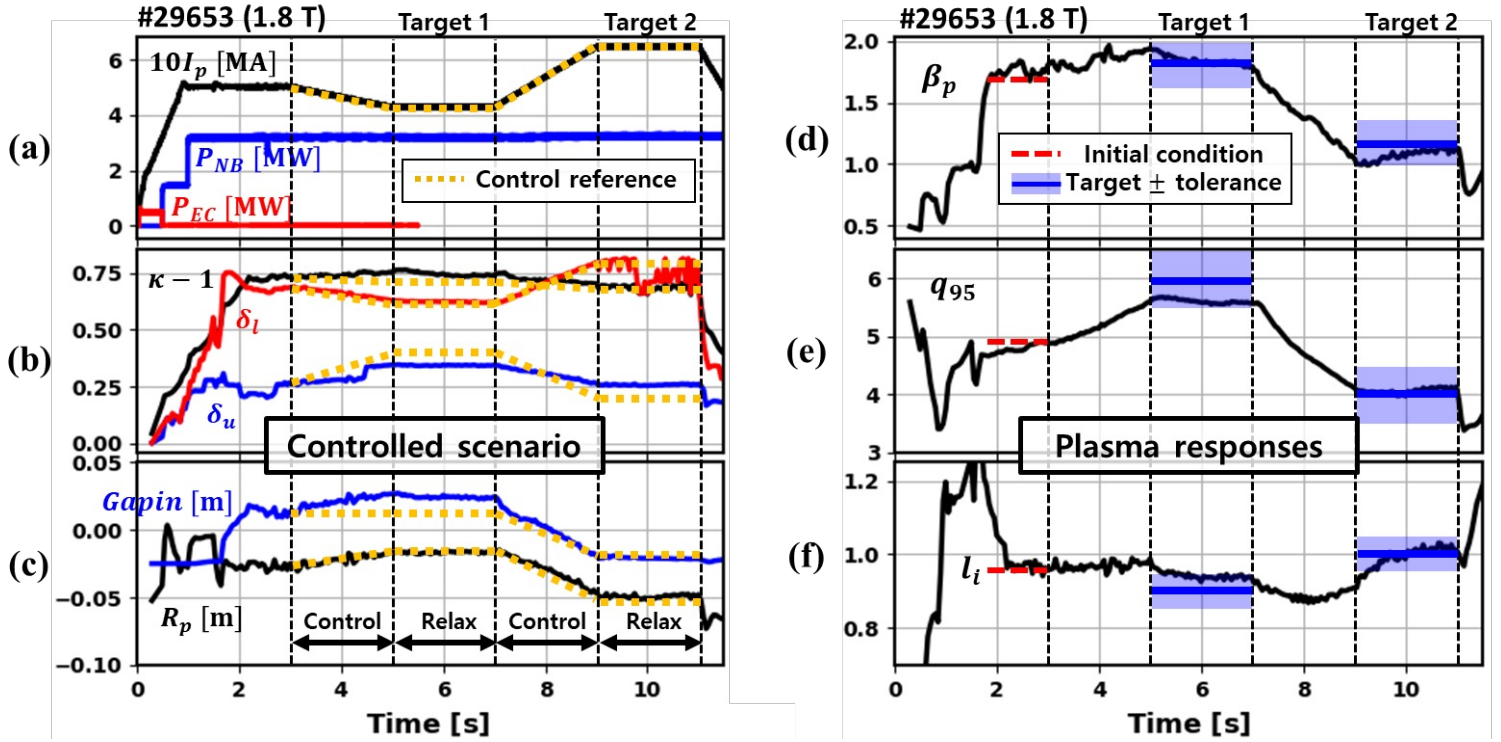
- **Validation in the KSTAR experiment**

Target setting

(β_p, q_{95}, l_i)

$= (1.8, 6.0, 0.9)$

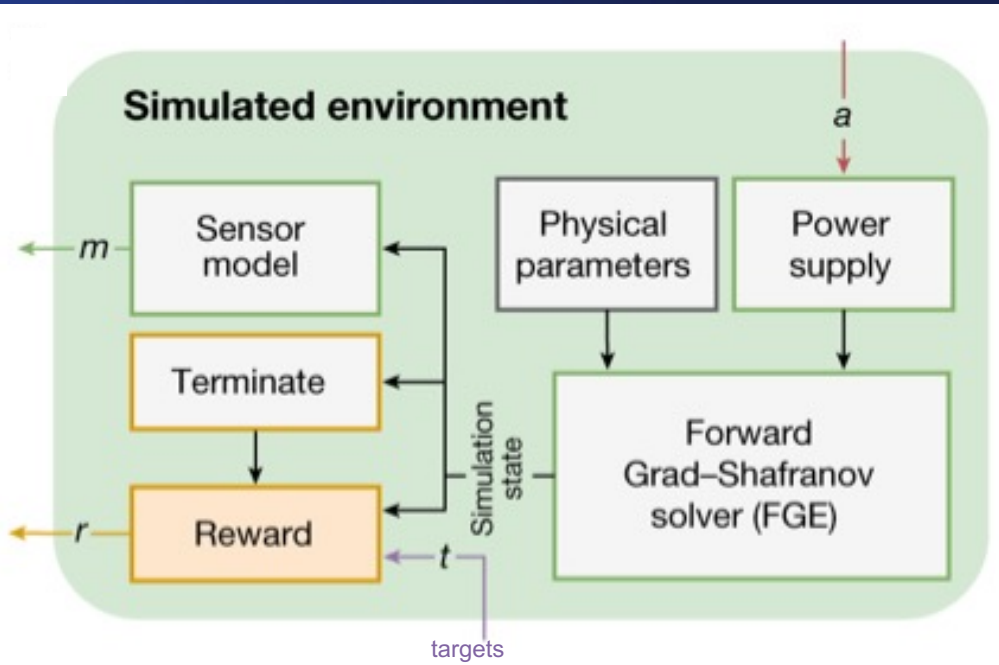
$\& \ (1.2, 4.0, 1.0)$

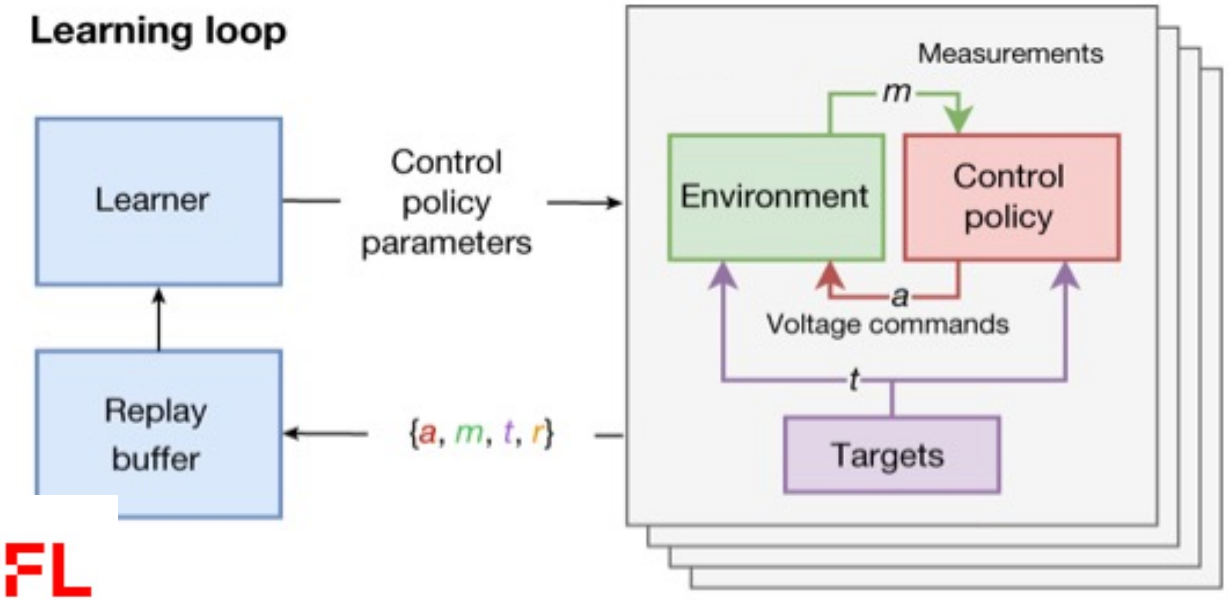


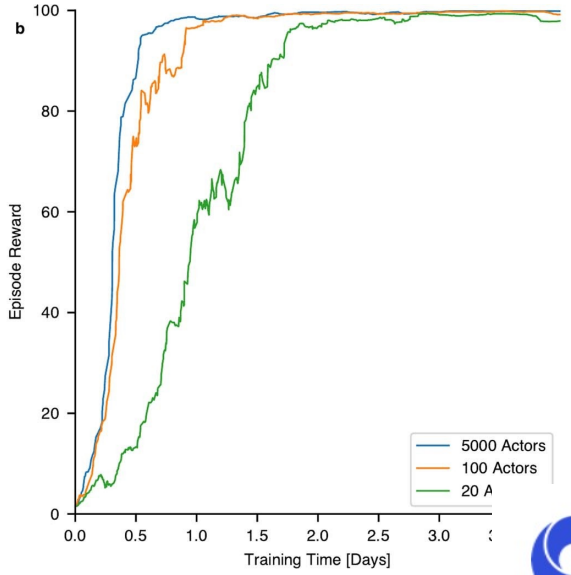
- After we set the target, the AI determines the operation trajectory. Then Jaemin an the experiment with that trajectory.
- The plasma response followed the preset targets.

Simulation based RL for Fusion Control: Free boundary Shape Control

- Free-boundary simulator
- Solve coupled system of equations:
- Grad-Shafranov equation
- Circuit equations for time evolution of currents
- Simple (feedforward, small) actor network to run in real time
- Complex (recurrent, large) critic network to learn system's dynamics



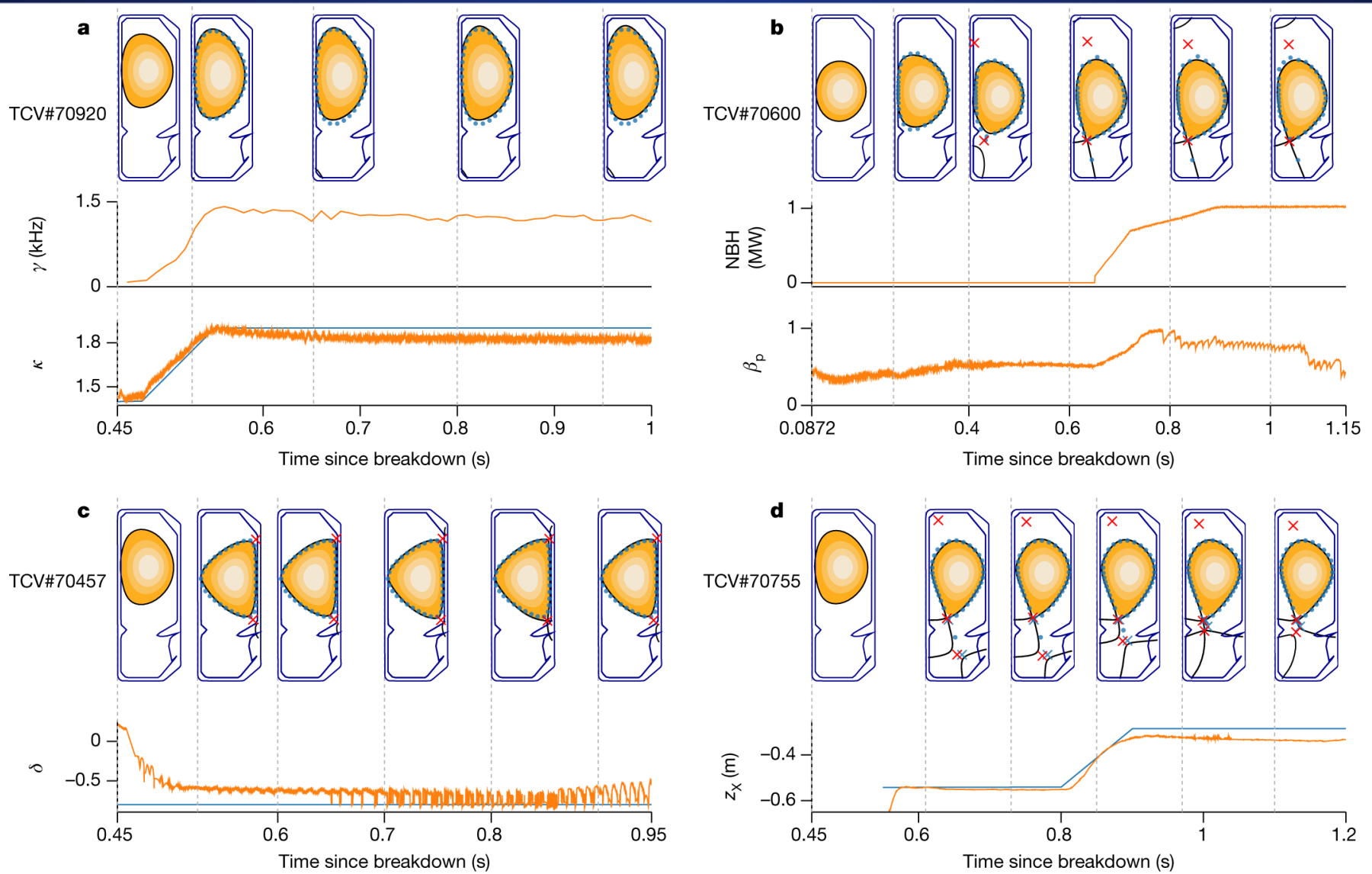




EPFL

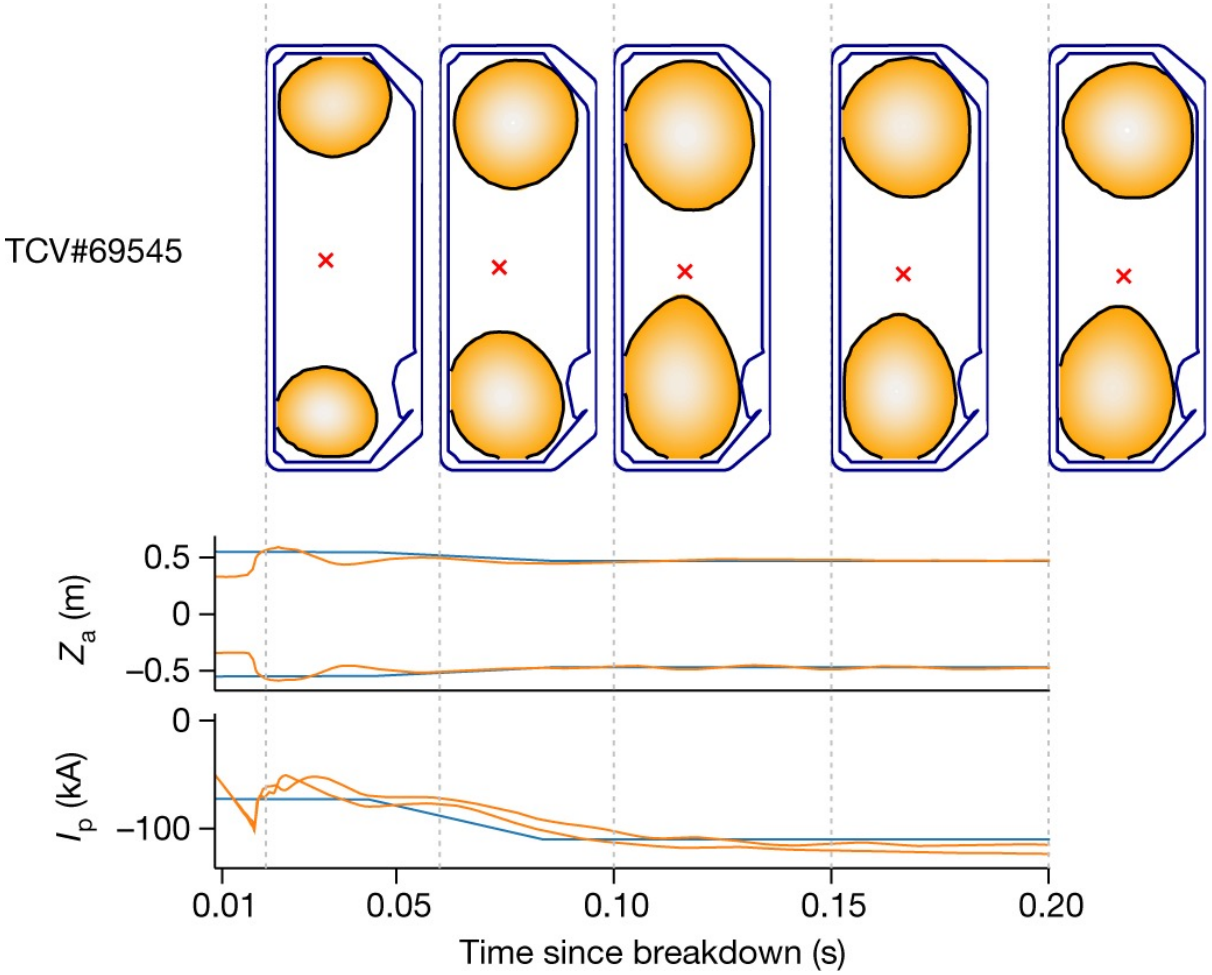
DeepMind

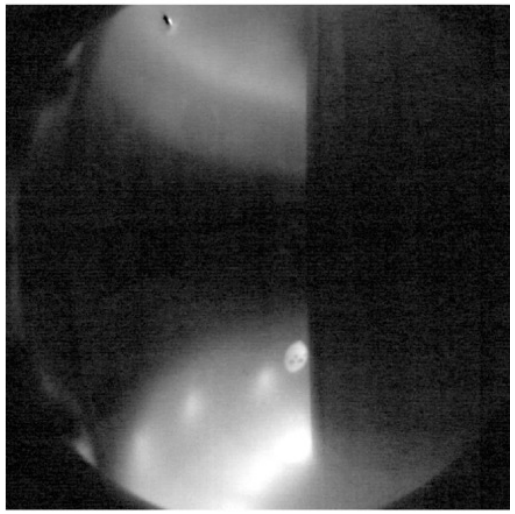
Various plasma shapes controlled in in TCV with reinforcement Learning



J. Degrave, F. Felici, et al. Nature 2022

Opening new frontiers for TCV: droplet plasmas





J. Degrave, F. Felici, et al. Nature 2022

Features of traditional / RL controllers

Traditional controllers (MIMO PID)	Our Reinforcement Learning implementation
Need to compute error for each control loop in real time	Single reward function, no explicit error signals or estimation
Need separate tuning of various control loops, using linear control techniques assuming (local) linearity	Joint solution to entire stabilization/control problem including any nonlinearities
Need control engineering + tokamak knowledge to break down control problems, design separate controllers	Domain knowledge is in simulator. Just define reward functions
Tuning of several control parameters	Reward function engineering
(Usually) Clear relation between parameters and aspects of control performance	Black-box agent
Integral control nominally gives zero steady-state error on desired quantities	No certainty of zero steady-state errors in case of external disturbances

Final Note on Robustness and Stability of Control

- Lots of people working on that
- Many recent progress by ML researchers (e.g. colleague at Princeton Prof. Anirudha Majumdar)
- E.g. - Fundamental Performance Limits for Sensor-Based Robot Control and Policy Learning https://arxiv.org/abs/2202.00129
- - Robust Control Under Uncertainty via Bounded Rationality and Differential Privacy https://ieeexplore.ieee.org/abstract/document/9811557
- Within this decade, my guess is that, we will have widely used metrics similar to the linear theory on the stability/robustness

Conclusions: Data-Based Control will gain more visibility going forward

- Lots of diagnostics information
- ML tools are maturing
- Initial applications of ML Control are begin tested
- They show promise
- We have many undergradute/graduate students and researchers working together in ML Control Project.
- Need to train more students!
- Interested in Data Science / ML for Fusion contact me ekolemen@Princeton.edu