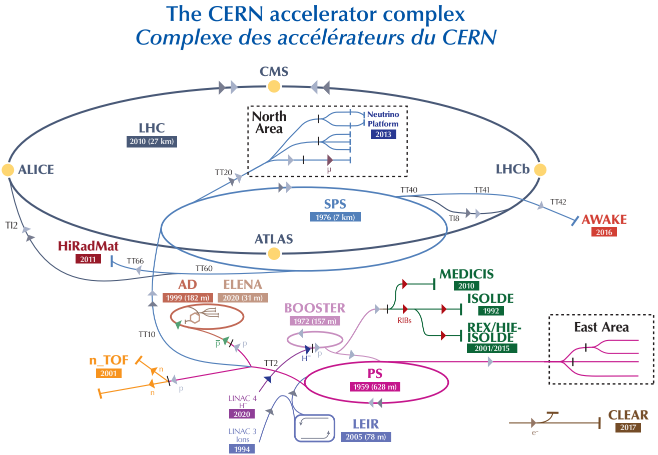
Machine Learning for Accelerators at CERN

A EURO-LABS Perspective

N. Madysa (CERN)

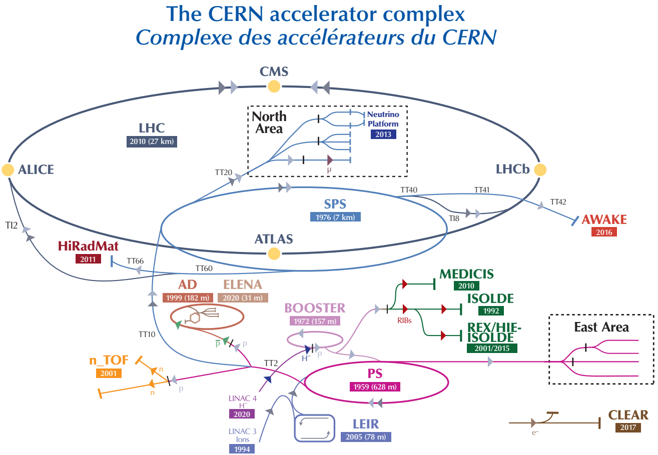
GANIL Community Meeting,
20 October 2022

- many accelerators, extremely diverse



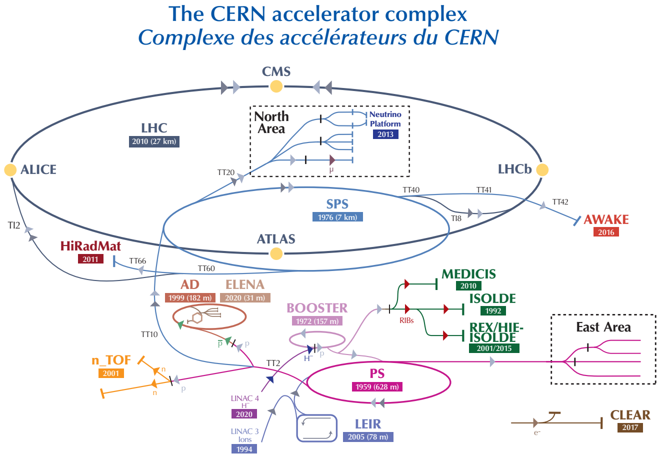
CERN

- many accelerators, extremely diverse
- uniform communication protocol (JAPC)



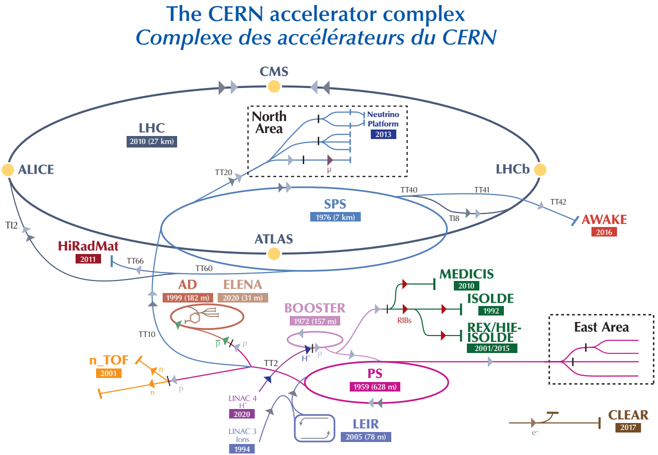
CERN

- many accelerators, extremely diverse
- uniform communication protocol (JAPC)
- lots of low-level problems already well automated

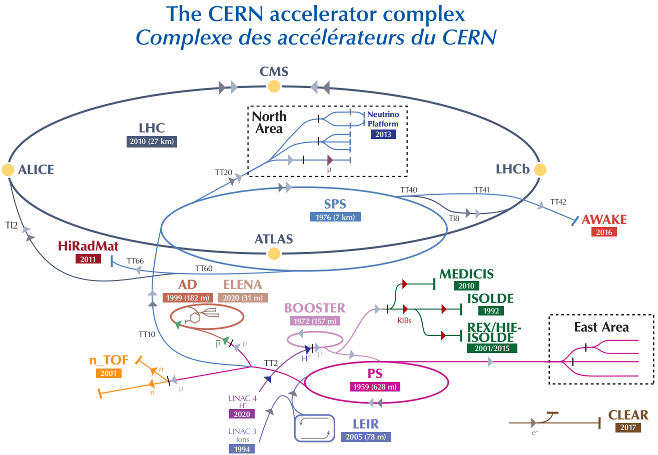


CERN

- many accelerators, extremely diverse
- uniform communication protocol (JAPC)
- lots of low-level problems already well automated
- but: many high-level problems still solved manually



- many accelerators, extremely diverse
- uniform communication protocol (JAPC)
- lots of low-level problems already well automated
- but: many high-level problems still solved manually
- better turnaround time and beam quality necessary to reach target integrated luminosity



CERN

Machine learning use cases:

- advanced modeling (supervised learning)

Machine learning use cases:

- advanced modeling (supervised learning)
- anomaly detection (semi-supervised learning)

Machine learning use cases:

- advanced modeling (supervised learning)
- anomaly detection (semi-supervised learning)
- beam scheduling (classical optimization)

Machine learning use cases:

- advanced modeling (supervised learning)
- anomaly detection (semi-supervised learning)
- beam scheduling (classical optimization)
- improved diagnostics (supervised & unsupervised learning)

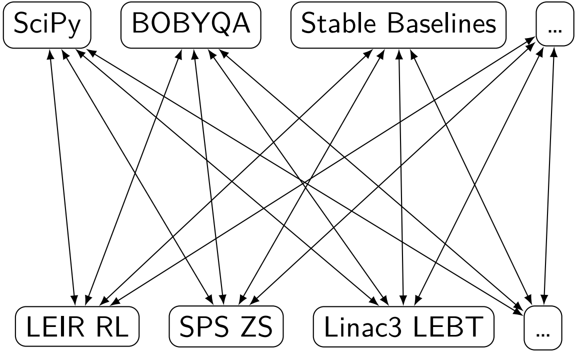
CERN

Machine learning use cases:

- advanced modeling (supervised learning)
- anomaly detection (semi-supervised learning)
- beam scheduling (classical optimization)
- improved diagnostics (supervised & unsupervised learning)
- accelerator controls (reinforcement learning, classical & optimization)

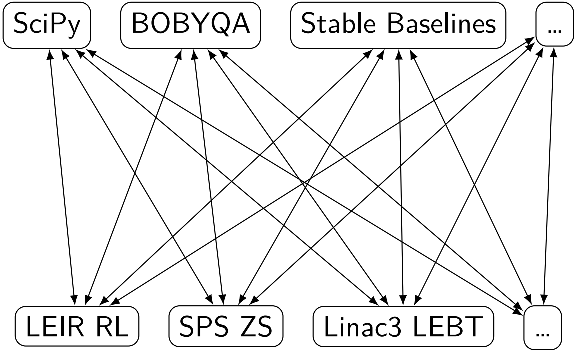
Motivation

- many different optimizers/APIs



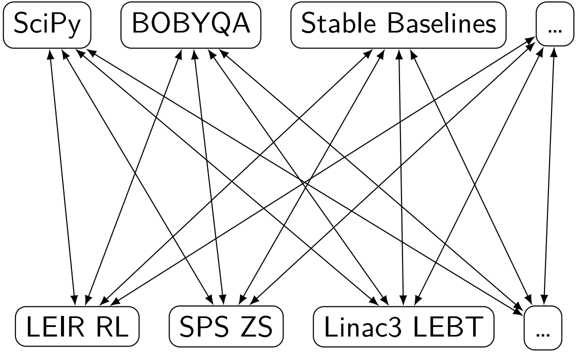
Motivation

- many different optimizers/APIs
- many different optimization problems



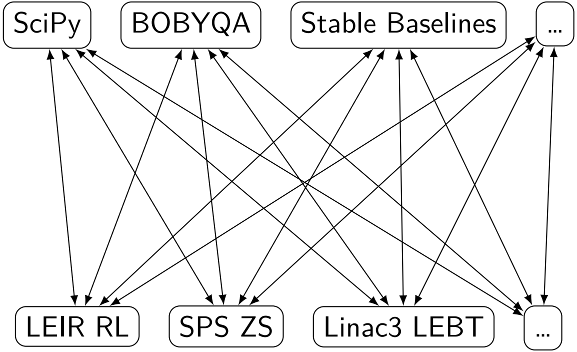
Motivation

- many different optimizers/APIs
- many different optimization problems
- each problem involves complex machine communication



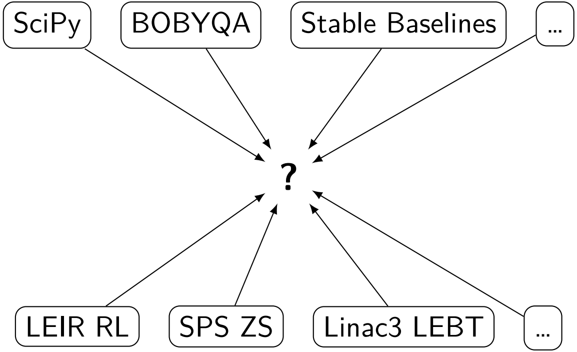
Motivation

- many different optimizers/APIs
- many different optimization problems
- each problem involves complex machine communication
- operators don't want to juggle Python scripts!



Motivation

- many different optimizers/APIs
- many different optimization problems
- each problem involves complex machine communication
- operators don't want to juggle Python scripts!



Motivation

Goals:

- provide ecosystem for accelerator optimization and control

Motivation

Goals:

- provide ecosystem for accelerator optimization and control
- provide compatibility with as many algorithms as possible

Motivation

Goals:

- provide ecosystem for accelerator optimization and control
- provide compatibility with as many algorithms as possible
- facilitate the progression
  manual tuning $\rightarrow$ *numerical optimization* $\rightarrow$ *machine learning*

Motivation

Goals:

- provide ecosystem for accelerator optimization and control
- provide compatibility with as many algorithms as possible
- facilitate the progression
  manual tuning $\rightarrow$ *numerical optimization* $\rightarrow$ *machine learning*

Guiding principles:

- be agnostic over machine, communication protocol or devices

Motivation

Goals:

- provide ecosystem for accelerator optimization and control
- provide compatibility with as many algorithms as possible
- facilitate the progression
  manual tuning → *numerical optimization* → *machine learning*

Guiding principles:

- be agnostic over machine, communication protocol or devices
- minimize boilerplate code that does not solve the problem

Motivation

Goals:

- provide ecosystem for accelerator optimization and control
- provide compatibility with as many algorithms as possible
- facilitate the progression
  manual tuning → *numerical optimization* → *machine learning*

Guiding principles:

- be agnostic over machine, communication protocol or devices
- minimize boilerplate code that does not solve the problem
- don't make people pay for features they don't use

Motivation

Goals:

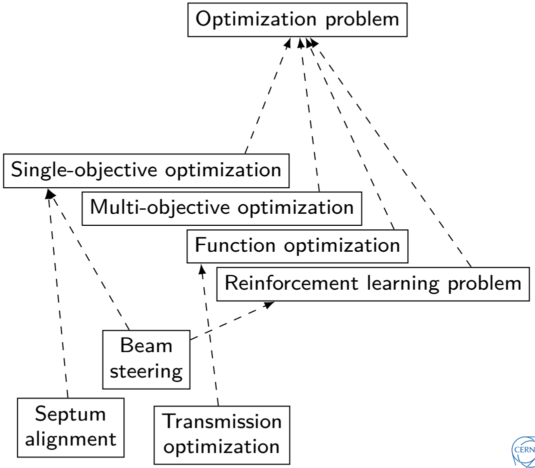
- provide ecosystem for accelerator optimization and control
- provide compatibility with as many algorithms as possible
- facilitate the progression
  manual tuning $\rightarrow$ *numerical optimization* $\rightarrow$ *machine learning*

Guiding principles:

- be agnostic over machine, communication protocol or devices
- minimize boilerplate code that does not solve the problem
- don't make people pay for features they don't use
- always leave an escape hatch open

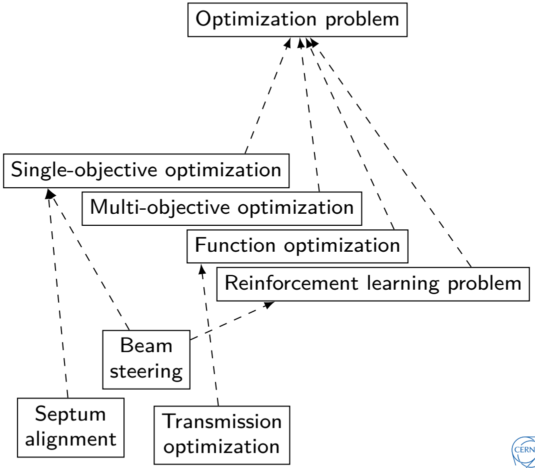
The Components: Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages



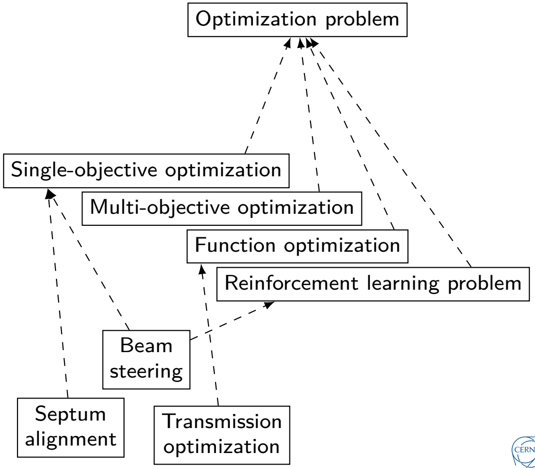
The Components: Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- inspired by OpenAI Gym



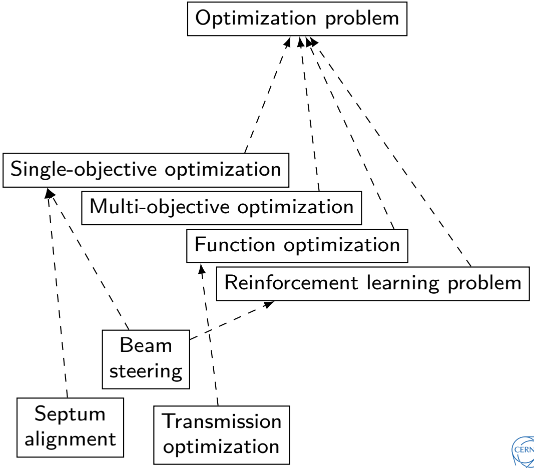
The Components: Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- inspired by OpenAI Gym
- extends their interfaces to numerical optimization



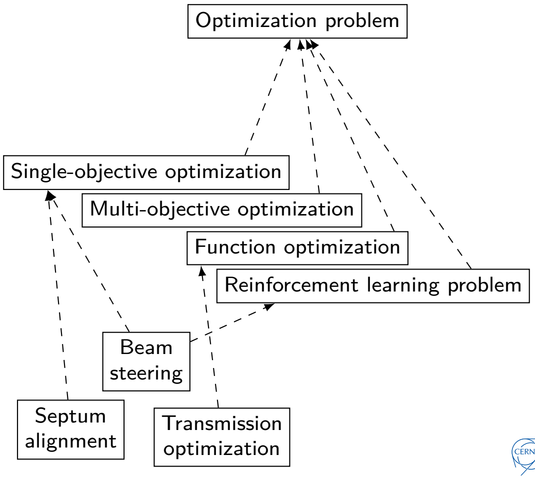
The Components: Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- inspired by OpenAI Gym
- extends their interfaces to numerical optimization
- extend Gym metadata system with CERN-specific info



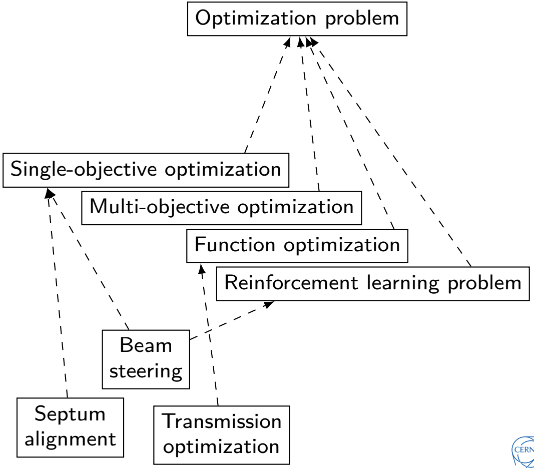
The Components: Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- inspired by OpenAI Gym
- extends their interfaces to numerical optimization
- extend Gym metadata system with CERN-specific info
  - which accelerator?



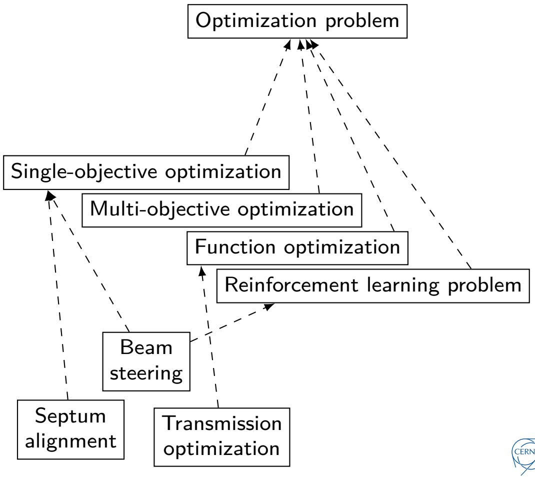
The Components: Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- inspired by OpenAI Gym
- extends their interfaces to numerical optimization
- extend Gym metadata system with CERN-specific info
  - which accelerator?
  - communicates with machines?



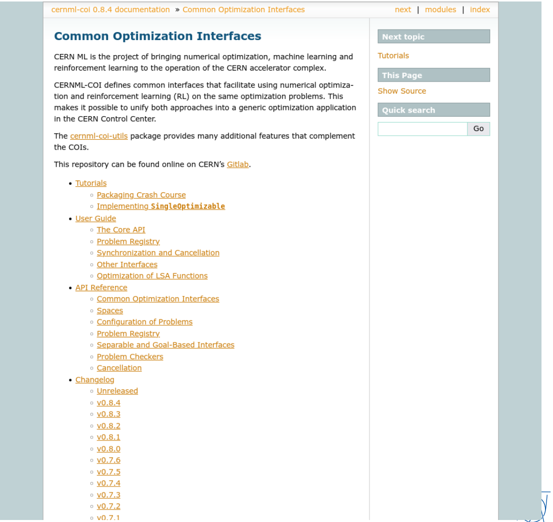
The Components: Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- inspired by OpenAI Gym
- extends their interfaces to numerical optimization
- extend Gym metadata system with CERN-specific info
  - which accelerator?
  - communicates with machines?
  - wants to plot additional data?



The Components: Common Optimization Interfaces (COI)

- standardized interfaces and adapters for various packages
- inspired by OpenAI Gym
- extends their interfaces to numerical optimization
- extend Gym metadata system with CERN-specific info
  - which accelerator?
  - communicates with machines?
  - wants to plot additional data?
- “20 % programming, 80 % documentation”


cernml-coi 0.8.4 documentation » Common Optimization Interfaces next | modules | index

Common Optimization Interfaces

CERN ML is the project of bringing numerical optimization, machine learning and reinforcement learning to the operation of the CERN accelerator complex.

CERNML-COI defines common interfaces that facilitate using numerical optimization and reinforcement learning (RL) on the same optimization problems. This makes it possible to unify both approaches into a generic optimization application in the CERN Control Center.

The cernml-coi-utils package provides many additional features that complement the COIs.

This repository can be found online on CERN's Gitlab.

- Tutorials
  - Packaging Crash Course
  - Implementing `SingleOptimizable`
- User Guide
  - The Core API
  - Problem Registry
  - Synchronization and Cancellation
  - Other Interfaces
  - Optimization of LSA Functions
- API Reference
  - Common Optimization Interfaces
  - Spaces
  - Configuration of Problems
  - Problem Registry
  - Separable and Goal-Based Interfaces
  - Problem Checkers
  - Cancellation
- Changelog
  - Unreleased
  - v0.8.4
  - v0.8.3
  - v0.8.2
  - v0.8.1
  - v0.8.0
  - v0.7.6
  - v0.7.5
  - v0.7.4
  - v0.7.3
  - v0.7.2
  - v0.7.1

Next topic

Tutorials

This Page

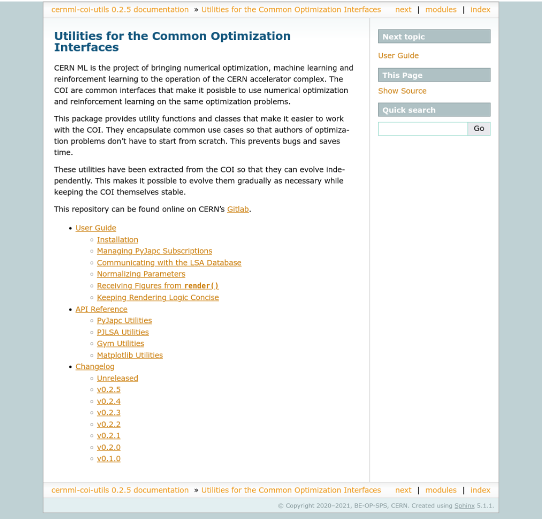
Show Source

Quick search

Go

The Components: COI Utilities

- separate package for faster versioning



cernml-coi-utils 0.2.5 documentation » Utilities for the Common Optimization Interfaces   next | modules | index

Utilities for the Common Optimization Interfaces

CERN ML is the project of bringing numerical optimization, machine learning and reinforcement learning to the operation of the CERN accelerator complex. The COI are common interfaces that make it posisble to use numerical optimization and reinforcement learning on the same optimization problems.

This package provides utility functions and classes that make it easier to work with the COI. They encapsulate common use cases so that authors of optimization problems don't have to start from scratch. This prevents bugs and saves time.

These utilities have been extracted from the COI so that they can evolve independently. This makes it possible to evolve them gradually as necessary while keeping the COI themselves stable.

This repository can be found online on CERN's Gitlab.

- User Guide
  - Installation
  - Managing PyJapc Subscriptions
  - Communicating with the LSA Database
  - Normalizing Parameters
  - Receiving Figures from `render()`
  - Keeping Rendering Logic Concise
- API Reference
  - PyJapc Utilities
  - PJLSA Utilities
  - Gym Utilities
  - Matplotlib Utilities
- Changelog
  - Unreleased
  - v0.2.5
  - v0.2.4
  - v0.2.3
  - v0.2.2
  - v0.2.1
  - v0.2.0
  - v0.1.0

Next topic

User Guide

This Page

Show Source

Quick search

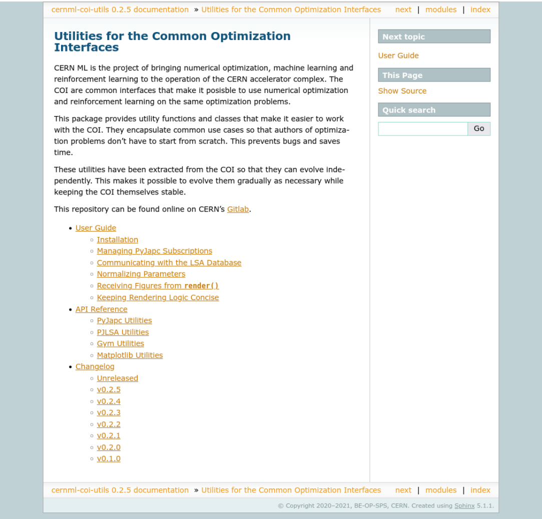
Go

cernml-coi-utils 0.2.5 documentation » Utilities for the Common Optimization Interfaces   next | modules | index

© Copyright 2020–2021, BE-OP-SPS, CERN. Created using Sphinx 5.1.1.

The Components: COI Utilities

- separate package for faster versioning
- encapsulate many common tasks



Utilities for the Common Optimization Interfaces

CERN ML is the project of bringing numerical optimization, machine learning and reinforcement learning to the operation of the CERN accelerator complex. The COI are common interfaces that make it posisble to use numerical optimization and reinforcement learning on the same optimization problems.

This package provides utility functions and classes that make it easier to work with the COI. They encapsulate common use cases so that authors of optimization problems don't have to start from scratch. This prevents bugs and saves time.

These utilities have been extracted from the COI so that they can evolve independently. This makes it possible to evolve them gradually as necessary while keeping the COI themselves stable.

This repository can be found online on CERN's Gitlab.

- User Guide
  - Installation
  - Managing PyJapc Subscriptions
  - Communicating with the LSA Database
  - Normalizing Parameters
  - Receiving Figures from `render()`
  - Keeping Rendering Logic Concise
- API Reference
  - PyJapc Utilities
  - PJLSA Utilities
  - Gym Utilities
  - Matplotlib Utilities
- Changelog
  - Unreleased
  - v0.2.5
  - v0.2.4
  - v0.2.3
  - v0.2.2
  - v0.2.1
  - v0.2.0
  - v0.1.0

Next topic

User Guide

This Page

Show Source

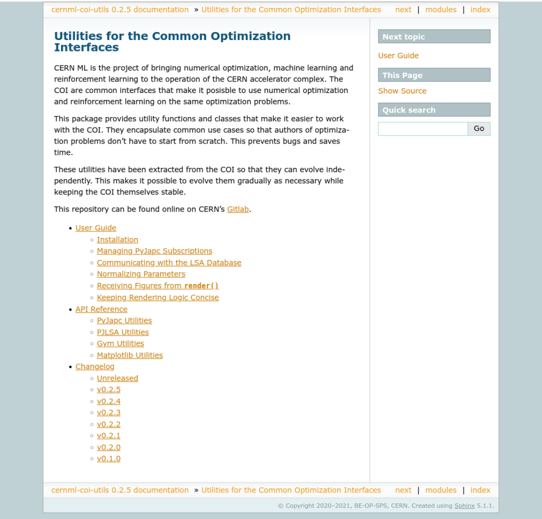
Quick search

Go

cernml-coi-utils 0.2.5 documentation » Utilities for the Common Optimization Interfaces next | modules | index

The Components: COI Utilities

- separate package for faster versioning
- encapsulate many common tasks
- removes repetitive tasks from the optimization problems



cernml-coi-utils 0.2.5 documentation » Utilities for the Common Optimization Interfaces   next | modules | index

Utilities for the Common Optimization Interfaces

CERN ML is the project of bringing numerical optimization, machine learning and reinforcement learning to the operation of the CERN accelerator complex. The COI are common interfaces that make it posisble to use numerical optimization and reinforcement learning on the same optimization problems.

This package provides utility functions and classes that make it easier to work with the COI. They encapsulate common use cases so that authors of optimization problems don't have to start from scratch. This prevents bugs and saves time.

These utilities have been extracted from the COI so that they can evolve independently. This makes it possible to evolve them gradually as necessary while keeping the COI themselves stable.

This repository can be found online on CERN's Gitlab.

- User Guide
  - Installation
  - Managing PyJapc Subscriptions
  - Communicating with the LSA Database
  - Normalizing Parameters
  - Receiving Figures from `render()`
  - Keeping Rendering Logic Concise
- API Reference
  - PyJapc Utilities
  - PJLSA Utilities
  - Gym Utilities
  - Matplotlib Utilities
- Changelog
  - Unreleased
  - v0.2.5
  - v0.2.4
  - v0.2.3
  - v0.2.2
  - v0.2.1
  - v0.2.0
  - v0.1.0

Next topic

User Guide

This Page

Show Source

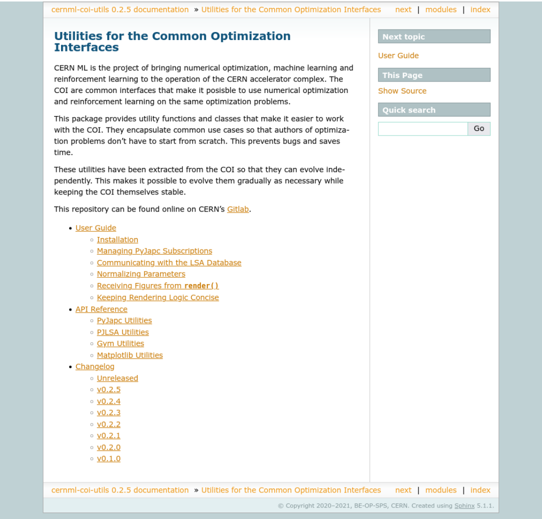
Quick search

Go

cernml-coi-utils 0.2.5 documentation » Utilities for the Common Optimization Interfaces   next | modules | index

The Components: COI Utilities

- separate package for faster versioning
- encapsulate many common tasks
- removes repetitive tasks from the optimization problems
- modular, only adds dependencies for what you use



cernml-coi-utils 0.2.5 documentation » Utilities for the Common Optimization Interfaces next | modules | index

Utilities for the Common Optimization Interfaces

CERN ML is the project of bringing numerical optimization, machine learning and reinforcement learning to the operation of the CERN accelerator complex. The COI are common interfaces that make it posisble to use numerical optimization and reinforcement learning on the same optimization problems.

This package provides utility functions and classes that make it easier to work with the COI. They encapsulate common use cases so that authors of optimization problems don't have to start from scratch. This prevents bugs and saves time.

These utilities have been extracted from the COI so that they can evolve independently. This makes it possible to evolve them gradually as necessary while keeping the COI themselves stable.

This repository can be found online on CERN's Gitlab.

- User Guide
  - Installation
  - Managing PyJapc Subscriptions
  - Communicating with the LSA Database
  - Normalizing Parameters
  - Receiving Figures from `render()`
  - Keeping Rendering Logic Concise
- API Reference
  - PyJapc Utilities
  - PJLSA Utilities
  - Gym Utilities
  - Matplotlib Utilities
- Changelog
  - Unreleased
  - v0.2.5
  - v0.2.4
  - v0.2.3
  - v0.2.2
  - v0.2.1
  - v0.2.0
  - v0.1.0

Next topic

User Guide

This Page

Show Source

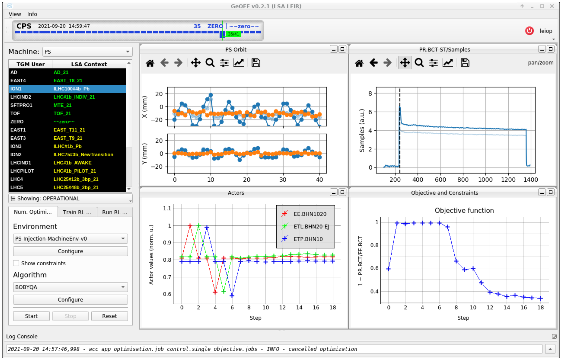
Quick search

Go

cernml-coi-utils 0.2.5 documentation » Utilities for the Common Optimization Interfaces next | modules | index

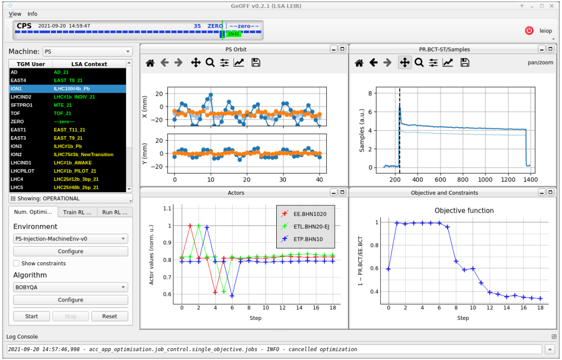
© Copyright 2020–2021, BE-OP-SPS, CERN. Created using Sphinx 5.1.1.

The Components: Generic Optimization Frontend & Framework (GeOFF)



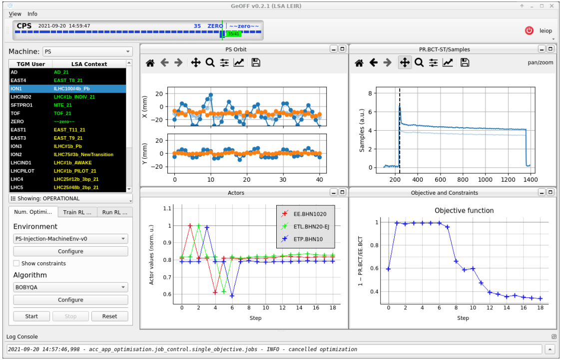
- lists, configures and runs optimization problems

The Components: Generic Optimization Frontend & Framework (GeOFF)



- lists, configures and runs optimization problems
- built-in list of optimizers

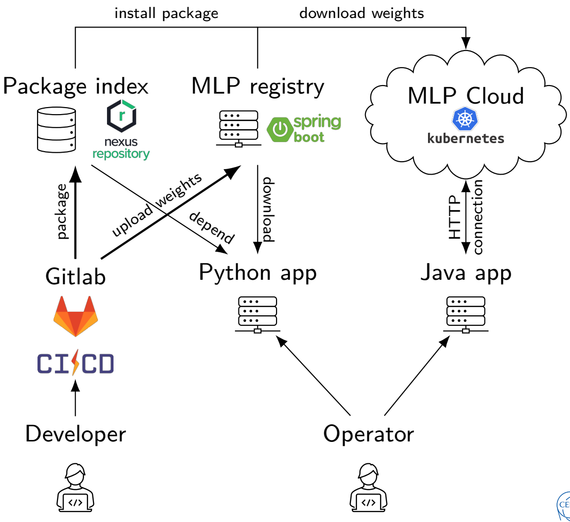
The Components: Generic Optimization Frontend & Framework (GeOFF)



- lists, configures and runs optimization problems
- built-in list of optimizers
- optimization problems are loaded as plugins pre-packaged or at runtime

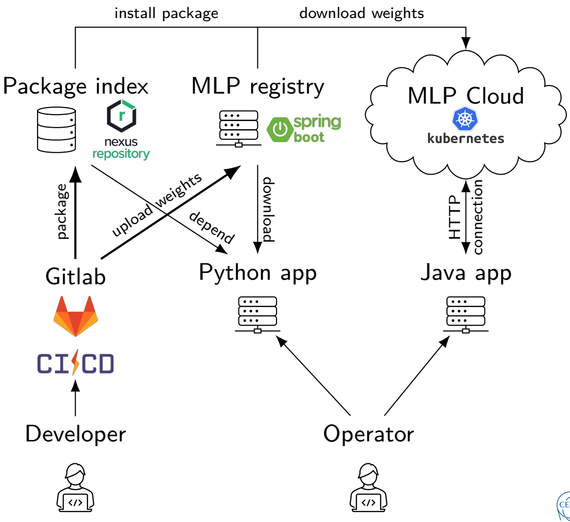
The Components: Machine Learning Platform (MLP)

- infrastructure for versioned and reliable storage of ML models



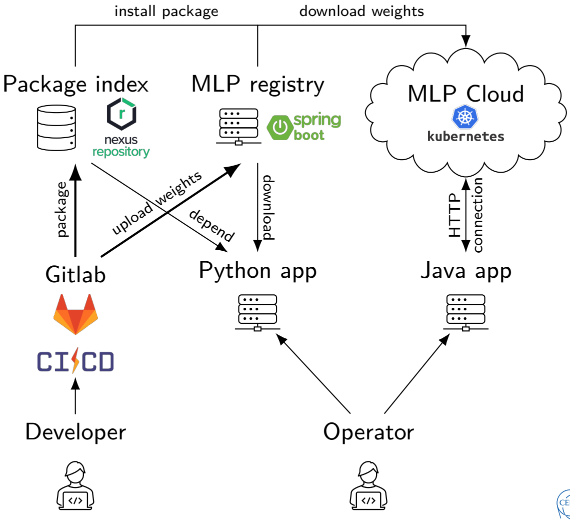
The Components: Machine Learning Platform (MLP)

- infrastructure for versioned and reliable storage of ML models
- separates model code from trained parameters



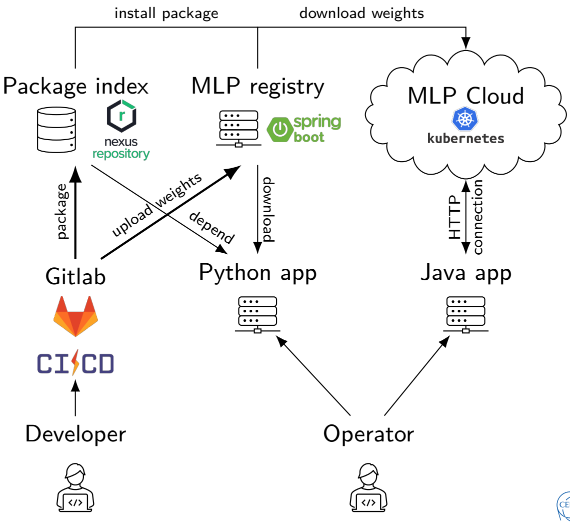
The Components: Machine Learning Platform (MLP)

- infrastructure for versioned and reliable storage of ML models
- separates model code from trained parameters
- Python-focused but framework-agnostic



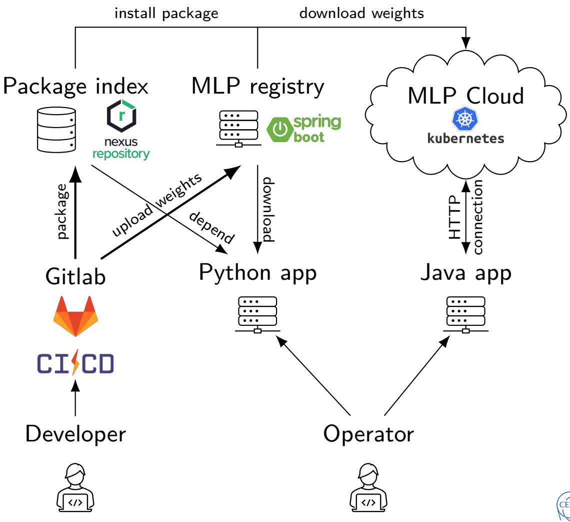
The Components: Machine Learning Platform (MLP)

- infrastructure for versioned and reliable storage of ML models
- separates model code from trained parameters
- Python-focused but framework-agnostic
- two modes of deployment



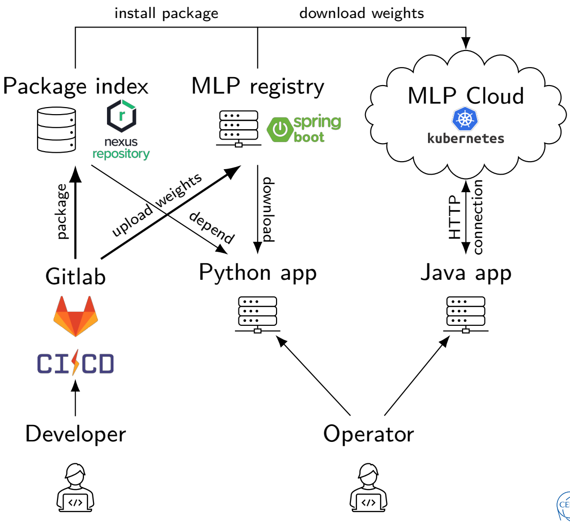
The Components: Machine Learning Platform (MLP)

- infrastructure for versioned and reliable storage of ML models
- separates model code from trained parameters
- Python-focused but framework-agnostic
- two modes of deployment
  - embedded in Python app



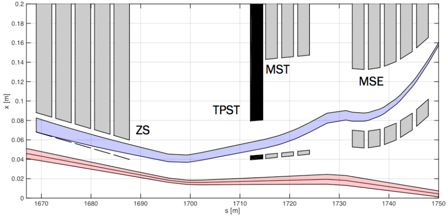
The Components: Machine Learning Platform (MLP)

- infrastructure for versioned and reliable storage of ML models
- separates model code from trained parameters
- Python-focused but framework-agnostic
- two modes of deployment
  - embedded in Python app
  - standalone as REST server usable from any kind of app



Use Case: SPS Septum Alignment via Numerical Optimization

- Alignment of electromagnetic septum, 9 DoF

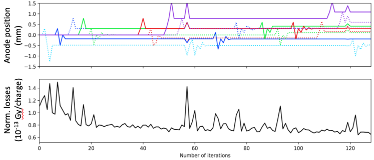


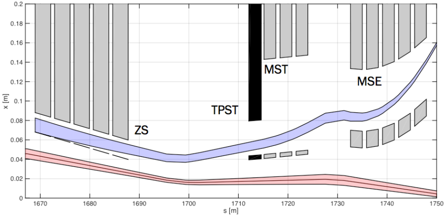
Time spent aligning:

before: $\sim 8\,\mathrm{h}$

Use Case: SPS Septum Alignment via Numerical Optimization

- Alignment of electromagnetic septum, 9 DoF
- 2018: test of *Powell* algorithm





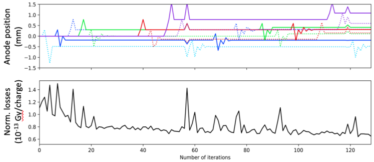
Time spent aligning:

before: $\sim 8\,\mathrm{h}$

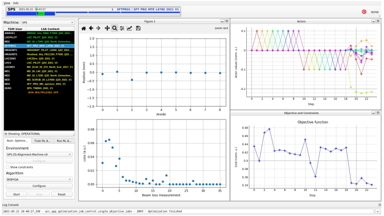
2018: $\sim 45\,\mathrm{min}$

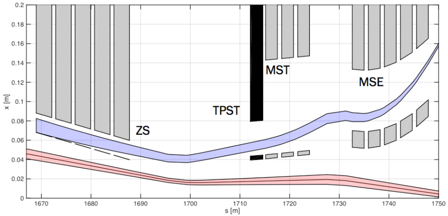
Use Case: SPS Septum Alignment via Numerical Optimization

- Alignment of electromagnetic septum, 9 DoF
- 2018: test of *Powell* algorithm



- 2021: *BOBYQA* algorithm in GeOFF





Time spent aligning:

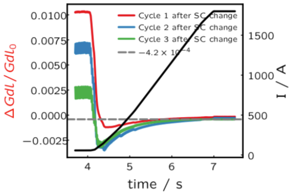
before: $\sim 8\,\mathrm{h}$

2018: $\sim 45\,\mathrm{min}$

2021: $\sim 10\,\mathrm{min}$

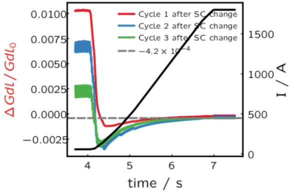
Use Case: SPS Quadrupole Hysteresis Prediction via LSTMs

- big issue for multi-cycling machines like SPS



Use Case: SPS Quadrupole Hysteresis Prediction via LSTMs

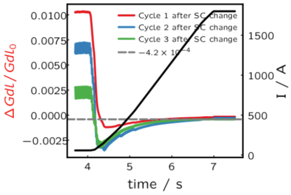
- big issue for multi-cycling machines like SPS
- extremely small effect ($< 1\,\%$), difficult to measure



Use Case: SPS Quadrupole Hysteresis Prediction via LSTMs

- big issue for multi-cycling machines like SPS
- extremely small effect ($< 1\,\%$), difficult to measure

$\Rightarrow$ still measurably affects the tune!



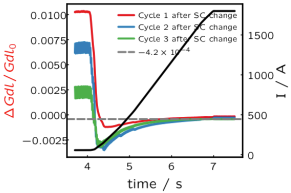
Use Case: SPS Quadrupole Hysteresis Prediction via LSTMs

- big issue for multi-cycling machines like SPS
- extremely small effect ($< 1\,\%$), difficult to measure

$\Rightarrow$ still measurably affects the tune!

Solution:

- measure the field in lab



Use Case: SPS Quadrupole Hysteresis Prediction via LSTMs

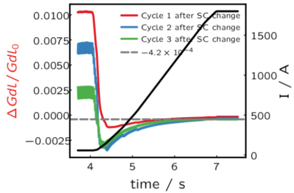
- big issue for multi-cycling machines like SPS
- extremely small effect ($< 1\,\%$), difficult to measure

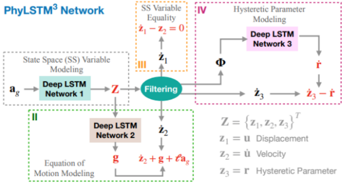
$\Rightarrow$ still measurably affects the tune!

Solution:

- measure the field in lab
- train physics-inspired LSTM on function $B(I(t), t)$

$$\mathcal{L} = \frac{1}{N} \sum \left(\alpha \left\| y_n - \bar{y} \right\|_2^2 + \beta \left\| \dot{y}_n - \dot{\bar{y}} \right\|_2^2 + \gamma \left\| \ddot{\bar{y}} - \mathrm{NN}(x, \dot{y}) \right\|_2^2 \right)$$

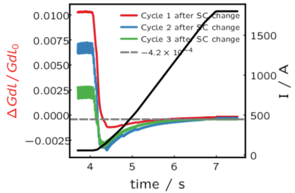




Use Case: SPS Quadrupole Hysteresis Prediction via LSTMs

- big issue for multi-cycling machines like SPS
- extremely small effect ($< 1\,\%$), difficult to measure

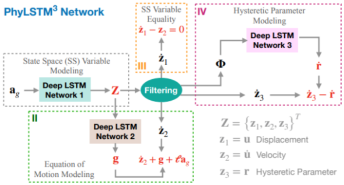
$\Rightarrow$ still measurably affects the tune!

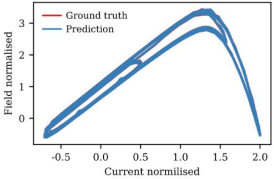


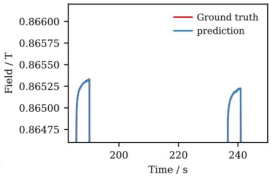
Solution:

- measure the field in lab
- train physics-inspired LSTM on function $B(I(t), t)$

$$\mathcal{L} = \frac{1}{N} \sum \left(\alpha \left\| y_n - \bar{y} \right\|_2^2 + \beta \left\| \dot{y}_n - \dot{\bar{y}} \right\|_2^2 + \gamma \left\| \ddot{\bar{y}} - \mathrm{NN}(x, \dot{y}) \right\|_2^2 \right)$$







Use Case: SPS Quadrupole Hysteresis Prediction via LSTMs

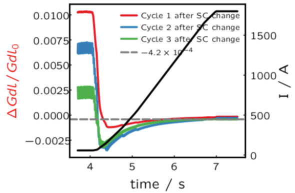
- big issue for multi-cycling machines like SPS
- extremely small effect ($< 1\,\%$), difficult to measure

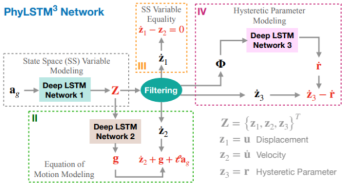
$\Rightarrow$ still measurably affects the tune!

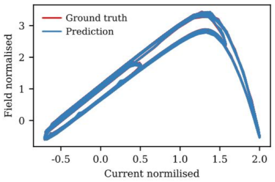
Solution:

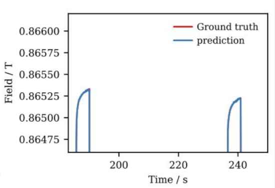
- measure the field in lab
- train physics-inspired LSTM on function $B(I(t), t)$

$$\mathcal{L} = \frac{1}{N}\sum\left(\alpha\left\|y_n - \bar{y}\right\|_2^2 + \beta\left\|\dot{y}_n - \dot{\bar{y}}\right\|_2^2 + \gamma\left\|\ddot{\bar{y}} - \mathrm{NN}(x, \dot{y})\right\|_2^2\right)$$





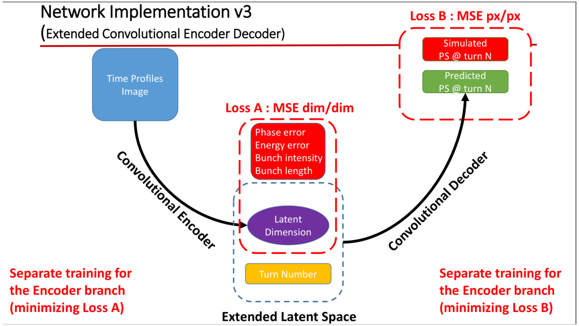


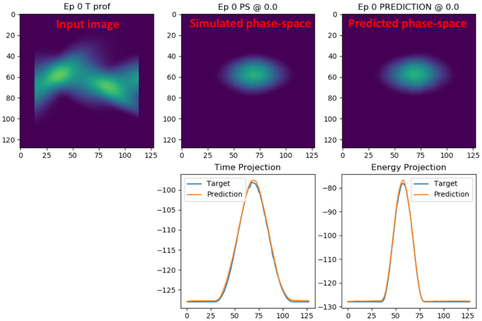


next milestones: predict Q, Q' and generalize to more magnets

Use Case: LHC Longitudinal Parameters Tomography via VAEs

- beam performance estimation requires longitudinal beam parameters

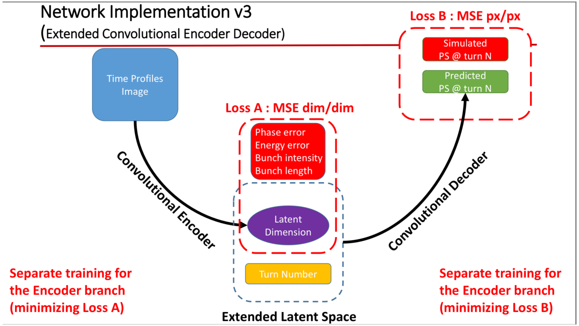


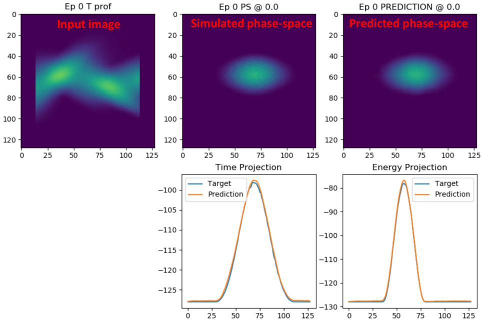


G. Trad and T. Argyropoulos

Use Case: LHC Longitudinal Parameters Tomography via VAEs

- beam performance estimation requires longitudinal beam parameters
- currently calculated via fits of longitudinal bunch profiles

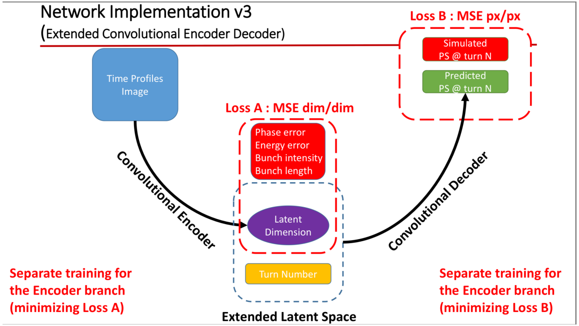


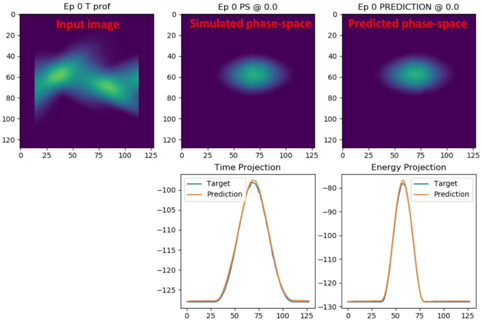


G. Trad and T. Argyropoulos

Use Case: LHC Longitudinal Parameters Tomography via VAEs

- beam performance estimation requires longitudinal beam parameters
- currently calculated via fits of longitudinal bunch profiles
- extremely time-consuming: can only be done online for single bunch

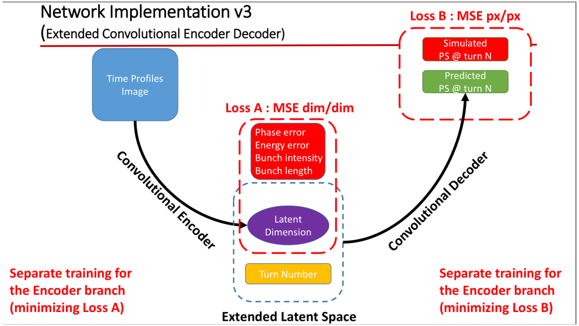


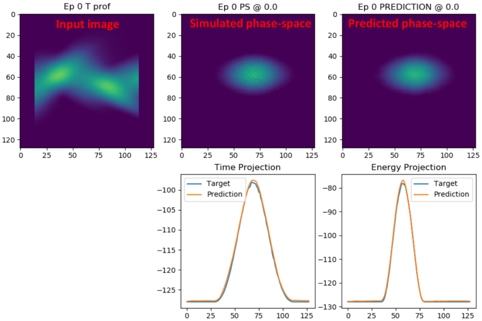


G. Trad and T. Argyropoulos

Use Case: LHC Longitudinal Parameters Tomography via VAEs

- beam performance estimation requires longitudinal beam parameters
- currently calculated via fits of longitudinal bunch profiles
- extremely time-consuming: can only be done online for single bunch
- ML can speed this up enough to do it bunch-by-bunch





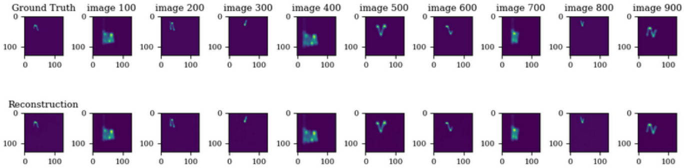
G. Trad and T. Argyropoulos

Use Case: Beam Dump Pattern Feature Extraction via CNNs

Goal: classify dump kicker failures in SPS and LHC based on beam dump pattern

Use Case: Beam Dump Pattern Feature Extraction via CNNs

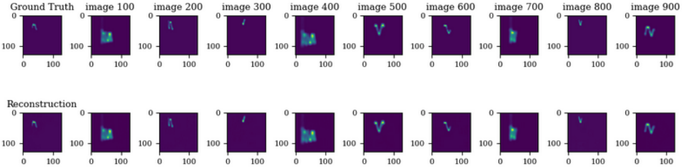
Goal: classify dump kicker failures in SPS and LHC based on beam dump pattern



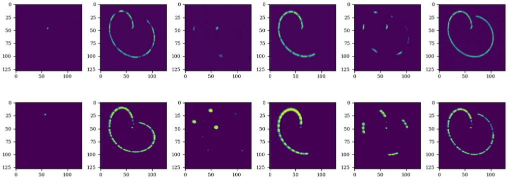
Model results in simulated data

Use Case: Beam Dump Pattern Feature Extraction via CNNs

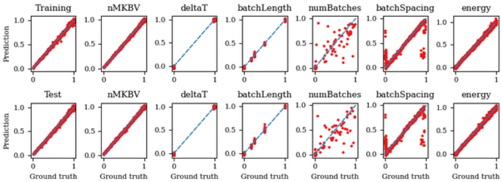
Goal: classify dump kicker failures in SPS and LHC based on beam dump pattern



Model results in simulated data



Model trained on simulation and applied on real data

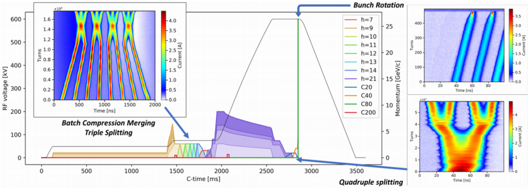


Extraction of physical information from images

F. Velotti and B. Goddard

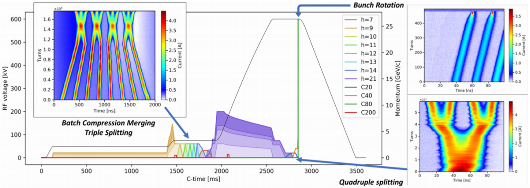
Use Case: PS RF Manipulations via RL

- **LHC beam production** requires quadruple splitting at $26\,\mathrm{GeV}/c$ in PS



Use Case: PS RF Manipulations via RL

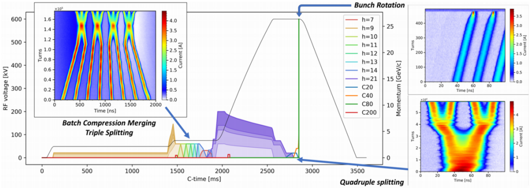
- **LHC beam production** requires quadruple splitting at $26\,\mathrm{GeV}/c$ in PS
- **RF phase errors** introduce spread in bunch-by-bunch intensity and emittance



Use Case: PS RF Manipulations via RL

- **LHC beam production** requires quadruple splitting at $26\,\mathrm{GeV}/c$ in PS
- **RF phase errors** introduce spread in bunch-by-bunch intensity and emittance

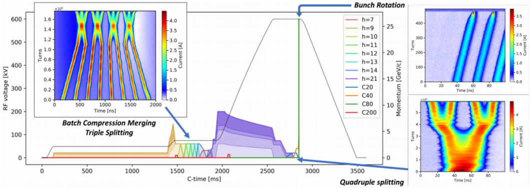
$\Rightarrow$ RL agent corrects phases for uniform bunches

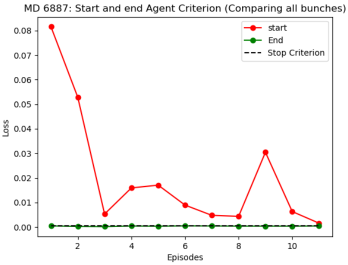


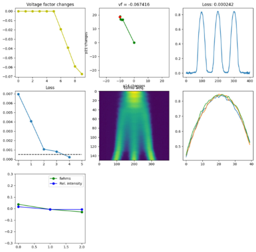
Use Case: PS RF Manipulations via RL

- **LHC beam production** requires quadruple splitting at $26\,\mathrm{GeV}/c$ in PS
- **RF phase errors** introduce spread in bunch-by-bunch intensity and emittance

$\Rightarrow$ RL agent corrects phases for uniform bunches





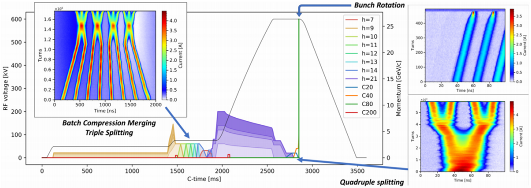


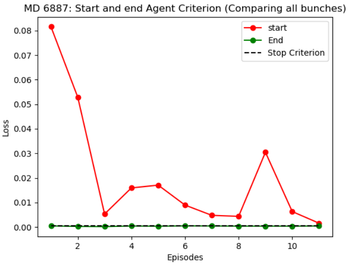
- faster than classical optimization due to reuse of experience

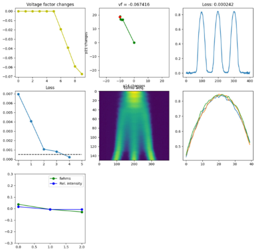
Use Case: PS RF Manipulations via RL

- **LHC beam production** requires quadruple splitting at $26\,\mathrm{GeV}/c$ in PS
- **RF phase errors** introduce spread in bunch-by-bunch intensity and emittance

$\Rightarrow$ RL agent corrects phases for uniform bunches





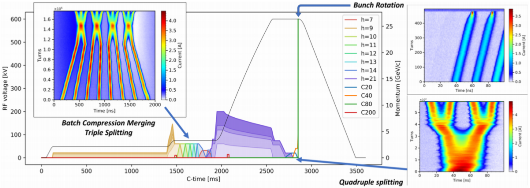


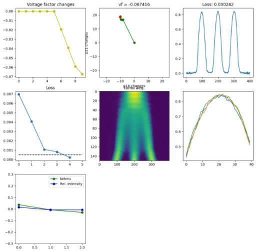
- faster than classical optimization due to reuse of experience
- trained on simulation, evaluated on real machine

Use Case: PS RF Manipulations via RL

- **LHC beam production** requires quadruple splitting at $26\,\mathrm{GeV}/c$ in PS
- **RF phase errors** introduce spread in bunch-by-bunch intensity and emittance

$\Rightarrow$ RL agent corrects phases for uniform bunches





- faster than classical optimization due to reuse of experience
- trained on simulation, evaluated on real machine
- episode length $n \in [2, 18]$, $\bar{n} = 8.46$

J. Wulff et. al. (2021, 2022)

GeOFF Use Cases

- Linac3: steering of beam transfer line

GeOFF Use Cases

- Linac3: steering of beam transfer line
- Linac4: 2 expert tools

GeOFF Use Cases

- Linac3: steering of beam transfer line
- Linac4: 2 expert tools
- PSB: operations (WIP) & commissioning
  - bunch recombination at PSB ejection
  - resonance compensation
  - RF optimization
  - injection to PS

GeOFF Use Cases

- Linac3: steering of beam transfer line
- Linac4: 2 expert tools
- PSB: operations (WIP) & commissioning
  - bunch recombination at PSB ejection
  - resonance compensation
  - RF optimization
  - injection to PS
- PS: used during commissioning
  - resonance compensation
  - transition gamma jump circuits
  - septa alignment for slow extraction

GeOFF Use Cases

- Linac3: steering of beam transfer line
- Linac4: 2 expert tools
- PSB: operations (WIP) & commissioning
  - bunch recombination at PSB ejection
  - resonance compensation
  - RF optimization
  - injection to PS
- PS: used during commissioning
  - resonance compensation
  - transition gamma jump circuits
  - septa alignment for slow extraction
- LEIR: used during commissioning
  - transfer lines (from Linac3, to PS)
  - injection bumps
  - phase adjustment of RF cavities

GeOFF Use Cases

- Linac3: steering of beam transfer line
- Linac4: 2 expert tools
- PSB: operations (WIP) & commissioning
  - bunch recombination at PSB ejection
  - resonance compensation
  - RF optimization
  - injection to PS
- PS: used during commissioning
  - resonance compensation
  - transition gamma jump circuits
  - septa alignment for slow extraction
- LEIR: used during commissioning
  - transfer lines (from Linac3, to PS)
  - injection bumps
  - phase adjustment of RF cavities
- SPS: expert tool & operations
  - tune adjustments
  - septa alignment for slow extraction
  - spill noise reduction
  - splitter optimization
  - injection kicker optimization
  - crystal shadowing

GeOFF Use Cases

- Linac3: steering of beam transfer line
- Linac4: 2 expert tools
- PSB: operations (WIP) & commissioning
  - bunch recombination at PSB ejection
  - resonance compensation
  - RF optimization
  - injection to PS
- PS: used during commissioning
  - resonance compensation
  - transition gamma jump circuits
  - septa alignment for slow extraction
- LEIR: used during commissioning
  - transfer lines (from Linac3, to PS)
  - injection bumps
  - phase adjustment of RF cavities
- SPS: expert tool & operations
  - tune adjustments
  - septa alignment for slow extraction
  - spill noise reduction
  - splitter optimization
  - injection kicker optimization
  - crystal shadowing
- used at almost all accelerators
  - ISOLDE: lots of homogeneous devices $\Rightarrow$ CPS Optimizer
  - LHC: fast acquisition, high safety req. $\Rightarrow$ bespoke algorithms

GeOFF Use Cases

- Linac3: steering of beam transfer line
- Linac4: 2 expert tools
- PSB: operations (WIP) & commissioning
  - bunch recombination at PSB ejection
  - resonance compensation
  - RF optimization
  - injection to PS
- PS: used during commissioning
  - resonance compensation
  - transition gamma jump circuits
  - septa alignment for slow extraction
- LEIR: used during commissioning
  - transfer lines (from Linac3, to PS)
  - injection bumps
  - phase adjustment of RF cavities

- SPS: expert tool & operations
  - tune adjustments
  - septa alignment for slow extraction
  - spill noise reduction
  - splitter optimization
  - injection kicker optimization
  - crystal shadowing

- used at almost all accelerators
  - ISOLDE: lots of homogeneous devices $\Rightarrow$ CPS Optimizer
  - LHC: fast acquisition, high safety req. $\Rightarrow$ bespoke algorithms
- most often used as expert tool

Conclusion:

- machine learning isn't coming to CERN — it's there!

Conclusion:

- machine learning isn't coming to CERN — it's there!
  - diagnostics, fault detection and modeling: use ML in production

Conclusion:

- machine learning isn't coming to CERN — it's there!
  - diagnostics, fault detection and modeling: use ML in production
  - controls: dominated by classical optimization (but also RL!)

Conclusion:

- machine learning isn't coming to CERN — it's there!
  - diagnostics, fault detection and modeling: use ML in production
  - controls: dominated by classical optimization (but also RL!)
- supported by ecosystem of independent projects and efforts (many not named here)

Conclusion:

- machine learning isn't coming to CERN — it's there!
  - diagnostics, fault detection and modeling: use ML in production
  - controls: dominated by classical optimization (but also RL!)
- supported by ecosystem of independent projects and efforts (many not named here)

  MLP: model storage and versioning

Conclusion:

- machine learning isn't coming to CERN — it's there!
  - diagnostics, fault detection and modeling: use ML in production
  - controls: dominated by classical optimization (but also RL!)
- supported by ecosystem of independent projects and efforts (many not named here)

  MLP: model storage and versioning

  COI: uniform interfaces for optimization and RL

Conclusion:

- machine learning isn't coming to CERN — it's there!
  - diagnostics, fault detection and modeling: use ML in production
  - controls: dominated by classical optimization (but also RL!)
- supported by ecosystem of independent projects and efforts (many not named here)

  MLP: model storage and versioning

  COI: uniform interfaces for optimization and RL

  GeOFF: framework for optimizers, tasks and monitoring

Conclusion:

- machine learning isn't coming to CERN — it's there!
  - diagnostics, fault detection and modeling: use ML in production
  - controls: dominated by classical optimization (but also RL!)
- supported by ecosystem of independent projects and efforts (many not named here)

  MLP: model storage and versioning
  COI: uniform interfaces for optimization and RL
  GeOFF: framework for optimizers, tasks and monitoring

The future:

- modularize GeOFF, make it independent of GUI

Conclusion:

- machine learning isn't coming to CERN — it's there!
  - diagnostics, fault detection and modeling: use ML in production
  - controls: dominated by classical optimization (but also RL!)
- supported by ecosystem of independent projects and efforts (many not named here)

  MLP: model storage and versioning
  COI: uniform interfaces for optimization and RL
  GeOFF: framework for optimizers, tasks and monitoring

The future:

- modularize GeOFF, make it independent of GUI
- adapt GeOFF to be available outside of CERN (EURO-LABS)