Towards artificial intelligence in accelerator operation

Simon Hirlaender, Niky Bruchon, Verena Kain, Alexander Scheinker,

B. Goddard, G. Valentino, F. Velotti and the ML Coffee

Overview

⇨ What is reinforcement learning (RL)?

⇨ Motivation and History of RL at CERN

⇨ Introduction to RL at accelerator operation

- ⇨ How does it work?
- ⇨ Overview of the methods and concepts - Model free reinforcement learning (MFLR)
- ⇨ Tailored method: NAF

⇨ Applications - trajectory steering:

- ⇨ AWAKE studies
- ⇨ Linac4 studies

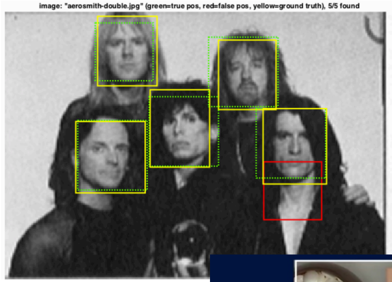
⇨ Applications - model based reinforcement learning (MBRL):

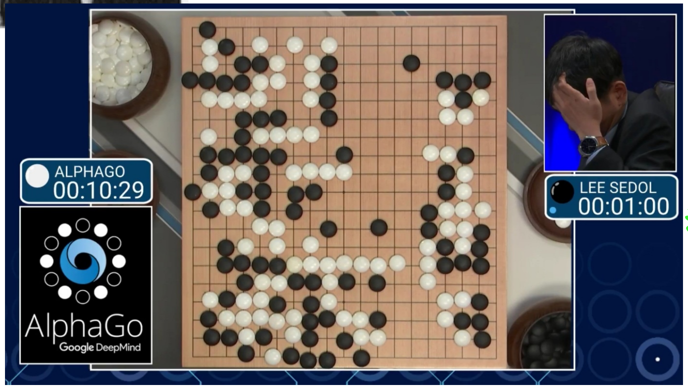
- ⇨ Uncertainties in deep models - probabilistic deep learning
- ⇨ The AE-Dyna idea
- ⇨ Applications at FERMI

⇨ Conclusions and discussion

⇨ More AWAKE studies*

Progress of artificial intelligence





Recognition /Prediction

Face++ 旷视
商汤 sensetime
DEEPGLINT 格灵深瞳

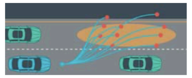
Decision making

Google DeepMind
UCL
OpenAI
covariant

Multi agent A.I. and distributed decision making

Bots Routine and Navigation

Distributed Optimisation



Multi-objectives optimisation

https://rlchina.org

PARIS LODRON UNIVERSITÄT SALZBURG · PARACELSUS MEDIZINISCHE PRIVATUNIVERSITÄT · salzburgresearch · FH Salzburg · LAND SALZBURG · CERN

AI = RL?

"Intelligence is the **computational** part of the ability to achieve **goals** in the world”

John McCarthy

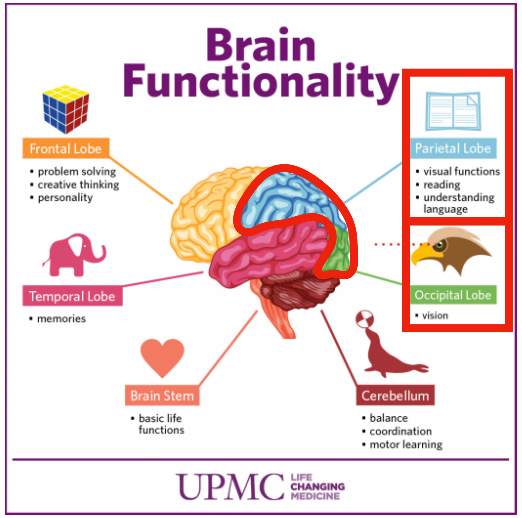
The **computer** is closely related to the computational part, but how to tell the computer our **goal**?

The **reward hypothesis:**

“All of what we mean by **goals** and purposes can be well thought of as maximisation of the expected value of the cumulative sum of a received scalar signal (**reward**).”

Decision making in machine learning

Supervised learning/modelling

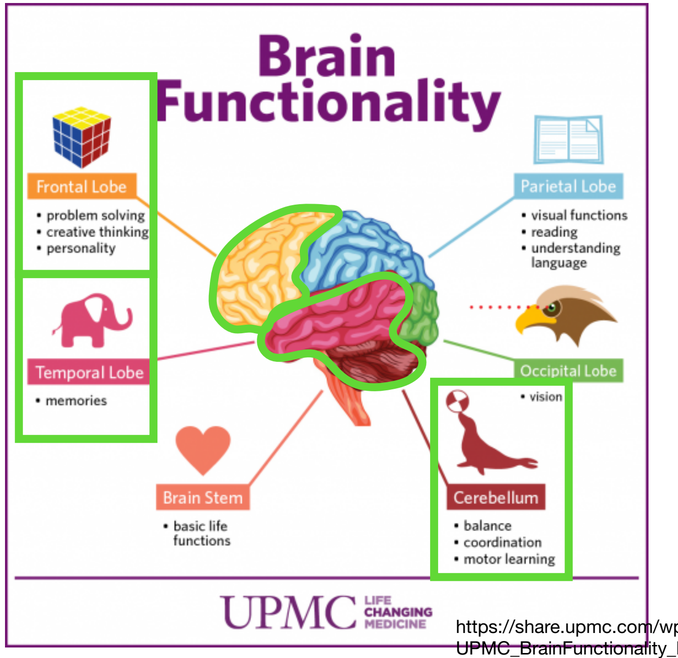


Handcrafted?

Prediction → Decision

Feedback?

Reinforcement learning



https://share.upmc.com/wp-content/uploads/2014/10/UPMC_BrainFunctionality_FINAL-524x524.png

Direct sequential decisions (only objective is provided - reward).

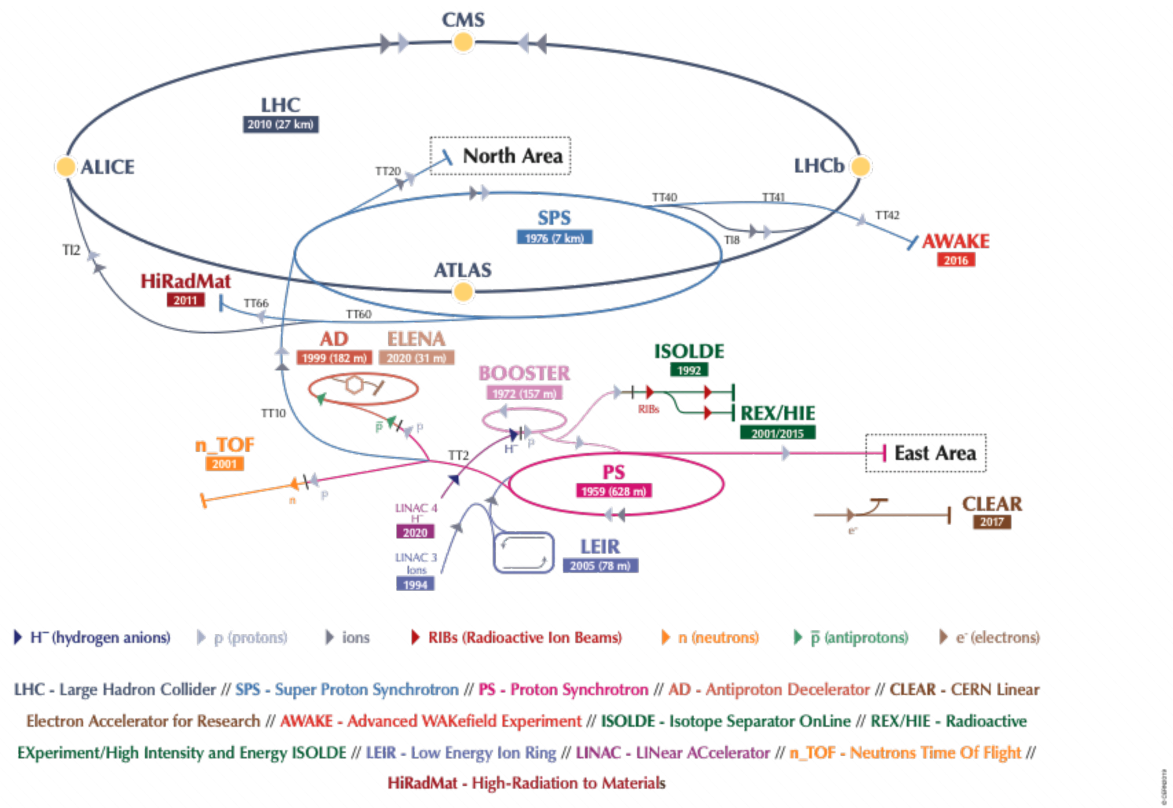
RL = optimising a sequential decision process

Motivation at CERN

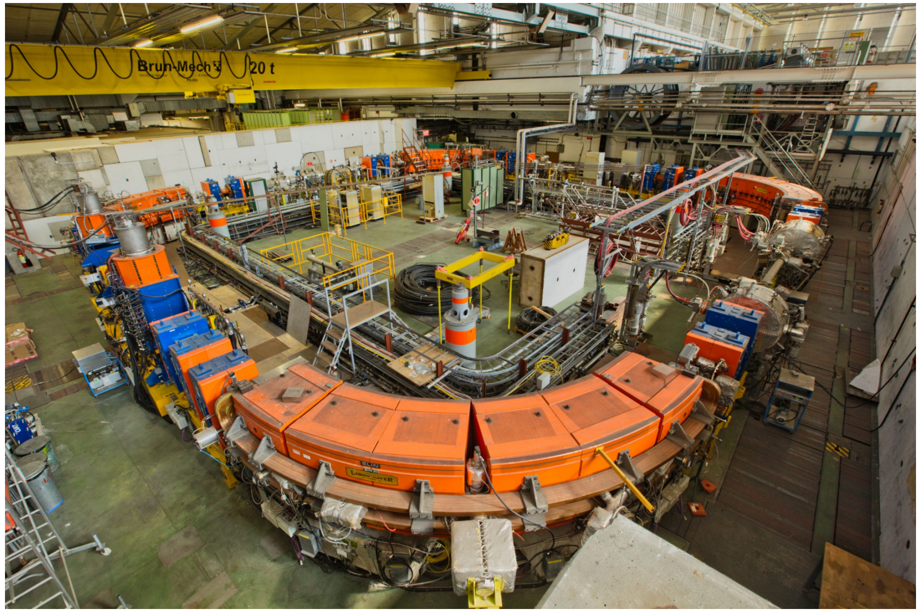
RL history @ CERN accelerators

Hard to predict/model e.g.:

- Space charge dominated beam dynamics in LINACs
- E-cooling setting-up
- Transmission optimization during transition crossing
- Alignment of electrostatic septa with many degrees of freedom



Example: Operating the Low Energy Ion Ring (LEIR)

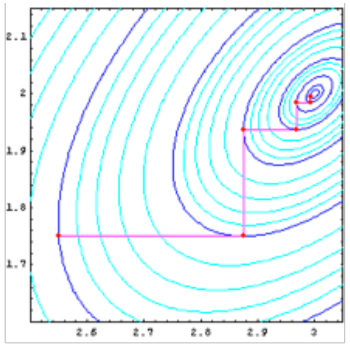


Complex system per design:

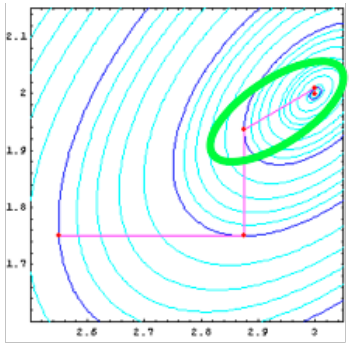
- **Several hours of manually maintaining/restoring the performance**
- **Introducing automatised optimisations**
- **Collaborations with other groups as Linac3…**

Partly cured with numerical optimisers

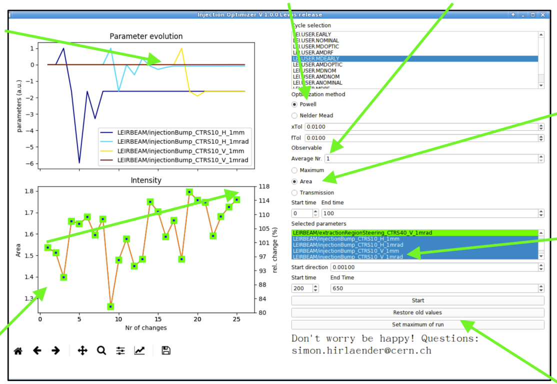
- Usage of classical DFO (derivative free optimiser) algorithms: Powell, Simplex, etc… (~from the 1960)
- Simple interfaces, scalable, robust…



Classical taxi-cab policy - search along fixed directions (human approach)



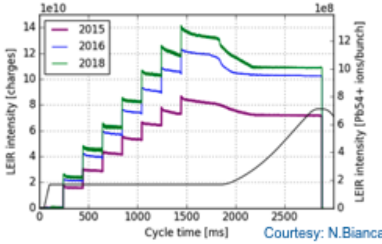
Powell's policy - take the direction of the average change



Highlights - Optimisation

Achievements - LEIR

- 2018: record injected intensity into LEIR (and LHC)
- Fast recovery after LEIR machine stops and drifts
- Reproducible performance

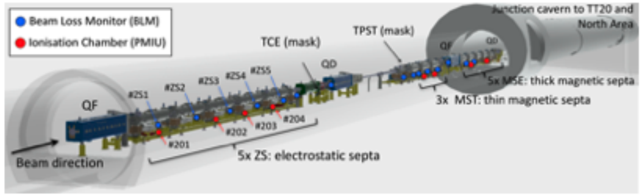


Courtesy: N.Biancacci

Result LHC 2018 for LEIR extracted intensity

75 ns	Mean $/10^{10}$c	Typical$/10^{10}$c	LIU$/10^{10}$c
LHC run	8.9	9.4	8.8

http://cds.cern.ch/record/2715365/



Example: automatic alignment of electro-static septum for slow extraction at the SPS

- 5 3.5 m long tanks with moveable anodes
  - 9 degrees of freedom to optimize; goal: minimize losses in extraction channel
  - Constrained to protect the hardware

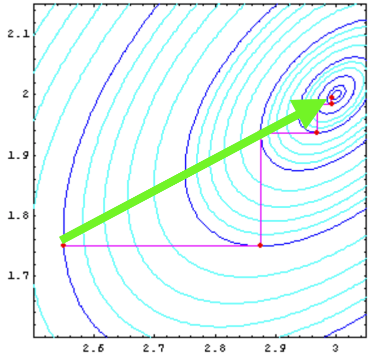
Reduced alignment time from ~ 8 h (quasi- manual scans) to ~ 45 minutes

Normalized losses

https://doi.org/10.18429/JACoW-IPAC2019-THPRB080

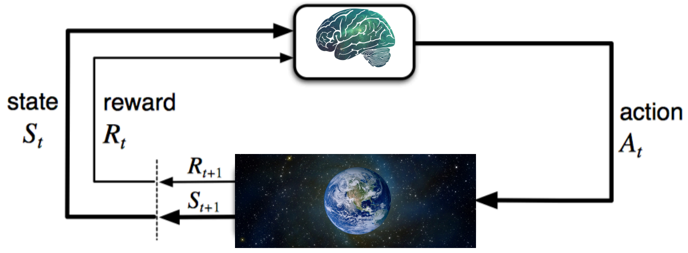
Natural next step: Reinforcement Learning!

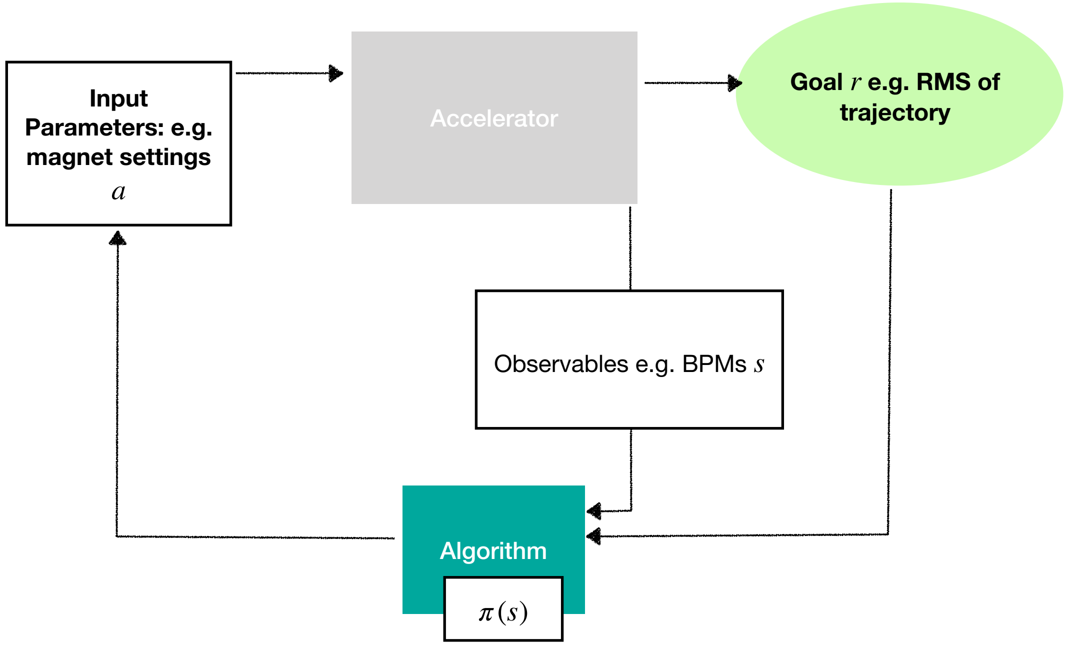
- Larger class of problems is covered.
- Optimization problems are not solved from scratch each time.
- Given data can be used.
- Possible insights into underlying (physical) structures of the problem.



Reinforcement learning

How does RL-work?





- Self-learning algorithm (agent) is learning from interactions with the environment:
  - Init: Read current state S_0 and taken an action A_0
  - Loop until finished:
    - Read S_t and R_t and set observe new state S_{t+1}.

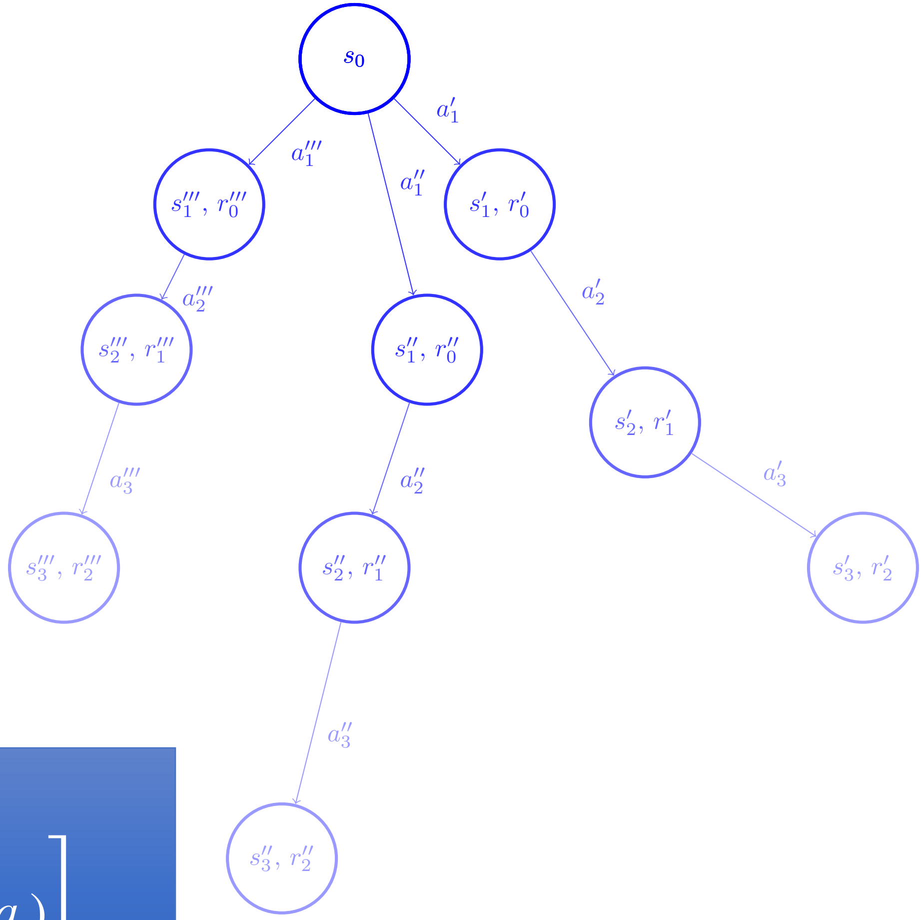
Finally we obtain the function $\pi(s) \mapsto a$, the policy.

Terminology:

- Markov decision process: $\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{T}, d_0, \gamma)$
- Dynamics of the system: $\mathscr{T}(s_{t+1} \mid s_t, a_t)$
- Trajectory is a sequence $\tau = (s_0, a_0, s_1, a_1, \ldots s_H, a_H)$
- The trajectory distribution using the policy $\pi(a_t \mid s_t)$ is:

$$p_\pi = d_0(s_0) \prod_{t_0}^{H} \pi(a_t, s_t) \mathscr{T}(s_{t+1} \mid s_t, a_t)$$

- Goal learn a function $\pi(a_t \mid s_t)$!



Expected return:

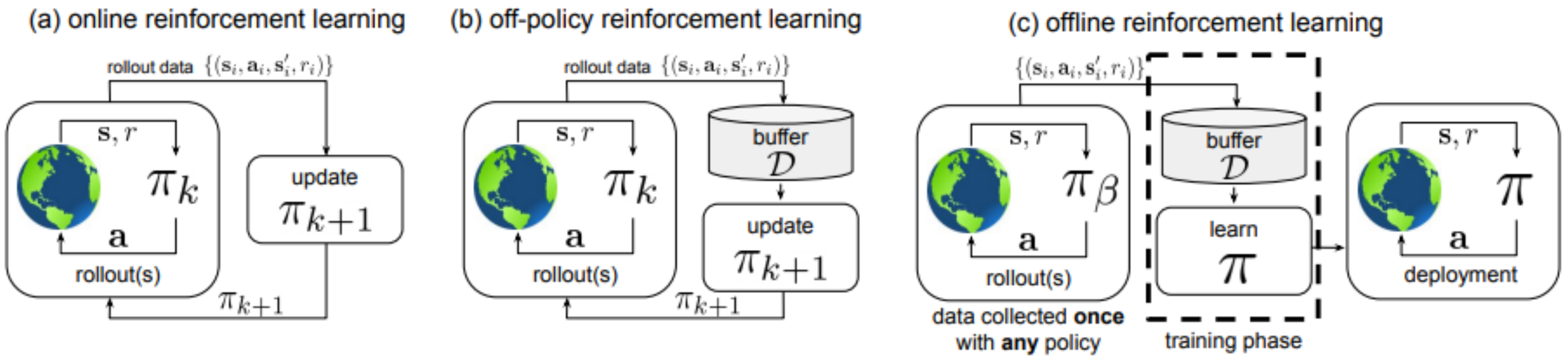
$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi} \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right]$$

Maximise return:

$$\max J(\pi) = \mathbb{E}_{\tau \sim p_\pi} \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right]$$

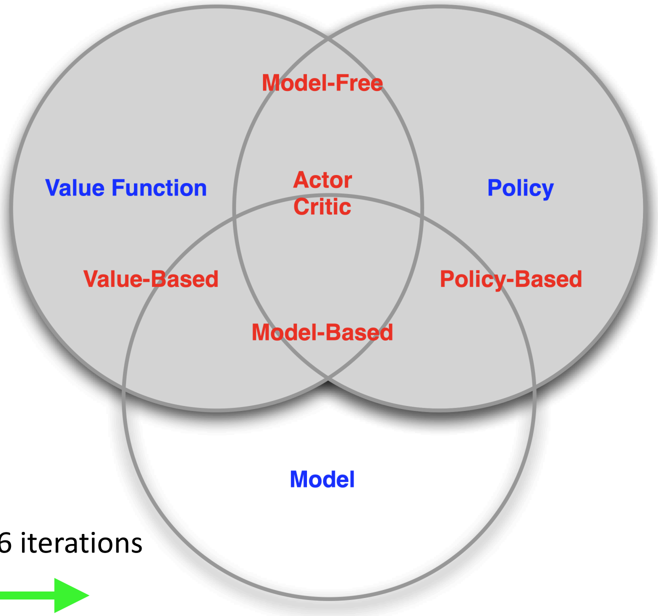
Taxonomy of RL

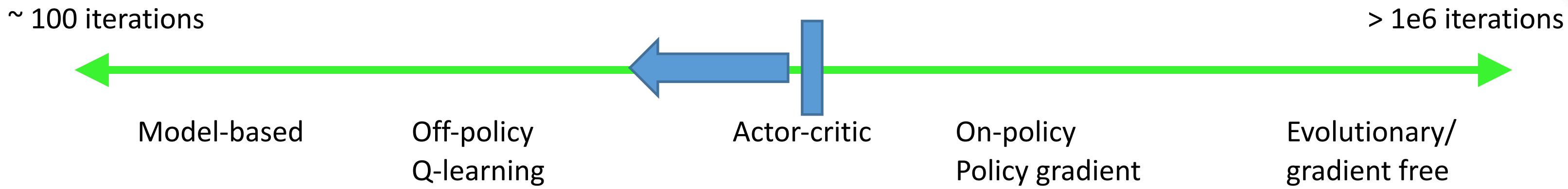
- Main schemes of training:



https://arxiv.org/pdf/2005.01643

- Main approaches of RL:

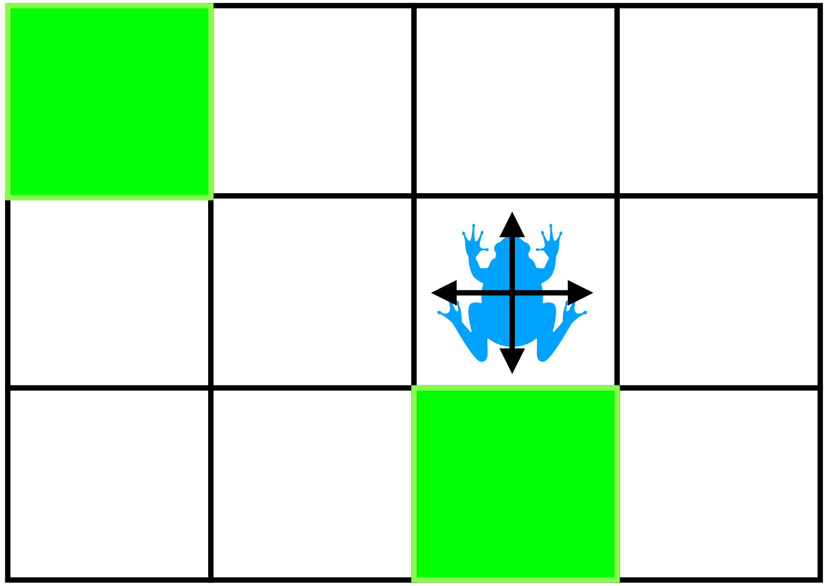




In accelerators each iteration is expensive!

Concept: State Value function

Frog wants to reach quickly the gras!

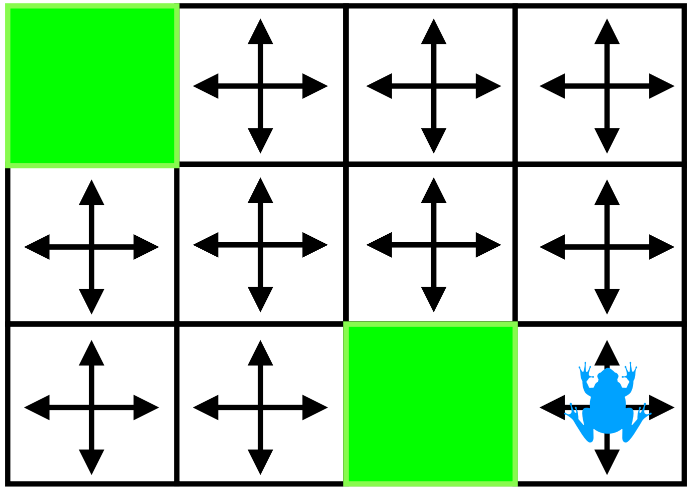


- Landing on white tiles gives a reward of -1.
- Landing on green tiles gives a reward of 0.

$$V^{\pi}(s_t) = \mathbb{E}_{\tau \sim p_{\pi}(\tau|s_t)} \left[\sum_{t=t'}^{H} \gamma^{'t-t} r(s_t, a_t) \right]$$

Concept: State Value function

$$\pi(s_t) = \text{random } a_t$$



0.0	-4.45	-5.88	-6.6
-4.29	-5.0	-4.75	-5.8
-5.33	-4.29	0.0	-4.19

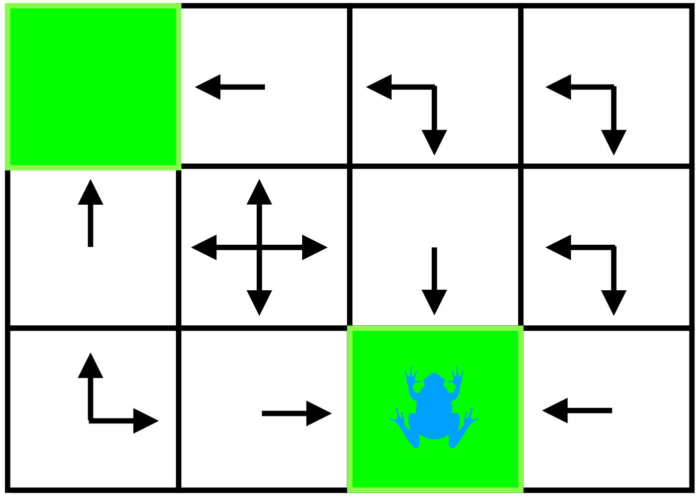
How good is a state s_t following a policy π?

$$V^{\pi}(s_t) = \mathbb{E}_{\tau \sim p_{\pi}(\tau|s_t)} \left[\sum_{t=t'}^{H} \gamma^{t'-t} r(s_t, a_t) \right]$$

Concept: State-Action Value function

How good is an action a_t in a state s_t following a policy π?

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\tau \sim p_{\pi}(\tau | s_t, a_t)} \left[\sum_{t='t}^{H} \gamma^{t'-t} r(s_t, a_t) \right] = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_{t+1} | s_t, a_t)} \left[V^{\pi}(s_{t+1}) \right]$$



0.0	-1.0	-1.9	-2.71
-1.0	-1.9	-1.0	-1.9
-1.9	-1.0	0.0	-1.0

$\pi(s_t) = \pi^*(s_t)$ optimal policy = $\underset{a_t}{\operatorname{argmax}} Q^*(s_t, a_t)$

Policy based

Value based

Dynamic programming

$$V^{\pi}(s_t) = \mathbb{E}_{\tau \sim p_\pi(\tau|s_t)}\left[\sum_{t='t}^{H} \gamma^{'t-t} r(s_t, a_t)\right]$$

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\tau \sim p_\pi(\tau|s_t, a_t)}\left[\sum_{t='t}^{H} \gamma^{'t-t} r(s_t, a_t)\right]$$

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t), a_{t+1} \sim \pi(a_{t+1}, s_{t+1})}\left[Q^{\pi}(s_{t+1}, a_{t+1})\right]$$

Bellman operator: $\overrightarrow{Q}^{\pi} = \mathscr{B}\overrightarrow{Q}^{\pi}$ has a unique fixed point!

Insert $\pi(a_t) = \delta(a_t - \underset{a_t}{\text{argmax}} Q(s_t, a_t))$:

$$Q^{*}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)}\left[\max Q^{*}(s_{t+1}, a_{t+1})\right],$$

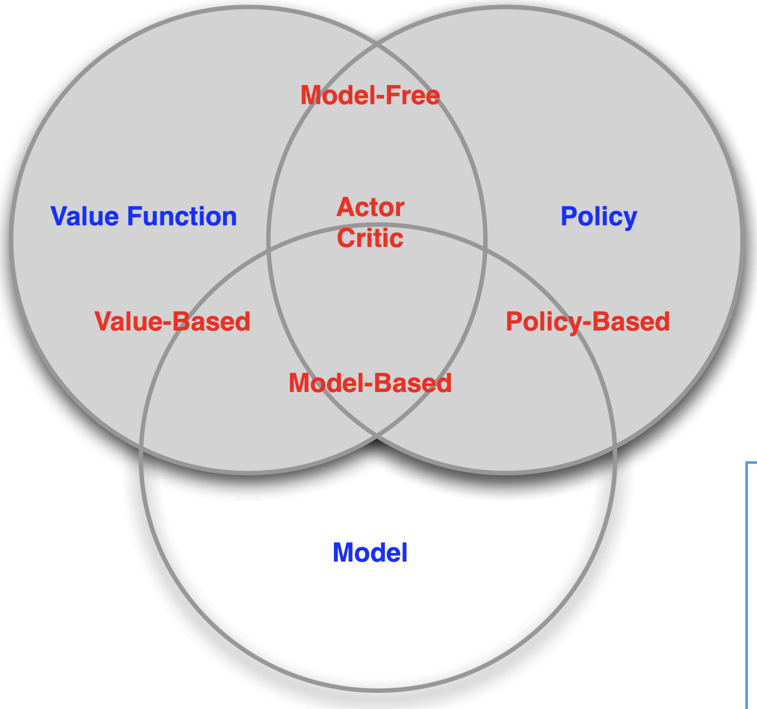
Bellman optimality equation:

$$\overrightarrow{Q} = \mathscr{B}^{*}\overrightarrow{Q}$$

Nonlinear (due to max operator).

Policy gradient and approximative dynamic programming

Functions are approximated with high capacity function approximators as ANNs.



$$\max J(\pi) = \mathbb{E}_{\tau \sim p_\pi}\left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t)\right]$$

Policy gradient methods - mainly parametrise the policy:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}}\left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t)\right]$$

Approximative dynamic programming methods - parametrise the value function Q-learning:

$$\min_\phi (\overrightarrow{Q}_\phi - \mathscr{B}^* \overrightarrow{Q}_\phi)^2$$

Actor-critic methods - parametrise both:

$$Q_\phi^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t), a_{t+1} \sim \pi_\theta(a_{t+1}, s_{t+1})}\left[\sum_{t='t}^{H} \gamma^{'t-t} r(s_{t+1}, a_{t+1})\right]$$

Model free algorithms

Policy gradients:

- <u>Stable performance</u>
- Some convergence guarantees
- Not sample efficient

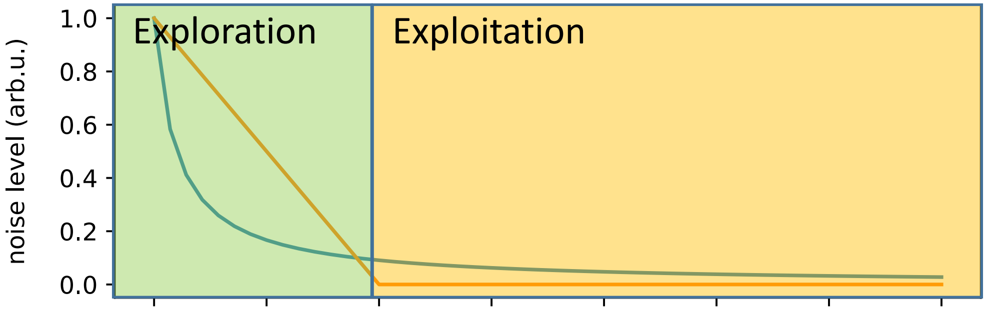
- Examples:
  - TRPO
  - PPO
  - PG

Q-learning/Actor-critic

- Generally unstable performance
- <u>Sample efficient</u> - reuse data in reply buffer

- Examples:
  - DDPG
  - TD3
  - NAF
  - SAC

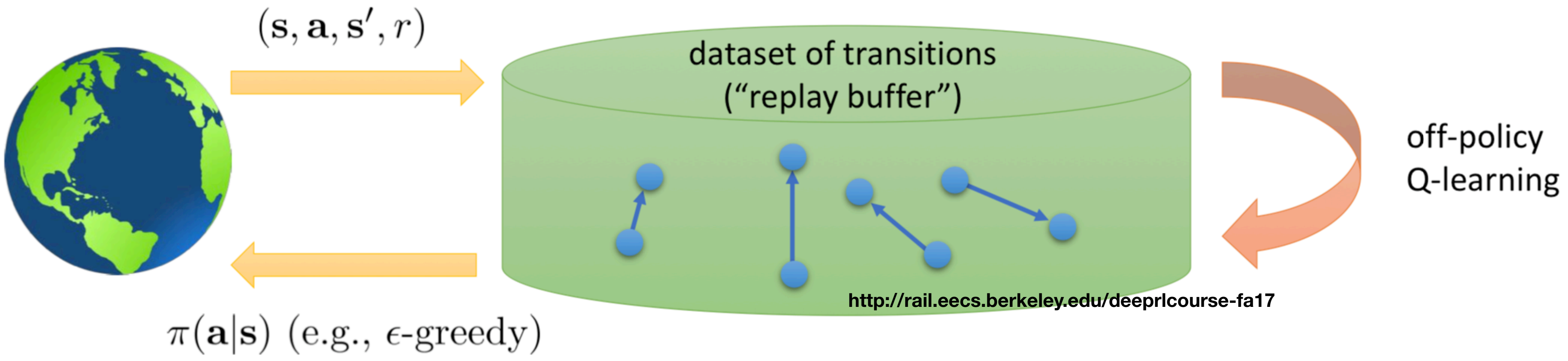
How is Q-learning done?



$$L(\phi) = (\overrightarrow{Q}_\phi - \mathscr{B}^* \overrightarrow{Q}_\phi)^2$$

Bootstrapping

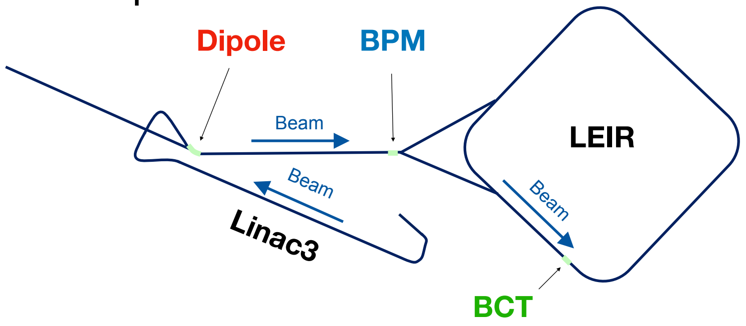
$$L(\phi_n) = \underbrace{\Big(\overbrace{R_{t+1} + \gamma \max_{a'} Q^\pi(S_{t+1}, a'; \phi_{n-1})}^{\textbf{Estimate including new information}} - Q^\pi(S, A; \phi_n)\Big)^2}_{\textbf{Bellman update}}$$



http://rail.eecs.berkeley.edu/deeprlcourse-fa17

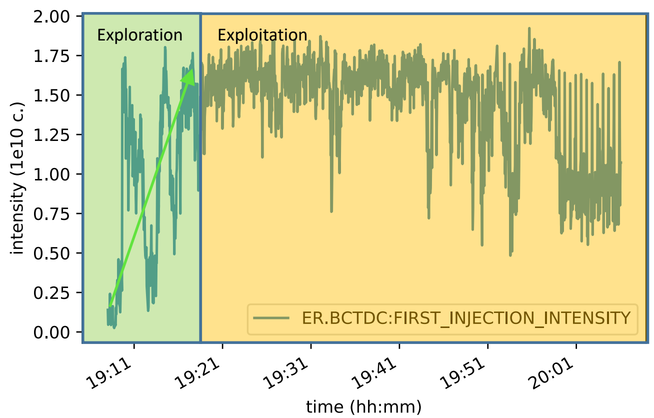
First Deep RL agent @CERN

The position of the elements



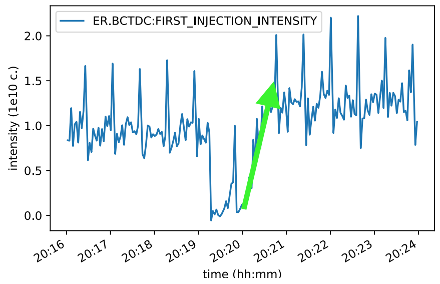
- The reward: intensity of BCT
- The state: position of the beam at BPM
- The action: change by $[+\Delta, -\Delta, 0]$ of dipole

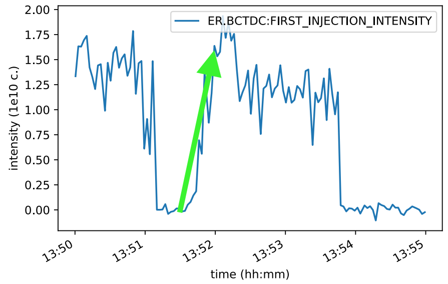
Training from scratch on LEIR approx. 600-iterations (measurement):



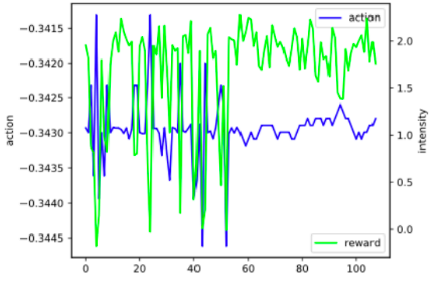
- The agent was able to learn the task without any initial knowledge and applied it successfully to different situations - reproducibility.
- **Discrete Q-learning approach (DQN).**
- Double (DQN) - around 100 interactions.
  - [Playing Atari with Deep Reinforcement Learning]

Apply the learned: real life test on different situations: recovered after a few moves although strongly perturbed

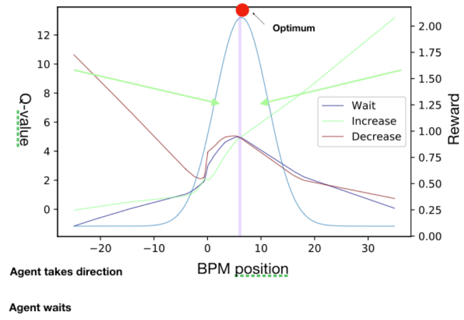




Training - run after tuning with Powell



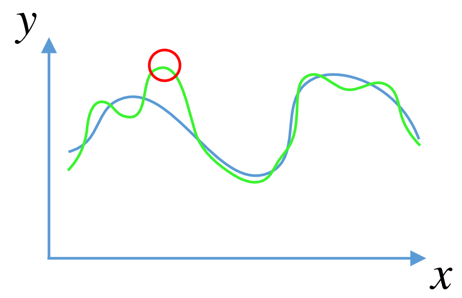
After training



Code: simon.hirlaender@sbg.ac.at

Q-learning issues:

- The max operator causes troubles in the Bellman update:



$$L(\phi) = (\vec{Q}_\phi - \mathscr{B}^*\vec{Q}_\phi)^2$$

$$Q^*(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)} \left[\max_a Q^*(s_{t+1}, a) \right]$$

$$\pi^*(s_t) = \underset{a}{\operatorname{argmax}} Q^*(s_t, a)$$

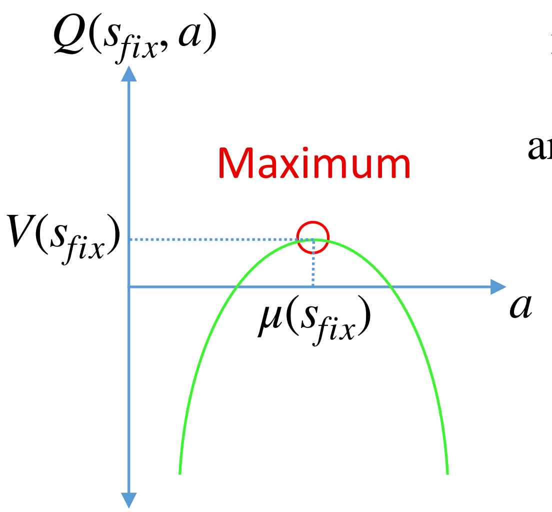
Simplify Q function - Why? Tailored solution

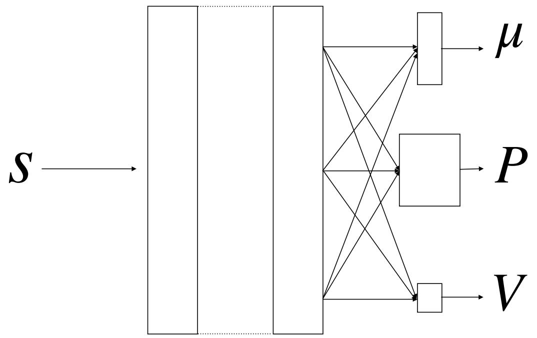
Normalised advantage function (NAF) Q learning algorithm

$$Q_\phi(s,a) = -\frac{1}{2}(a-\mu_\phi(s))^T P_\phi(s)(a-\mu_\phi(s)) + V_\phi(s)$$

$$\max_a Q_\phi(s,a) = V_\phi(s)$$

$$\operatorname*{argmax}_a Q_\phi(s,a) = \mu_\phi(s)$$





- Simple analytical form to (automatically) get best action.
- NAF is a “natural” extension of the DQN algorithm to continuous problems.
- NAF is an off-policy algorithm.
- <u>Highly sample efficient.</u>
- <u>Low of representational power.</u>

https://arxiv.org/abs/1603.00748

Prioritised experience replay - PER

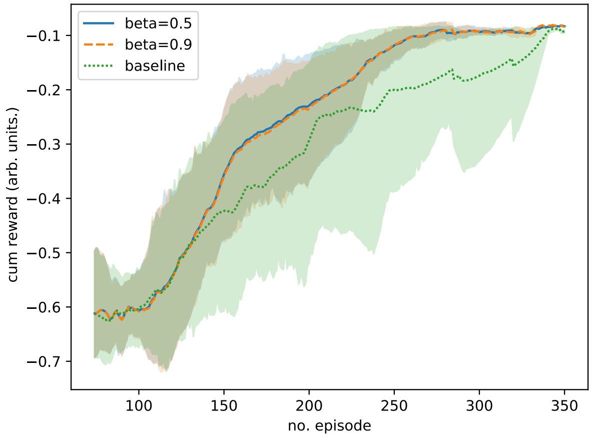
- **Weighting the tuples in the buffer by the Bellmann-error:**

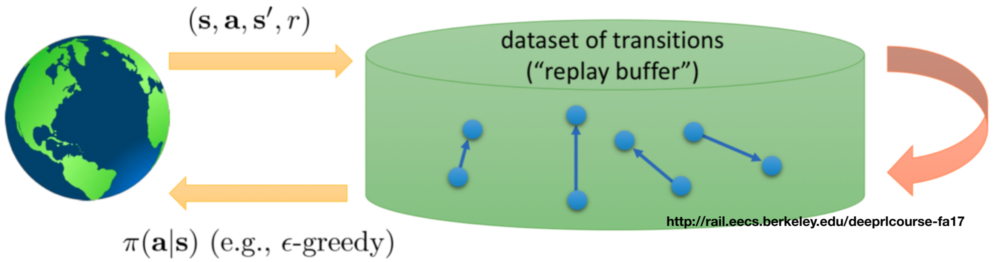
$$|\delta_i| = (\overrightarrow{Q}_\phi - \mathcal{B}^* \overrightarrow{Q}_\phi)^2$$

- $$p_i = |\delta_i| + \epsilon, P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

- pip install pernaf own implementation as used for all CERN the tests.

Stabler training





Code: simon.hirlaender@sbg.ac.at

Applications

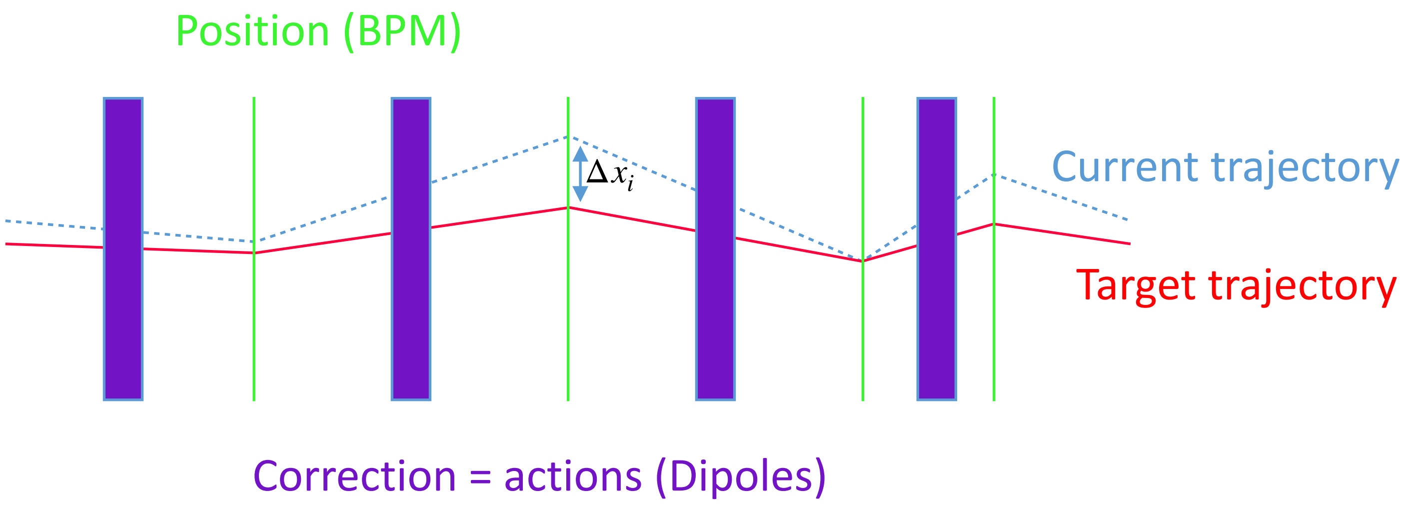
Trajectory steering

- ⇨ **Target: trajectory steering - correct the trajectory in as little steps as possible.**
- ⇨ **AWAKE - model given**
- ⇨ **Linac4 - no model given**

State = $\{\Delta x_1, \Delta x_2, \ldots \Delta x_n\}$

$\Delta x_i := x_{i\text{current}} - x_{i\text{target}}$

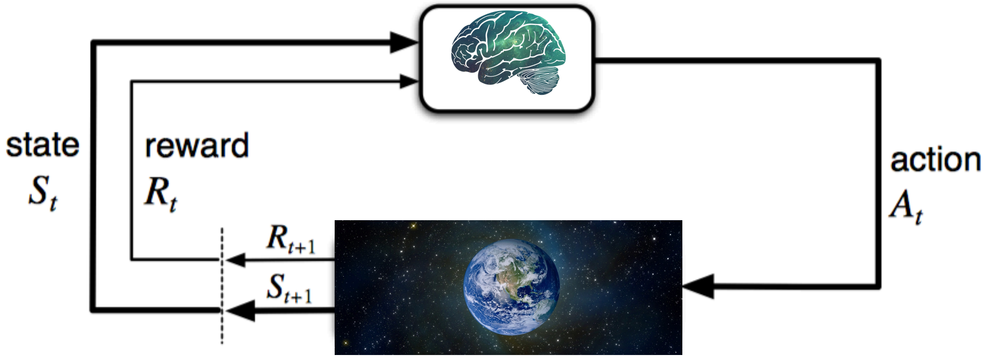
Actions = $\{k_0, k_1, k_2 \ldots, k_n\}$, limited k_{max}

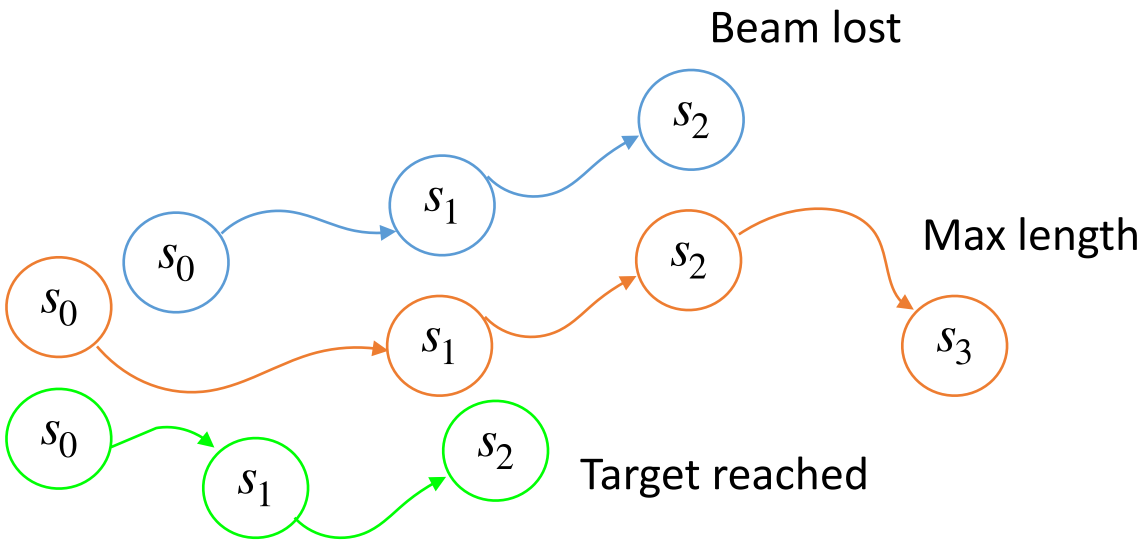


Reward $\propto -\sum_i^N \Delta x_i^2$

Episodic training

- **Loop until finished:**
  - **Randomly init the position s_0.**
  - **Take action a_t and measure new state s_{t+1}.**
  - **Define a target threshold r_{min} and stop if $r < r_{min}$ or step number reaches a maximum or beam is lost.**

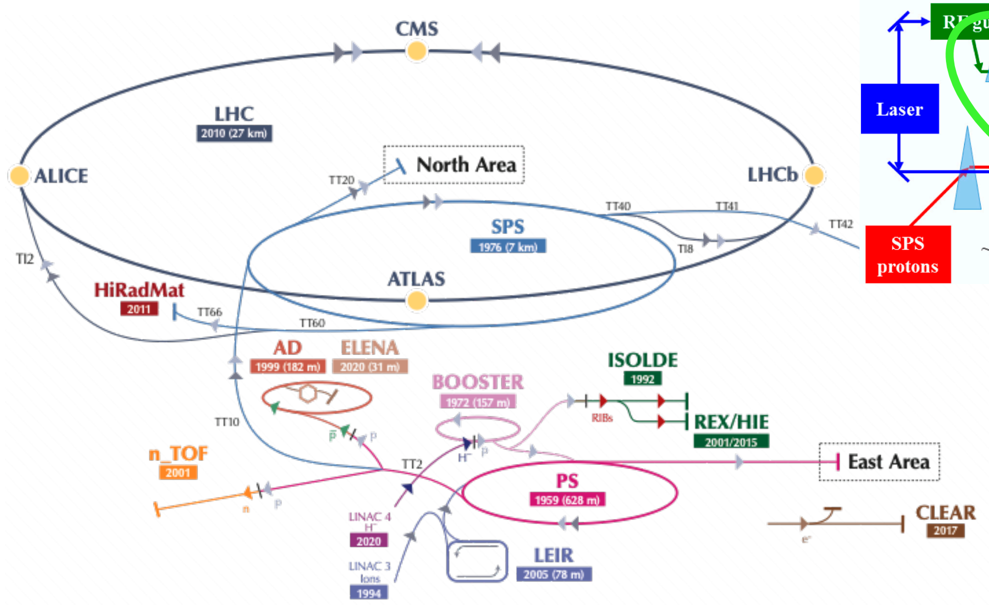


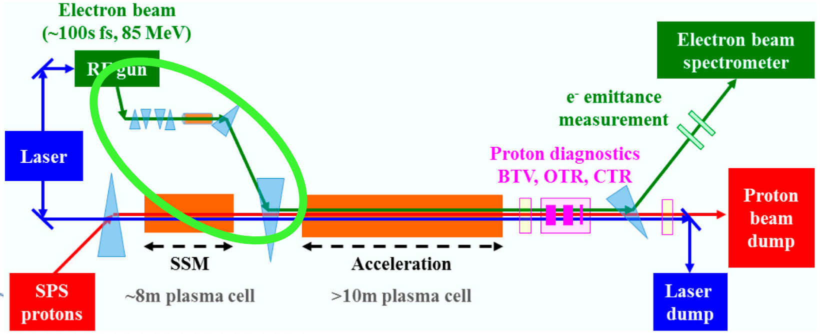


AWAKE

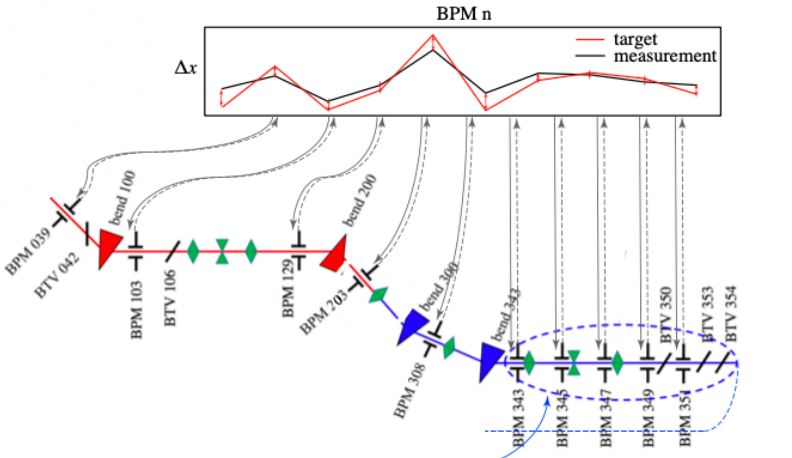
AWAKE Overview

Proof of principle proton driven plasma wake field accelerator





https://doi.org/10.1016/j.nima.2018.02.044

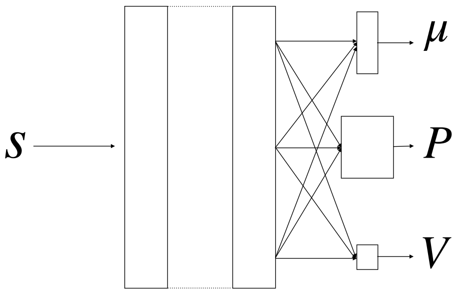


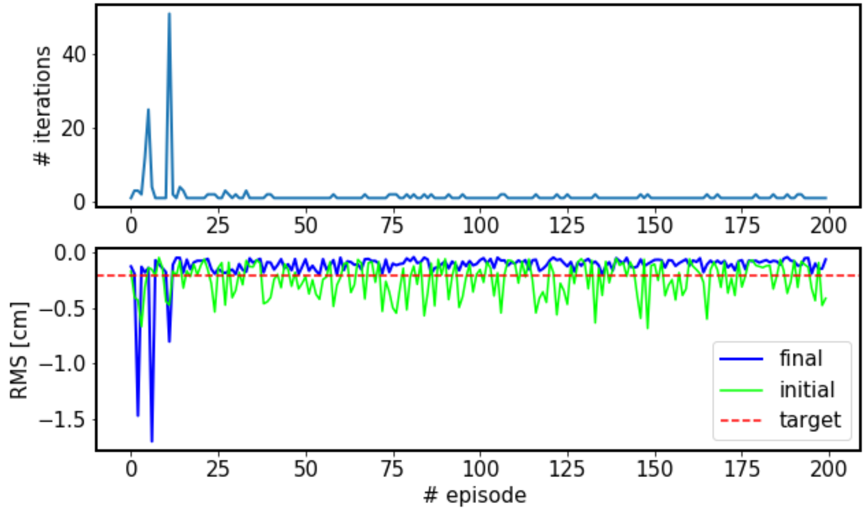
Courtesy Alexander Scheinker

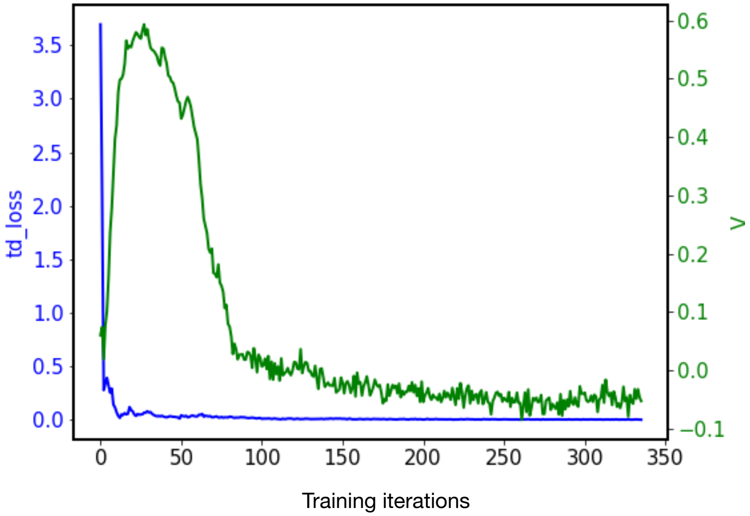
Training - machine only

- The AWAKE transfer line!
- 10 DOF - magnets a to steer electron beam, 10 BPM to measure the trajectory s, RMS to be minimised.
- Problem learned in ~ 200 iterations with threshold of 1 mm.

$$\max_a Q_\phi(s, a) = V_\phi(s)$$



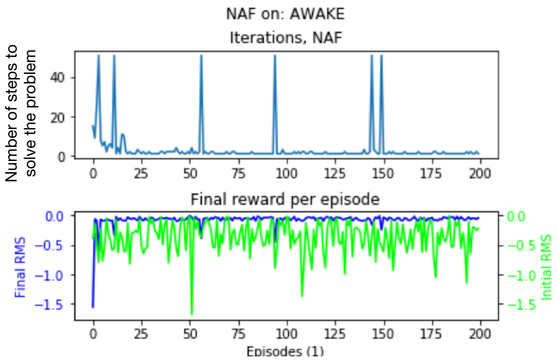


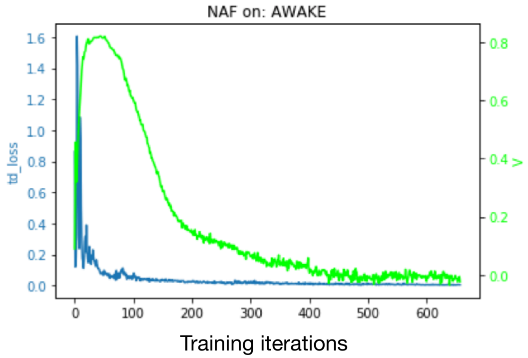


Model to machine

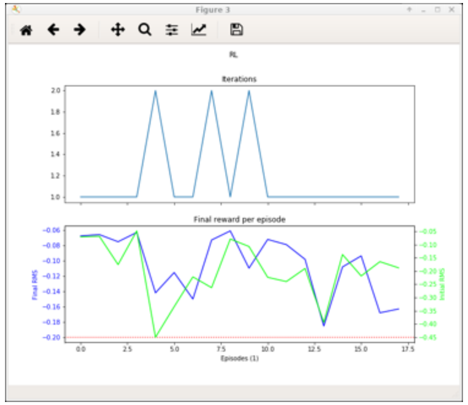
- The AWAKE transfer line - **we have a model**!

Training of the controller on the simulation





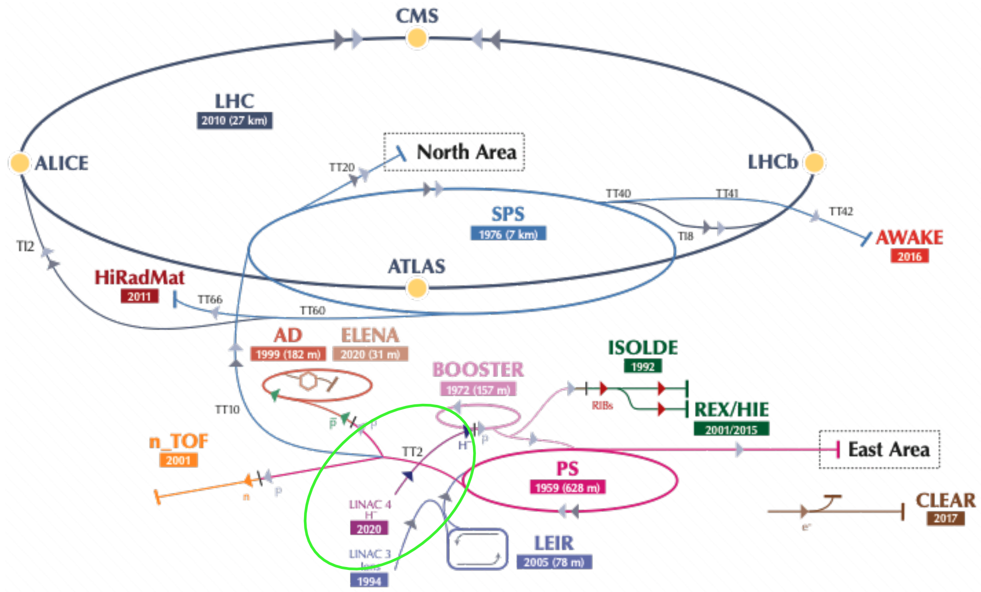
Applied on the machine.
Training a meta policy: Fine tuning could be done directly with a **small number of interactions!**

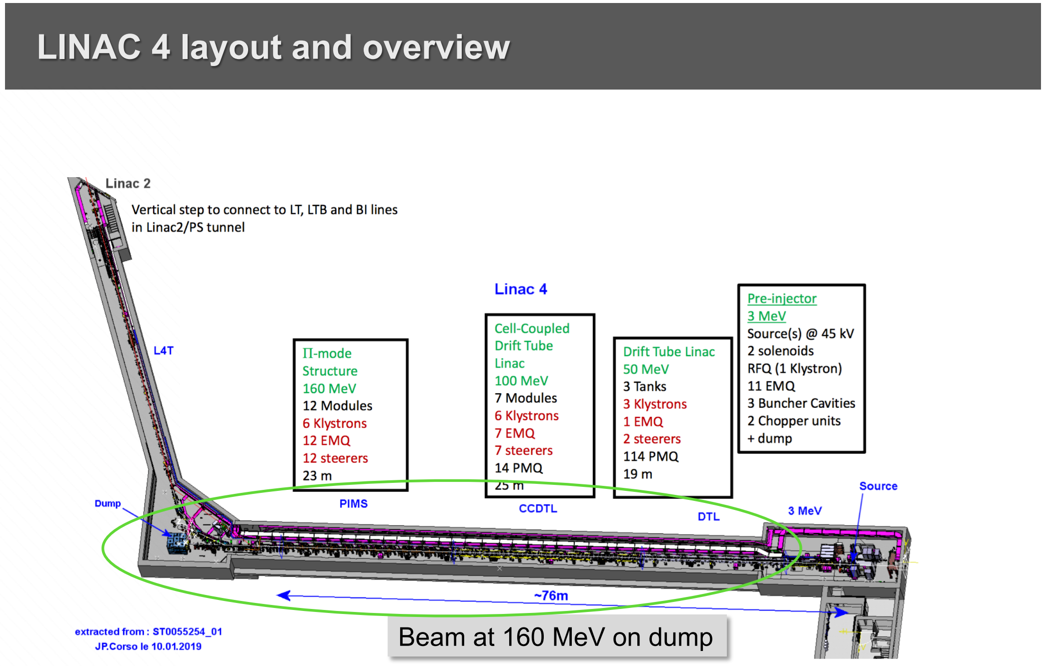


Linac 4

Linac4 Overview

- 16 DOF - magnets a to steer H^- ion beam, 16 BPM to measure the trajectory s, RMS to reference trajectory to be minimised.



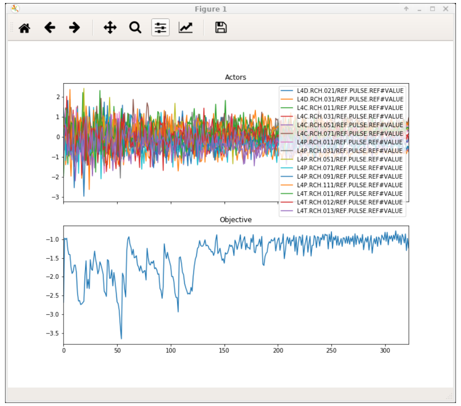


- Repetition rate 0.833 Hz (one shot/BP)

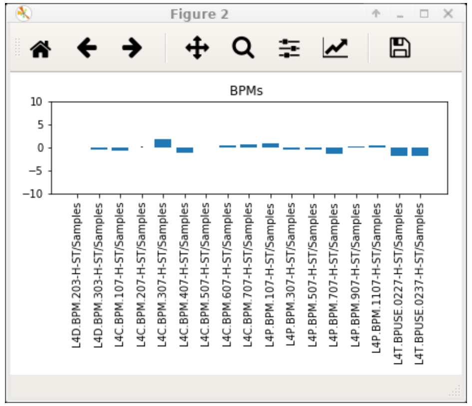
Linac 4 results

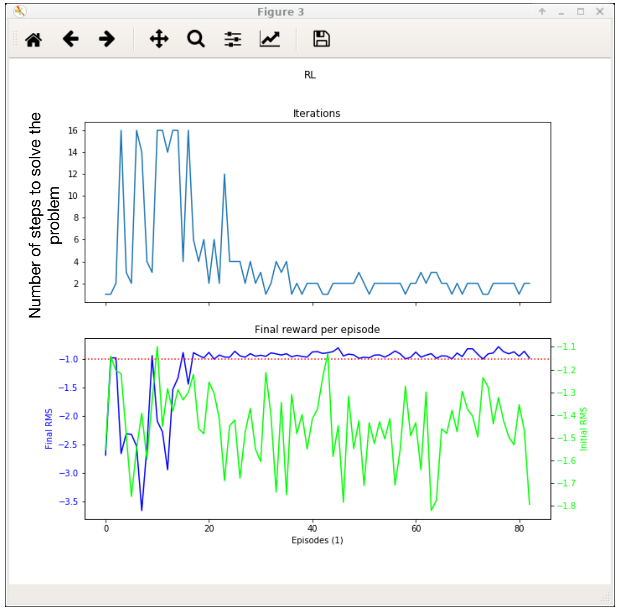
- The controller converges after ~250 steps.
- **First successful test of deep-reinforcement-learning @ Linac4.**

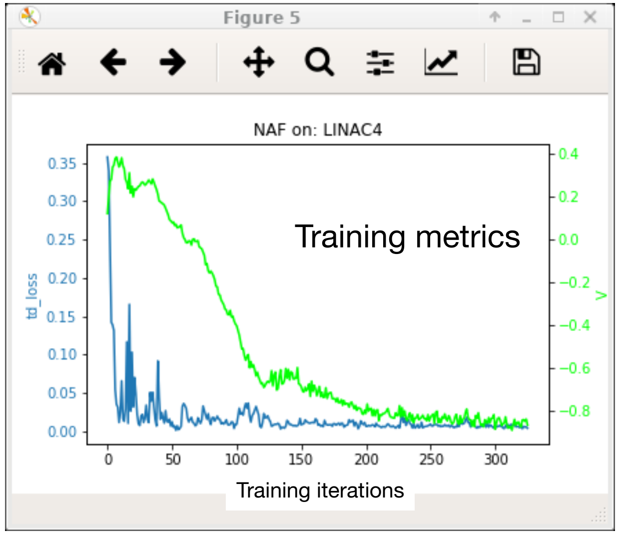
Action space



Observation space

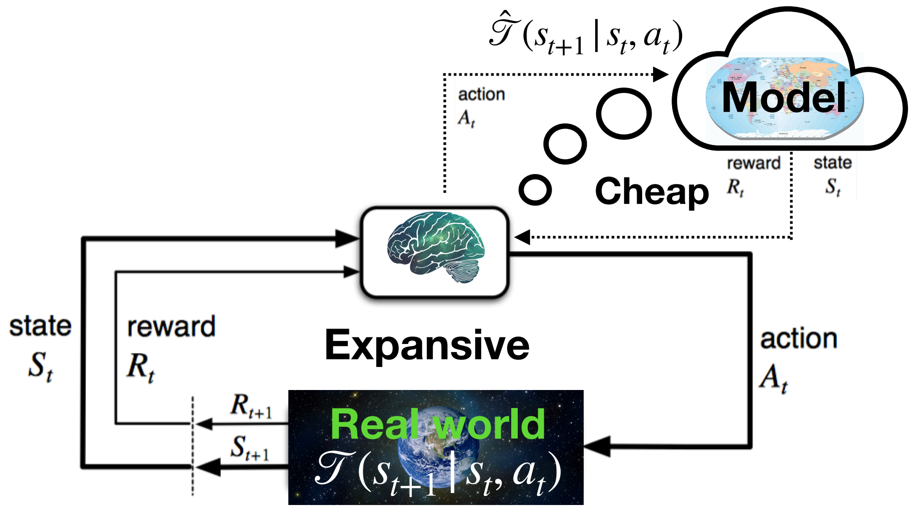




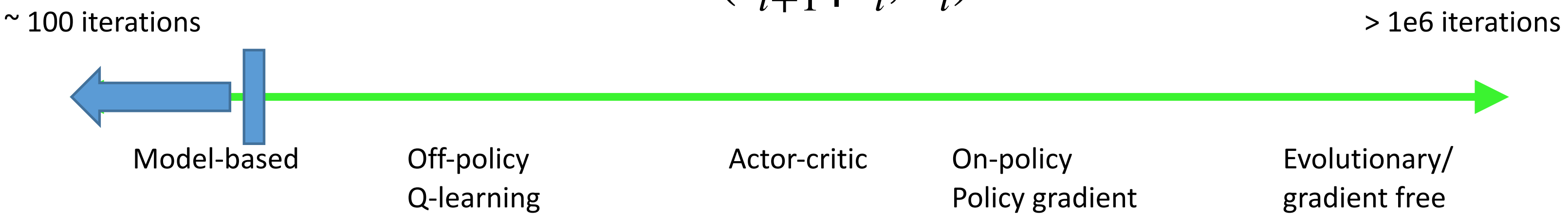


Model based reinforcement learning MBRL

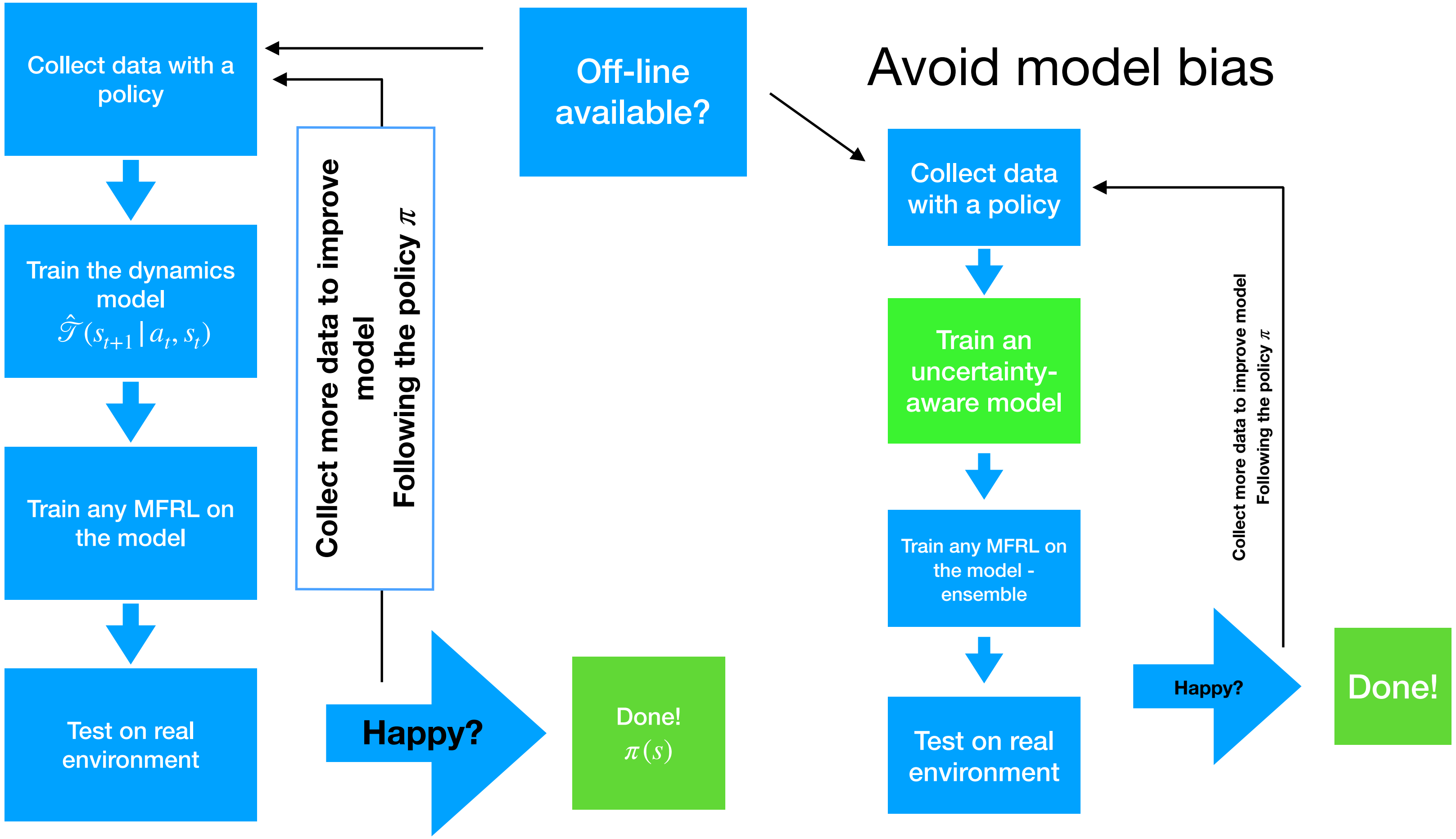
Overview MBRL



- If the interaction with the system is expensive.
- Dynamics of the system: $\mathcal{T}(s_{t+1} | s_t, a_t)$ is approximated via $\hat{\mathcal{T}}(s_{t+1} | s_t, a_t)$.



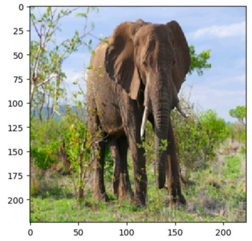
Dyna style work flow



Uncertainty aware modelling

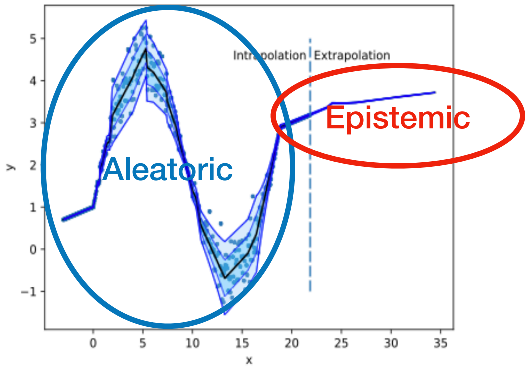
There is no elephant in the room?

- The high performant VGG16-CNN trained on ImageNet data fails to see the elephant in the room :



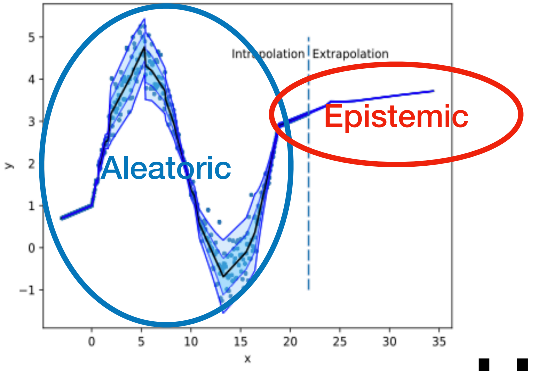
The big DL-lie:

$$P(train) = P(test)$$



- Star Trek motto: “Boldly go where no one has gone before.”

Probabilistic deep learning



How to capture uncertainties of the data?

Answered by probabilistic models

How to capture uncertainties of the model?

Answered by Bayesian statistics

Classic: Maximum likelihood approach
The mother of all loss functions

The Maximum Likelihood Principle Mantra:

Choose the parameter(s) of the model so that the observed data has the highest likelihood (lowest negative log likelihood NLL)

or

Tune the parameter of a model such that the resulting model can produce the observed data with higher probability than all other models with different parameter values

- It is still a probability!

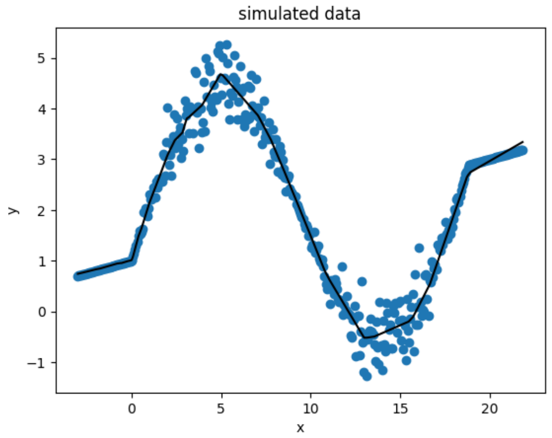
Aleatoric uncertainty: probabilistic modelling

- How to fit a probabilistic model: use a neural network (NN) to determine the parameters of the predicted conditional probability distribution (CPD) for the outcome.

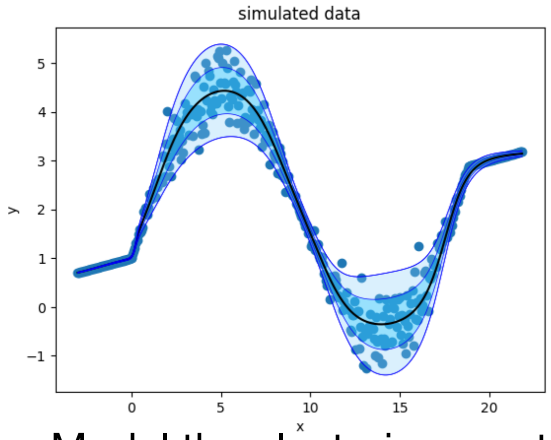
  1. You pick an appropriate distribution model for the outcome.

  2. You set up an NN that has as many output nodes as the model has parameters.

  3. You derive from the picked distribution the negative-log-likelihood function and **use that function as the loss function to train the model**.



Standard output



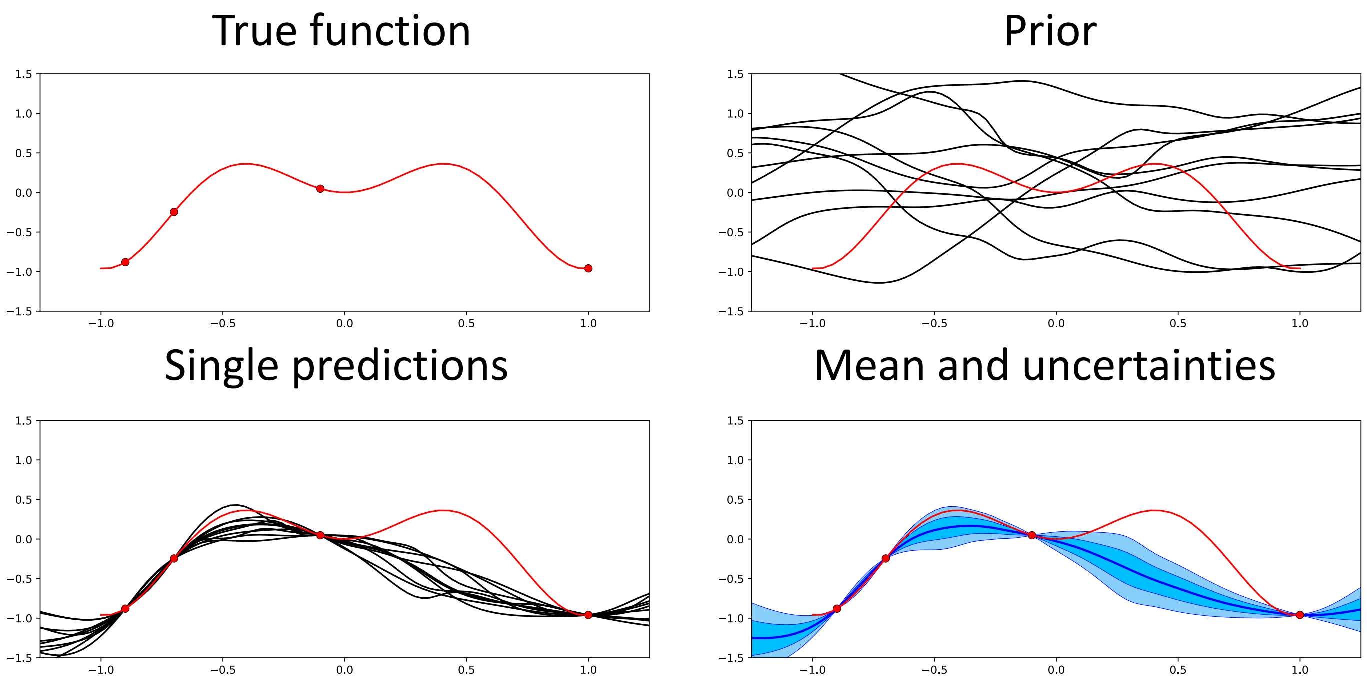
Model the aleatoric uncertainty (heteroscedastic)

Anchored ensembles (AE)

- ➪ Capture both uncertainties
- ➪ Mimics true Bayesian posterior
- ➪ The aleatoric error is taken into account via anchored regularisation.

https://arxiv.org/abs/1810.05546

- ➪ Fast and intuitive!!!!



Epistemic uncertainty

- The Bayesian interpretation of uncertainty is a lack of knowledge.
- More data → less uncertainty

likelihood
probability distribution of the observed data given a parameter value

(how probable are the observed data for this parameter value?)

prior
probability distribution of the parameter independantly from any observation

(prior knowledge: how probable are each value of the parameter before any observation?)

Bayesian rule: $$p(\theta \mid x) = \frac{p(x \mid \theta)\, p(\theta)}{p(x)}$$

posterior
probability distribution of the parameter given the observed data

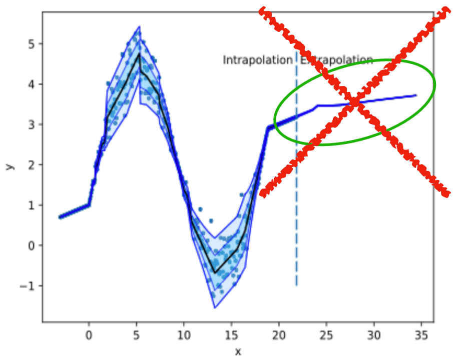
(updated knowledge: how probable are each value of the parameter given the observed data?)

evidence
probability distribution of the observed data independantly from any parameter value

(how probable is it to observe these particular data?)

https://Miro.medium.com/max/1260/1*04pd7c6QIHXYHgAelzzWlg@2x.png

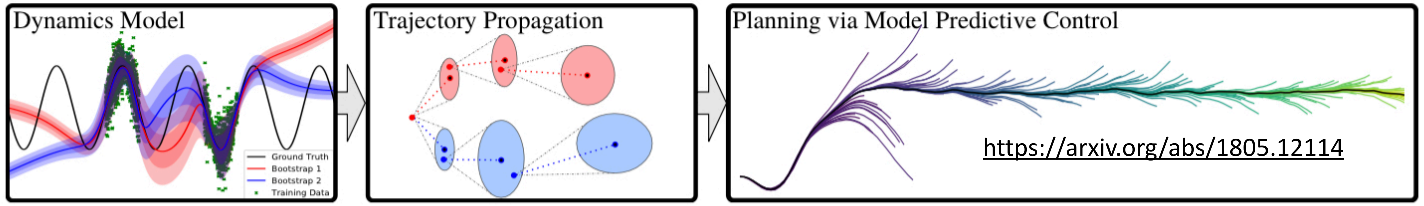
Extrapolation of a standard approach fails

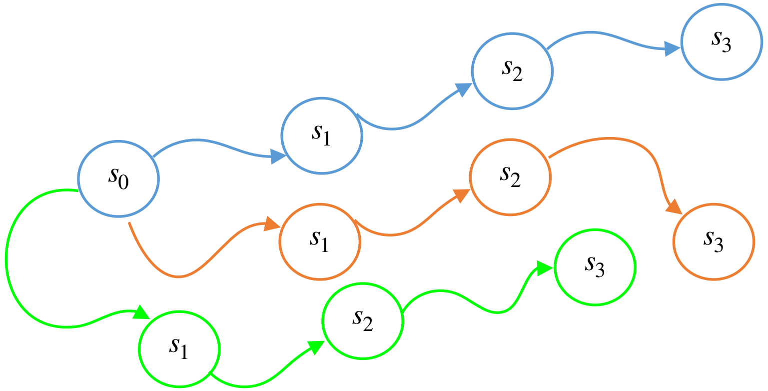


Ways to compute the probability:

1. Variational inference: parametrised approximation - cheap but not accurate.
2. Markov Chain Monte Carlo (MCMC) - sampling. Accurate but expensive.
3. Drop out: https://arxiv.org/abs/1506.02142
4. Ensemble techniques.

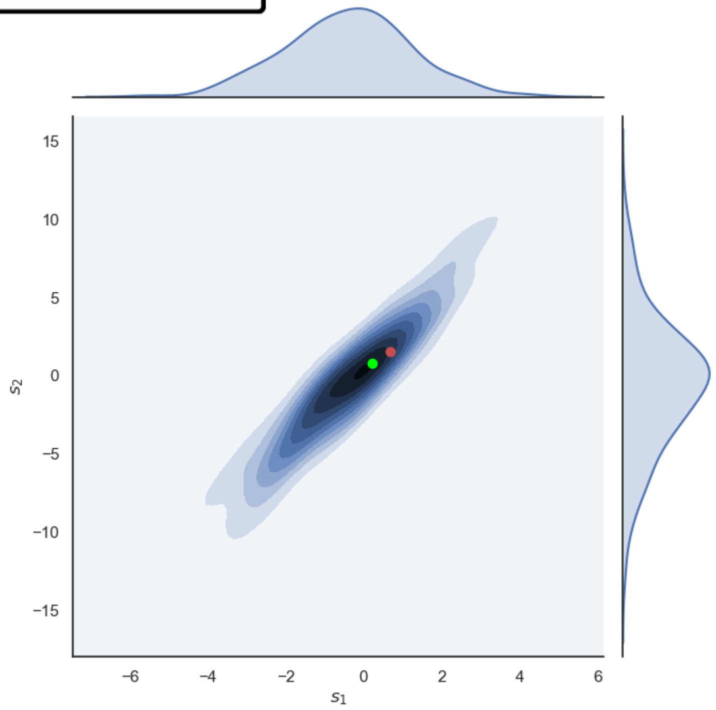
How to take the uncertainty into account?





- Take each step a model randomly
- Take the lowest reward/mean
- Randomly select a model for each episode

Example: AWAKE steering

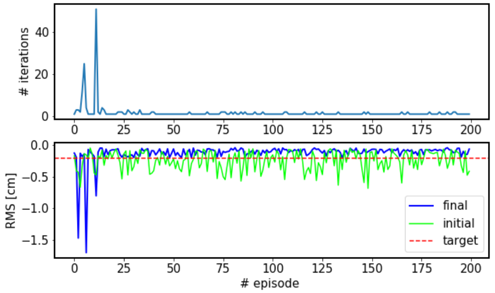


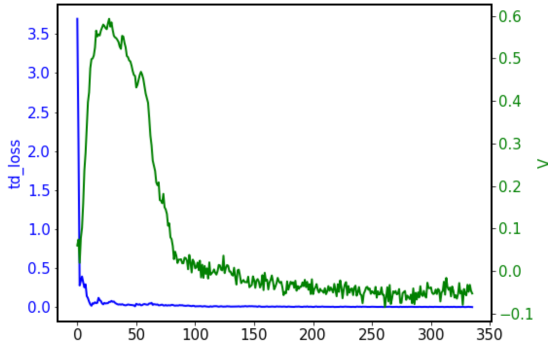
- Only two elements (s_1, s_2) and an action (a_1, a_2) to (s_1', s_2').
- A multivariate normal distribution to represent the posterior probability.
- We obtain a transition probability $s \overset{p(s)}{\mapsto} s'$.
- Gets non-feasibly for longer trajectories…

Applications

Comparing the performance

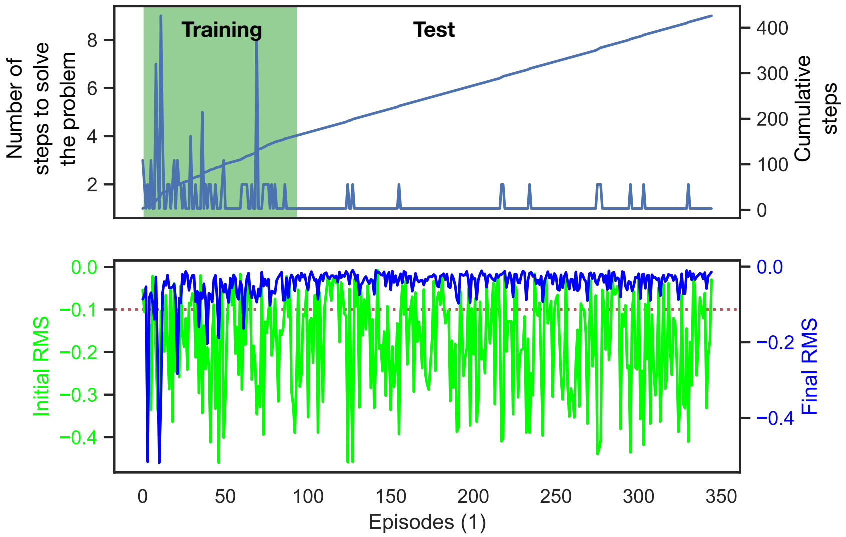
Highly sample efficient algorithm
learning the controller directly PER-NAF
Assumes specific problem structure





Done on the machine!

Highly sample efficient algorithm
learning the controller indirectly on a
model ensemble
**Ability to overcome limitations while
being sample efficient**

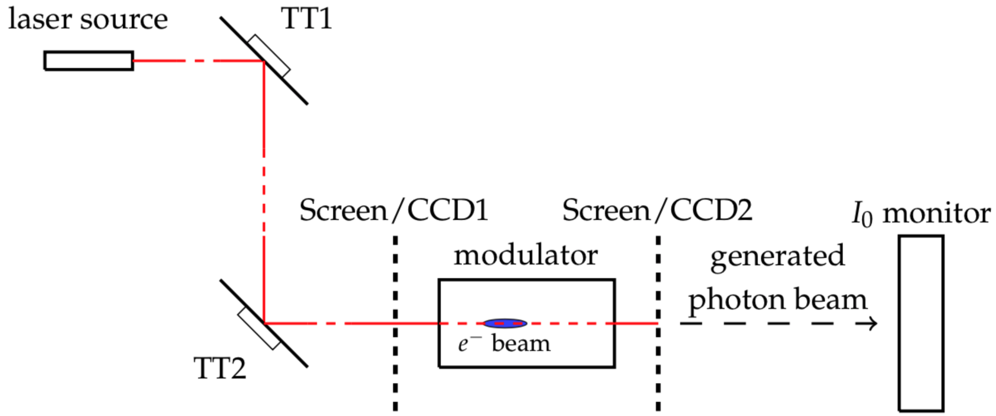


Simulation!

In principle reduction up to factor 3!

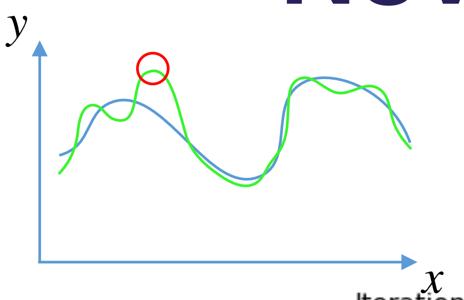
PARIS LODRON UNIVERSITÄT SALZBURG
PARACELSUS MEDIZINISCHE PRIVATUNIVERSITÄT
salzburgresearch
FH Salzburg
LAND SALZBURG
CERN

The Fermi FEL Studies

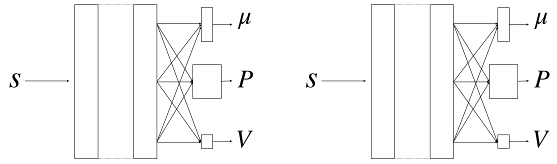


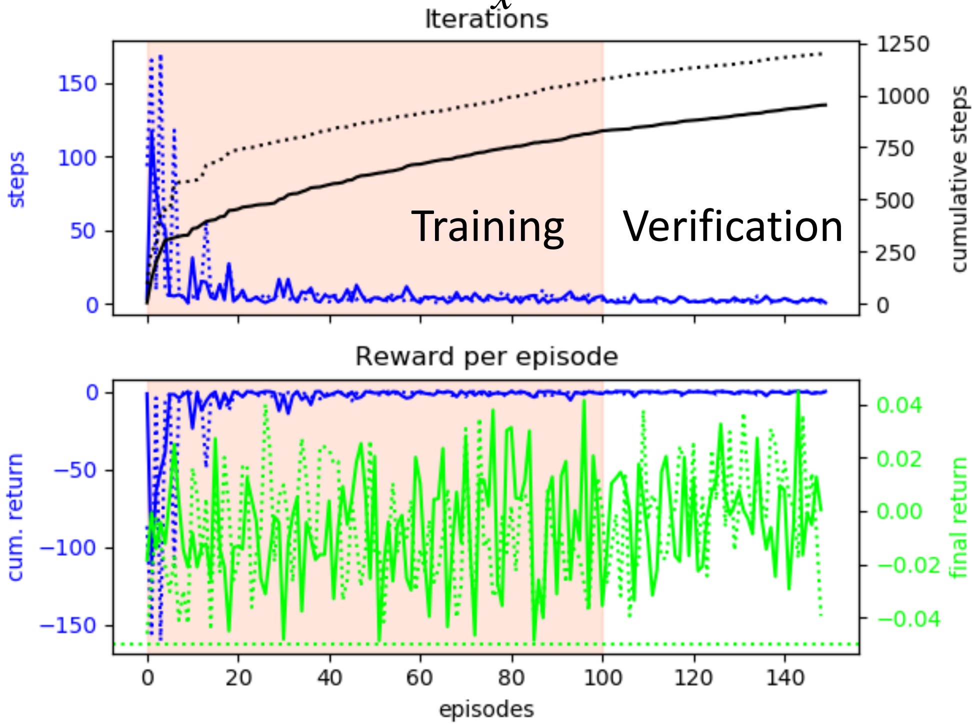
- Laser is interfering with electron beam
- 4 DOF in action and state
- Nonlinear in nature

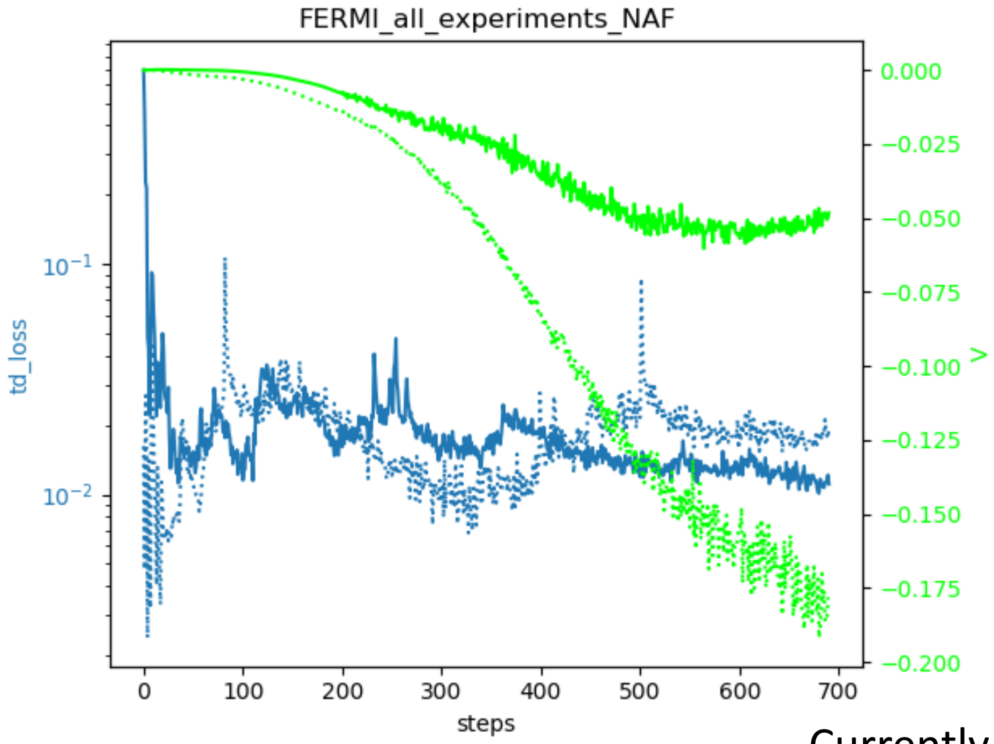
Novel NAF2 on FERMI - FEL



Double network - solid
Singe network - dashed



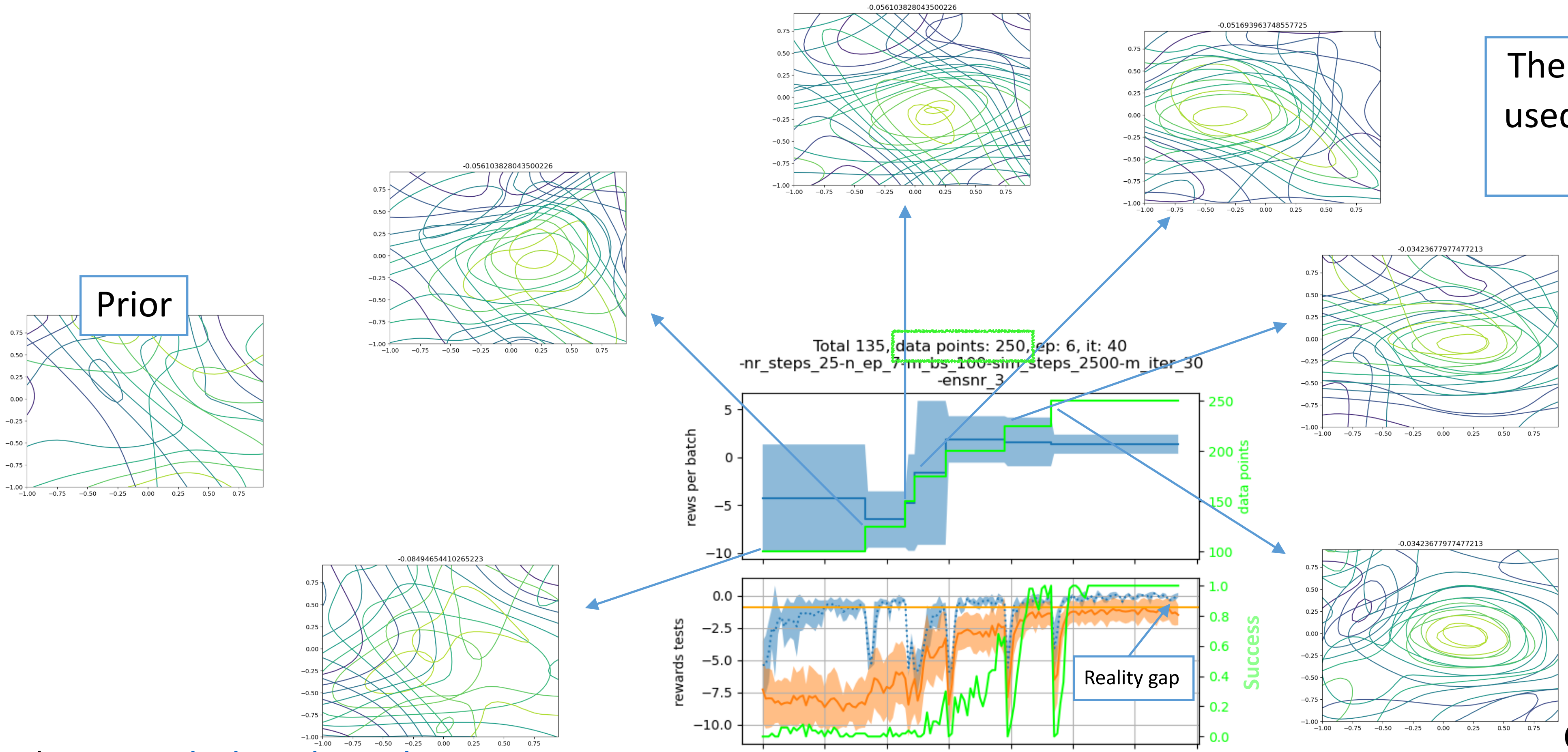




Currently published

Code: simon.hirlaender@sbg.ac.at

Novel AE-Dyna
Theoretical aspects - learning evolution

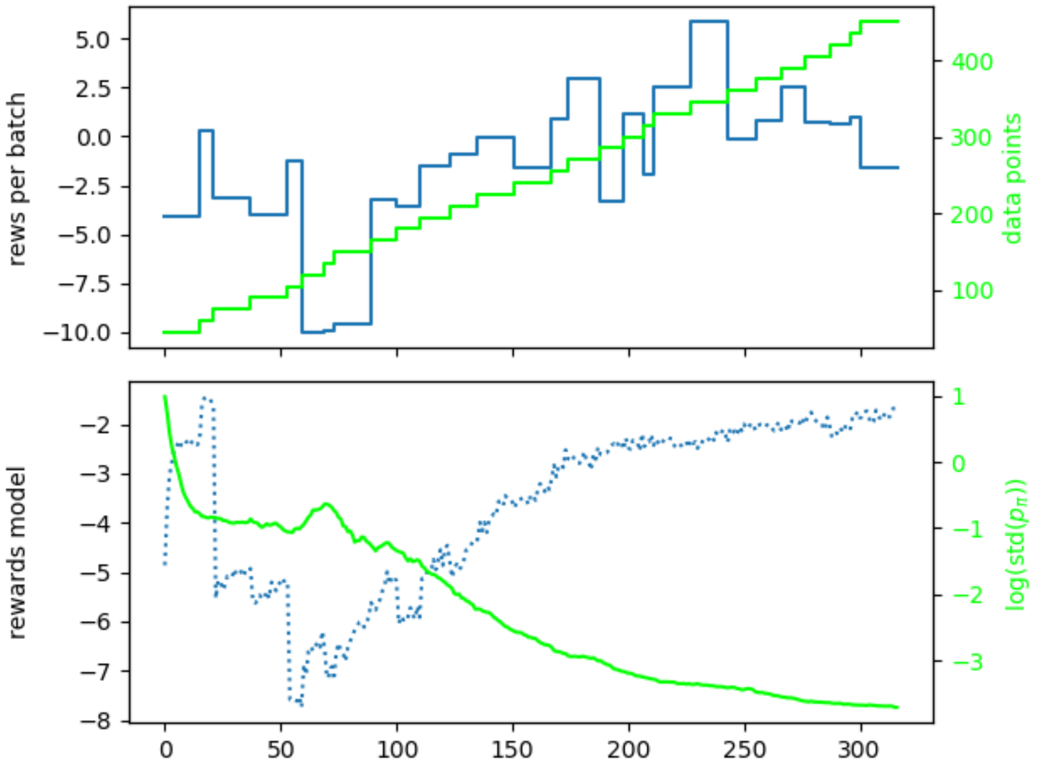


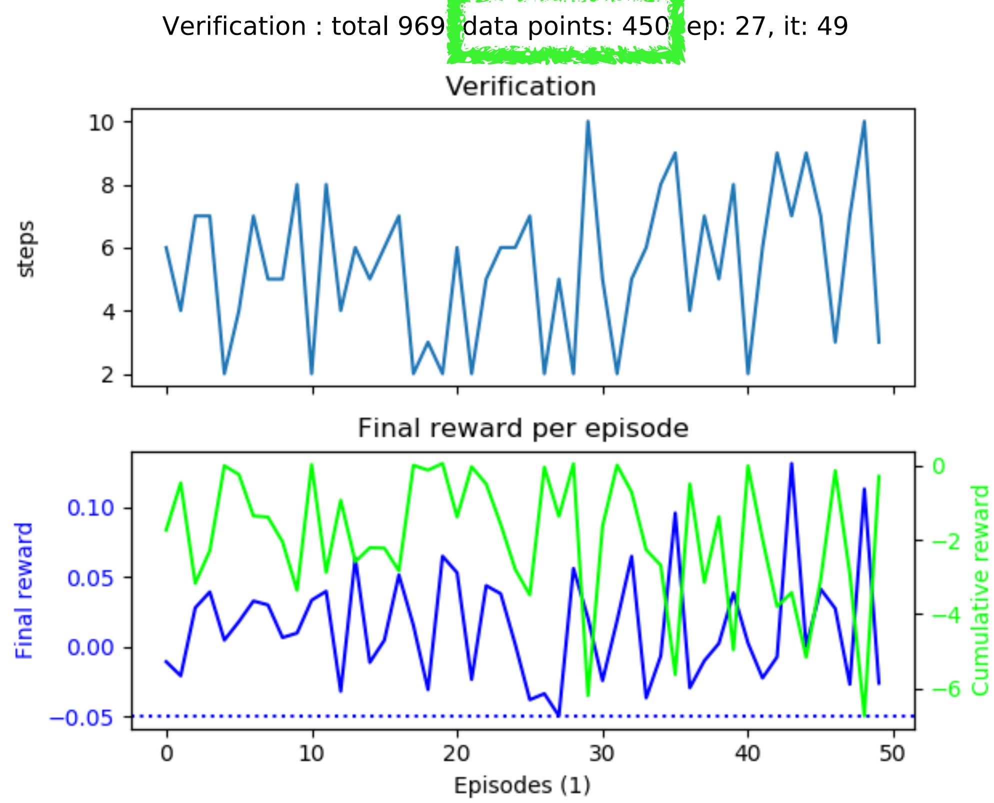
The Soft actor critic is used to maximise also the entropy

Currently published

Code: simon.hirlaender@sbg.ac.at

ME-TRPO variant on FERMI - FEL





Code: simon.hirlaender@sbg.ac.at

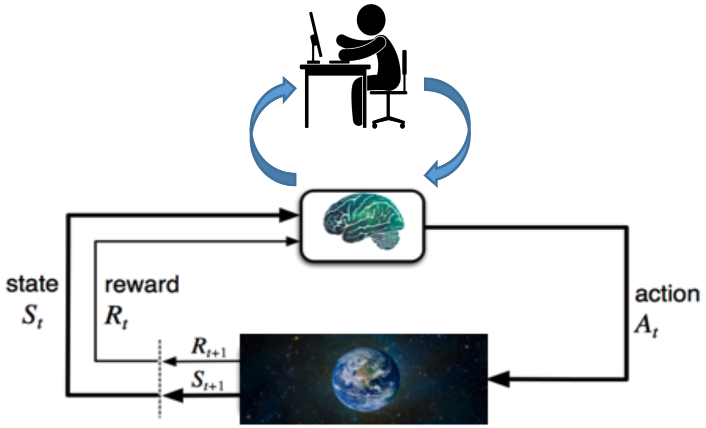
Currently published

Conclusions

- RL is powerful - enormous potential for many applications in accelerator operation:

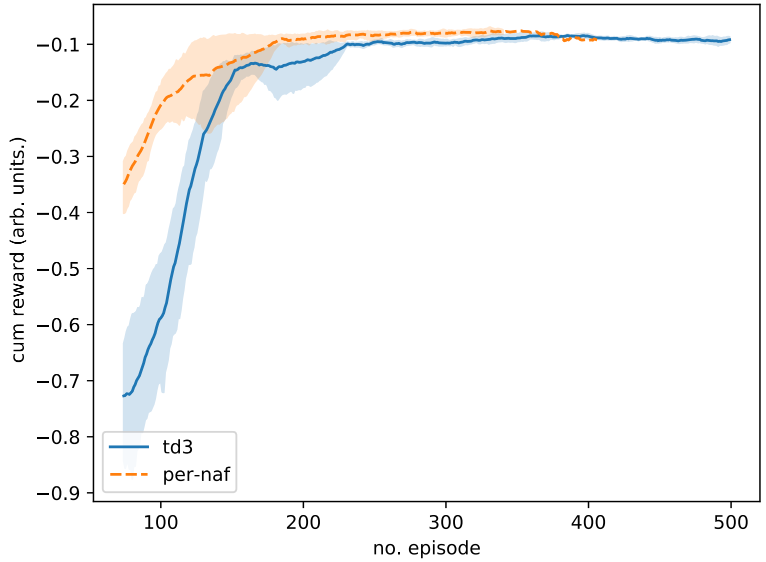
Optimiste a sequence of decissions

- RL is a highly popular research topic
- RL is specific - adjust method - needs some experience.

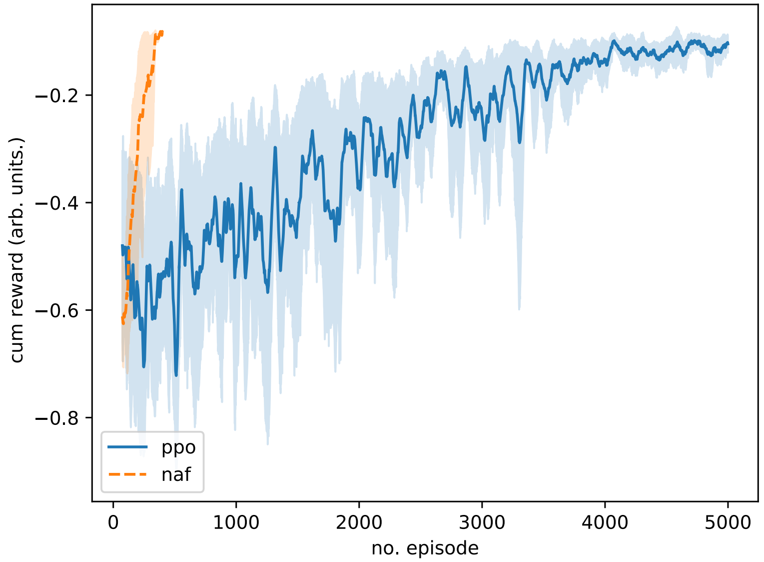


PER-NAF vs. state of the art

TD3 - offline Twin Delayed DDPG



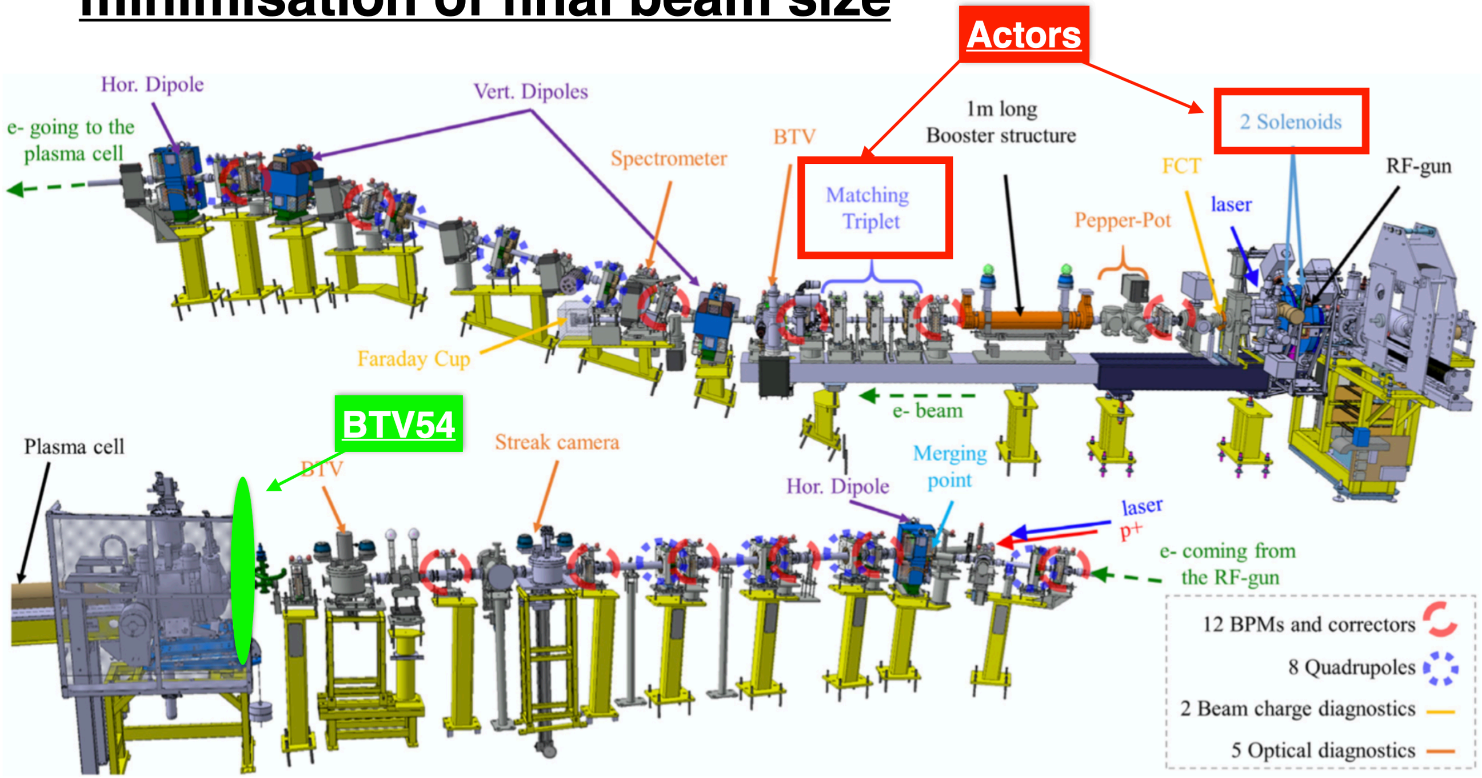
Proximal Policy Optimization (on-policy):



Additional AWAKE studies

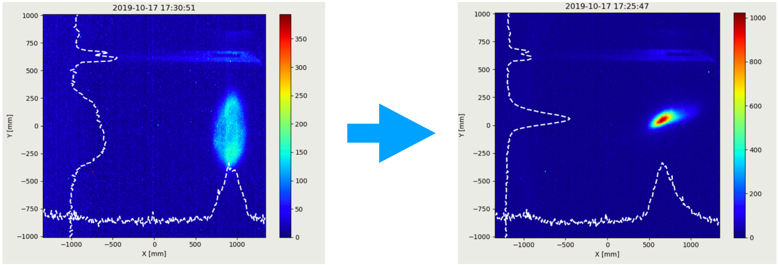
Auto-matching of AWAKE

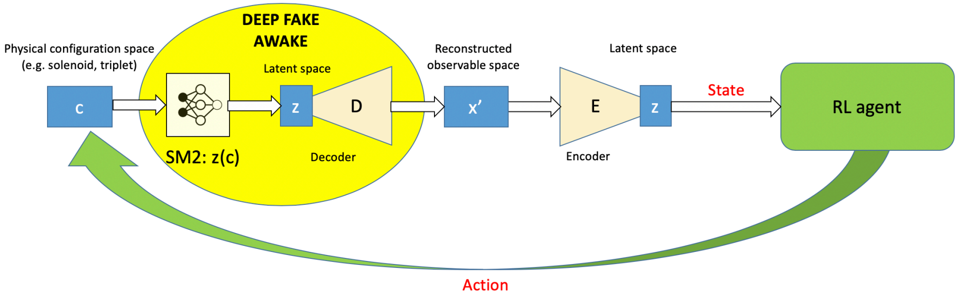
- Thanks to the convexity of the final beam size at the end of the TL wrt optics initial conditions => **minimisation of final beam size**

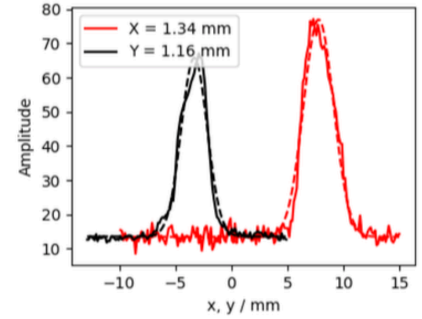


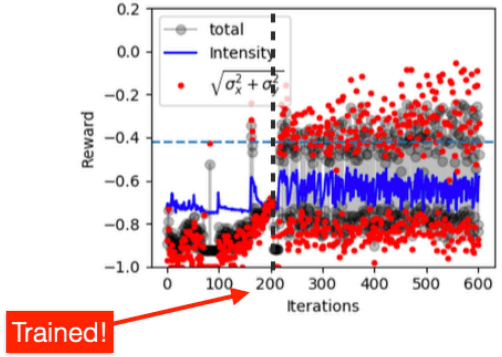
VAE for off-line RL agents training

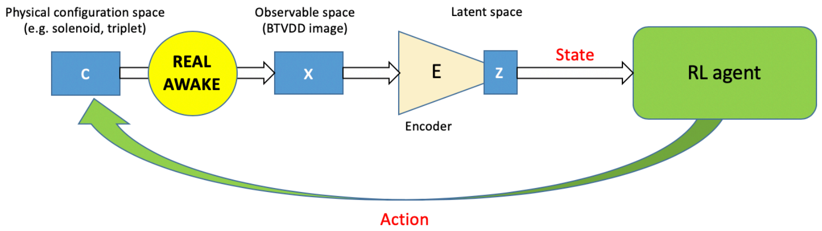
- Variational Auto-Encoder (VAE) used in AWAKE
- Basically, <u>**VAE gives access to latent space representation**</u> of the CNN of the input images
- In AWAKE, we exploited this in 2 ways:
  - 1) to build a data-based model of the TL (at least of the parameters we were interested in)
  - 2) to extract features from images to be used as environment observables





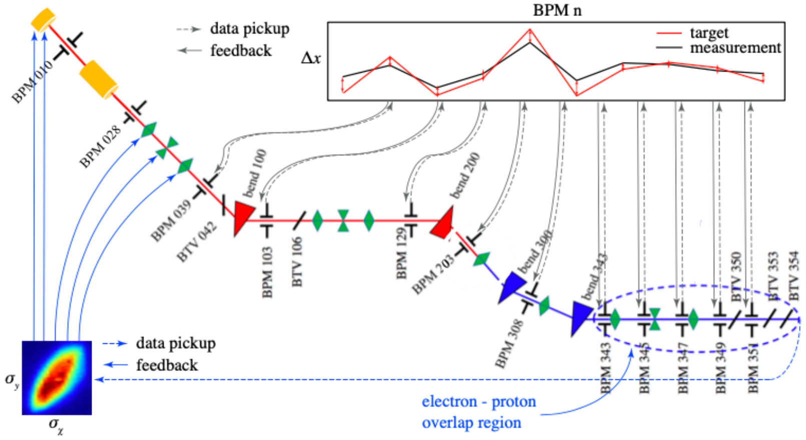


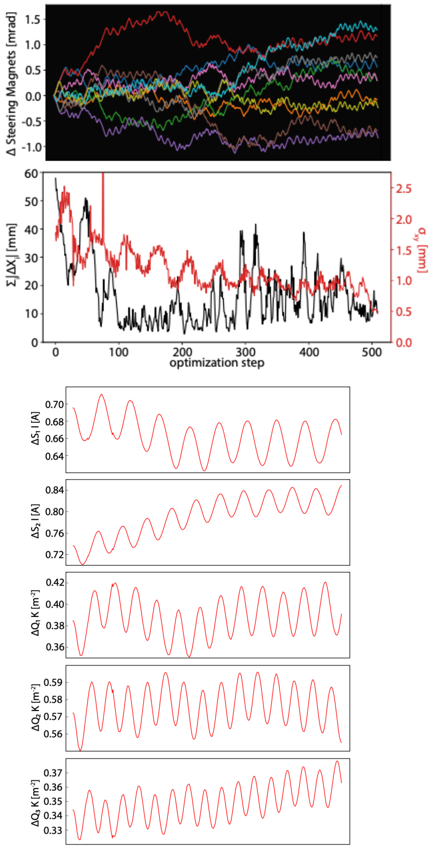




Extremum seeking ES

- Running two independent instances:
  - One is stabilising the trajectory
  - The other one is optimising the emittance





Courtesy of Alexander Scheinker