Machine Learning in Accelerator Physics

Dr. Andrea Santamaria Garcia

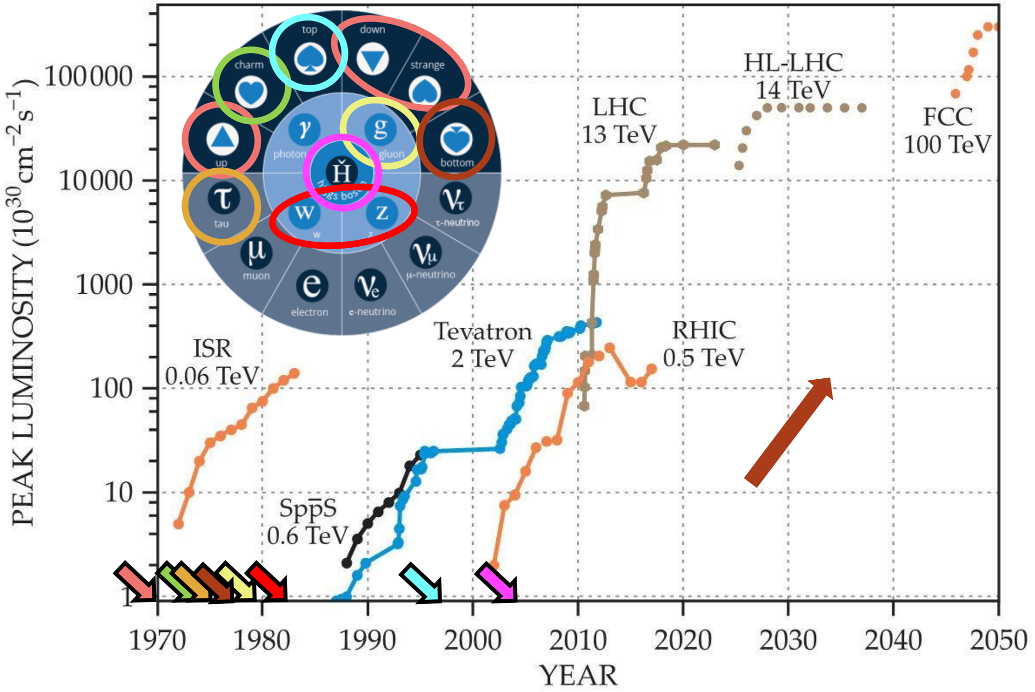
Lecturer in AI for particle accelerators
University of Liverpool & Cockcroft Institute

EUROPEAN AI FOR FUNDAMENTAL PHYSICS CONFERENCE
EuCAIFCon 2025

The particle accelerator roadmap 🧭

Ability to generate new particles via high-energy collisions

Energy frontier



Ability to probe atomic structures

Photon science

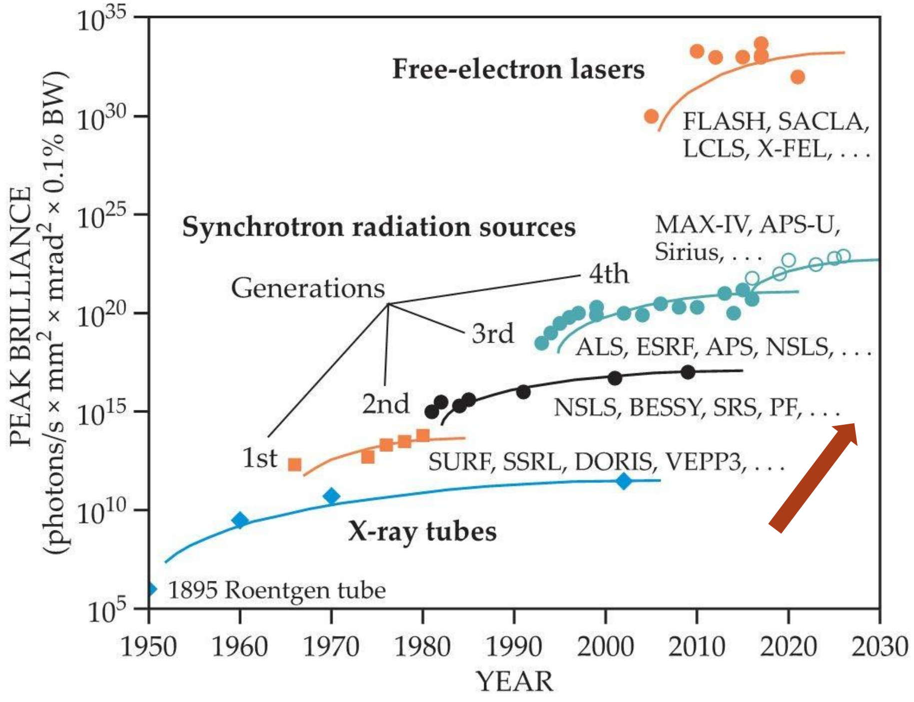


Image source: "Particle beams behind physics discoveries" (Physics Today)

Technological innovation is needed to keep up with the upcoming challenges!

Trends and challenges of frontier accelerators

Denser beams for higher luminosity and brilliance

- Complex beam dynamics
- Complex accelerator design and operation

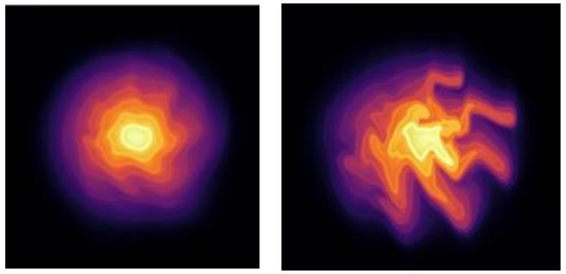


Image: KIT

Larger circular colliders for higher energies

- Orders of magnitude more signals
- Machine protection limits

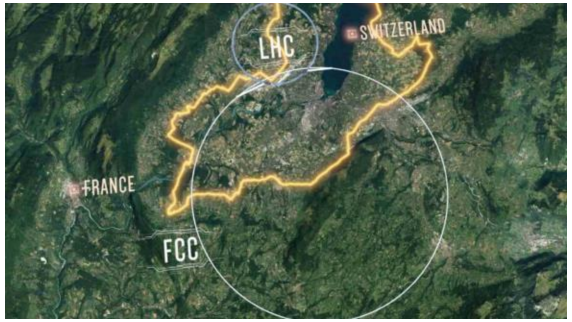


Image: CERN

Compact plasma accelerators with higher gradients

- Tight tolerances
- High-quality beams required

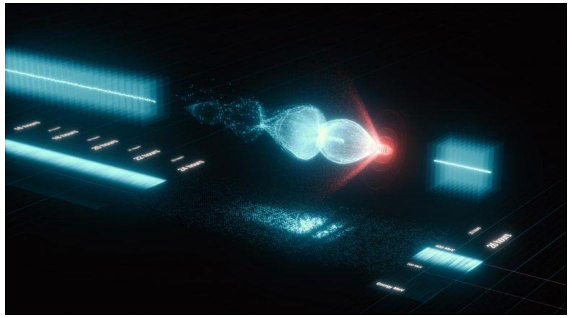
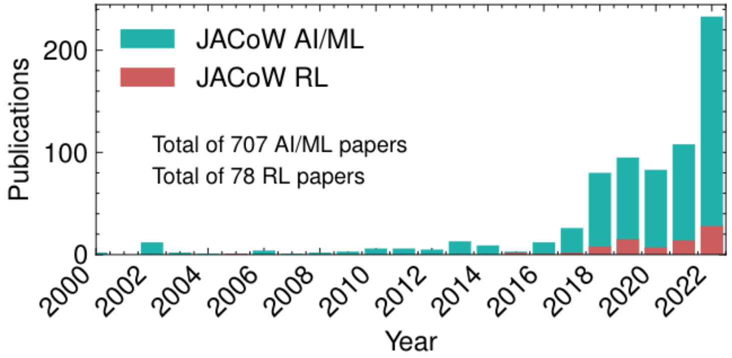
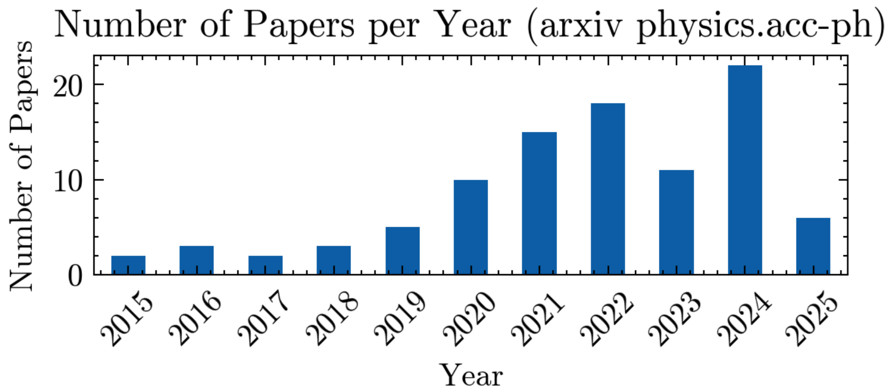


Image: DESY

FLS2023-TH3D3

Machine learning in particle accelerators





https://github.com/MALAPA-Collab/AccML-LivingReview/tree/main

Task	Goal	Methods/Concepts	Examples[1]
Detection	Detect outliers and anomalies in accelerator signals for interlock prediction, data cleaning	• Anomaly detection • Time series forecasting • Clustering	• Collimator alignment • Optics corrections • SRF quench detection
Prediction	Predict the beam properties based on accelerator parameters	• Virtual diagnostics • Surrogate models • Active learning	• Beam energy prediction • Accelerator design • Phase space reconstruction
Optimization	Achieve desired beam properties or states by tuning accelerator parameters	• Numerical optimizers • Bayesian optimization • Genetic algorithm	• Injection efficiency • Radiation intensity
Control	Control the state of the beam in real time in a dynamically changing environment	• Reinforcement learning • Bayesian optimization • Extremum Seeking	• Trajectory steering • Instability control

[1] non-exhaustive

ML in Accelerators Community and Roadmap

Yearly community meetings called **MaLAPA**
(Machine LeArning for Particle Accelerators)

Strategy

- Define control system needs for LLMs, agents, robotics
- Create a central, forward-looking strategy with ethics and safety
- Productionise and demonstrate existing ML/AI tools
- Collect impact metrics and success stories

MaLAPA Community Actions

- Share software, datasets, and tutorials via the MaLAPA repo
- Promote joint development over siloed tools
- Establish test problems to benchmark ML methods
- Join the MaLAPA Discord for year-round collaboration

April 2025, 5th edition, CERN

https://indico.cern.ch/event/1382428/

https://github.com/MALAPA-Collab

ML trends in particle accelerators

MLOps Challenges

- No common MLOps framework in the community yet
- Entire ML lifecycle needs better support
- Start with human-in-the-loop systems; aim for higher autonomy
- Need for tutorials and success case studies

Digital Twin Framework

- Virtual accelerator: synchronized, predictive, actionable.
- Supports: anomaly detection, optimization, safe experimentation.
- Needs:
  - Shared modelling libraries
  - Interfaces for integration with control systems
  - Monitoring and update pipelines
  - Community standardization efforts

LLMs and Agentic Interfaces

- Serve as operator interfaces
- Parse logs, discover correlations
- Act as collaborative agents during design
- Vision: 'LLM-based expert pools' for collective knowledge and feedback

ML trends in particle accelerators

Tuning and control

- Bayesian optimisation
- Reinforcement learning

PHYSICAL REVIEW ACCELERATORS AND BEAMS **27**, 084801 (2024)

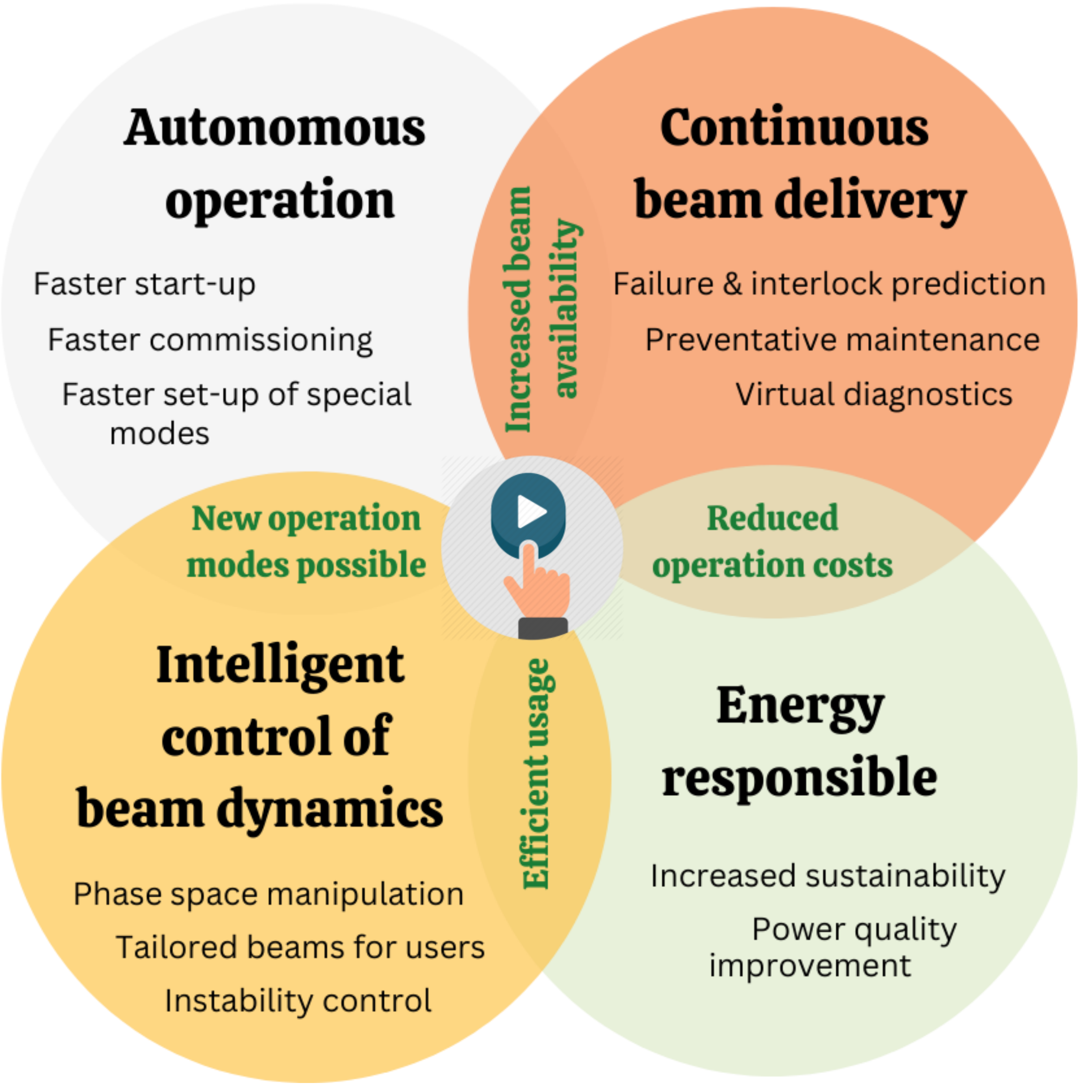
Review Article

Bayesian optimization algorithms for accelerator physics

Ryan Roussel,[1,*] Auralee L. Edelen,[1] Tobias Boltz,[1] Dylan Kennedy,[1] Zhe Zhang,[1]
Fuhao Ji,[1] Xiaobiao Huang,[1] Daniel Ratner,[1] Andrea Santamaria Garcia,[2] Chenran Xu,[2]
Jan Kaiser,[3] Angel Ferran Pousa,[3] Annika Eichler,[3,4] Jannis O. Lübsen,[4]
Natalie M. Isenberg,[5] Yuan Gao,[5] Nikita Kuklev,[6] Jose Martinez,[6] Brahim Mustapha,[6]
Verena Kain,[7] Christopher Mayes,[8] Weijian Lin,[9] Simone Maria Liuzzo,[10]
Jason St. John,[11] Matthew J. V. Streeter,[12] Remi Lehe,[13] and Willie Neiswanger[14]

https://doi.org/10.1103/PhysRevAccelBeams.27.084801

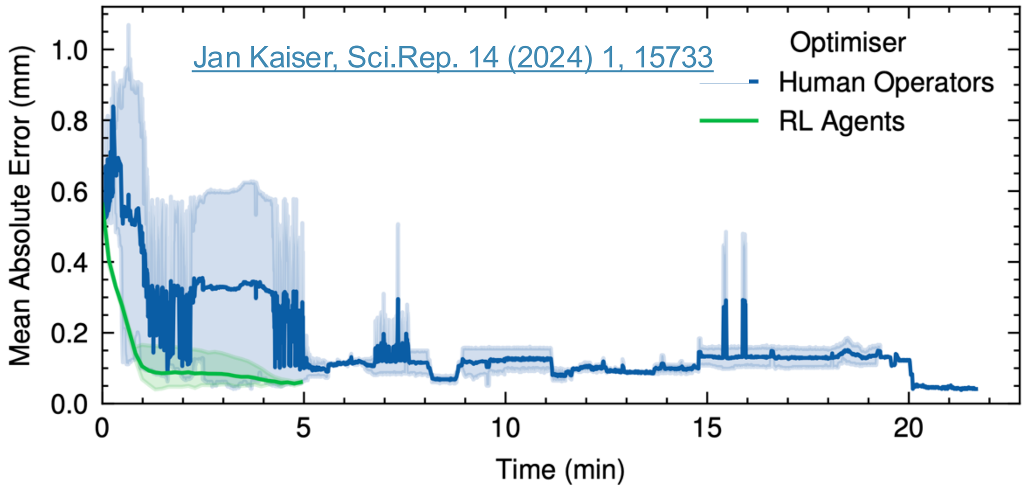
A vision for future accelerators, driven by ML



FLS2023-TH3D3

What is the path to true autonomous accelerators? (with continuous, robust, and safe control)

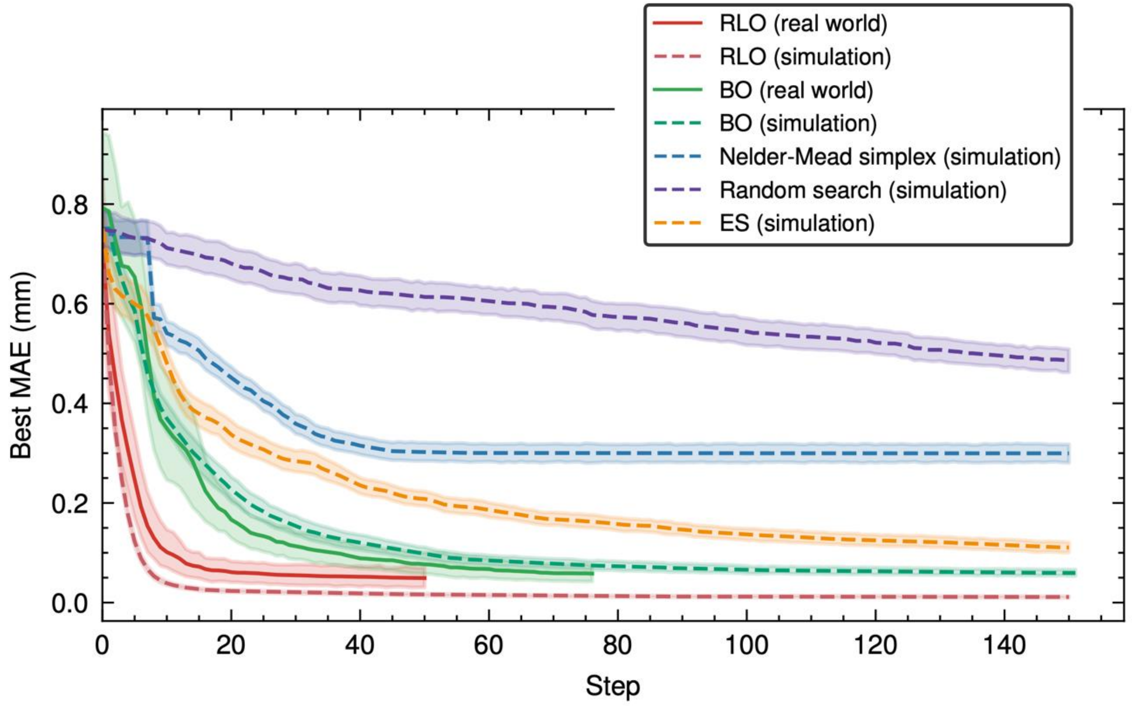
Maybe reinforcement learning



Why RL for particle accelerators?

Reinforcement learning can:

- **Adapt dynamically** to changing environments
- **Scale better** to high-dimensional problems than other methods
- Consider **delayed consequences**
- Perform **closed-loop control in real time**
- **Converge faster** than any other methods after training



Jan Kaiser, Sci.Rep. 14 (2024) 1, 15733

RL is a promising and powerful framework for adaptive, goal-directed behaviour in complex environments

MACHINE LEARNING

SUPERVISED LEARNING

Classification, prediction, forecasting
computer learns by example

Spam detection
Weather forecasting
Housing prices prediction
Stock market prediction

UNSUPERVISED LEARNING

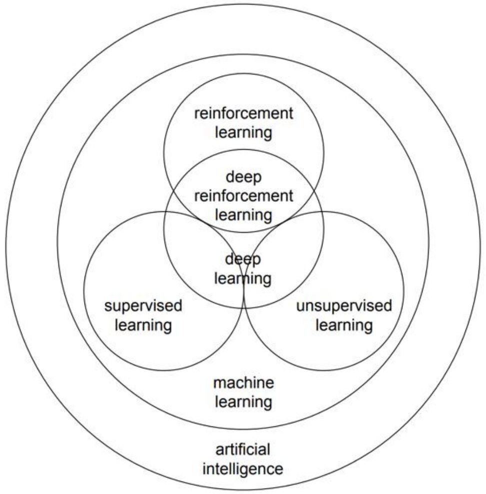
Segmentation of data
computer learns without prior information about the data

Medical diagnosis
Fraud (anomaly) detection
Market segmentation
Pattern recognition

REINFORCEMENT LEARNING

Real-time decisions
computer learns through trial and error

Self-driving cars
Make financial trades
Gaming (AlphaGo)
Robotics manipulation



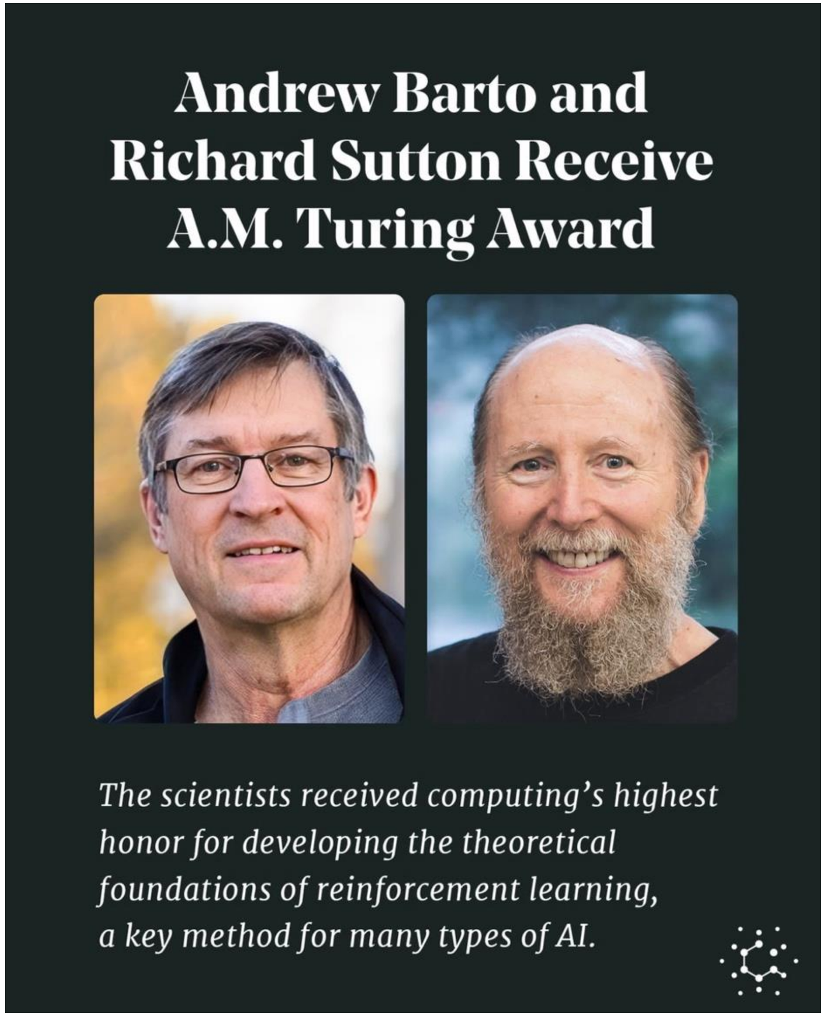
Reinforcement learning 🧠

More than machine learning

BEHAVIOUR LEARNING

Psychology (classical conditioning)
Neuroscience (reward system)
Economics (game theory)
Mathematics (operations research)
Engineering (optimal control, planning)

Reinforcement learning



Quanta Magazine

What we understand today as RL (established in the 1980s) inherits concepts from:

- **Trial-and-error learning**
  Behavioural basis 🧠
  Learning emerges through repeated interaction, reward feedback, and adaptation

- **Optimal control**
  Mathematical framework 📐
  Markov decision processes (MDPs), Markov property, Bellman equation, partially observable MDPs (POMDPs), value function, policy function, dynamic programming

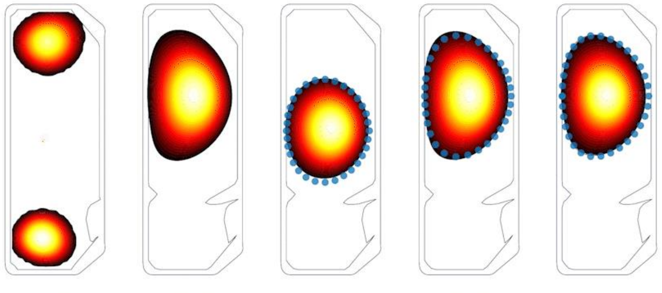
- **Temporal difference learning**
  Adaptability and scalability 🔄
  Enables prediction and learning from partial experiences

So, what can RL do *in practice*?

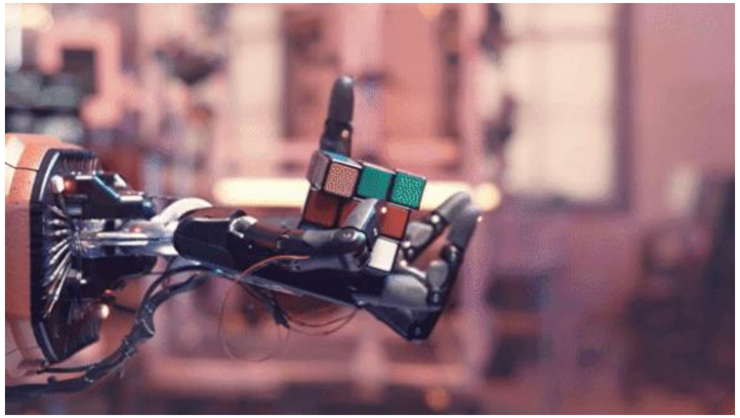
Modern RL = **deep RL**, which allows sequential decision making in continuous and infinite environments thanks to function approximation with deep neural networks.

https://arxiv.org/abs/1707.02286

Control the plasma in a tokamak fusion reactor



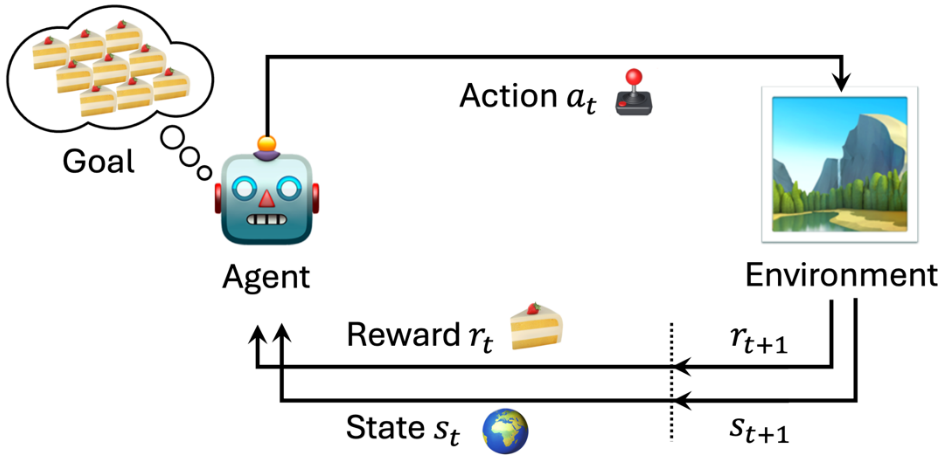
https://www.deepmind.com/publications/playing-atari-with-deep-reinforcement-learning



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

https://openai.com/index/openai-five/

RL in a nutshell



An agent (algorithm) learns through trial-and-error by interacting with a dynamic environment

Action space ***Reward function***

$$(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$$

State space ***Transition probability*** ***Discount factor***

Stochastic decision making is modelled by Markov decision processes (MDPs), a 5-tuple

Agent

1. Executes **action a_t** ***free-will***
2. Receives **observation s_t** ***perception***
3. Receives scalar **reward r_t** ***motivation***

Reward

Scalar feedback signal r_t that indicates how well the agent is doing at step t

Cumulative reward (return) $\mathcal{G}_t(\tau) = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad \gamma \in [0, 1)$

Goal

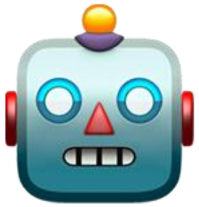
Maximisation of cumulative reward $\mathcal{G}_t$ through selected actions

Simple concept from which intelligent behaviour emerges

"Reward is enough" by Silver et al. (2021)

RL in a nutshell

How does the agent "learn"?



What behaviours perform well in this environment?

Policy: agent's behaviour function (how it picks its actions)

$$\pi : \mathcal{S} \rightarrow \mathcal{A}$$

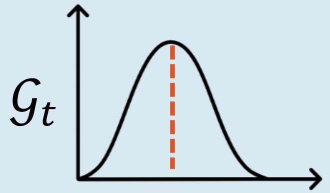
$$\pi(s) = a$$

$$\pi(a|s) = \mathbb{P}[a|s]$$

Estimate the utility of taking actions in particular states of the environment (evaluation of the policy)

Value function: how good each state and/or action are

$\mathcal{V}^{\pi}$= state-value function $\mathcal{Q}^{\pi}$ = action-value function



$\mathcal{V}^{\pi}, \mathcal{Q}^{\pi}$ are an estimation of where the return distribution is centered

- **Prediction**: evaluate the future given a policy
- **Control**: optimise the future (find the best policy)

$$\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} \mathbb{E}_{\pi}[\mathcal{G}_t]$$

where π^* is the optimal policy

RL in a nutshell

How does the agent “learn”?

At every time step t:

1. **The agent is in state s_t**
2. **The agent selects an action $a_t \sim \pi(a|s)$**

   This action is chosen based on the agent’s current policy π, which may prioritise actions that maximise expected future reward, e.g.:

$$a_t = \arg\max_a Q^\pi(s_t, a)$$

3. **The environment returns:**
   - Next state s_{t+1}
   - Reward r_t
4. **The agent learns from the experience (s_t, a_t, r_t, s_{t+1})**

   Value-based methods: update value estimates (assess value of action)

   Policy-based methods: directly improve the policy (how to act)

Types of learning

Online: data is actively collected during training

Offline: learns from a fixed dataset (supervised learning)

Simulation-based: training in a virtual or simulated environment

Experiment-based: direct interaction with a real-world system

On-policy: policy is updated from data collected by the current version of the policy

Off-policy: can learn from data generated by a different policy

Main challenges of RL deployment

Policy and value functions are approximated by deep neural networks (DNNs)

$$\max_{\theta} J(\pi_\theta) = \max_{\theta} \mathbb{E}_{\pi_\theta}[\mathcal{G}_t] \qquad \theta \leftarrow \theta + \alpha \nabla J(\pi_\theta)|_\theta$$

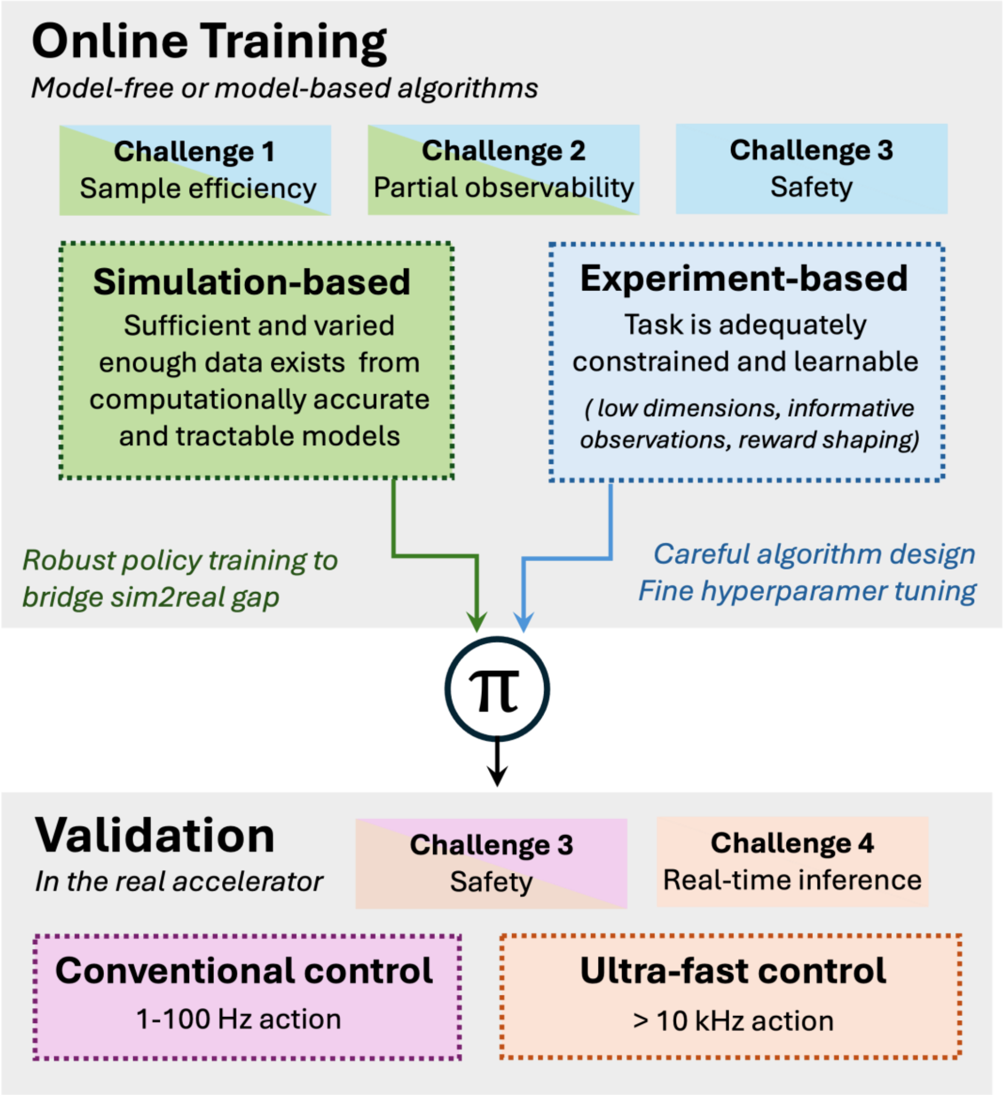
Generalisation capabilities

→ quantity and quality of data

No real **convergence** guarantees

Training instability due to:

- Bootstrapped value targets
- Function approximation bias (net. architecture, weight initialization, training dynamics)
- Hyperparameter sensitivity (high variance in performance across random seeds)



Challenge 1: sample efficiency

Sample efficiency ↗
Training cost ↘

Sample efficiency: number of interactions with the environment required to achieve a certain level of performance during the decision-making process

	✅	❌
Model-free, on-policy *Policy gradient: REINFORCE* *Actor-critic: PPO, A2C*	▪ *Simple implementation* ▪ *Good for continuous action*	▪ *Poor sample efficiency* ▪ *Large variance if unclipped*
Model-free, off-policy, value based *DQN*	▪ *Sample efficient* ▪ *Efficient in discrete envs*	▪ *Unstable (function appr.)* ▪ *Limited to discrete or low-dimensions*
Model-free, off-policy, actor-critic *DDPG, TD3, SAC*	▪ *Sample efficient* ▪ *Good for continuous action* ▪ *Stable*	▪ *Hard to tune* ▪ *Hyperparameter sensitivity* ▪ *Overestimation bias*
Model-based RL	*Very high sample efficiency*	*Model is hard to train, complex to tune, brittle & sensitive*

?

Challenge 1: sample efficiency

How does this play in practice?

Idealised setting ---------------------------→ ***Noisy, unpredictable dynamics***

sim2real gap

Training

Simulation-based

Sufficient and varied enough data exists from computationally accurate and tractable models

Diff. simulations (Cheetah), DNN surrogates, GPU-accelerated

~10^3-10^6 interactions

→ π →

Validation

In the real accelerator

*Need **robust** policies!*

Domain randomization: train on "perturbed" environments

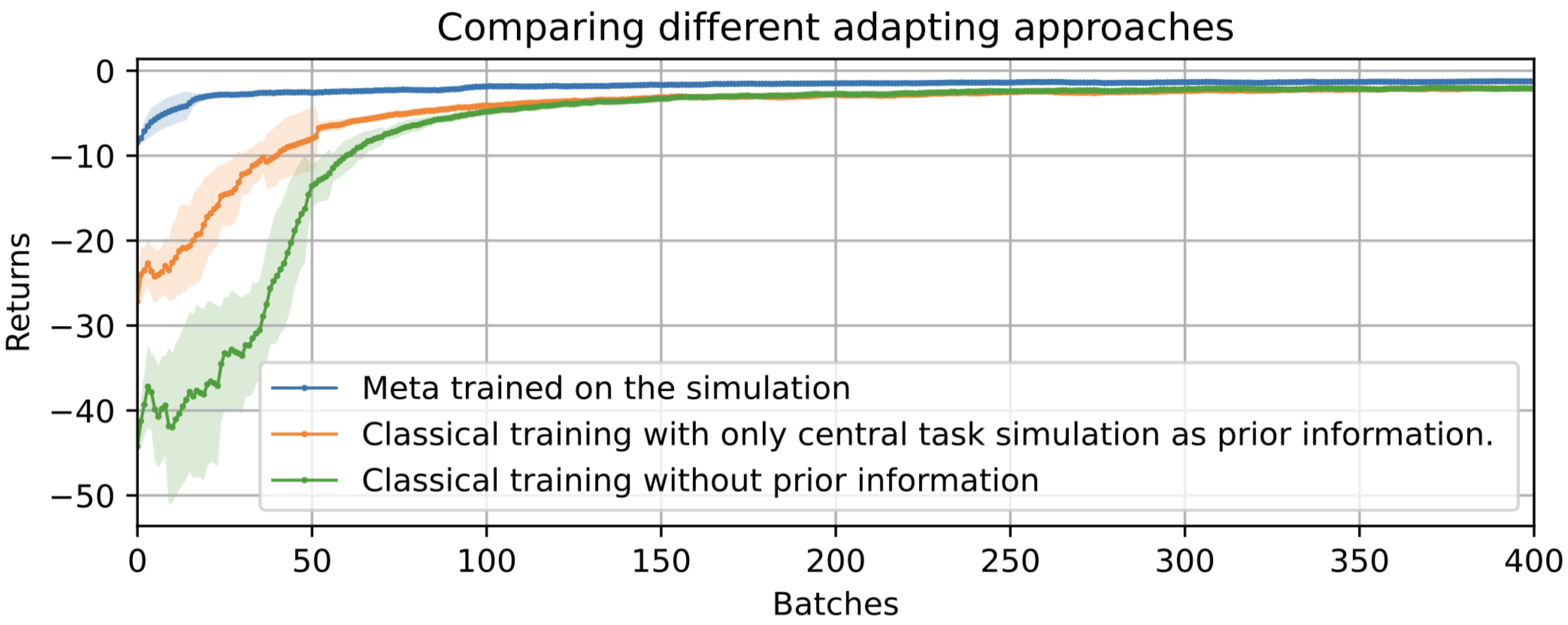
Meta-RL: learn an adaptable policy that can quickly specialize with minimal fine tuning

More robust and sample efficient in validation (real machine) but requires more samples (simulation)

Challenge 1: sample efficiency

How does this play in practice?

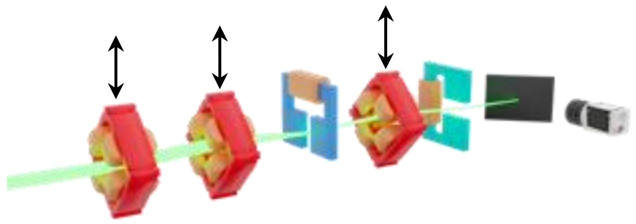
Beam steering task at AWAKE beamline
10 H dipoles, 10 V dipoles, 10 BPMs → ideal trajectory



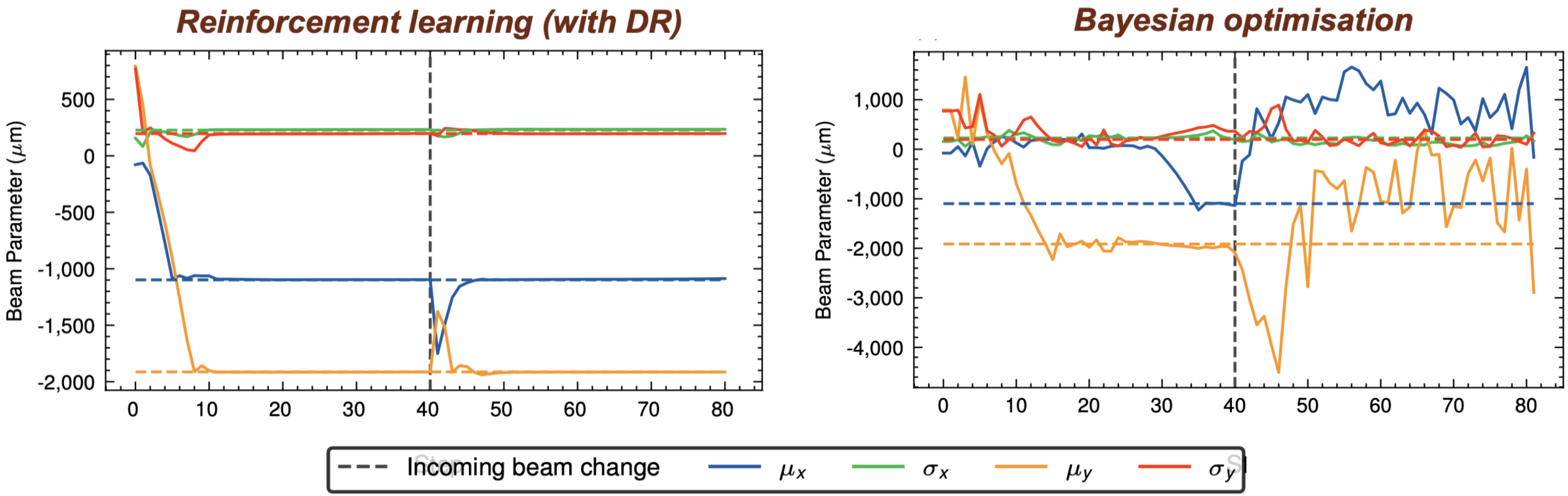
[Towards few-shot reinforcement learning in particle accelerator control, *JACoW* IPAC2024 (2024) TUPS60]

Robustness & sample efficiency

How does this play in practice?



Beam steering and focusing task at ARES linear accelerator
3 quadrupoles, 2 correctors → target beam size and position on a screen
Recovery from sudden change in incoming beam



[Reinforcement learning-trained optimisers and Bayesian optimisation for online particle accelerator tuning, *Sci.Rep.* 14 (2024) 1, 15733](#)

Challenge 1: sample efficiency

How does this play in practice?

Training

Experiment-based

Task is adequately constrained and learnable

(low dimensions, informative observations, reward shaping)

Very rare! Only a handful of cases

FERMI, AWAKE, Linac4, KARA

- ~10^3 real-world interactions required for training
- Low-dimensional action and observation spaces
- Dense reward
- Very sensitive to hyperparameter choices
- Hard to find dedicated beamtime
- Safety concerns

N. Bruchon “Basic reinforcement learning techniques to control the intensity of a seeded free-electron laser”, Electronics, vol. 9, no. 5, 2020

S. Hirlaender “Model-free and Bayesian ensembling model-based deep reinforcement learning for particle accelerator control demonstrated on the FERMI FEL”, arXiv:2012.09737, 2022.

V. Kain “Sample-efficient reinforcement learning for CERN accelerator control”, Phys. Rev. Accel. Beams, vol. 23, no. 12, p. 124 801, 2020.

L. Scomparin “Preliminary results on the reinforcement learning-based control of the microbunching instability” IPAC2024-TUPS61

Challenge 2: partial observability

Fully observable environments

The agent directly observes the true state of the environment, which includes everything relevant

state of the agent (belief)

$$\mathcal{O}_t = \mathcal{S}_t^a = \mathcal{S}_t^e$$

observation

true state of the environment

Partially observable environments

The agent receives partial observations and must create its own state representation

$$\mathcal{O}_t \neq \mathcal{S}_t^a \neq \mathcal{S}_t^e$$

partial, noisy, filtered

Stacking recent observations to approximate motion

Example: autonomous driving

$\mathcal{S}_t^e$: we know all cars exact positions, road friction, weather conditions, etc.

$\mathcal{O}_t$: pixels from cameras, GPS signal, lidar?

what the agent can "sense"

$\mathcal{S}_t^a$: estimated positions and speeds based on past observations

what the agent "believes" the environment is

Challenge 2: partial observability

Ideal setting

State fully observable

- MDP (finite, discrete)
- Model known
- Value function exact
- Optimal policy computable

We can completely solve the control problem and find the **optimal policy** π^*

vs

Real world

State partially observable

- POMDP (infinite, continuous)
- Model unknown or learned
- Value function approximated
- Policy approximated

We just want **good-enough policies** that are robust, generalizable, sample-efficient, and safe

Challenge 2: partial observability

Ideal setting

State fully observable

- MDP (finite, discrete)
- Model known
- Value function exact
- Optimal policy computable

vs

Real world

State partially observable

- POMDP (infinite, continuous)
- Model unknown or learned
- Value function approximated
- Policy approximated

Classical dynamic programming

- Bellman equations + greedy action.
- Policy evaluation, policy improvement, value iteration.
- Non-tractable for large state and action spaces.

Modern RL (deep RL)

- One sample does not return the true expected value (noisy reward).
- The same action does not always lead to the same next state.
- We don't know the true state (only observed).

Challenge 2: partial observability

Ideal setting

State fully observable

- MDP (finite, discrete)
- Model known
- Value function exact
- Optimal policy computable

vs

Real world

State partially observable

- POMDP (infinite, continuous)
- Model unknown or learned
- Value function approximated
- Policy approximated

Partial observability will always be a challenge in particle accelerator deployment, but can be mitigated with:

- Frequent and informative observations
- Memory (e.g., recurrent architectures) or a learned model
- Well-structured state representation
- Low-frequency decision making

Challenge 3: safety

Exploration vs exploitation dilemma:

We want to learn the **optimal behaviour** and for that we need to behave non-optimally to **explore** the state-action space.

→ ***Hard safety*** cannot be ensured in high-dimensional continuous state spaces!

Hard safety in RL, especially during exploration, is an active area of research

Soft safety can be implemented:

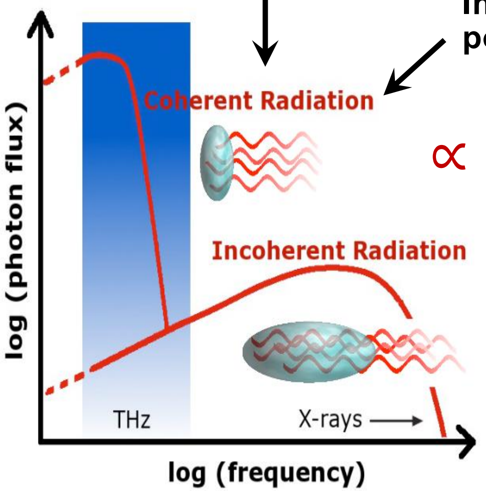
- Shielding
- Reward shaping
- Uncertainty-aware planning

Trade-offs between safety, optimality, and sample efficiency.

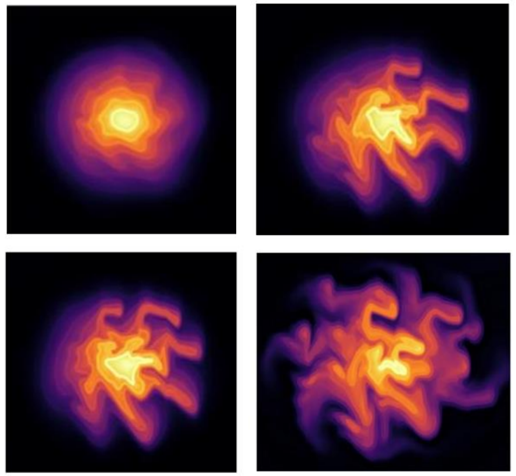
My recommendation: do experiment-based training only in safe machines (low energy, electrons) or have an excellent interlock system.

Challenge 4: real-time inference

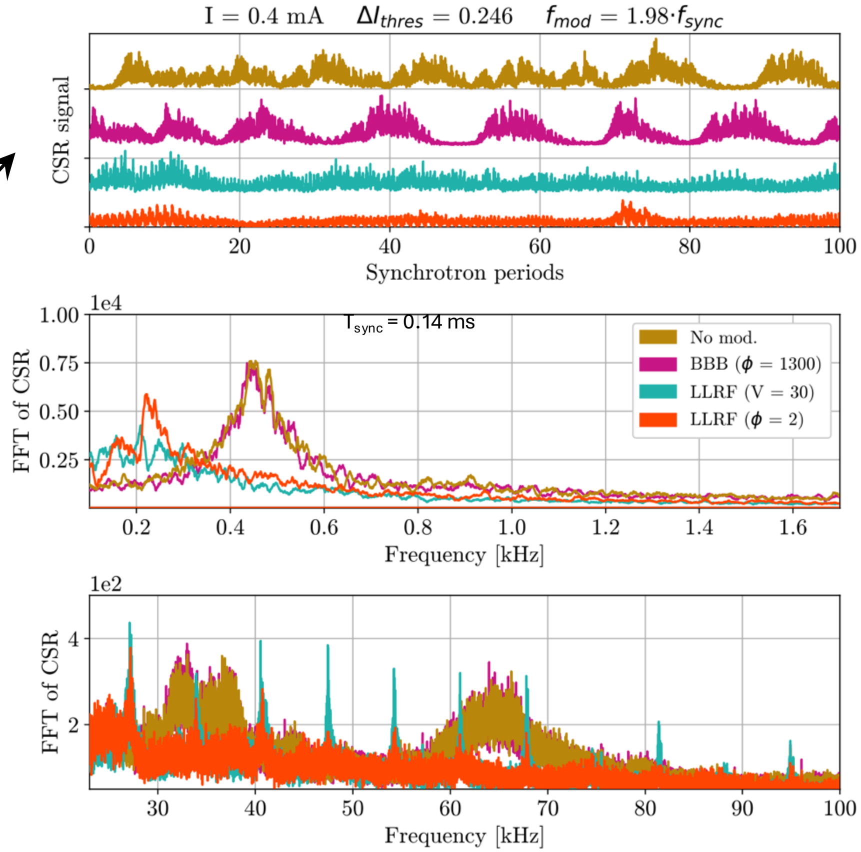
Control of the microbunching instability



Low-α_c optics → MBI



Bursting can be controlled with RF modulations

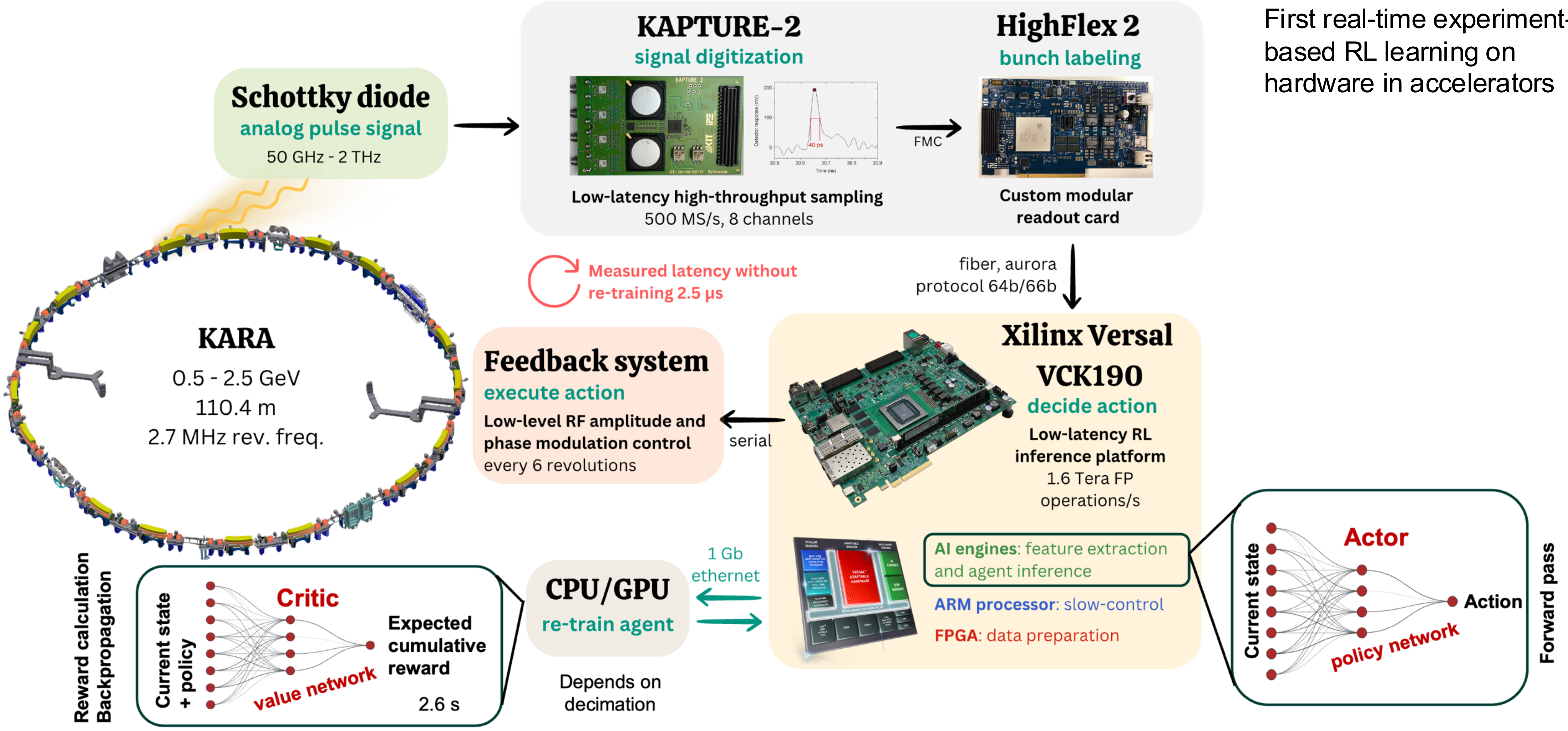


Challenge 4: real-time inference

[Doctoral thesis L. Scomparin]

Control of the microbunching instability

First real-time experiment-based RL learning on hardware in accelerators



Challenge 4: real-time inference

[Doctoral thesis L. Scomparin]

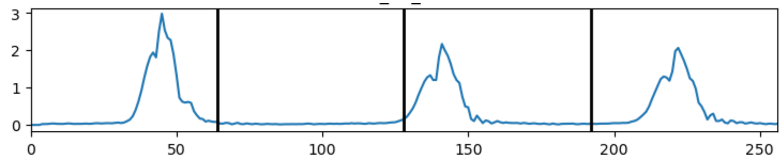
Control of the microbunching instability

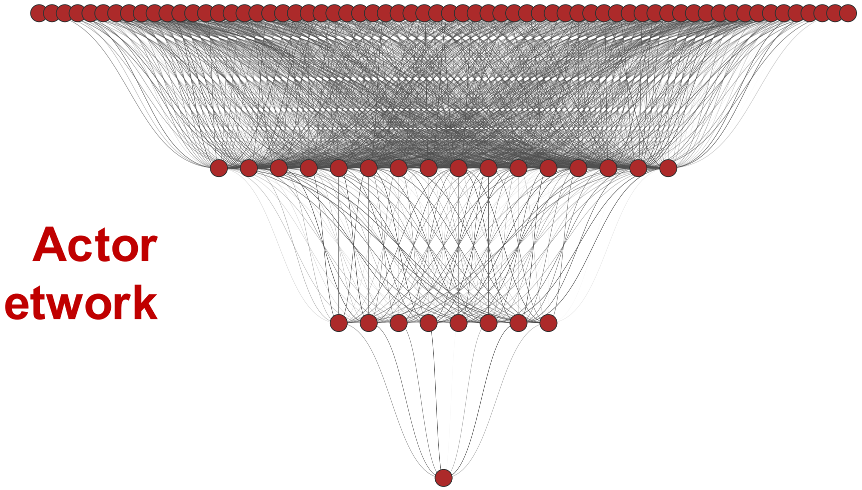
Algorithm: vanilla PPO from Stable Baselines3

Observation

Last 64 THz signal samples (decimated)

Circular buffer (keeps last 64 samples in memory)





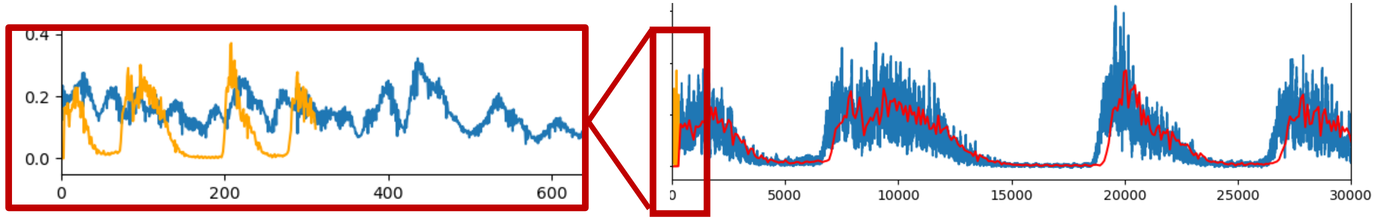
Actor network

Action (voltage value)

Interpolated (smoothed) to avoid sharp steps (cavity interlocks)

Decimation (two stages):

- Controls timescale agent "sees"
  - Rate of action = decimation x T_{rev}
- Makes infererence & training easier (smaller networks)
- We decimate 16 x 6 = 96 (take a sample every 96 revolutions)
  - ➢ We act every 96 x T_{rev} ~ **28 kHz** ~ 0.25 x T_{sync} ~ 36 μs
  - ➢ We use 440 samples per second



Filtering to remove spurious content + decimation

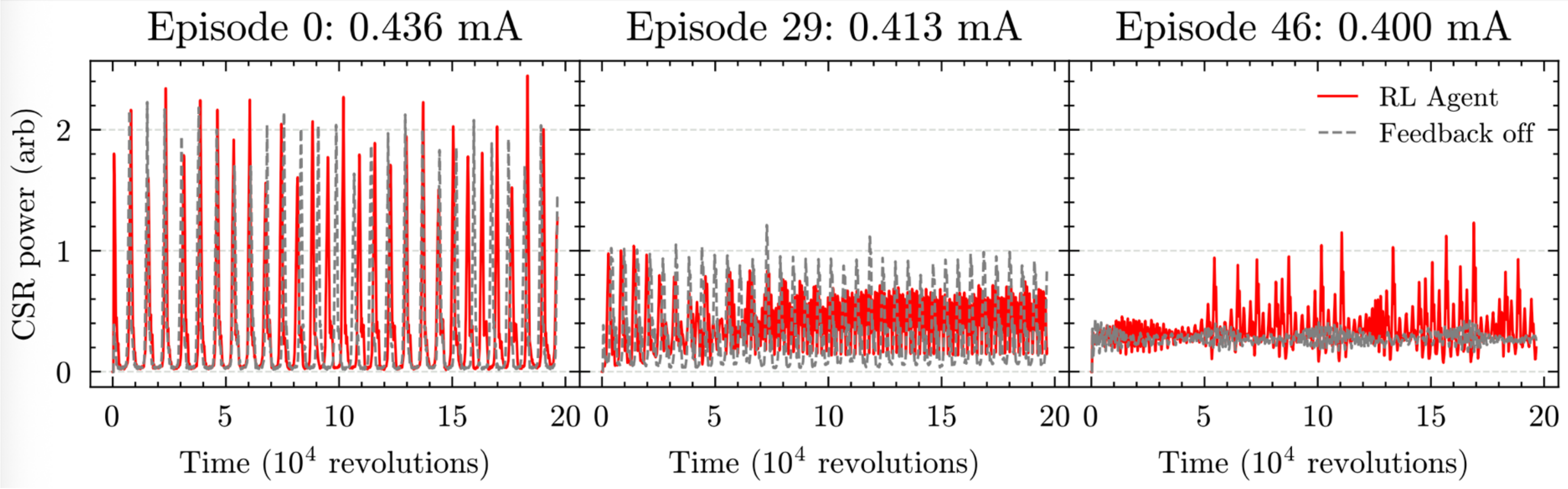
Strategy:

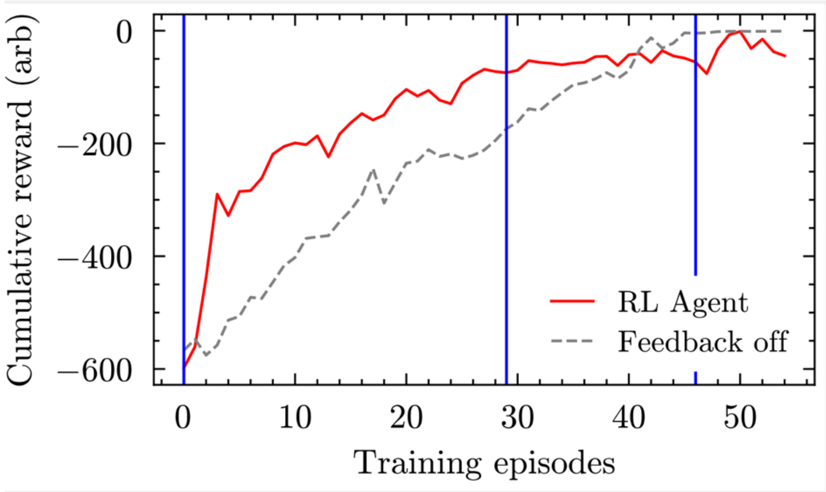
1. Agent acts during 2048 steps (samples of decimated signal)
2. Agent stops and is re-trained in a CPU (takes ~2.6 s)
   - ➢ We train every (2048 x 96) T_{rev} = 509 T_{sync}
3. New weights are sent to Versal board and agent starts again

Challenge 4: real-time inference

Doctoral thesis L. Scomparin

Control of the microbunching instability





Some improvements but conditions (charge and instability mode) change too rapidly for the agent to properly adapt → needs memory

Challenge 4: real-time inference

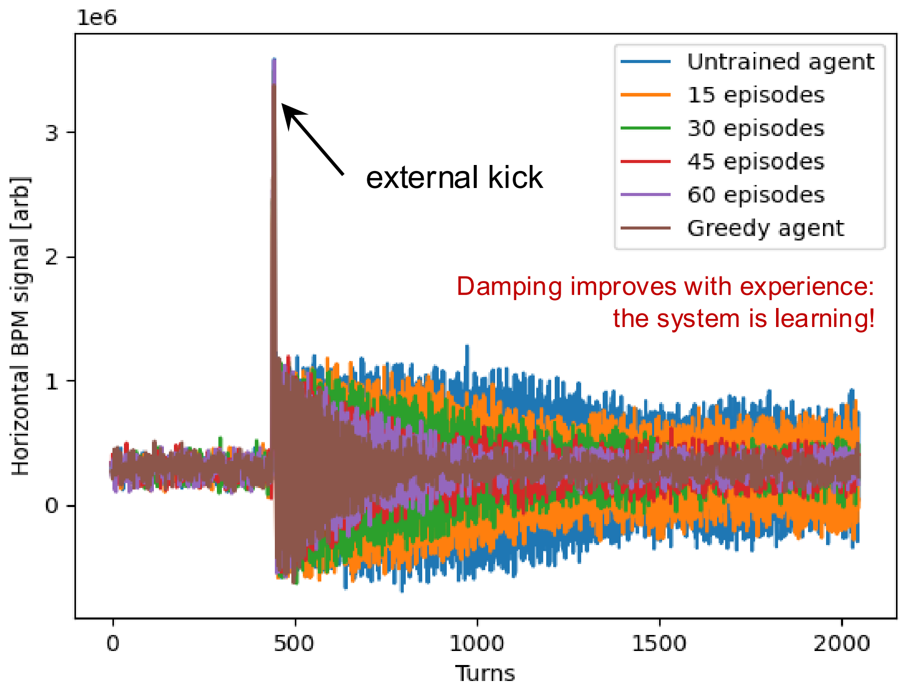
Doctoral thesis L. Scomparin

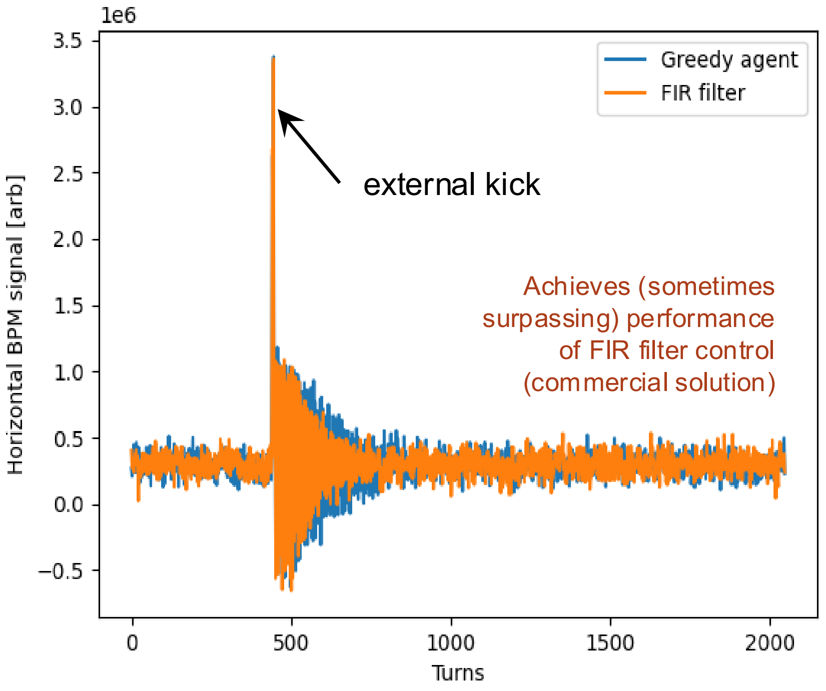
Same concept applied to a simpler problem

Algorithm: Vanilla PPO from Stable Baselines3
Actor & critic architecture: 8-16-1
Reward: metric of the beam position (low as possible)
Observation: last 8 BPM samples

Strategy:

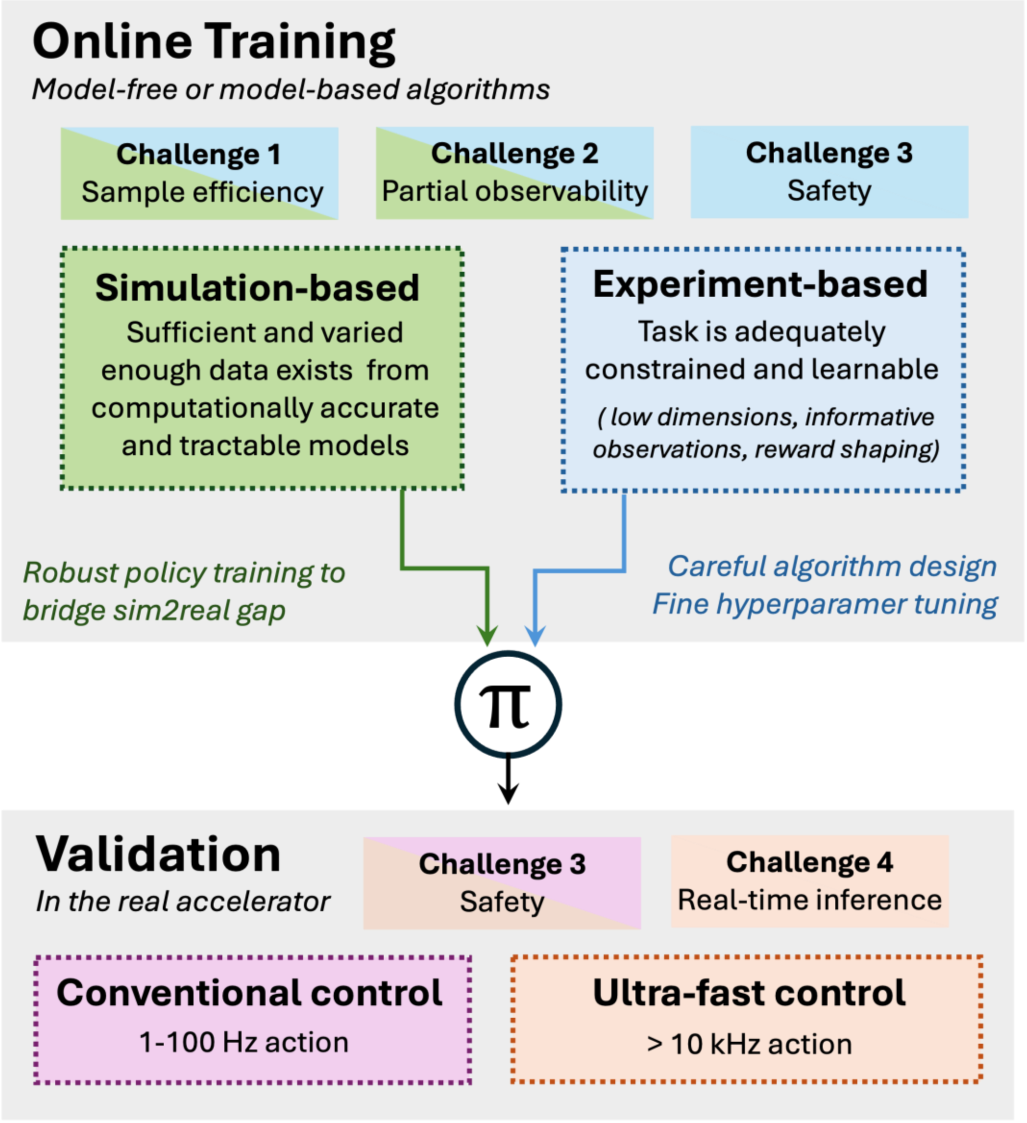
1. Agent acts during 2048 turns (0.74 ms)
2. Agent stops and is re-trained in a CPU (~2.6 s)
3. New weights are sent to Versal board and agent starts again





Main challenges of RL deployment

In particle accelerators



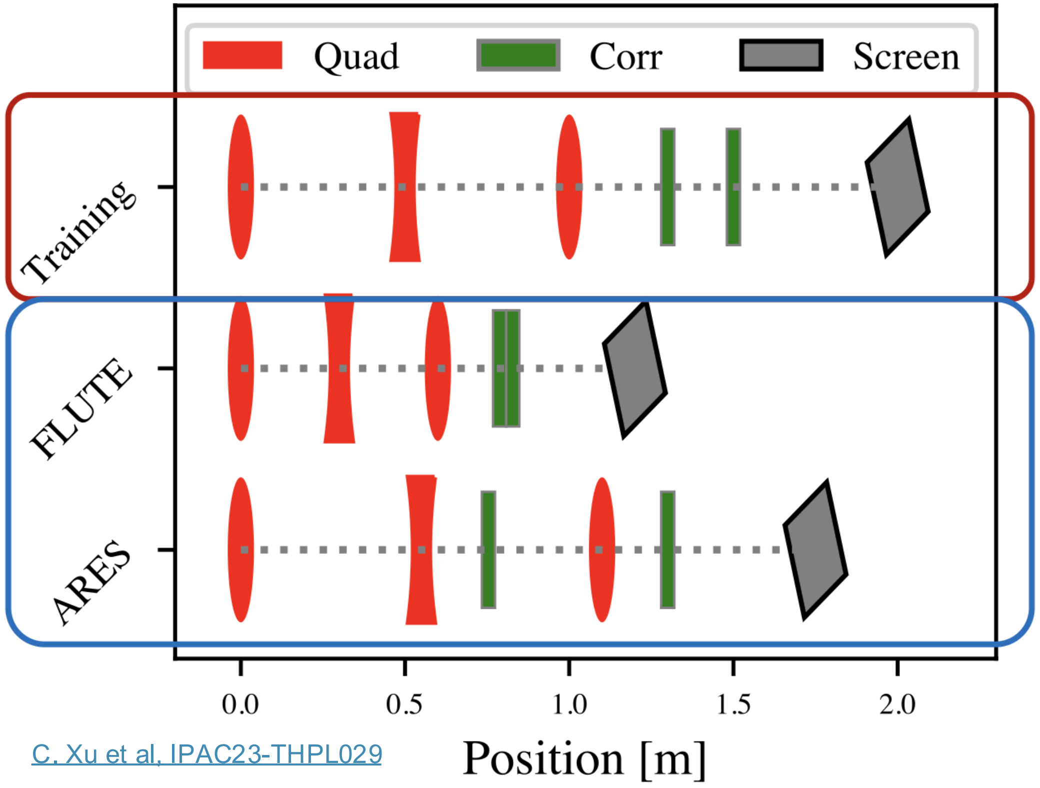
RL is a promising and powerful framework for adaptive, goal-directed behaviour in complex environments…

…that requires careful design!

Can it be an intelligent accelerator co-pilot? 🤖

Future directions

Lattice-agnostic RL



C. Xu et al, IPAC23-THPL029

Used during training, with randomized positions but following order (=DR)

Test lattices
Re-training with new lattice only 2% of the original training samples

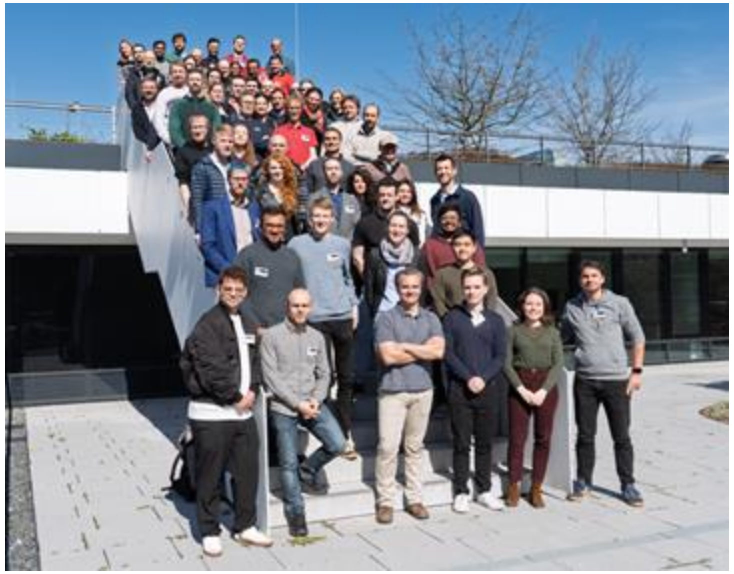
Multi agent RL, hierarchical RL, explainable RL, more model-based RL

The Reinforcement Learning for Autonomous Accelerators Collaboration

RL4AA

Join our Discord

Yearly targeted workshops



RL4AA'25 at DESY

Github: https://github.com/RL4AA

Discord: https://discord.gg/rudtJaeW

Website: https://rl4aa.github.io/

Youtube: https://www.youtube.com/@RL4AACollaboration

Paper: DOI:10.18429/JACoW-IPAC2024-TUPS62

Annika Eichler, Christian Contreras, Christian Hespe, Simon Hirlaender, Jan Kaiser, Sabrina Pochaba, Borja Rodriguez Mateos, Andrea Santamaria Garcia, Chenran Xu

Next year RL4AA'26 workshop:

University of Liverpool (25th – 27th April 2026)

Cheetah: differentiable beam dynamics code

Python package for beam dynamics simulations based on PyTorch

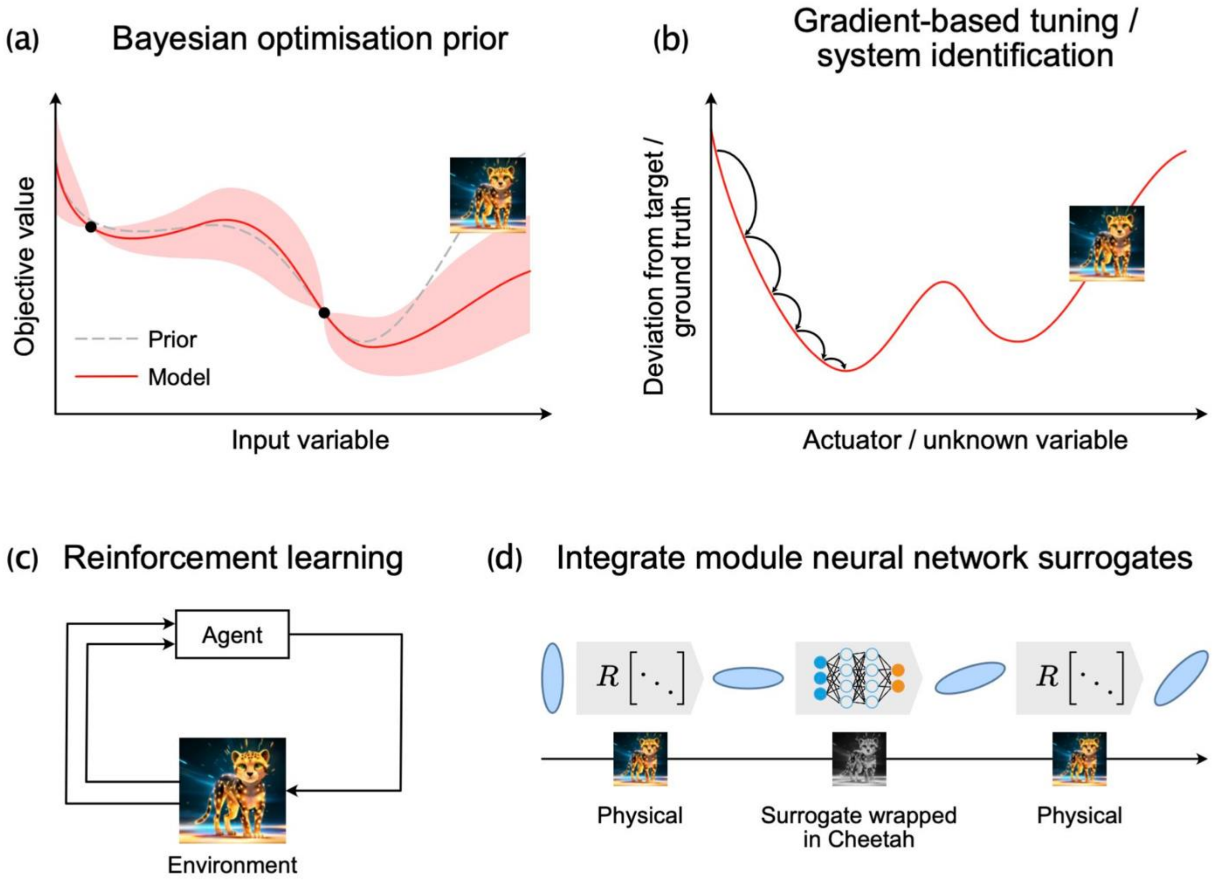
```
pip install cheetah-accelerator
```

Ultra-fast compute: Cheetah can run order of magnitude faster than some other codes.

Differentiability: supports automatic differentiation for all its computations.

Full **GPU support** and integrates seamlessly with ML models built in PyTorch.

https://github.com/desy-ml/cheetah



J. Kaiser et al. *Bridging the gap between machine learning and particle accelerator physics with high-speed, differentiable simulations.* 10.1103/PhysRevAccelBeams.27.054601

The Cheetah collaboration

DESY.


Jan Kaiser


Christian Hespe


Annika Eichler

Argonne NATIONAL LABORATORY


Chenran Xu

UNIVERSITY OF LIVERPOOL

The Cockcroft Institute


A. Santamaría García

BERKELEY LAB


Grégoire Charleux


Axel Huebl


Remi Lehe

THE UNIVERSITY OF CHICAGO


J. P. Gonzalez-Aguilera

SLAC


Ryan Roussel


Daniel Ratner


Auralee Edelen


Jenny Morgan


Zihan Zhu

Thank you for your attention!

What questions do you have for me?

Some of our research:

- https://doi.org/10.1038/s41598-024-66263-y
- https://doi.org/10.1103/PhysRevAccelBeams.27.054601
- https://arxiv.org/abs/2409.16177
- https://doi.org/10.1007/978-3-031-65993-5_21
- https://meow.elettra.eu/81/doi/jacow-ipac25-thyd1/index.html

Intro to RL talk: https://doi.org/10.5281/zenodo.12649046

Dr. Andrea Santamaria Garcia
Lecturer at University of Liverpool
Cockcroft Institute

ansantam@liverpol.ac.uk

https://www.linkedin.com/in/ansantam/

https://github.com/ansantam

https://instagram.com/ansantam