Reinforcement Learning applied to Optimization of LHC beams in the CERN Proton Synchrotron

Presenter: Joel Wulff

Date of presentation: 2/11/2022

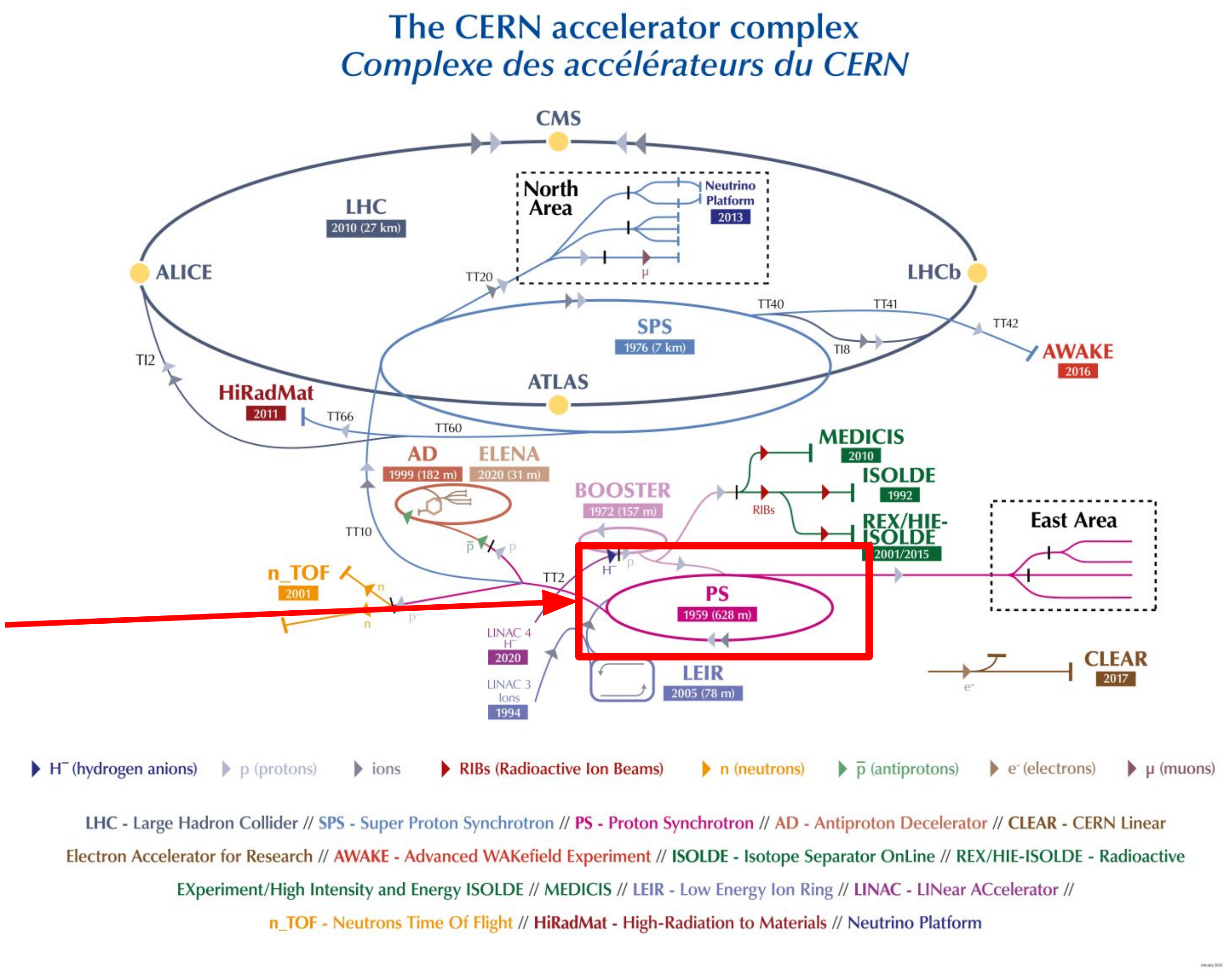
Acknowledgements: A. Lasheen,
CPS operation crew

Special thanks to the Center for Bright Beams for the student grant to attend this event

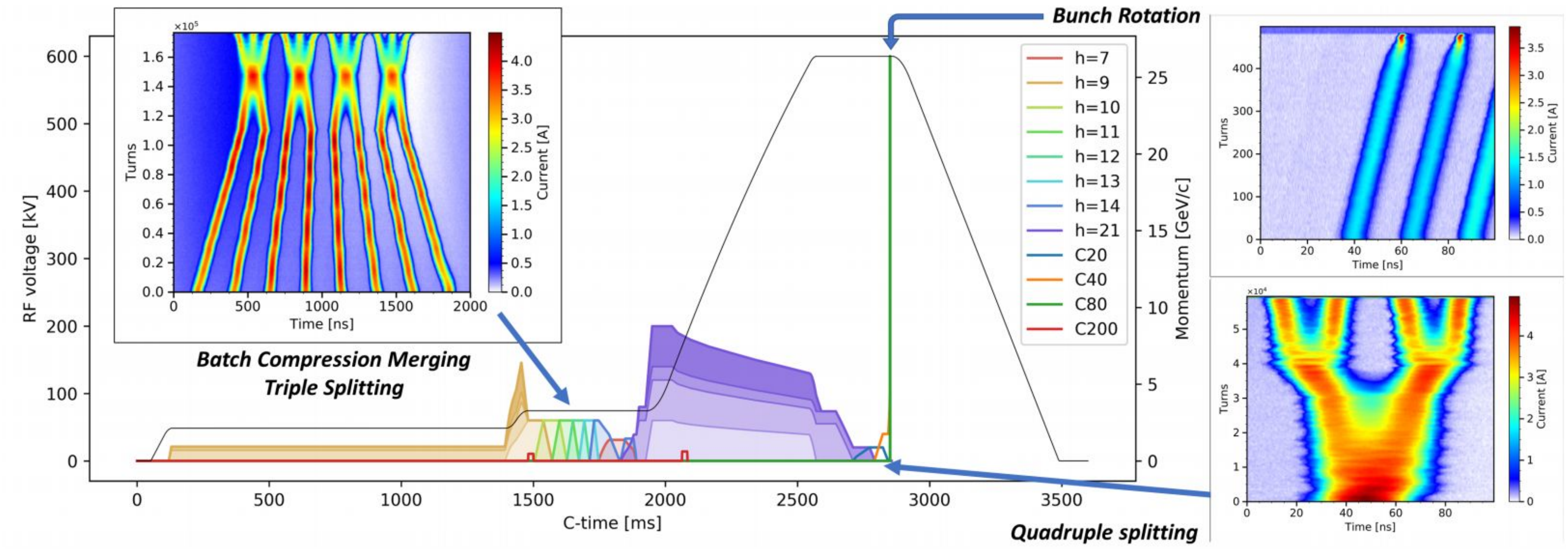
Part 1: Introduction and motivation

The CERN accelerator complex

- A complex combination of accelerators and experiments.
- Particles used for the LHC go through a cascade of 4 separate accelerators before injection.
  - The nominal bunch spacing of 25 ns is created in the Proton Synchrotron (PS) through a series of RF manipulations.
  - These manipulations need to be carefully optimized to create good quality beams for the LHC.



RF manipulations in the PS



All RF voltage programs and RF manipulations for the BCMS cycle, an LHC type beam.

- The PS has a large number of RF systems covering a wide range of RF harmonics, allowing for plenty of RF manipulations.
- The relevant parameters are the **RF amplitude** and **phase**, that can be adjusted for each harmonic to produce the desired bunch characteristics.

RF manipulations in the PS

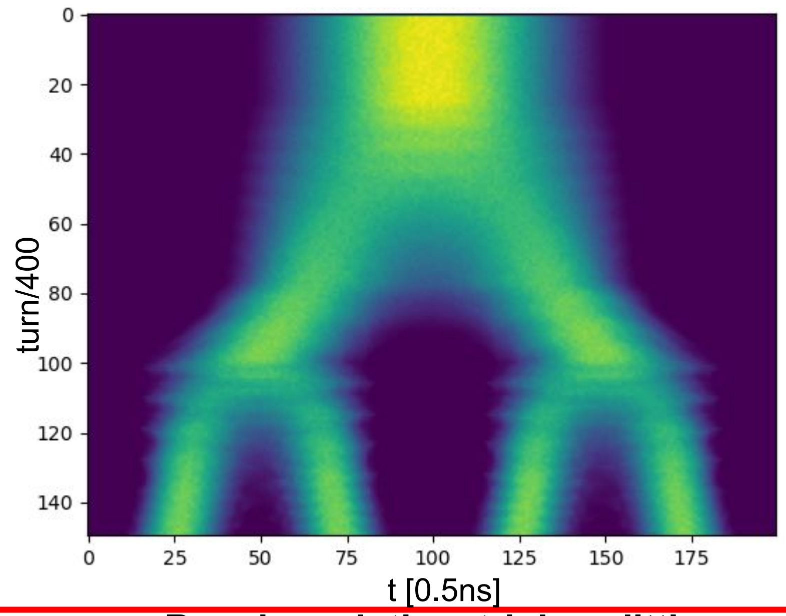
- Ongoing project at CERN: automate setup and optimization of RF manipulations in the PS

- Presently, settings are adjusted manually, which
  - Takes time,
  - Relies on operator experience,
  - Risks performance inconsistency due to qualitative judgements of when the beam is “good enough”.

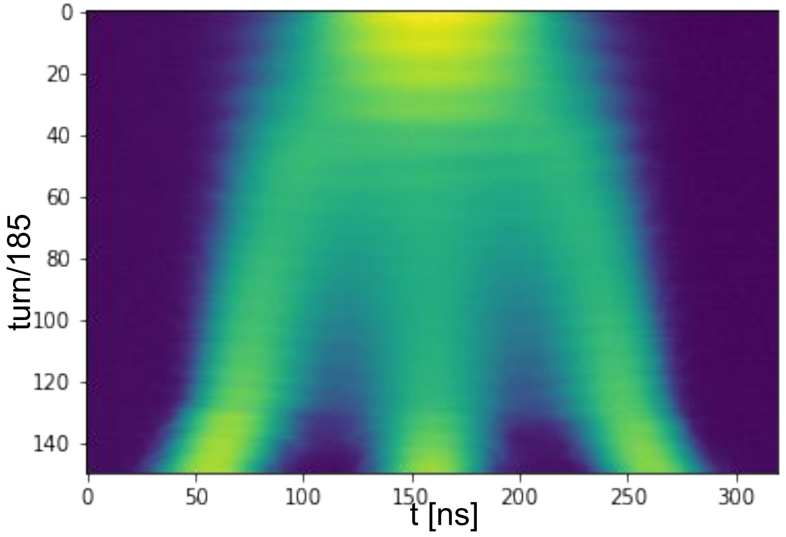
- Initial focus on RF splittings,
  - Quadruple splittings
  - Triple splittings

Promising results in both cases, however focus on triple splitting for this presentation!

Bunch evolution, Quad. splitting



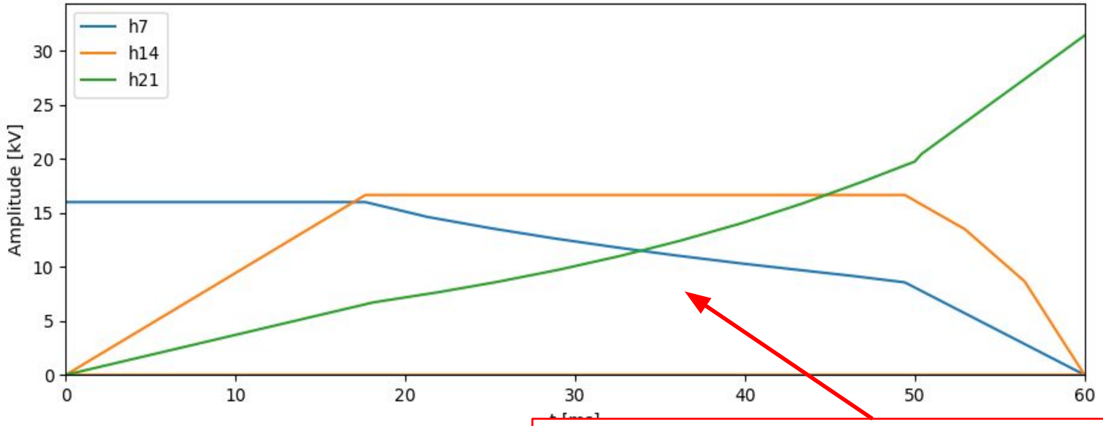
Bunch evolution, triple splitting



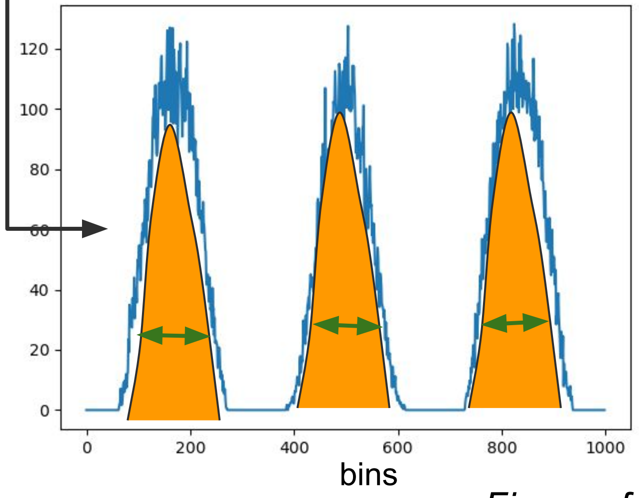
The triple splitting: Parameters, Observables and Goal

- **Three main parameters to optimize (chosen):**
  - Phases and voltage,
    ϕ_{14}, ϕ_{21} , **and** V_{14} **.**
- **Observables:**
  - Final bunch **profiles**, final bunch-by-bunch **length** + **intensity**.
- **Goal:**
  - All bunch-by-bunch observables equal after splitting.
  - Quality measured through Mean Square Error (MSE) between bunches after splitting.
    - **+ :** Single metric that judges overall splitting quality.
    - **- :** Many local minima…

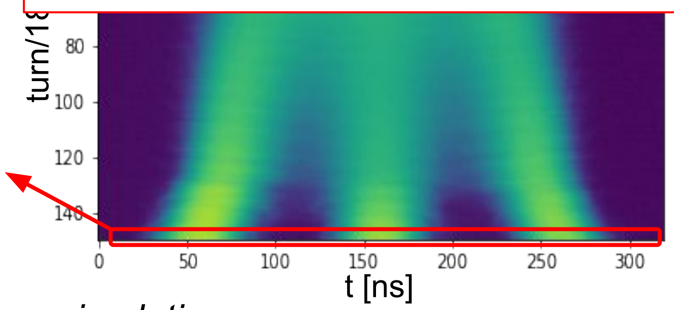
RF voltage program



Final profile



Three **simultaneously** active cavities with different voltages → **Non-linear** interactions, **difficult to optimize**...



Figures from simulations

Part 2: Automation through ML

Applying Reinforcement Learning to efficiently optimize the triple splitting

Overview of the setup: Two main ML components

Feature extractor (CNN)

RL-Agent(s)

Based on supervised learning and computer vision approach to **process more information and downscale it to simple, actionable parameters**.

Based on deep reinforcement learning to **train an agent to complete a task by taking correct actions**, i.e. actually optimizing a splitting.

The feature extractor

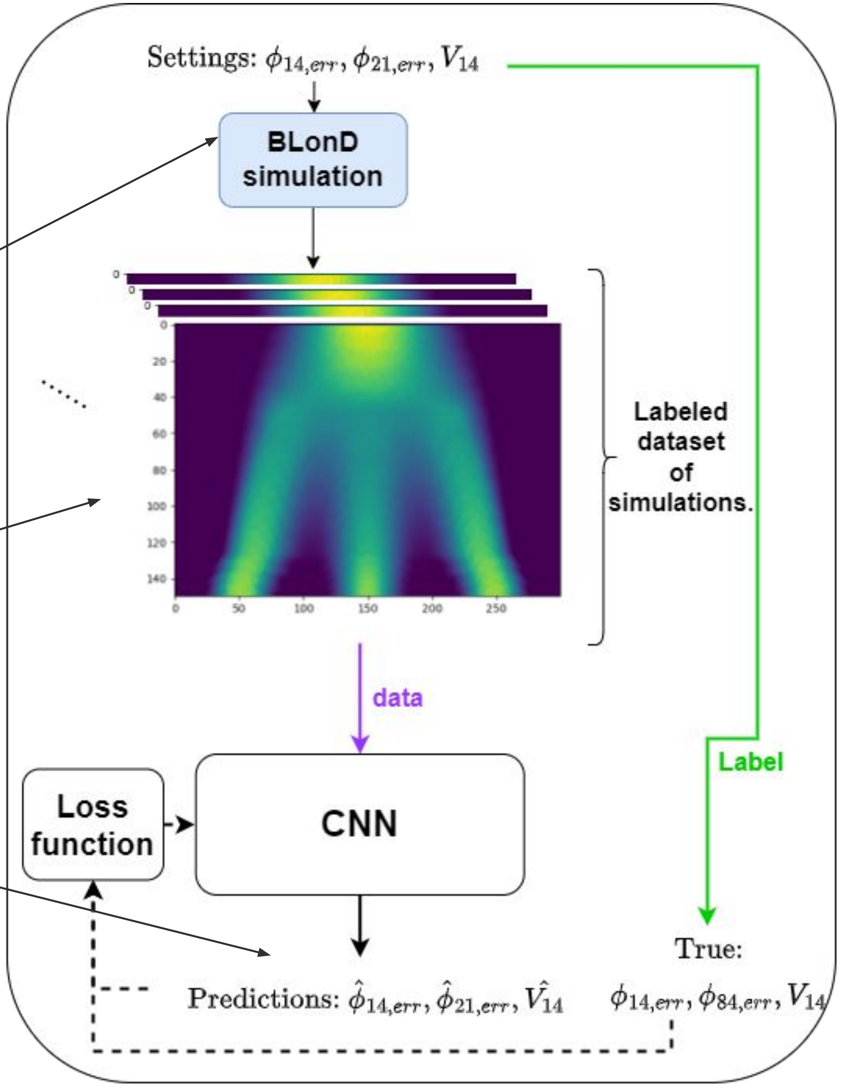
- A *supervised* **C**onvolutional **N**eural **N**etwork (CNN)

Simulated dataset using the BLonD tracking code
→ Necessary to acquire enough labeled data

Data: series of bunch profiles over time. Similar to an image.

Predicts $\phi_{14,err}, \phi_{21,err}, V_{14}$.

Works in simulation, with small prediction errors!

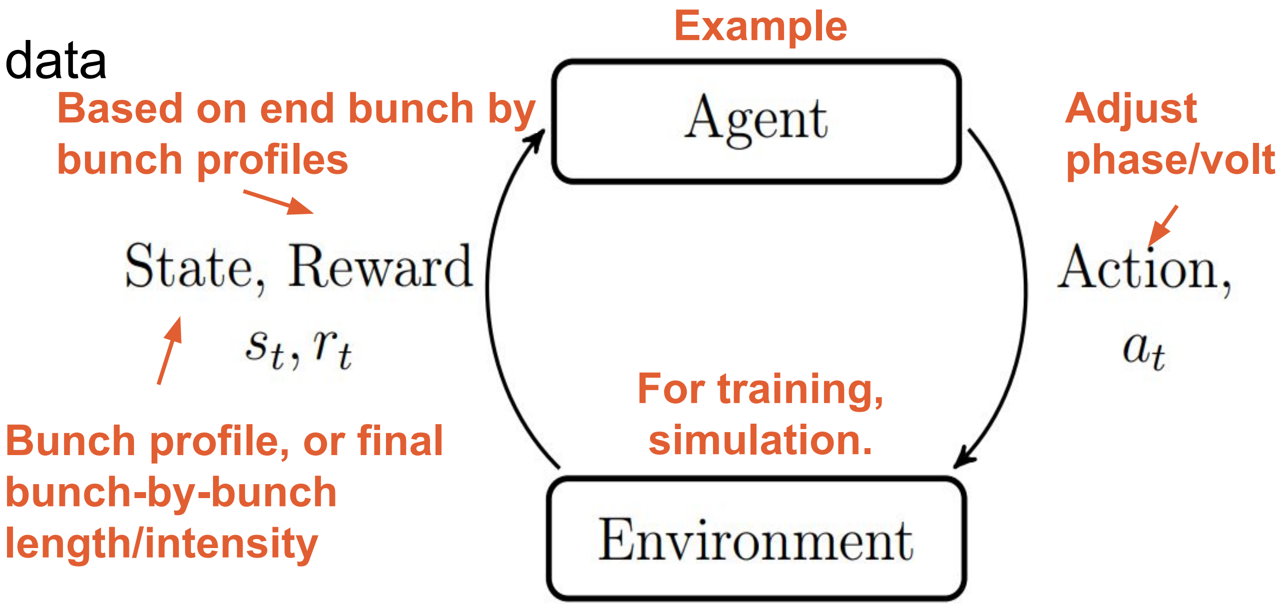


The RL-agent

- **Model-free: S**oft-**A**ctor **C**ritic (**SAC**)
    - Trained in an environment of simulated data
    - Acting on $\phi_{14}, \phi_{21}, V_{14}$.

- Several versions tested.
    - In this presentation only the **final triple splitting setup** is presented.

NOTE: all models (CNN/RL Agents) used have been trained on simulated data only!

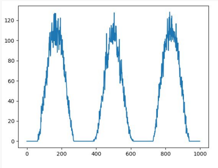


Agent-environment interaction loop

Trial and error: different attempted approaches

1. Optimize directly

Final profile



Extract final bunch lengths/intensities

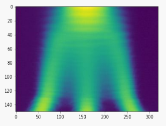
Input: BLs, Int. → RL Agent, $\{\phi_{14}, \phi_{21}, V_{14}\}$ → Sim. ✗

Failed already in simulation. Why? Perhaps not enough information…

2. Provide more info.

Final bunch lengths/intensities

Input:



→ Feature extractor (CNN) → RL Agent, $\{\phi_{14}, \phi_{21}, V_{14}\}$ → Sim. ✔ → Op. ✗

Feature extractor not accurate enough on ***real data****…*

3. Simplify through segmentation

Optimize sequentially, phase → voltage

~~RL Agent, $\{\phi_{14}, \phi_{21}, V_{14}\}$~~ → RL Phase Agent, $\{\phi_{14}, \phi_{21}\}$ → RL Volt Agent, $\{V_{14}\}$

Required decoupling of phases and voltages in loss function…

The phase and voltage losses

1. Phase Loss:
Compare only the outer two bunches. From beam dynamics, we know that for almost all combinations of phase offset and voltage factor we will observe a difference in their shapes.

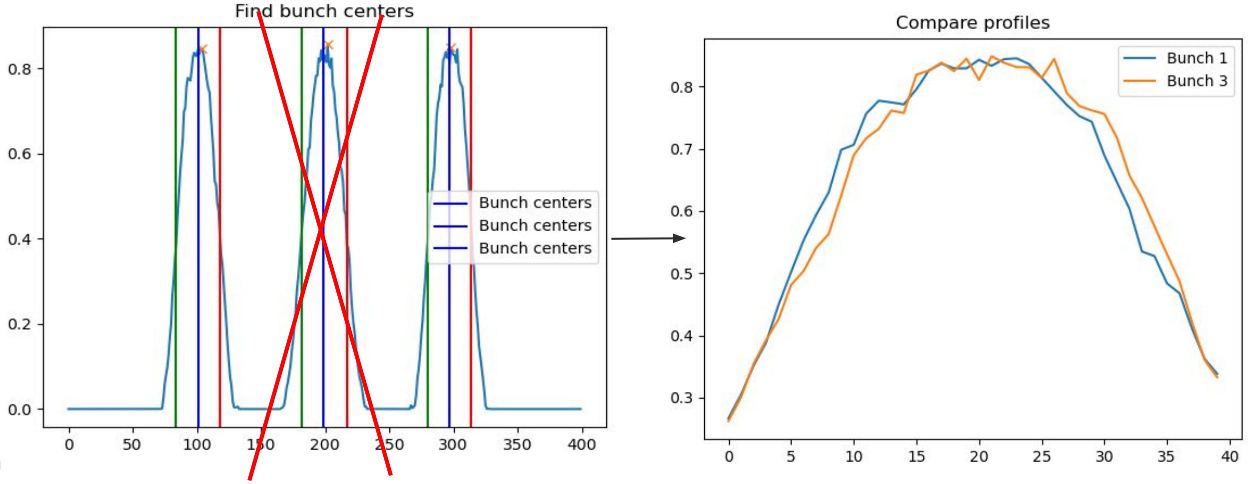
With optimal phase, they should **always** be identical!
Gives a semi-voltage agnostic loss.

2. Voltage Loss: Assume phase is already optimized,
→ Optimization reduced to a univariate problem.

Reuse original three-bunch comparison,
→ Provides a nice, approximately parabolic loss curve!

Note: See the extra slides for a scan of phase losses for phase errors at different fixed voltages.

Figure: Illustration of phase loss. Isolated outer bunches are compared through MSE.



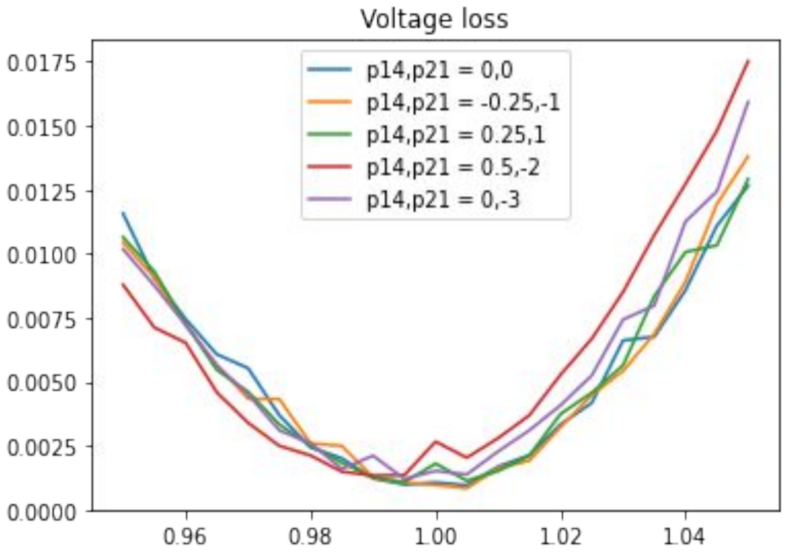


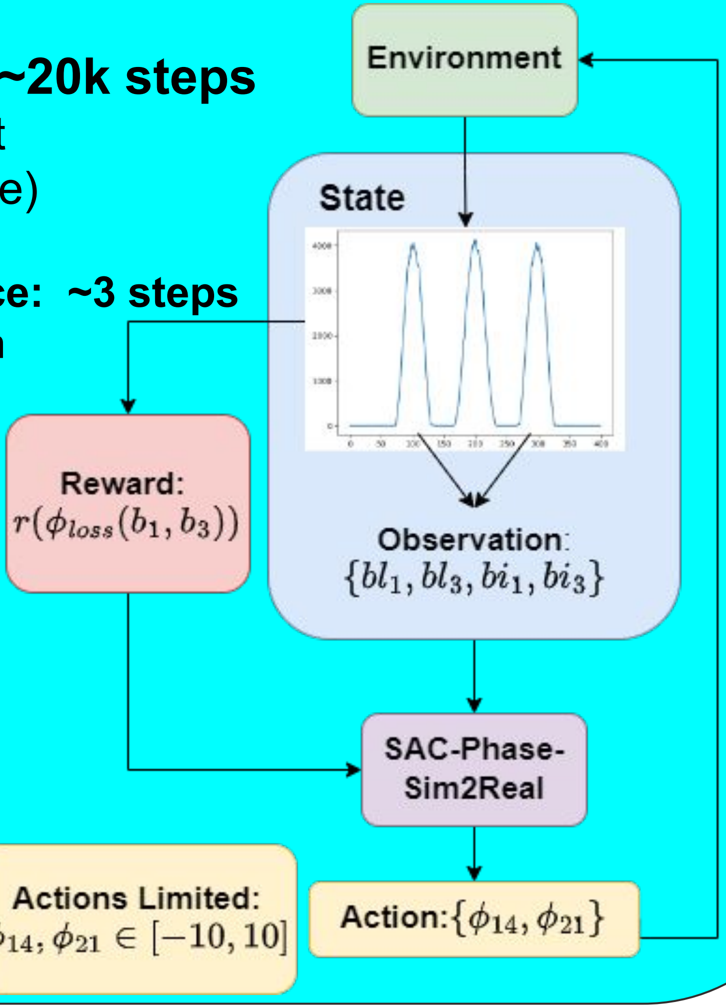
Figure: Scan of profile loss as a function of voltage factor for small residual phase errors.

Segmented RL-Agents: Setup and sim. results

Phase Agent

Training: ~20k steps
(60k to best performance)

Performance: ~3 steps
in simulation

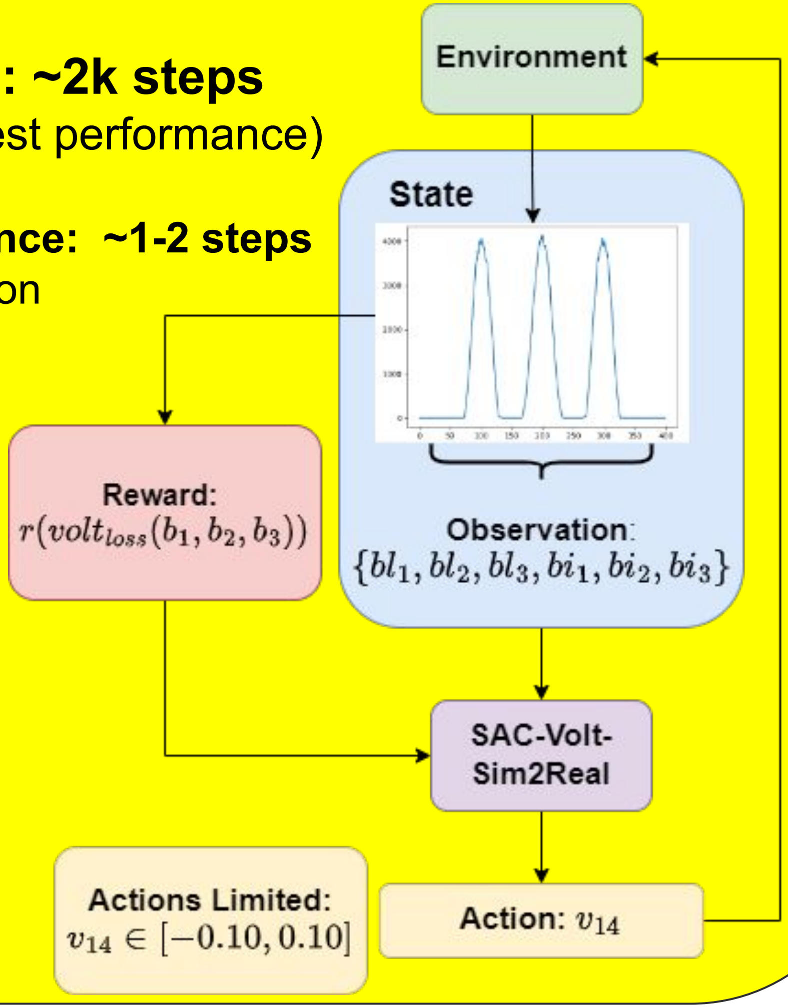


Work great in simulation, what about operation?

Voltage Agent

Training: ~2k steps
(20k to best performance)

Performance: ~1-2 steps
in simulation

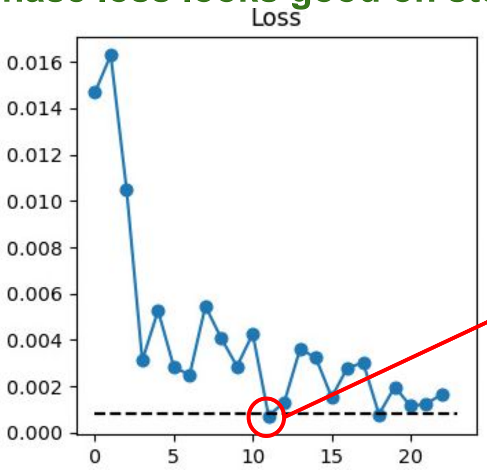


Segmented RL-Agents: direct application

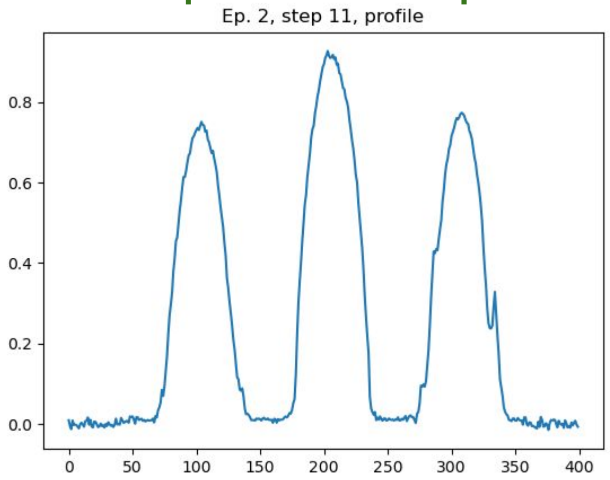
Initial test: Apply the pre-trained RL-Agents **directly** to the output from the PS, optimizing **Phase** → **Voltage**.

Unreliable → Succeeded most of the time, but not always. Why? An example…

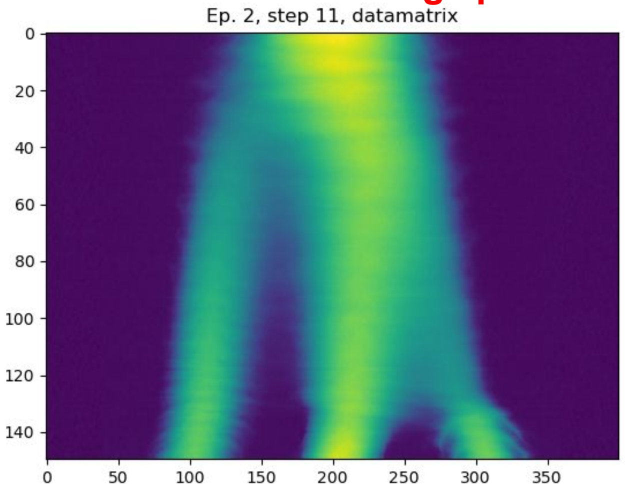
Phase loss looks good on step 11,



Final profile looks in phase



Bunch evolution shows large phase error!

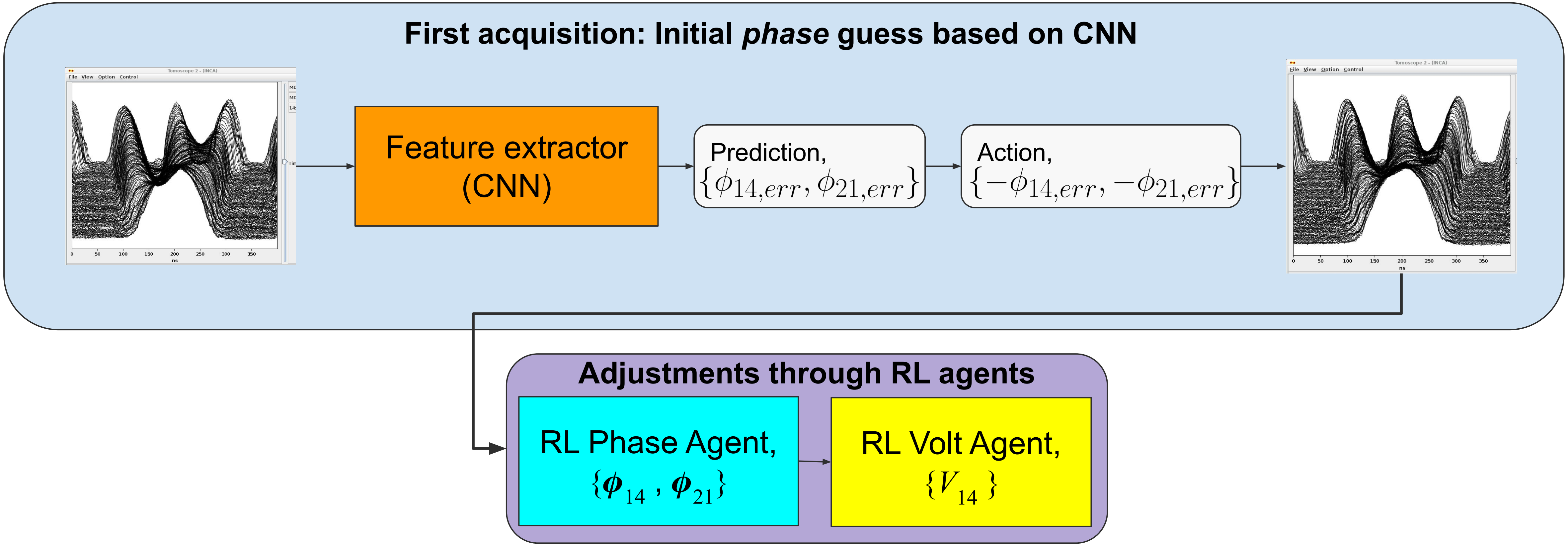


In few special cases, the information contained in final profile sometimes **not enough** to solve the problem. Could **more** information be leveraged to find a better initial condition?

→ **Yes**, by using the pre-trained feature extractor!

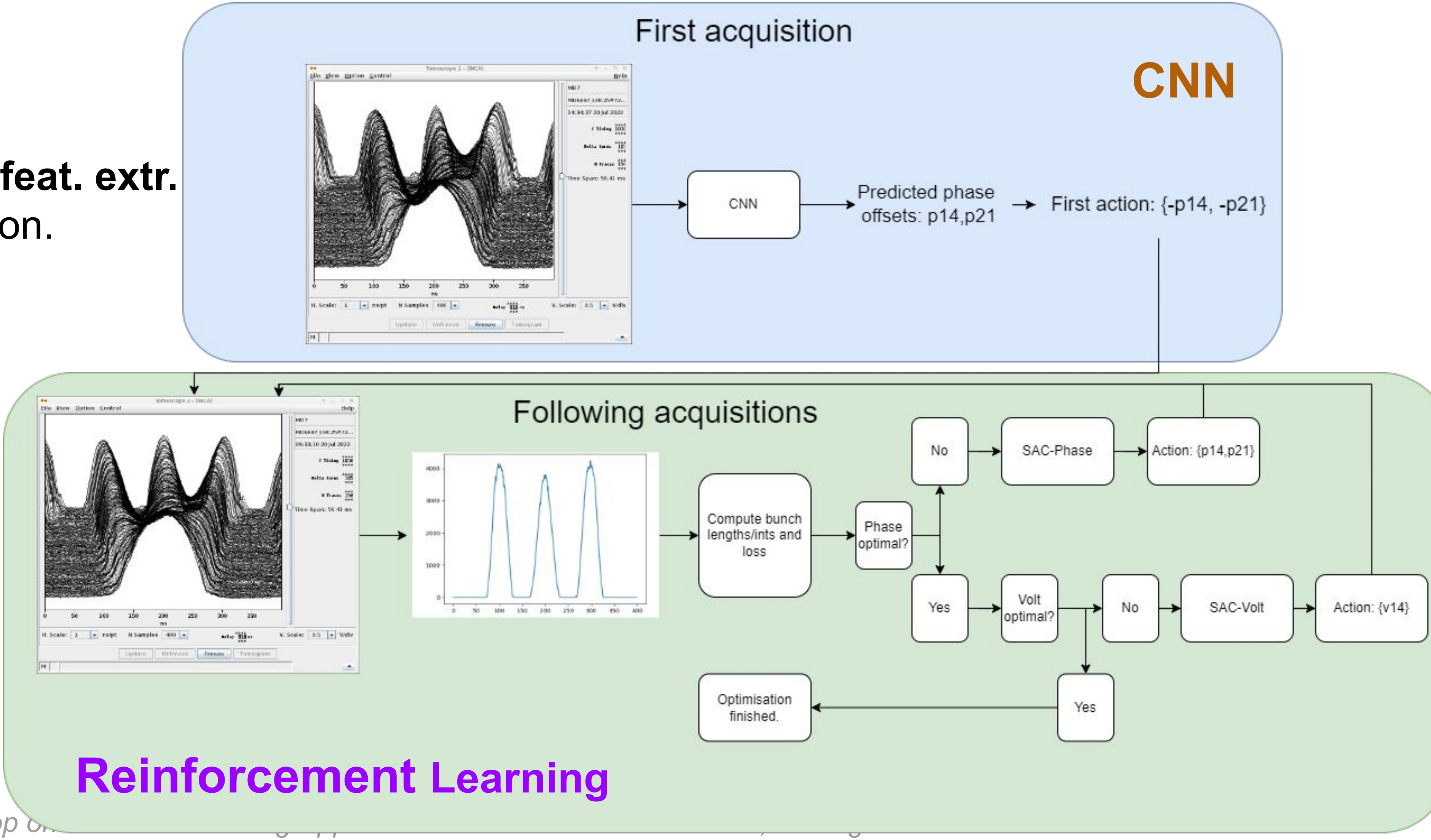
Segmented RL-Agents: Add initial guess from CNN

Feature extractor predicts phases from bunch profiles over the entire splitting (more info.)
→ can identify errors earlier in the bunch splitting otherwise not visible in the final profile,
→ is usually within 3-10 degrees of the true offset when predicting phase,
→ **can provide an initial guess leading to a better initial condition for the RL agents!**



Segmented RL-Agents: Final setup

First:
→ **Initial *phase* step from feat. extr.**
Provides good initial condition.



Followed by:
→ **SAC-phase optimizes phase**
→ **SAC-Volt optimizes voltage.**

Does it work?

Part 3: Operational results

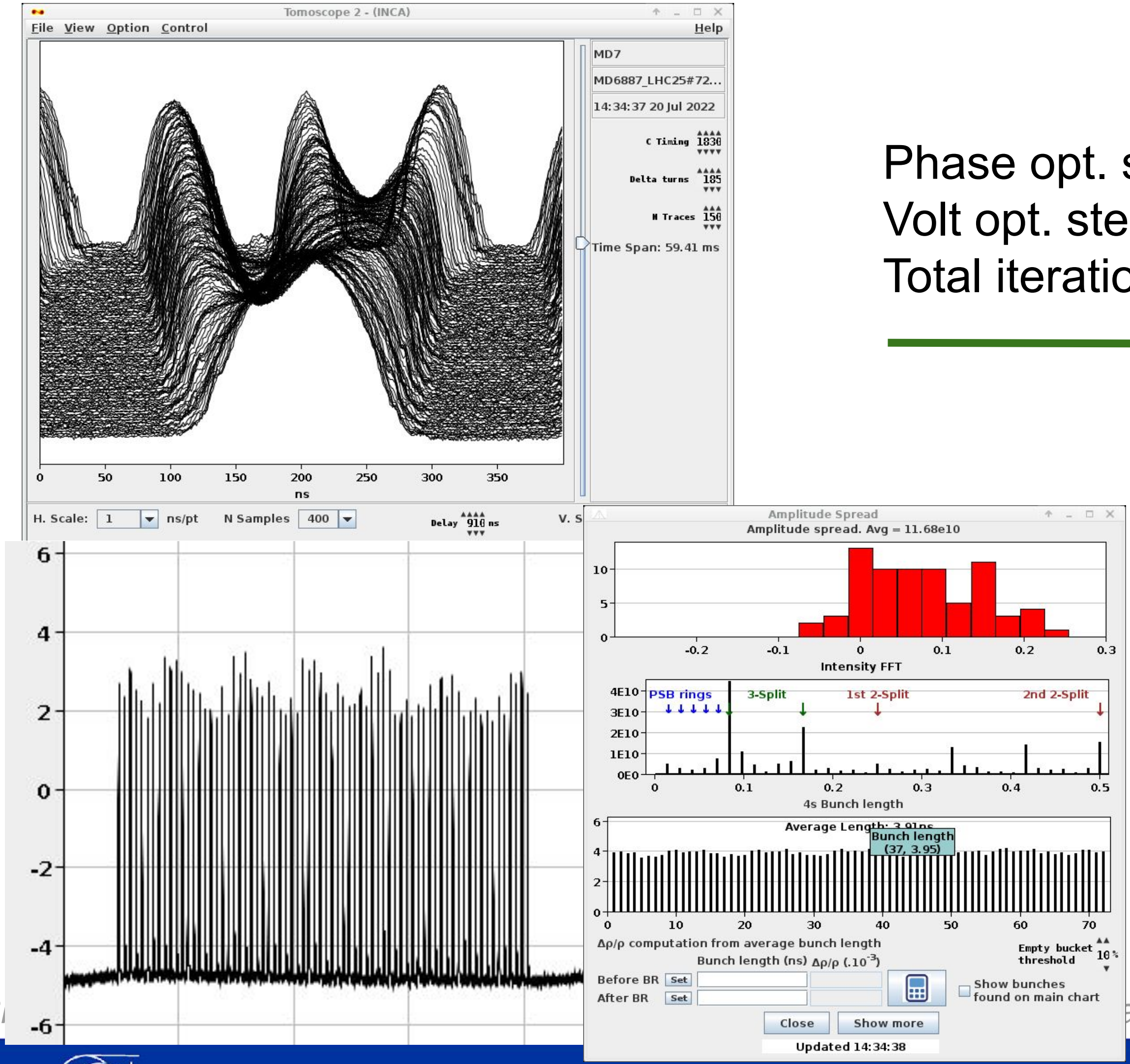
Applying simulation trained agents to the PS

Results: SAC-Phase/Volt-Sim2Real + Feat. extr.

Example episode:

Approx. initial offset: ϕ_{14} = 10, ϕ_{21} = -20, V_f = 1.08

Init

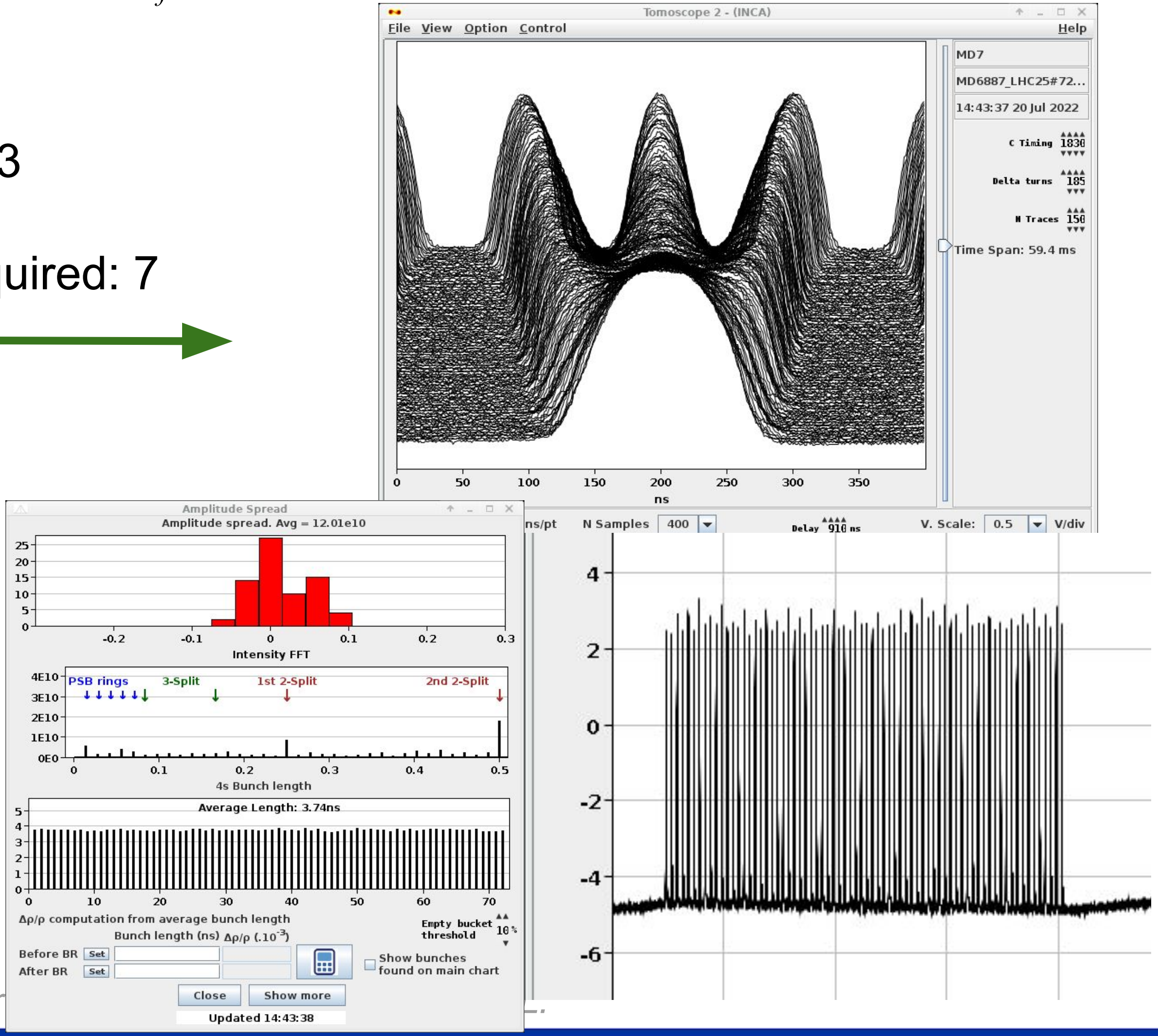


Phase opt. steps: 3

Volt opt. steps: 4

Total iterations required: 7

Final



Results: SAC-Phase/Volt-Sim2Real + Feat. extr.

- **<u>26 Full episodes collected</u>**
  - 15 using LHC25ns 72b beam at 1.3e11 int/bunch (nominal beam),
  - 2 using LHC25ns 72b beam at 1.6e11 int/bunch (higher intensity),
  - 4 using LHC25ns 72b beam at 2.6e11 int/bunch (intensity for HL-LHC),
  - 5 using BCMS 48b beam,
    - **<u>All episodes successful!</u>**

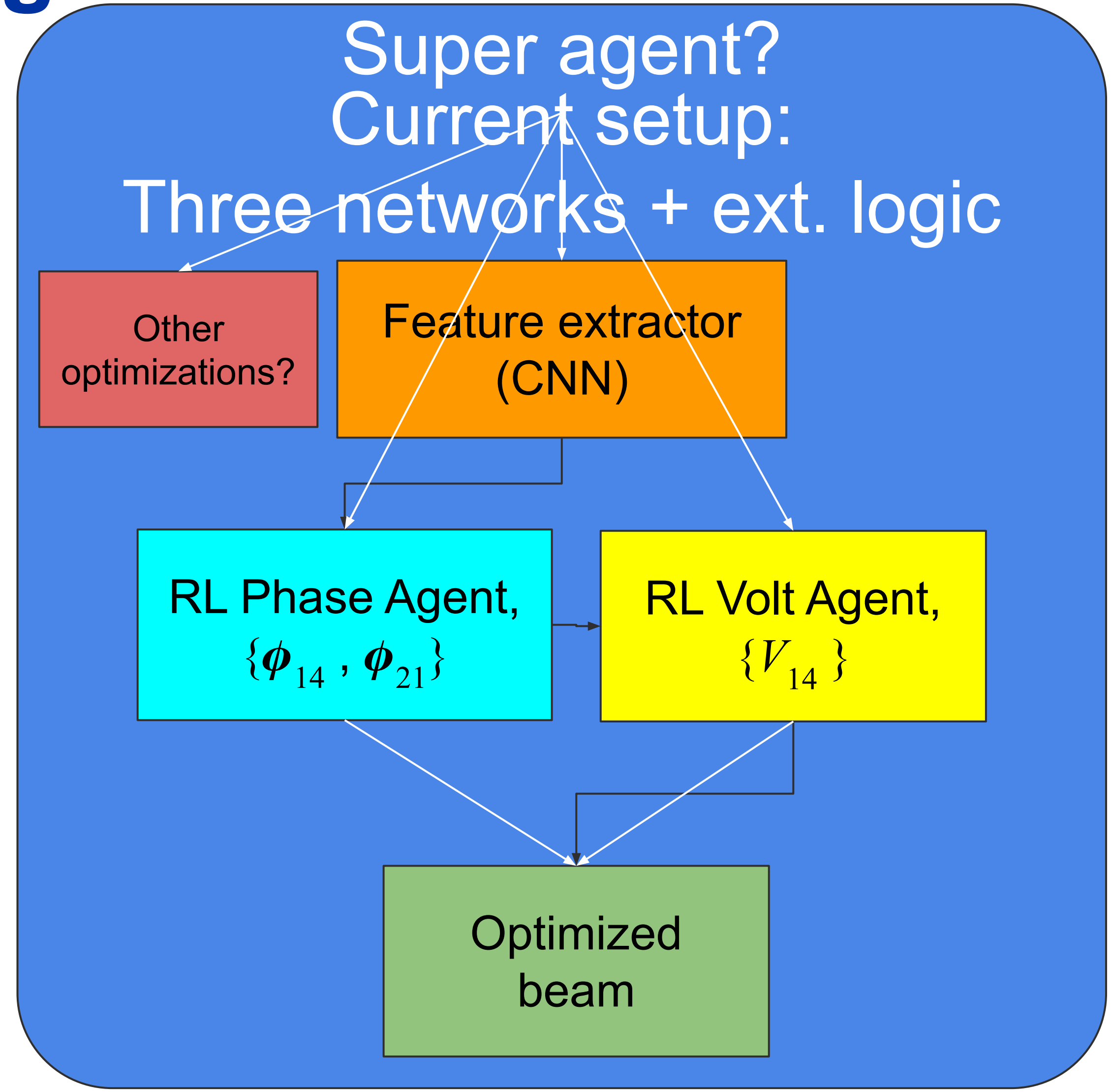
- Steps required for optimization:
  - Minimum: 2
  - Maximum: 18
  - Mean: 8.46
    - Note: number of steps required influenced heavily by initial state and restrictions on actions by agents.

Conclusion: Triple splitting agents

- **Consistent good performance** for
  - varying bunch intensities (1.3e11-2.6e11)
  - different beam types (72b, BCMS)

- **Consistently rivals operators/experts** in optimization steps
  - Averaging ~8.5 steps per optimization (with difficult initial conditions).

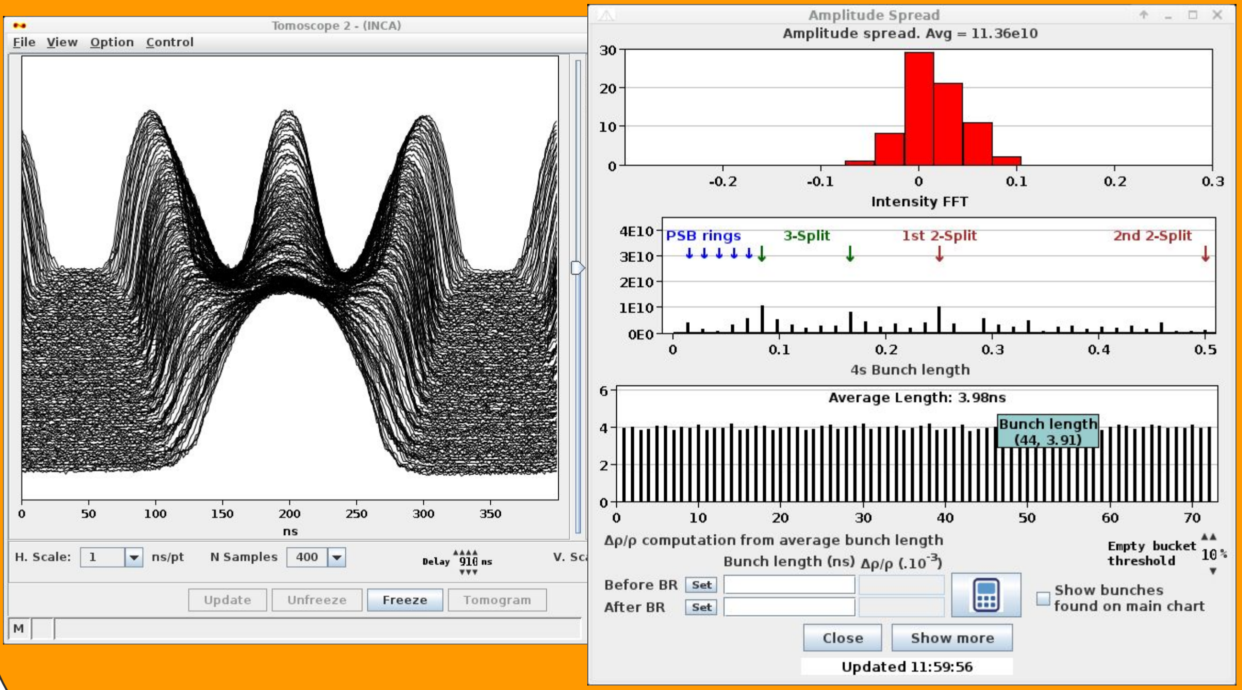
- **Future work**
  - Inclusion of **multi-bunch information**.
  - Finetune/retrain on **real data**
  - Other RF manipulations?
  - Investigate Hierarchical RL



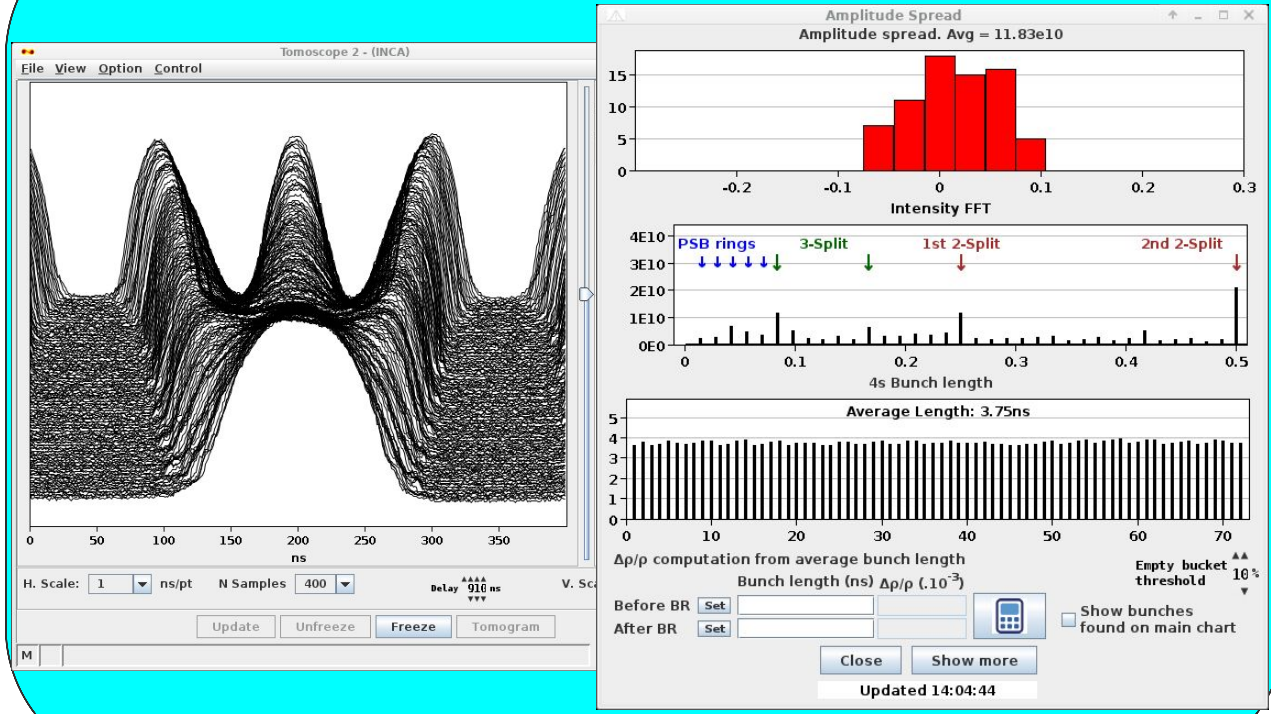
Thank you for listening! Questions?

More examples of splittings after optimisation:

Trisplit episode: Init {15,-5,0.96}. Optimised in 10 steps.



Trisplit episode: Init {10,-5,1.04}. Optimised in 8 steps.



Links and contact information

Additional information available in:

- Progress with RL for controlling RF manipulations in the PS, J. Wulff, 2022 ML community forum
- Reinforcement learning applied for RF manipulations in the PS, J. Wulff, 2021 ML Coffee
- Summer student technical note
- Optimization of RF manipulations in the PS, A. Lasheen and S. Johnston, 2020 ML Coffee

Contact information

Authors: Joel Wulff, joel.wulff@cern.ch or joel.wulff@hotmail.com,
Alexandre Lasheen, alexandre.lasheen@cern.ch

home.cern

Extra slides

Following this slide you will find many extra slides containing additional information for the interested individual. The order of them may be a bit confusing, but they could contain some interesting information for those of you who are extra interested.

For example, you can find some results from the agents used on the more simple quadruple splitting.

Cheers,
Joel Wulff

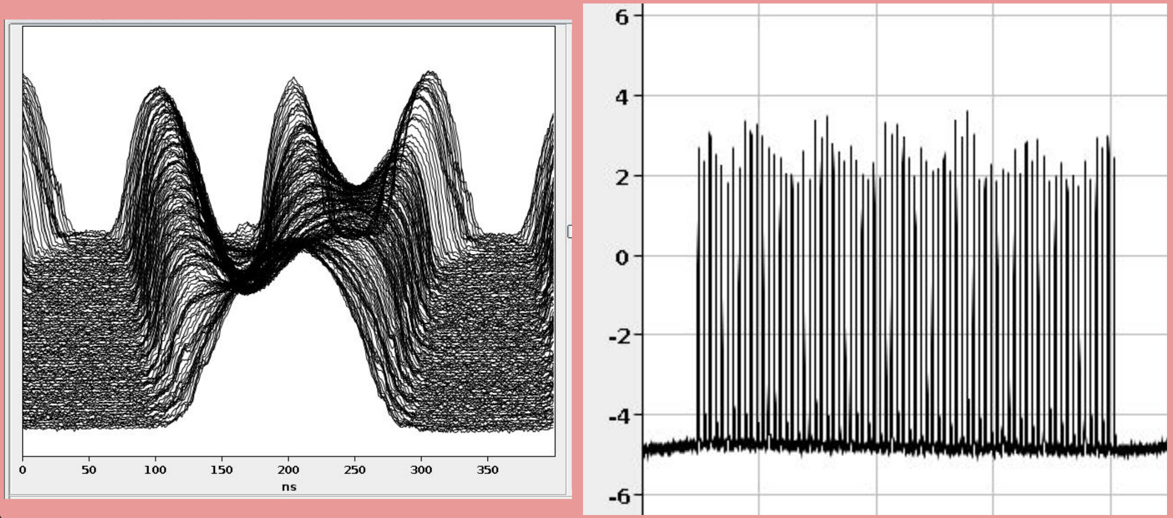
Extra results: 11 sample episodes

Episode	Init settings [*p14,p21, v14_offset*]	Phase opt.	Voltage opt.	Success
1	20, 20, -0.08	5	7	Yes!
2	-20, -20, 0.10	8	3	Yes!
3	-20, 20, 0.10	8	3	Yes!
4	20, -20, -0.01	9	5	Yes!
5	15,-5,-0.05	3	6	Yes!
6	15, -5, 0.05	1	1	Yes!
7 (2.6e11)	10, -10, -0.10	8	10	Yes!
8 (2.6e11)	-10, 10, 0.10	2	2	Yes!
9 (2.6e11)	10, 10, -0.05	5	10	Yes!
10 (BCMS)	10, -10, 0.05	3	2	Yes!
11 (BCMS)	-10, 10, -0.05	2	3	Yes!

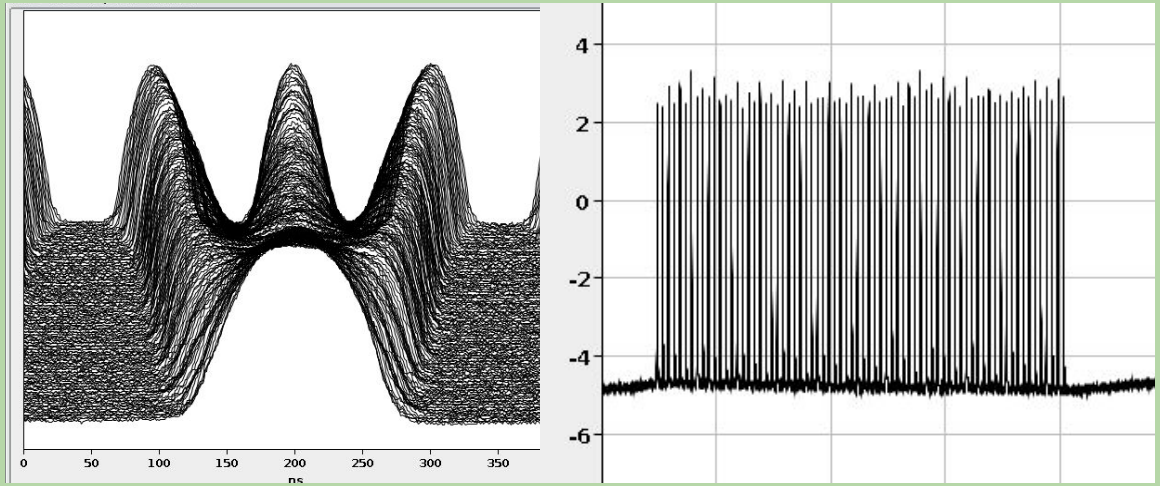
Figure: Start and end criterion for episodes 1-11. Computed using full profile loss, i.e. comparing all three bunches.

Extra: Examples of poor/optimized triple splitting

Poor splitting:
Uneven bunch characteristics, large variations along final bunch train



Optimized splitting:
Even bunch characteristics, small variations along final bunch train



Extra: Judging splitting quality, the loss function

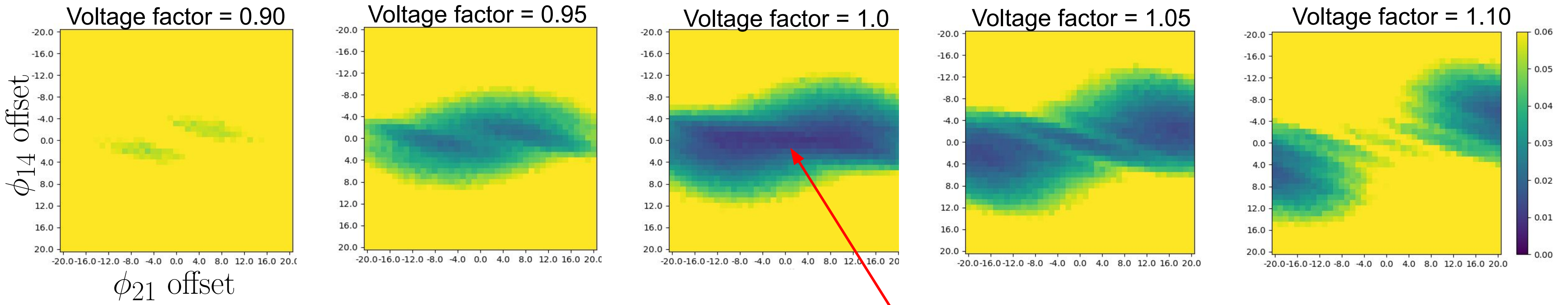


Figure: Clipped losses for different fixed voltages: when voltage is changed, optimal phase also changes.

- Scan of the three-bunch loss values while varying phase errors at **fixed** voltages
  - Shows how the "optimal" phase varies with the voltage setting.
  - Compare with phase loss on next slide!

Note: the "true" minimum over these different settings is still located in voltage factor 1.0 and phases 0, 0, as expected.

Extra: The phase loss, scanning phases for set voltages

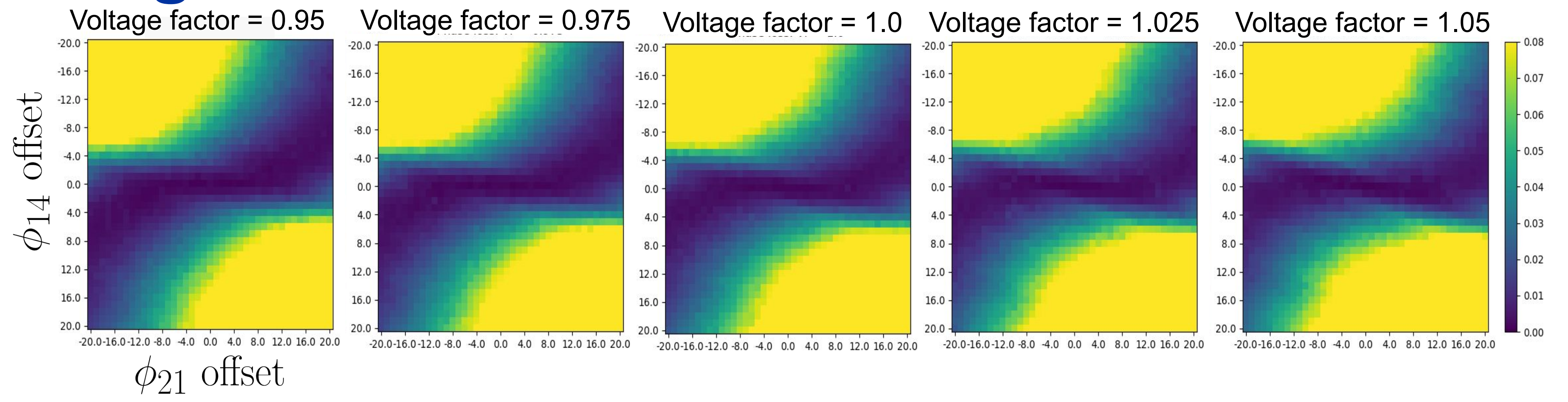


Figure: Clipped phase losses for different fixed voltages: when voltage is changed, optimal phase also changes.

- With the phase loss function, we no longer see the same variation in the loss landscape when varying voltage: as expected, the loss is (semi-) voltage agnostic.

Note: The quality of the triple splitting is much more dependent on the p14 phase setting than the p21.

Logbook entry with some initial/final states: http://elogbook.cern.ch/eLogbook/eLogbook.jsp?shiftId=1120696

Extra: Plots of phase/voltage optimization in example episode

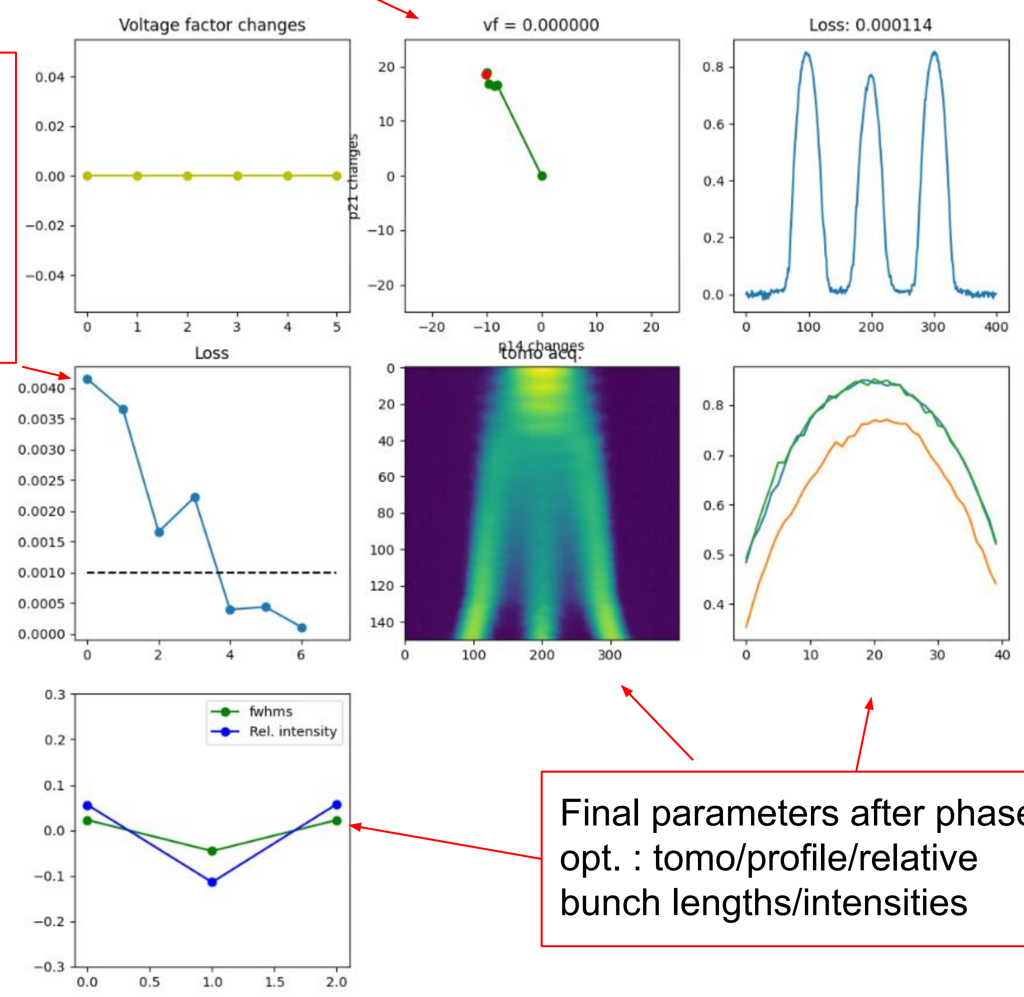
Example episode:
Approx. initial offset: p14 = 10, p21 = -20, vf = 1.08

Phase path: actions taken

NOTE: First step no action taken.

Phase loss during steps

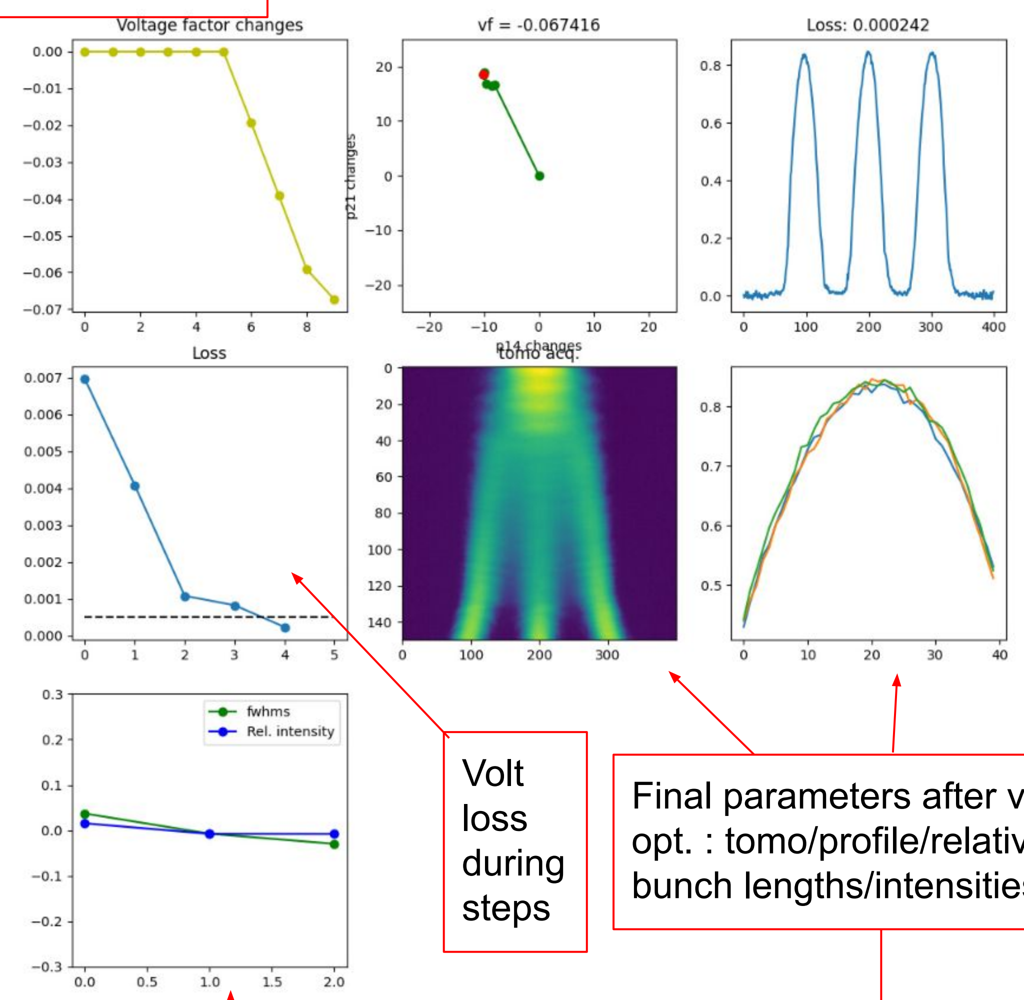
Phase optimisation



Final parameters after phase opt. : tomo/profile/relative bunch lengths/intensities

Voltage path: actions taken

Voltage optimisation



Volt loss during steps

Final parameters after volt opt. : tomo/profile/relative bunch lengths/intensities

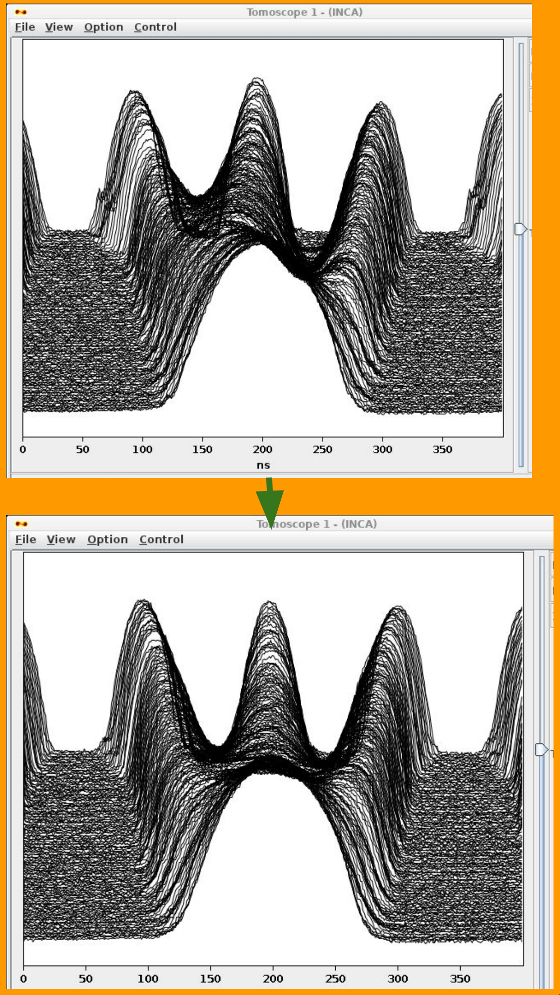
Example: Segmented RL-Agents only (no init. guess from CNN)

- Three initial episodes ran with the setup described in slide __
  - Two successes, one failure.
  - Generally slower than desired (>10 steps).

Episode	Init settings [p14,p21, v14_offset]	Phase opt.	Voltage opt.	Total steps	Comment	Success
1	-15,5, -0.07	12	3	15		Yes!
2	20,-20,-0.10	22+	-	n/a	Did not finish. Failed to optimise phase to a good degree.	No.
3	10,-10,-0.10	10	12	22		Yes!

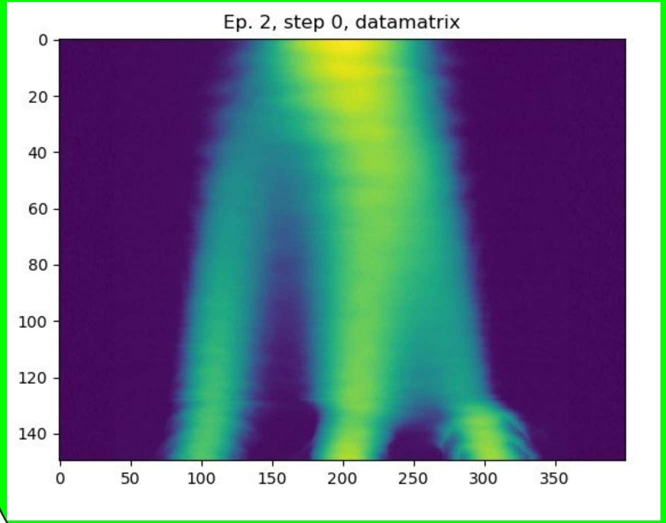
Why did the agents fail in this episode?
→ Explored in next slide

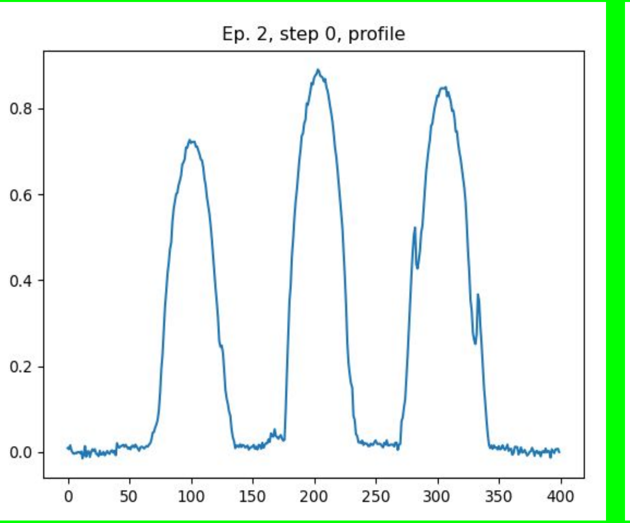
Figure: Init and end tomoscope acq. of ep. 1.

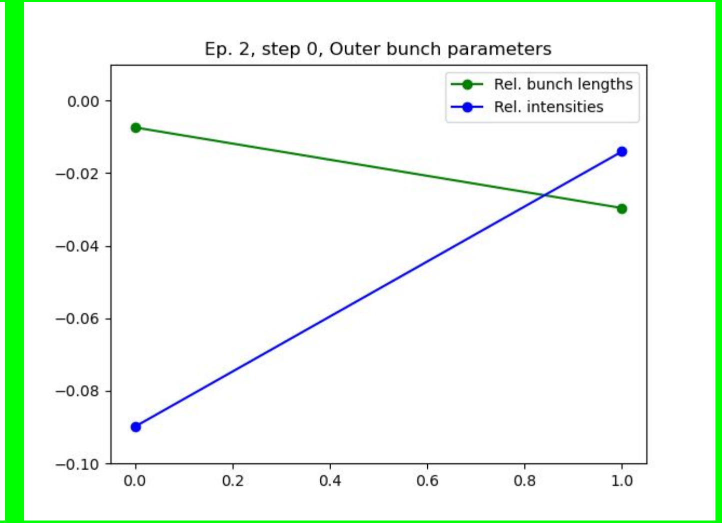


Example: Segmented RL-Agents only (no init. guess from CNN)

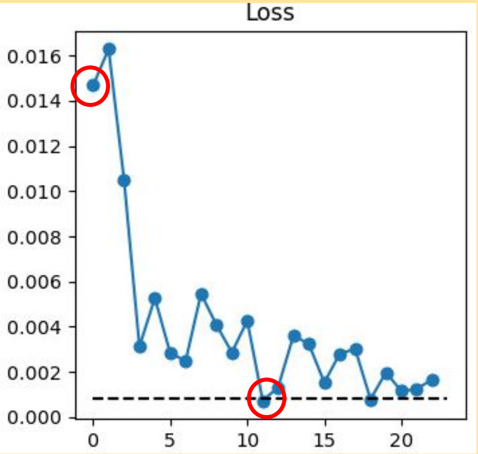
Initial acquisition: Start offset 20, -20, -0.10. Initial tomo looks very poor, final profile looks less poor.



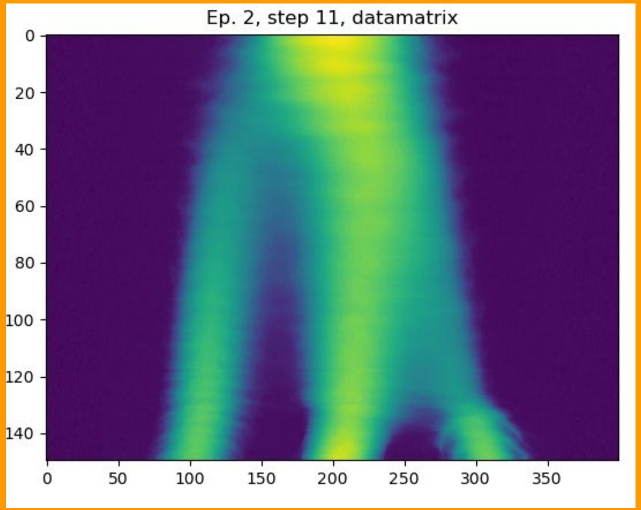


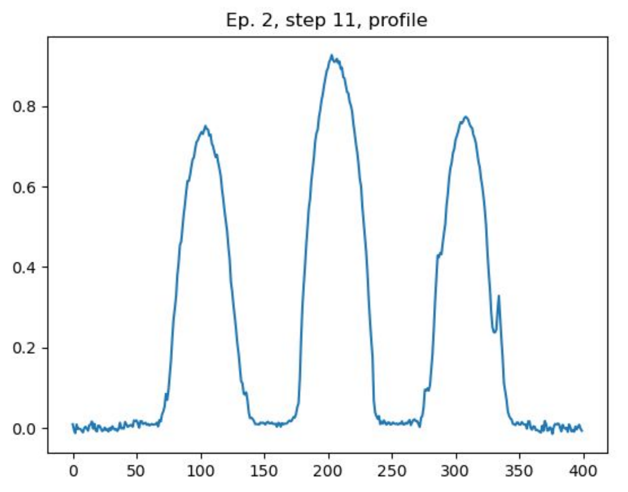


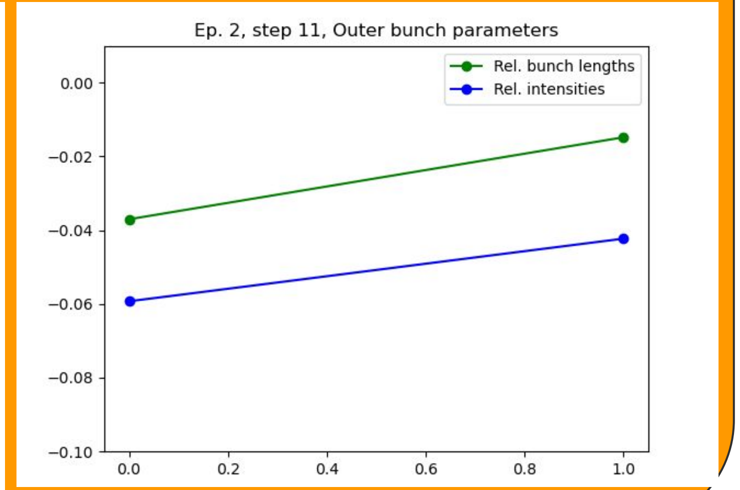
Phase loss during optimisation: We will look closer on steps 0 and 11.



Step 11: Phase loss below criteria, final profile has similar outer bunches. Rel. bunch lengths/intensities also similar. But, tomo acquisition shows large phase error remains → Error in observable!







Agent believes it is close to the minimum, but is actually far from it. A special case where the agent can get stuck!

Example: Segmented RL-Agents only (no init. guess from CNN), conclusion

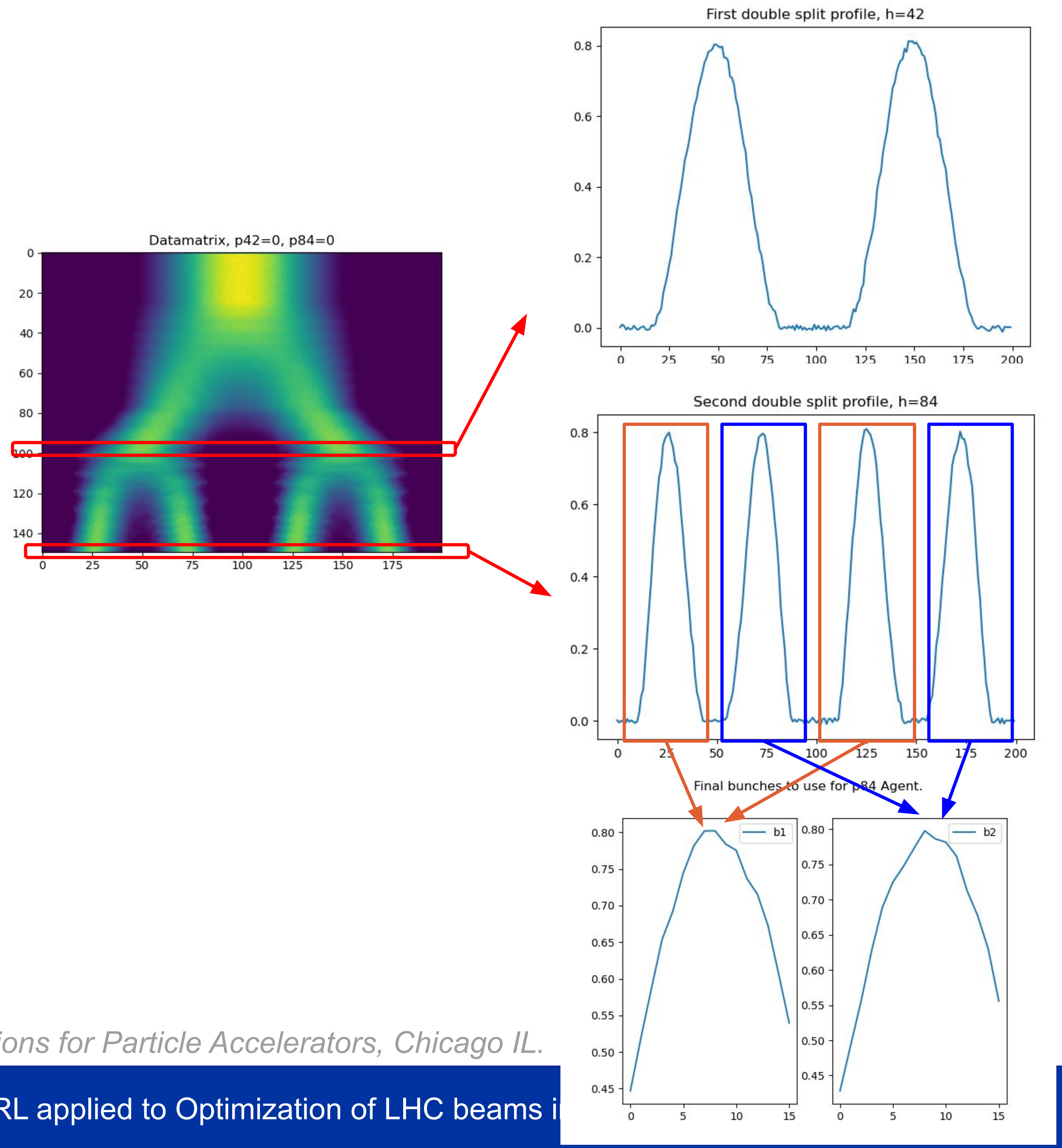
From this example, we know that the phase loss is not perfect:

- If the initial condition is too poor, the information contained in the final profile may not be enough to solve the problem → **The Agents may converge to a local minima.**
- Could we exploit the information in the full tomoscope acquisition in some way to achieve a better initial condition, where the final profile contains adequate information for the agents optimisation?
  → **Yes, by using the pre-trained feature extractor!**

Extra: Approach for Quadruple splitting

The inputs were designed to be taken from a tomoscope acq. (as the simulated data for this was already available).

- Calculates inputs and losses from profiles at different timings
    - h=42 agent uses the profile after the first splitting is complete, but the second has yet to start.
    - h=84 uses the final profile. We only care about differences caused by the second splitting, so we average together the first + third and the second + fourth bunches respectively to get only two bunches, representing the quality of the second double split.



Extra: Quadruple splitting: Training and simulation results

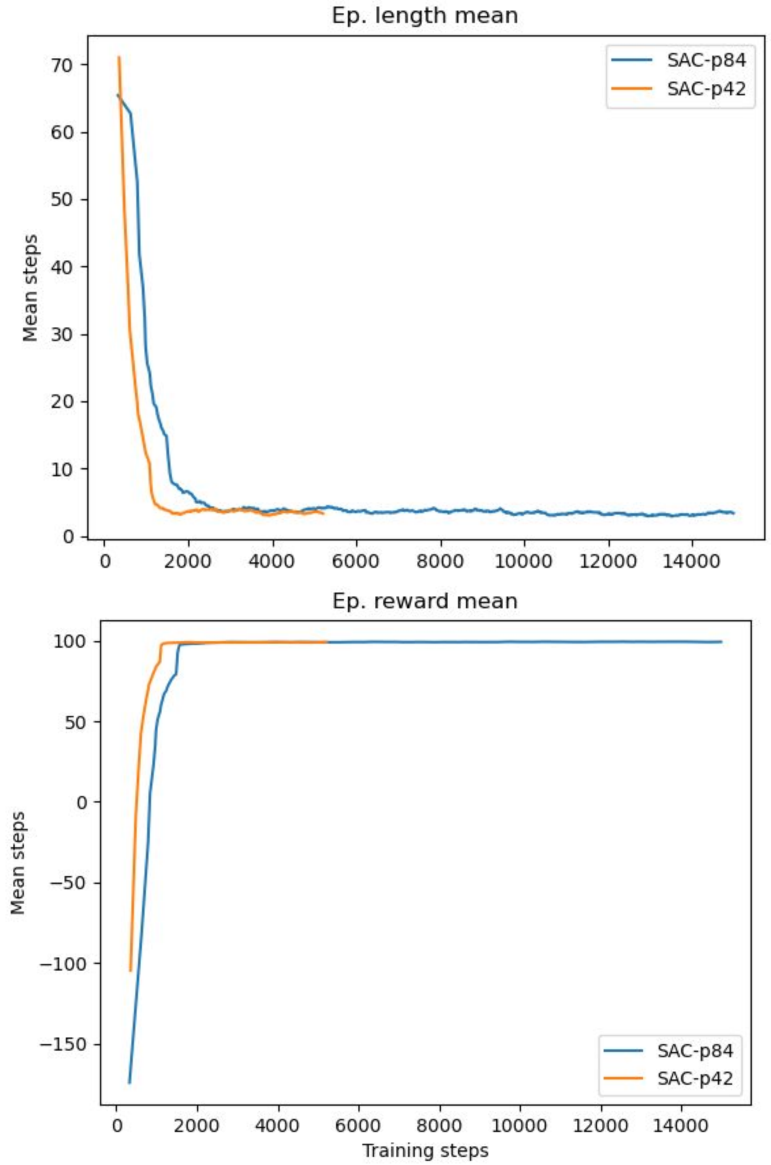
Both new models perform well in simulation

- **SAC-p42** converged to good policy in ~1.5k steps (3k before best model)
  - Optimising splitting in ~ 3.2 steps

- **SAC-p84** converged to good policy in ~2k steps (6.5k before best model)
  - Converging splitting in ~ 3.01 steps

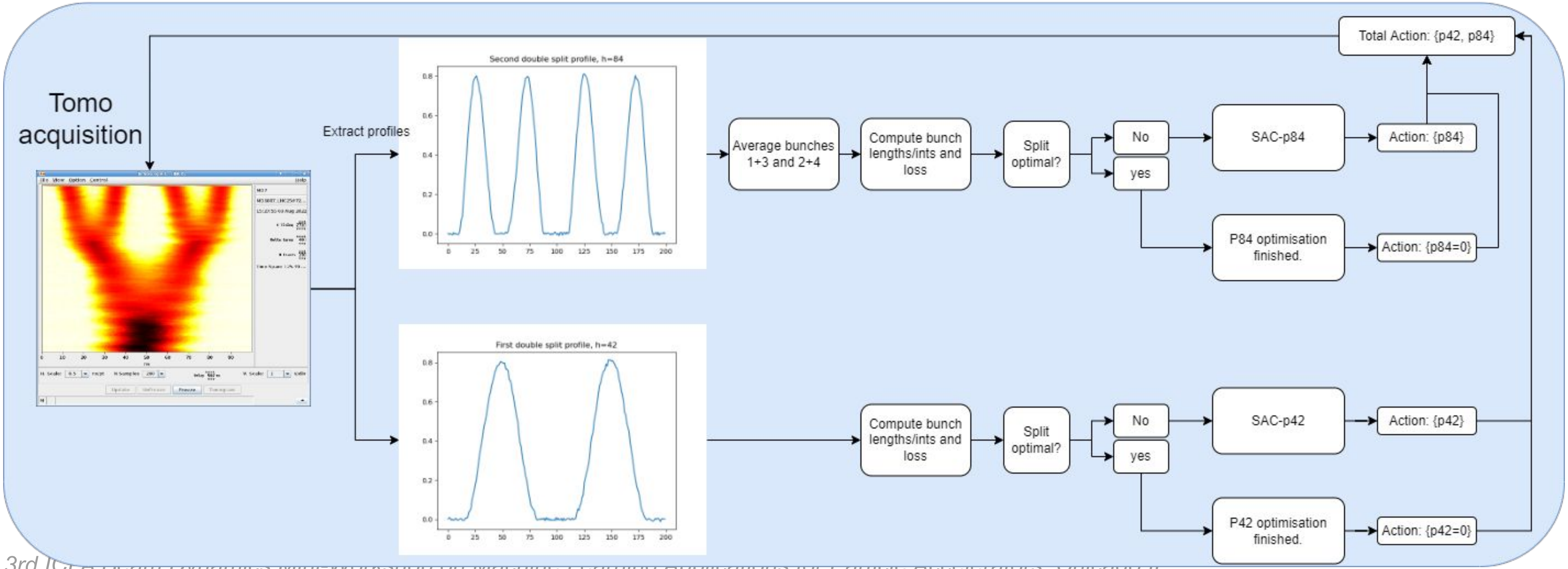
Two positives about new setup:

- Training fast enough to potentially train directly in the PS during MDs.
- No intrinsic need for full tomoscope acquisition (if the two profiles can be collected in some other way).



Extra: MD setup: Quadsplit, SAC-p42/p84

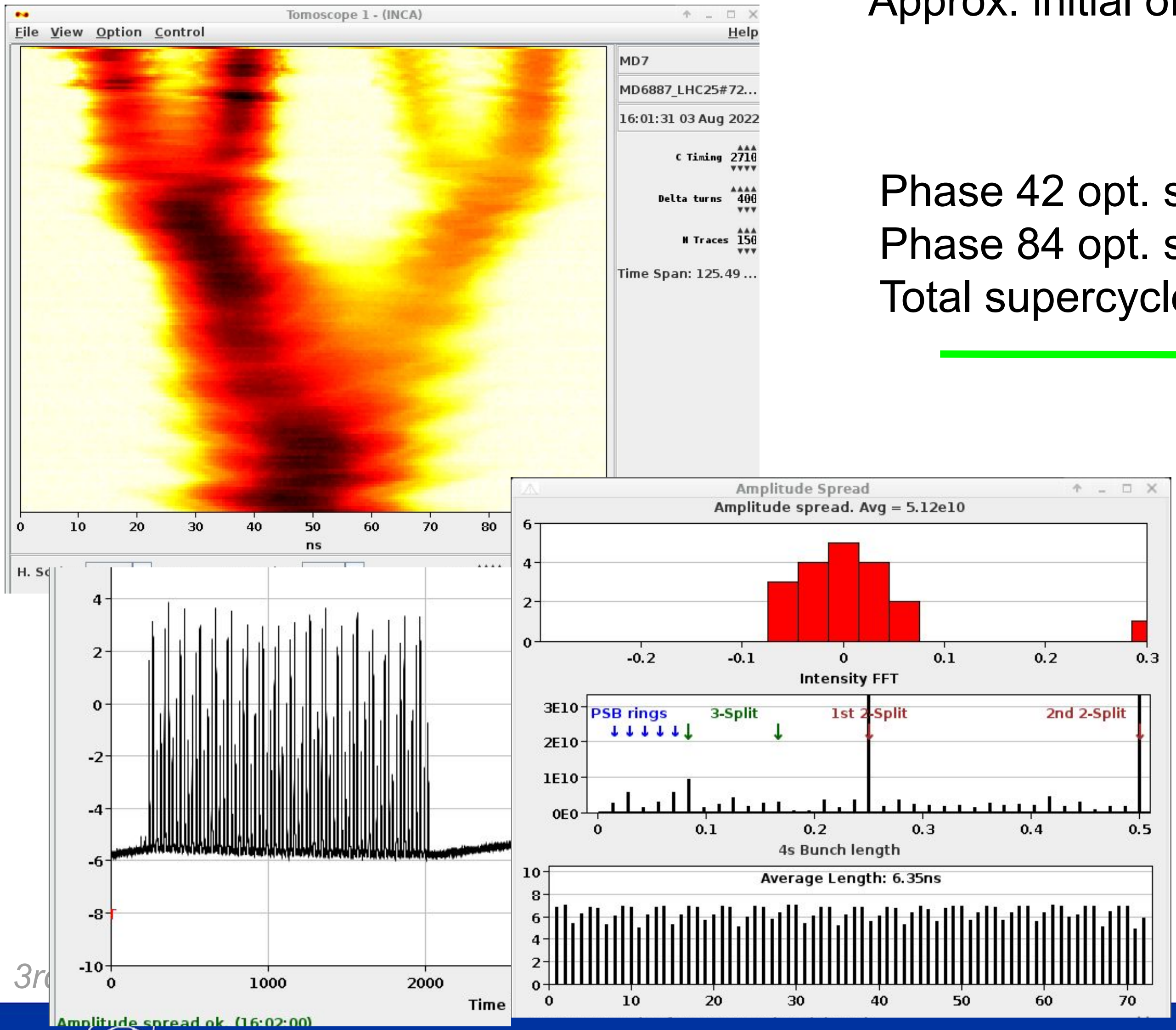
Figure: Flowchart of MD setup. Optimisations of p42 and p84 run in parallel. Optimisation finished when both splittings are optimal (at the same time)



3rd ICFA Beam Dynamics Mini-Workshop on Machine Learning Applications for Particle Accelerators, Chicago IL.

Extra: MD Result: SAC-p42/p84 (Quadsplit)

Init



Example episode

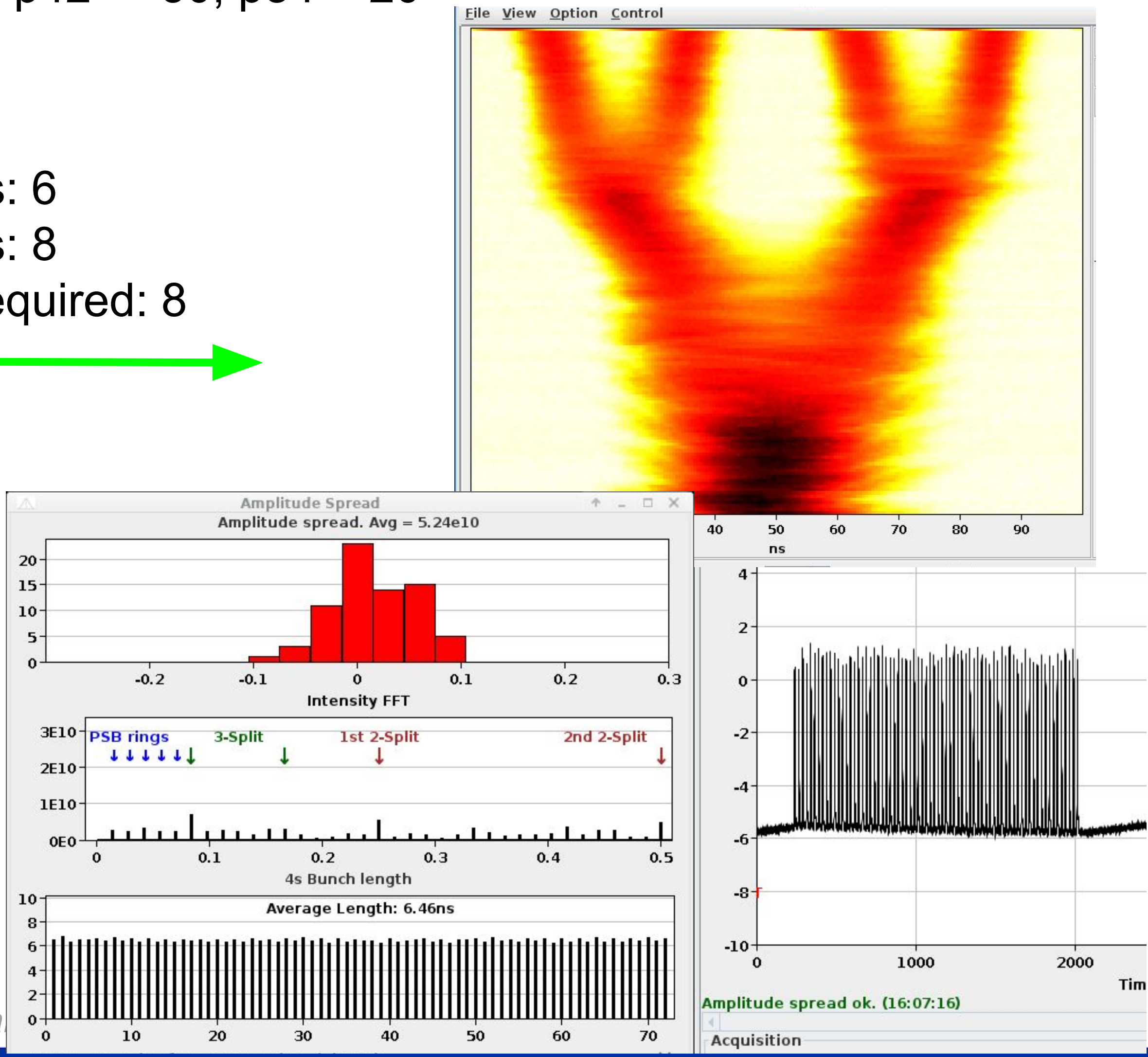
Approx. initial offset: p42 = -30, p84 = 20

Phase 42 opt. steps: 6

Phase 84 opt. steps: 8

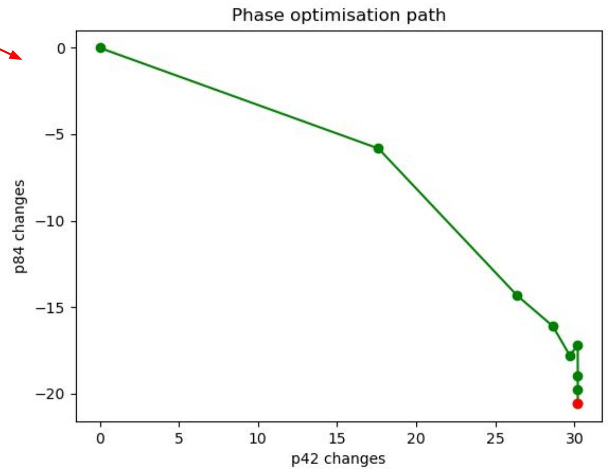
Total supercycles required: 8

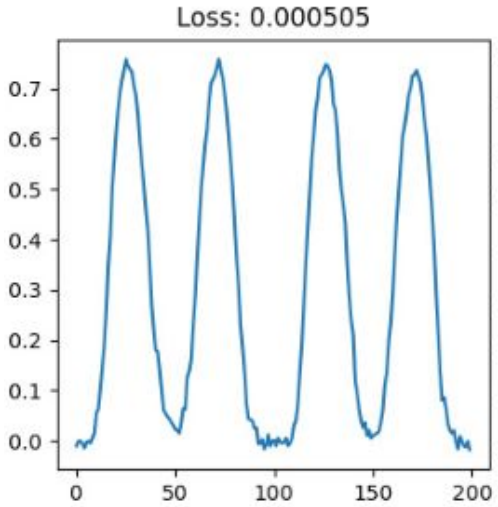
Final

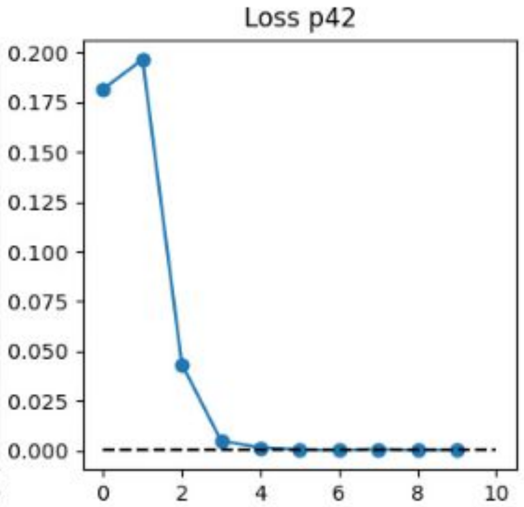


Extra: MD Result: Quadsplit, SAC-p42/p84

Phase path: actions taken





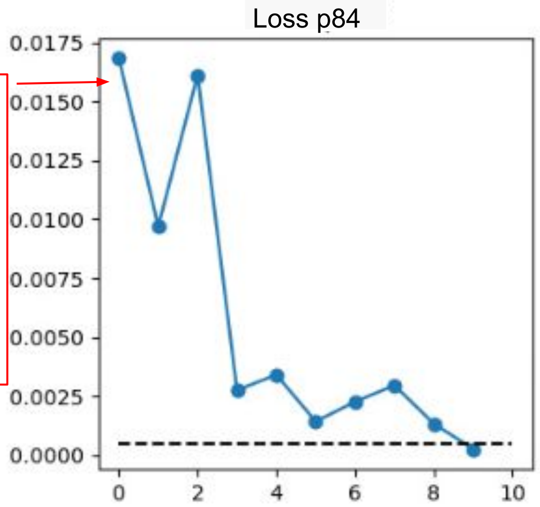


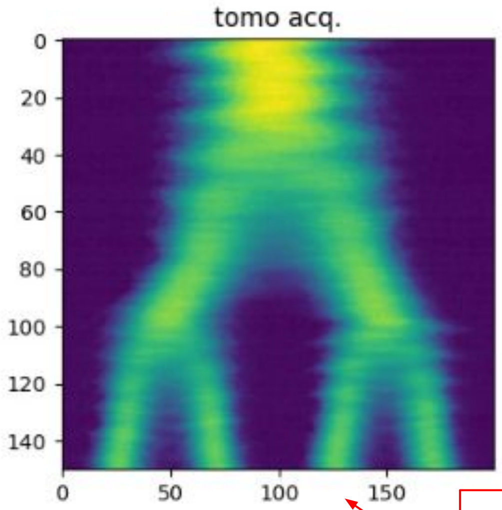
Example episode:

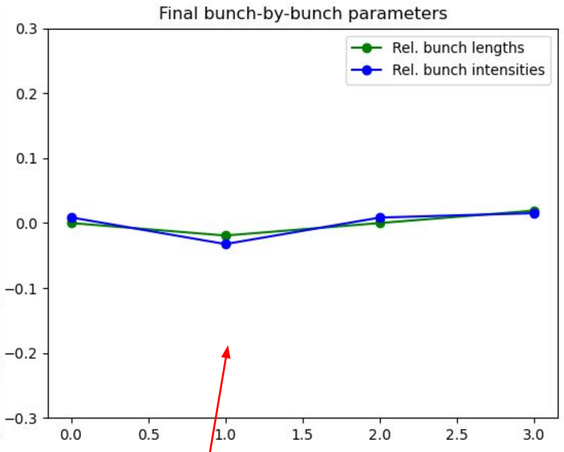
Despite a very large initial error, optimised in < 10 steps.

No need for feature extractor
→ No need for tomoscope!

NOTE: First step no action taken.







Final parameters after phase opt. : tomo/profile/relative bunch lengths/intensities

Extra: MD Result: SAC-p42/p84 (Quadsplit)

- 11 Full episodes collected
  - All using LHC25ns 72b beam at 1.3e11
    - **9 episodes reached criterion, 2 failed**
      - During the two failures the agents did not manage to reach the preset loss criterion. However, in both cases the splitting looked close to perfect on amp. spread → **Criterion may be set too low**, **not the agents**!

- Mean steps required for optimisation:
  - **SAC-p42: 4.0 steps.**
  - **SAC-p84: 4.64 steps.**
- Number of supercycles required (for both phases to be optimised):
  - **Mean: 5.27**
  - Min: 2
  - Max: 9
    - Note: Steps required influenced heavily by initial state and restrictions on actions by agents.
      - Maximum step size for the agents is 20 degrees.

Extra: MD Result: SAC-p42/p84 (Quadsplit)

Episode	Loss criterion	Init settings, [*p42, p84*]	p42 opt.	p84 opt.	Success
1	0.0008	-10, 10	3	4	Yes!
2	0.0008	10, -10	2	2	Yes!
3	0.0005	15, -15	1	2	Yes!
4	0.0005	-15, 15	3	3	Yes!
5	0.0005	-10, -10	4	3	Yes!
6	0.0005	-10, -10	(3) n/a	(2) n/a	No.
7	0.0005	20, -5	(2) n/a	(4) n/a	No.
8	0.0005	-12.5, 30	6	9	Yes!
9	0.0005	-30, 20	5	8	Yes!
10	42: 0.0006 84: 0.0008	5, -5	3	5	Yes!
11	42: 0.0006 84: 0.0008	-5, 5	5	9	Yes!

Reached criterions after (_) steps, but forced to continue until 5 steps. Then, did not reach criterion within 10 steps. Splitting looked good.

Figure: Start and end criterion for episodes 1-11. P42 and P84 losses shown separately.

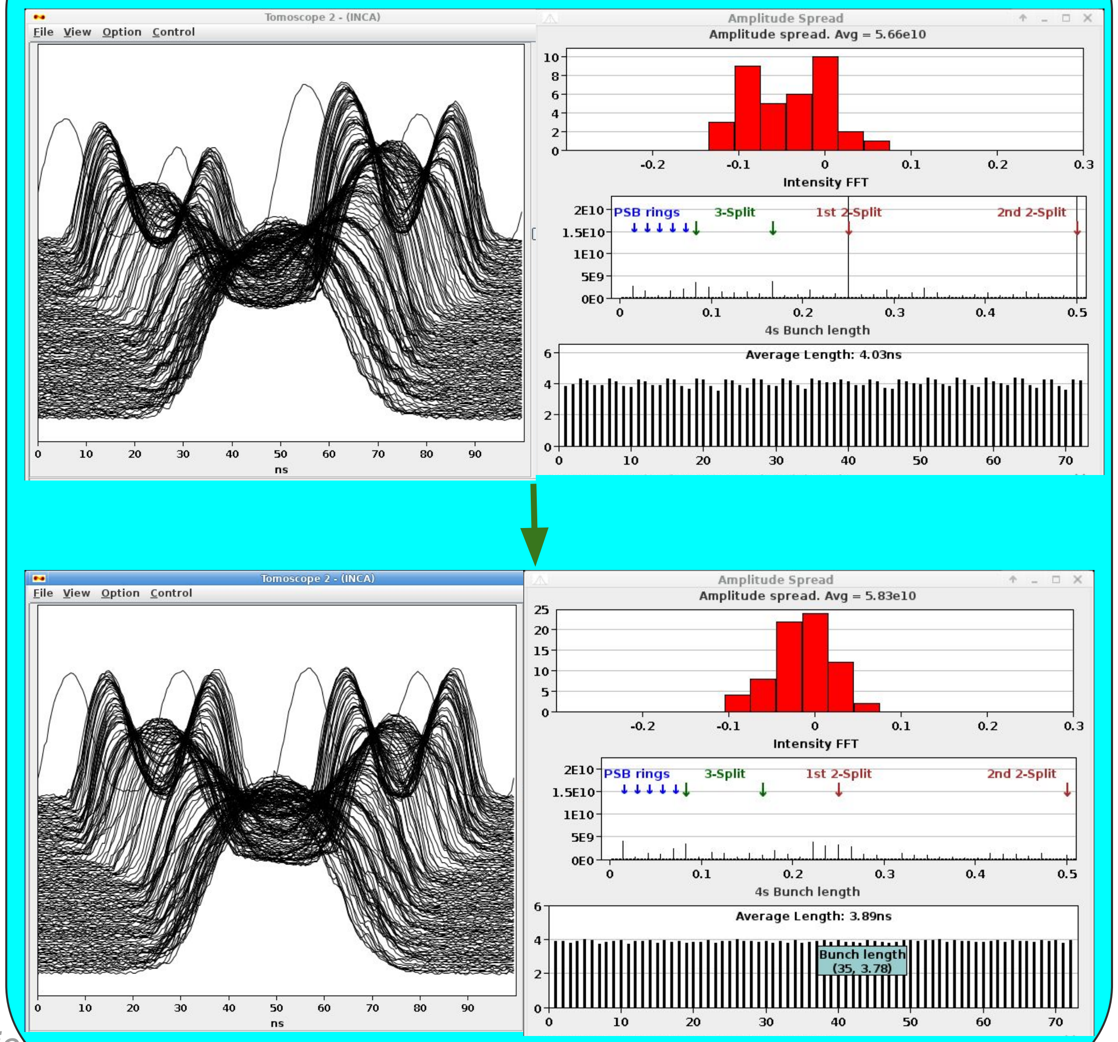
Extra: Conclusion: Quad splitting agents

- Good performance so far
  - Investigate criterion value for optimal splitting
  - Test consistency across beams/intensities

- Averaging ~5.27 steps per optimisation (of both phases, depending on initial conditions).

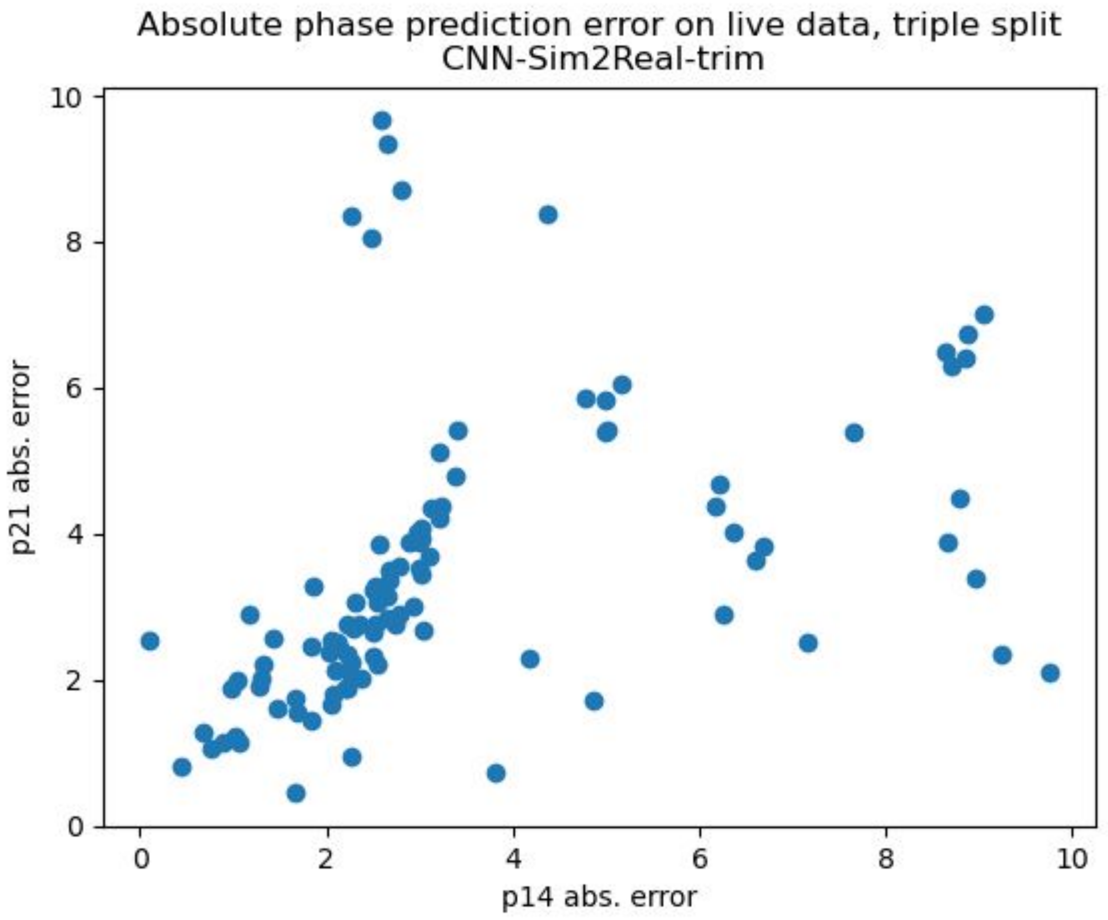
- Future work
  - Test using feature extractor
  - Benchmark against mathematical optimisation
    - Setup constructed simple enough for easy implementation in GeOFF.
    - Is RL overkill in this case?
  - **Offline RL**?
  - **Collection of labeled data**
  - **Hyperparameter** tuning of RL agents.

Init {15,-15}. Optimised in 2(!!) steps.

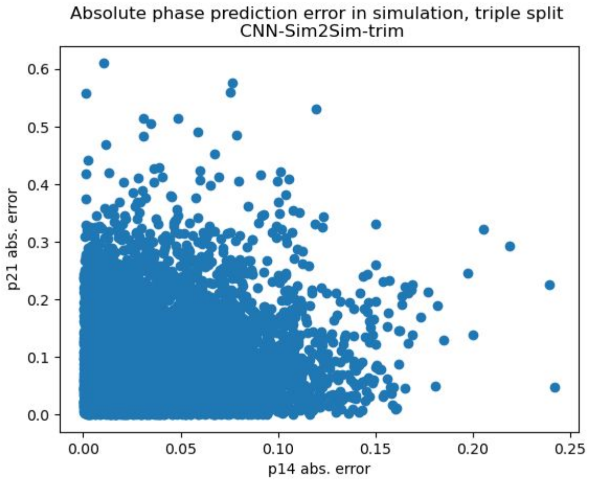


Feature extractor performance on real trisplit data

- Problem: CNN fails to generalise and is not accurate on real data.
  - Error is however most often only ~3 degrees, which means it can improve on large phase errors.
  - However, finetuning of phase becomes difficult. The agent is pre-trained with an almost perfect CNN, and trusts it too much



Compare with simulation accuracy <1 degree.



Extra: Feature extractors, gathering data

No dataset of real labeled acquisitions with available
→ **One had to be simulated.** Cern developed code **BLonD** was used.

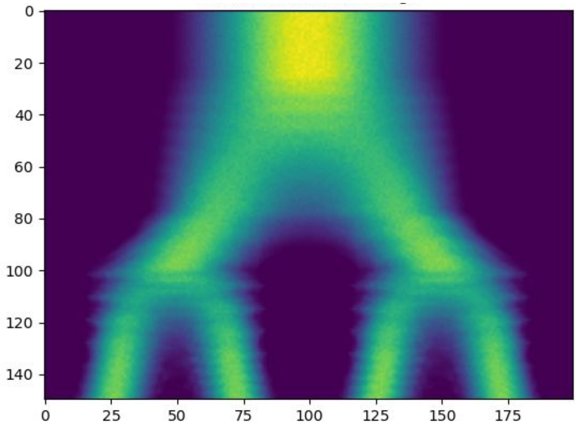
When creating this simulated data, much care was taken to resemble live acquisitions to reduce the sim/real domain gap:

1. **Adapting the resolution (ns/pt), number of traces, and timings to match those of the normal Tomoscope references for the quadsplit/trisplit.**
2. **Updated voltage programs** of simulation by acquiring the latest ones from the LSA settings (quadsplit) or acquiring a **reference of the design voltage** (trisplit).
3. **Added several data augmentation** steps during training to:
    a. **Normalise the data**: the absolute values of the simulation and the detectors don't line up, so the data is normalised before being used as input.
    b. **Add noise to the data**. This is done by approximating the noise of the machine by adding some Gaussian noise to the simulated data.
    c. **Moving the initial injection center** of the bunch +- a few ns. This is done as sometimes the beam jitters slightly compared to the tomoscope position, and we want the feature extractor to be agnostic to this.

Extra: Feature extractors, the datasets

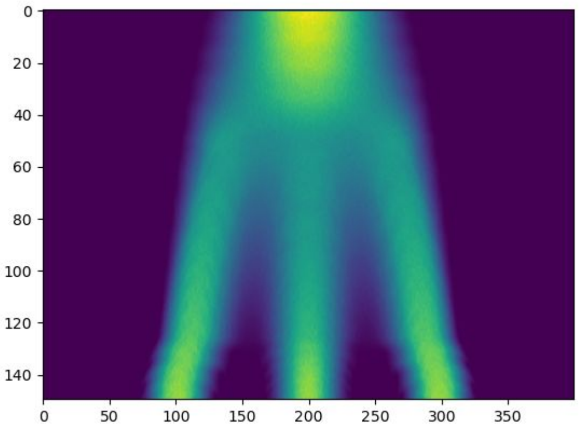
- **The Quad dataset:**
  - Scan of absolute phase errors in range $\phi_{h=42}$, $\phi_{h=84}$ **= [-30,30]**.
  - A total of 14641 samples in dataset.
- **The Tri dataset:**
  - Scan of absolute phase errors in range $\phi_{h=14}$, $\phi_{h=21}$, **= [-20,20]**, and voltage factors for *h=14* in range $v_{h=14}$ **= [0.95, 1.05]**.
  - A total of 59541 samples in dataset.
- Each sample stores **the entire datamatrix** of traces along with **the label** of the offset used to simulate it.
- A 9:1 training/validation split was used.
- Note: These same datasets are used for training of RL agents later, but then only extracted features such as end bunch-by-bunch length/intensities are given to the agents.

Datamatrix (150x200)



Label = [0, 0]

Datamatrix (150x400)



Label = [0, 0, 1.0]