HEPS Commissioning tools and machine learning

Yi Jiao (焦毅)
On behalf of the HEPS Commissioning Team

Jul. 13, 2026

Outline

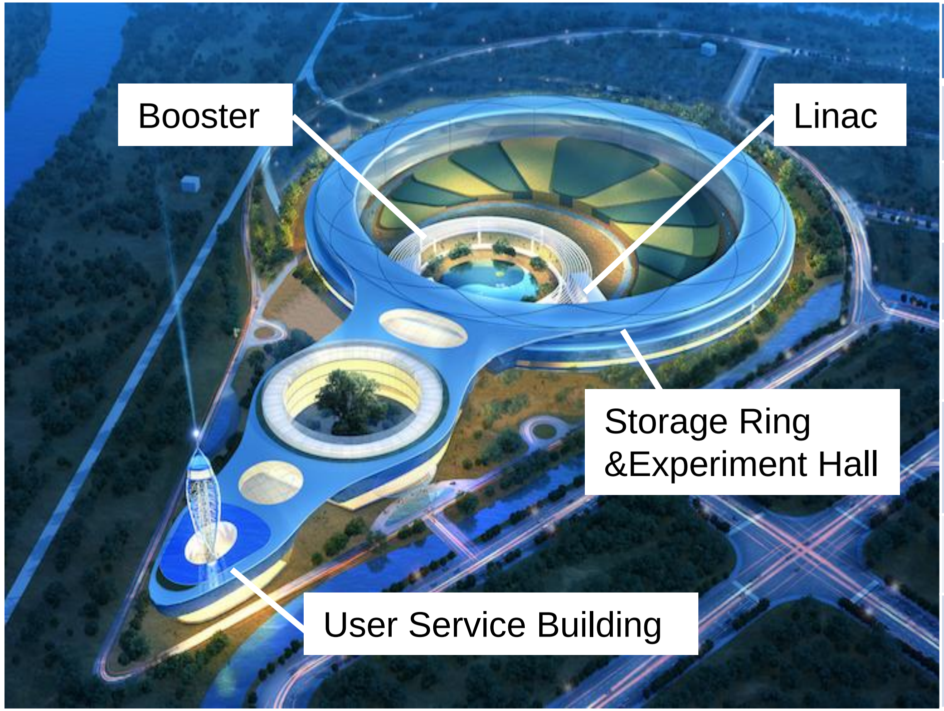
- HLA development requirements
- *Pyapas* for HLA development
- HLAs development for HEPS based on *Pyapas*
- Machine Learning Application at HEPS
- Conclusion

- HLA development requirements
- *Pyapas* for HLA development
- HLAs development for HEPS based on *Pyapas*
- Machine Learning Application at HEPS
- Conclusion

Introduction to HEPS

High Energy Photon Source

One of the world's brightest fourth-generation synchrotron light sources



	APS-U	ESRF-EBS	MAX-IV	Sirius	HEPS
E(GeV)	6	6	3	3	6
C(m)	1104	844.4	528	518	1360.4
Lattice	7BA	7BA	7BA	5BA	7BA
Cell	40	32	20	20	48
Emittance (pm*rad)	42	150	330	250	34
Brightness	$> 10^{22}$	$> 10^{22}$	$\sim 10^{21}$	$\sim 10^{21}$	$> 10^{22}$
Construction period	2018-2024	2015-2020	2010-2016	2015-2018	2019~2025

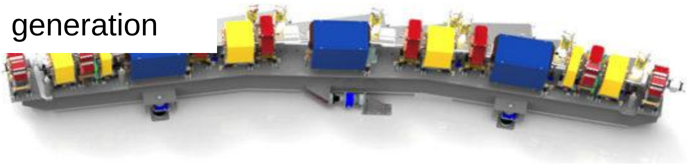
2025.10.29 HEPS has successfully passed the process qualification and acceptance

Introduction to HEPS

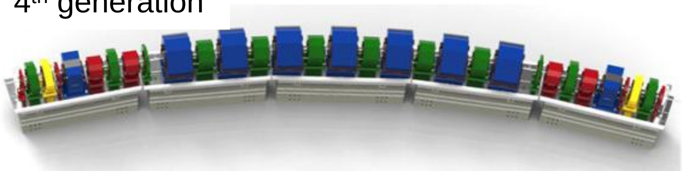
→The emittance is reduced by 1or 2 orders of magnitude, approaching the X-ray diffraction limit

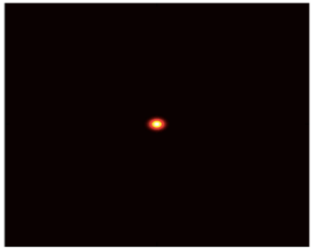
→The number of magnets increases by an order of magnitude

3rd generation



4th generation





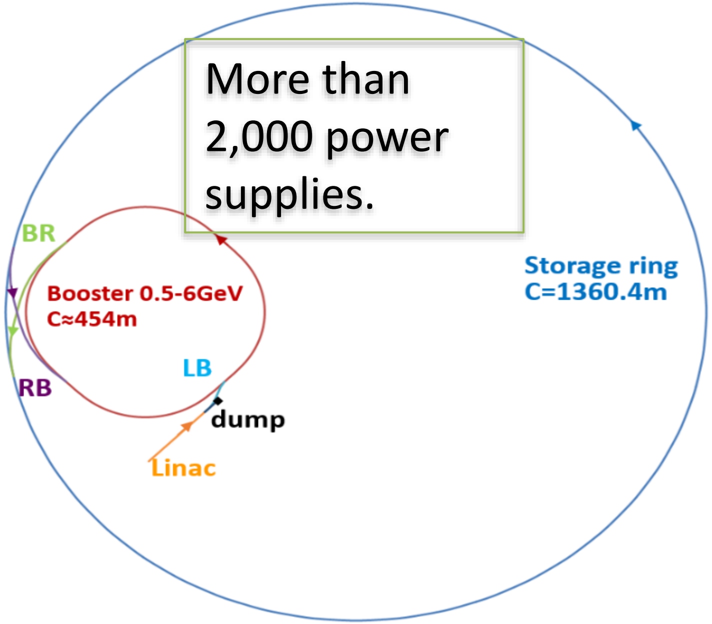
1 mm

Layout

Beam Profile

HEPS is currently the largest accelerator facility in China.

- High control complexity
- Large scale of control variables
- High precision control required
- Large number of HLAs required



High Level Application (HLA)

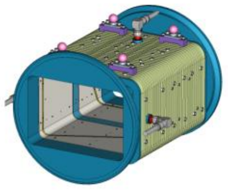
Application-layer: integrates low-level devices with physics algorithms to **monitor, analyze and control** accelerator operation — the accelerator’s **“central nervous system”.**

Low-Level Hardware

diagnostics • devices • raw signals


Camera / Screen


Wire Scanner


BPM


Magnet / Power Supply

Physics & Control Layer

Data fusion

Physics models + algorithms

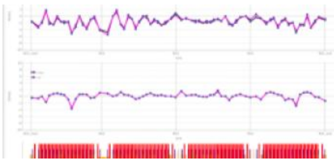
control communication

EPICS / APIs / HLA framework

High-Level Application

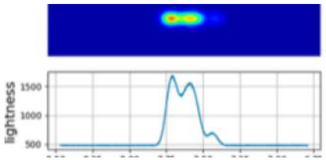
beam performance • operation decision • closed-loop control

Closed Orbit Correction

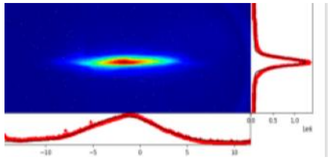


Lifetime Estimation

Energy / Spread Analysis



Emittance Measurement



Core message: HLA bridges physics models and accelerator devices, enabling simulation, optimization, feedback, and operation analysis over a large-scale control system.

High Level Application

What HLA does

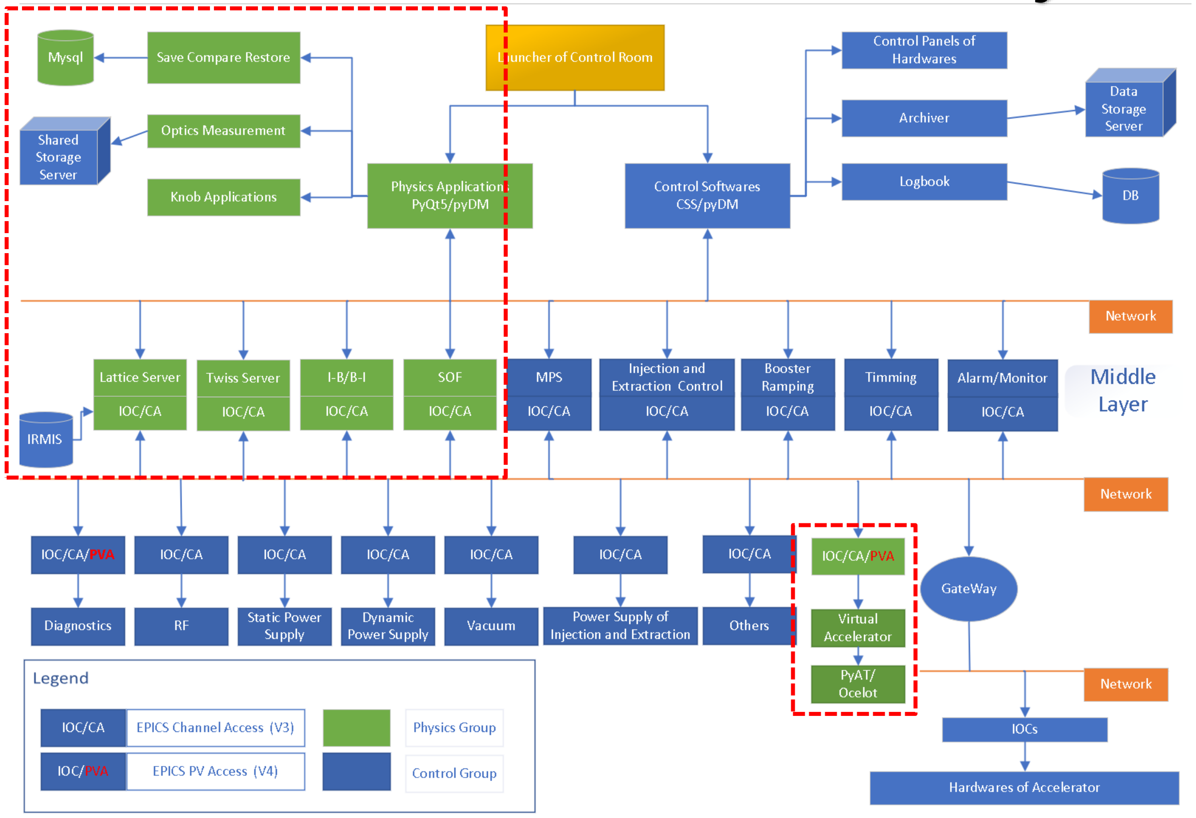
Focuses on **physics processes** rather than device-level control.

- Optimization
- Simulation
- Feedback
- Operation Analysis

Why it is challenging

- High control complexity
- Large-scale control variables
- Strict precision and reliability
- Many dedicated HLA programs

The overall structure of control system



Core message: HLA bridges physics models and accelerator devices, enabling simulation, optimization, feedback, and operation analysis over a large-scale control system.

High Level Application

A reusable software framework that packages communication, online physics models, visualization, deployment, and data services for accelerator high-level applications.

Development Challenges

Traditional HLA development often repeats the same engineering work for every new application.

Idea	Physics target & operator requirements
Algorithm	Optics / feedback / optimization logic
Implementation	Communication, GUI, server, database
Operation	Deployment, reliability, long-term maintenance

Key bottleneck:
duplicated engineering, fragmented interfaces, and difficult maintenance.

Systematic HLA Framework

Reusable Core

Framework Capabilities

1	**Hardware communication**	EPICS CA/PVA access, device abstraction, unified parameter interface
2	**Online physics models**	Invoke optics models, simulation, analysis and feedback algorithms online
3	**Interactive user interface**	Visual panels for tuning, monitoring, diagnostics and operator workflows
4	**Service deployment**	Server-side execution, scheduling, reliability and long-term operation
5	**Data management**	Real-time database connection, logging, archiving and reproducible records

Applications become faster to develop, easier to deploy, and more reliable to operate.

- HLA development requirements
- *Pyapas* for HLA development
- HLAs development for HEPS based on *Pyapas*
- Machine Learning Application at HEPS
- Conclusion

New HLA framework: *Pyapas*

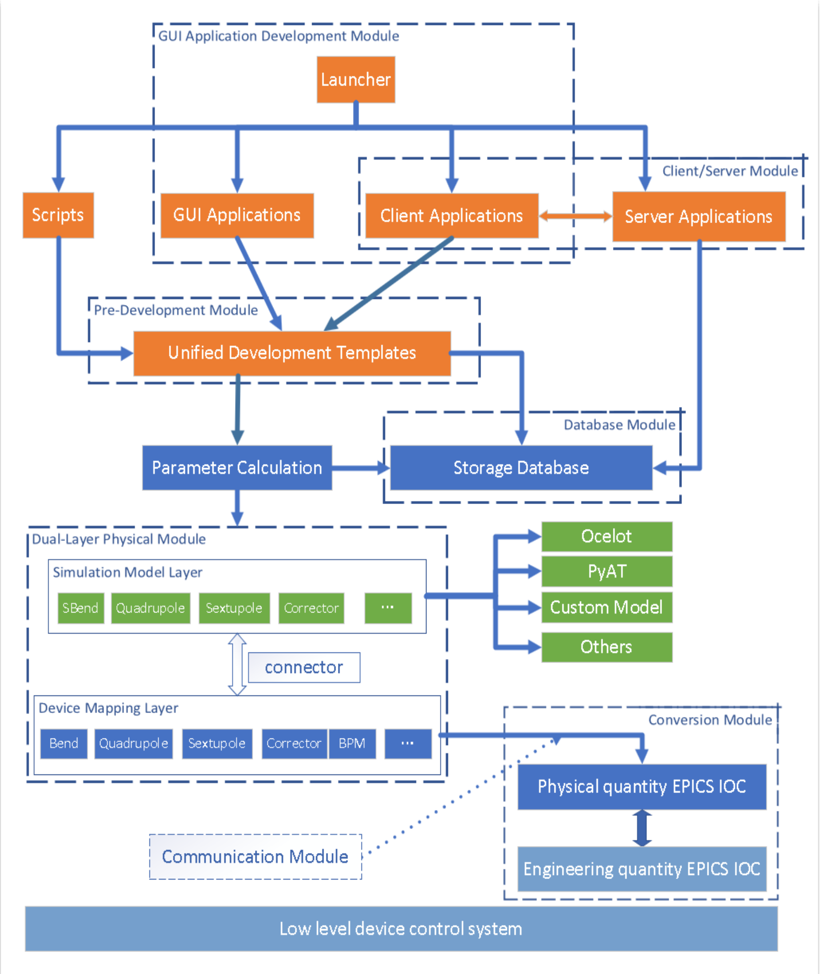
- A Pure Python Framework for High Level Application Development.

  Python-based Accelerator Physics Application Set (*Pyapas*)

  - **Modular Design**
  - **Model-Based**
  - **Based on Physical Quantities**

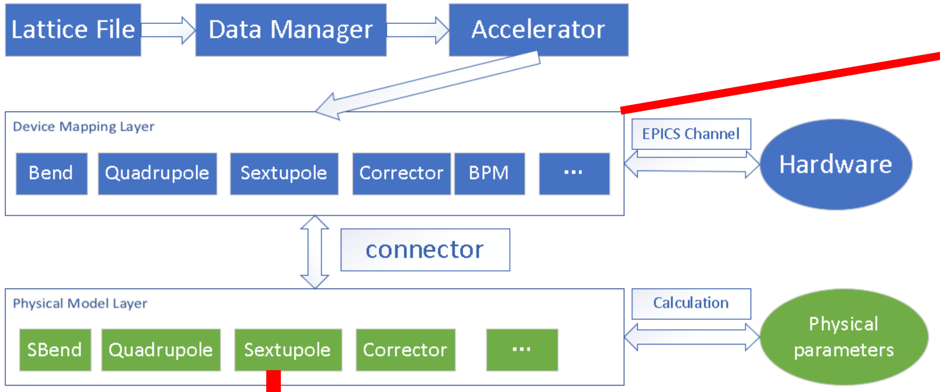
① Dual-layer physical module
② Friendly user interface module
③ Hardware communication module
④ Database connection module
⑤ Client-Server development module
⑥ Various phyiscal algorithm toolkits

Pyapas framework



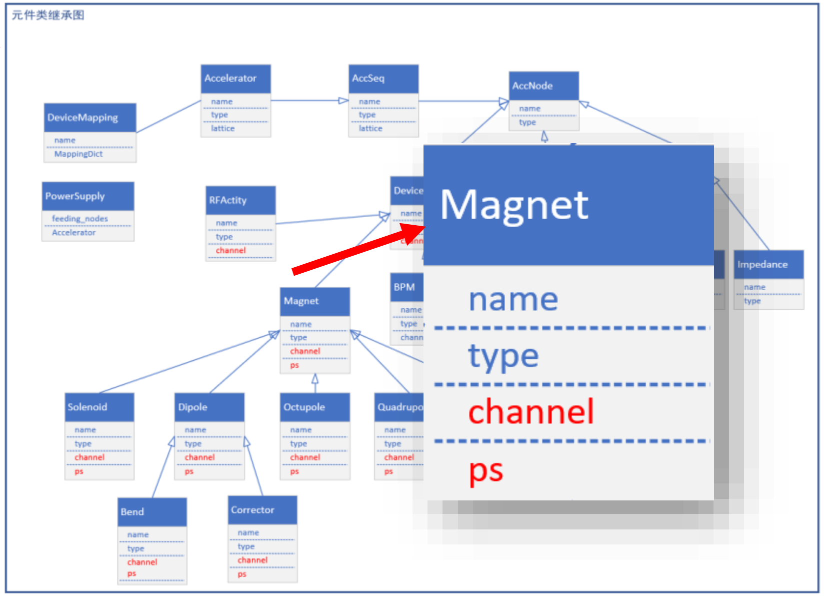
Xiaohan Lu, et al, "A new modular framework for high-level application development at HEPS", Journal of Synchrotron Radiation, 31: 385-393, 2024.

Low-coupling dual-layer physical module

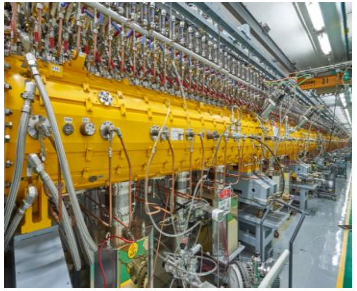


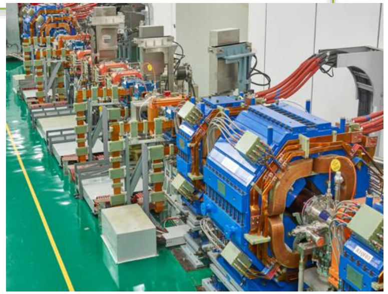
- Ocelot
- PyAT
- Custom model

The "Connector" is responsible for invoking simulation algorithms



Abstraction of the Hardware





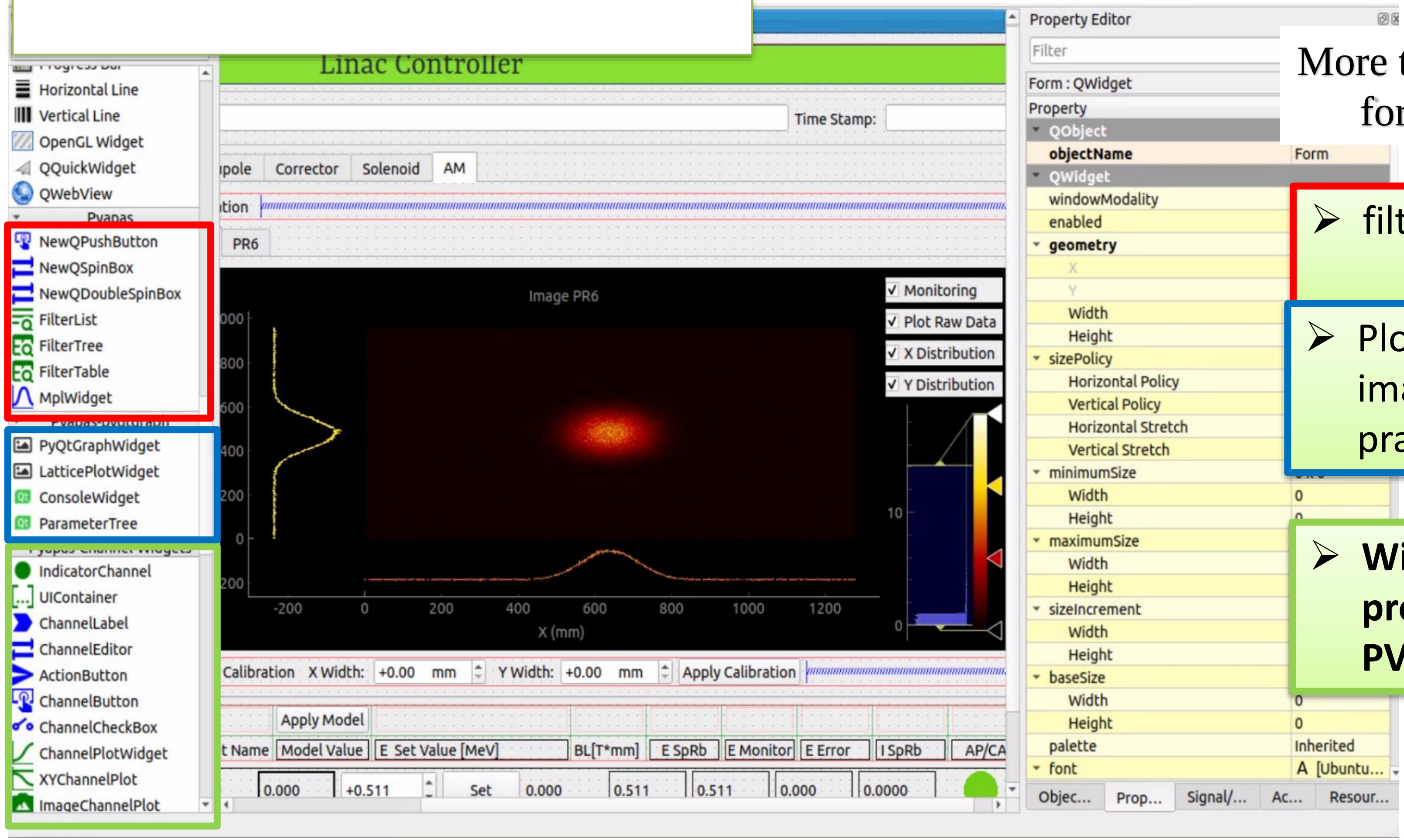
Astract Object:

Name/Position/Length/
Field/Channel/Type/PS/
Physical parameter

Low-Code development

Customize widgets for HLA development

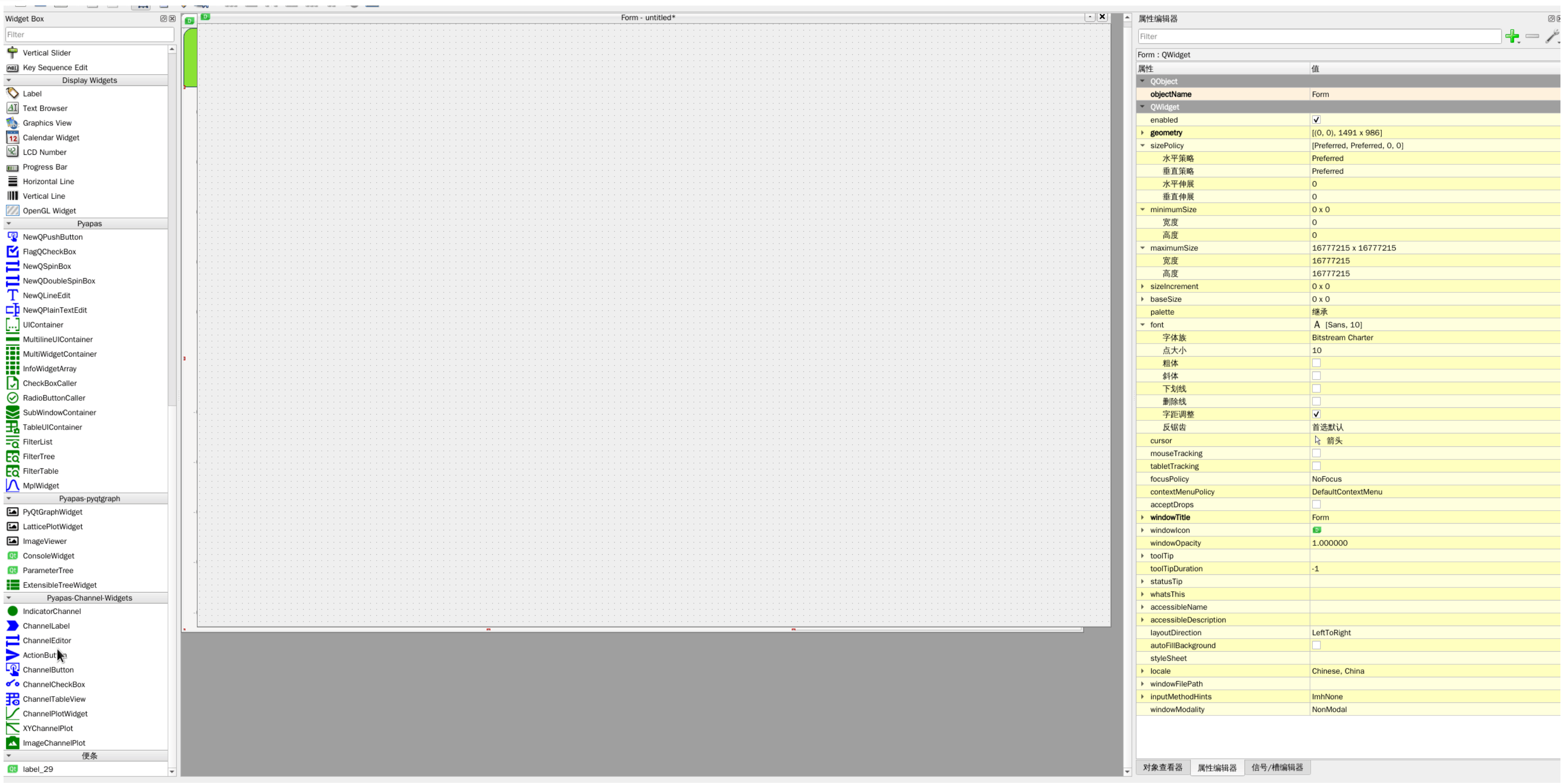
"Drag and Drop"



More than 20 pre-built widgets for rapid development.

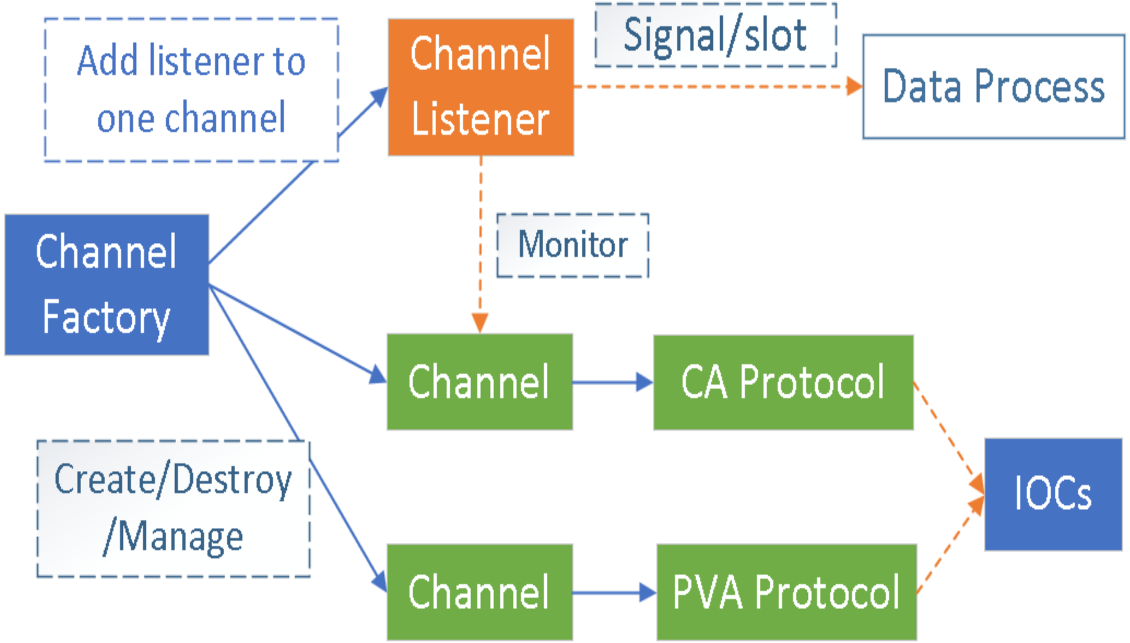
- ➢ filterable list, table, tree
- ➢ Plot widget for physical image, e.g. plot prameters with lattice
- ➢ **Widgets with channel property to interact with PV directly.**

Low-Code development



Real-time Data Interaction System

➢ Design the communication structure (active and passive channel data acquisition)）



The Channel class contains the EPICS communication interface responsible for specific communication with the IOC

- ✓ The 'Singleton Factory Pattern' centrally manages interaction channels in the program, avoiding redundant creation and improving interaction efficiency.
- ✓ The channel factory has namespace functionality

	NodeName	xAvgChannel	Value	yAvgChannel	
1	R01BPM01	R:BI:R01BPM01:x	-0.00024846263641131787	R:BI:R01BPM01:y	0.0
2	R01BPM02	R:BI:R01BPM02:x	-0.0007405004049405429	R:BI:R01BPM02:y	0.0
3	R01BPM03	R:BI:R01BPM03:x	-0.0003863998687952148	R:BI:R01BPM03:y	0.0
4	R01BPM04	R:BI:R01BPM04:x	-5.57367509261469e-05	R:BI:R01BPM04:y	0.0
5	R01BPM05	R:BI:R01BPM05:x	0.00019363297907922127	R:BI:R01BPM05:y	0.0
6	R01BPM06	R:BI:R01BPM06:x	0.0003999152633156065	R:BI:R01BPM06:y	0.0
7	R01BPM07	R:BI:R01BPM07:x	-0.0002012937695207042	R:BI:R01BPM07:y	0.0
8	R01BPM08	R:BI:R01BPM08:x	-0.0003712213026685325	R:BI:R01BPM08:y	0.0
9	R01BPM09	R:BI:R01BPM09:x	0.0003782940324338285	R:BI:R01BPM09:y	0.0
10	R01BPM10				0.0
11	R01BPM11				0.0
12	R01BPM12				0.0
13	R02BPM01				0.0
14	R02BPM02				0.0
15	R02BPM03				0.0
16	R02BPM04				0.0
17	R02BPM05				0.0
18	R02BPM06	[illegible]	[illegible]	[illegible]	0.0

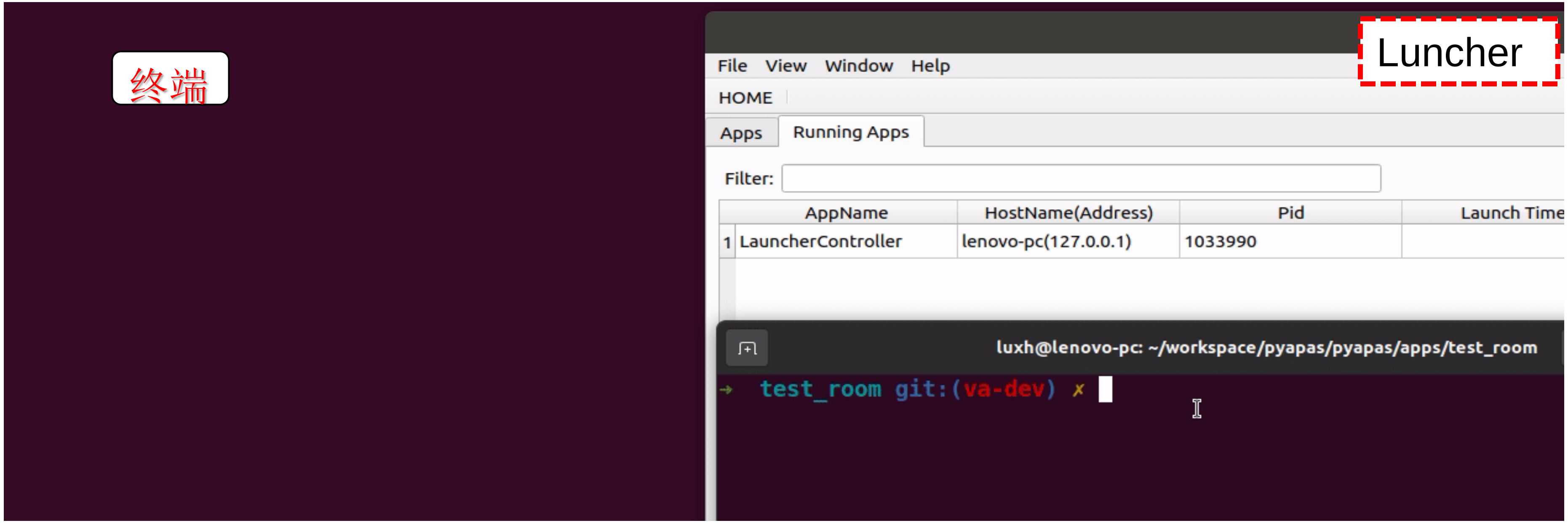
One application can smoothly run while monitoring 10,000 PVs

Client/Server Module

mDNS +XMLRPC

The system autonomously detects active Pyapas applications on the local area network and tracks their health and status.

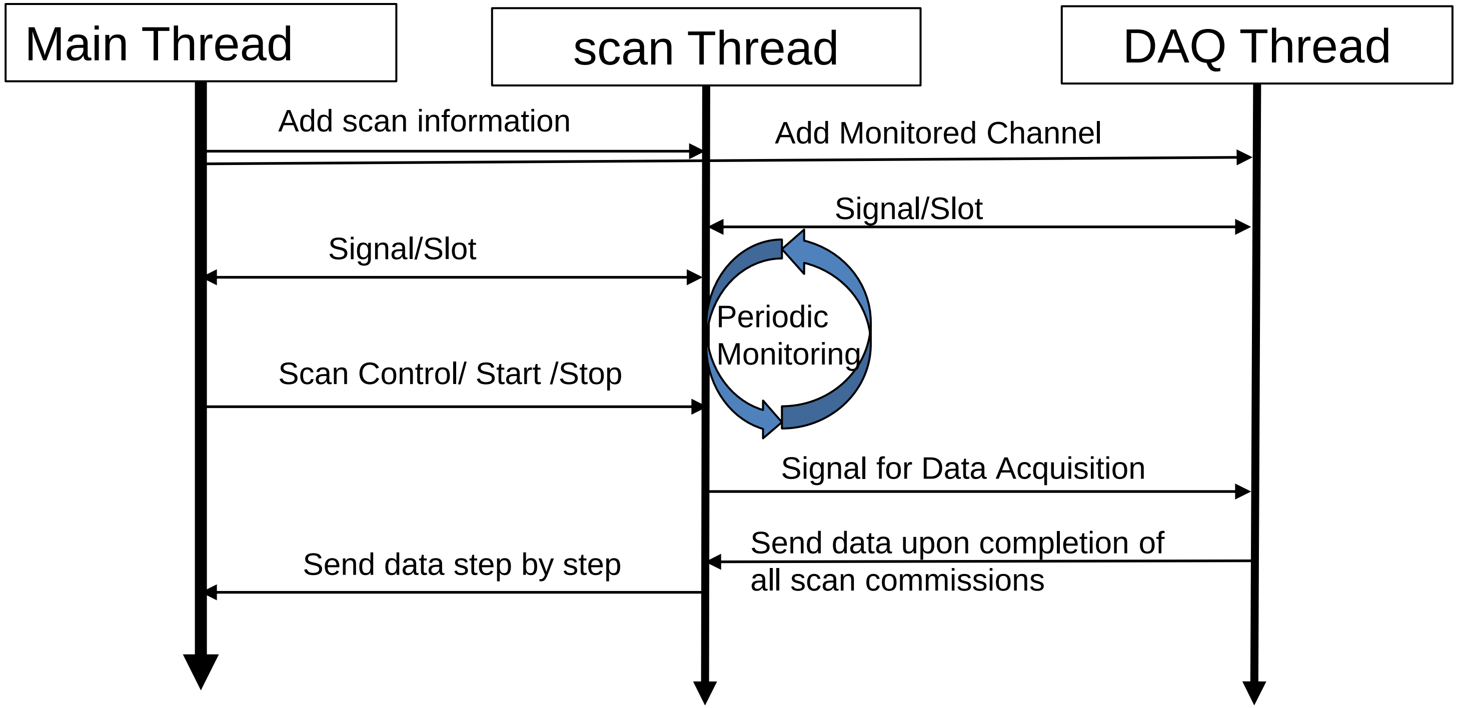
These features are packaged into a full-featured Server class, allowing any subclass to instantly become a functional server.



Multithreading-based scanner

Python's GIL locks up the UI during single-threaded scans, as the loop cannot be paused. This requires a forced close of the program, which is impractical and risky in a live machine setting.

A universal scanner was designed to address this limitation



- HLA development requirements
- *Pyapas* for HLA development
- **HLAs development for HEPS based on *Pyapas***
- Machine Learning Application at HEPS
- Conclusion

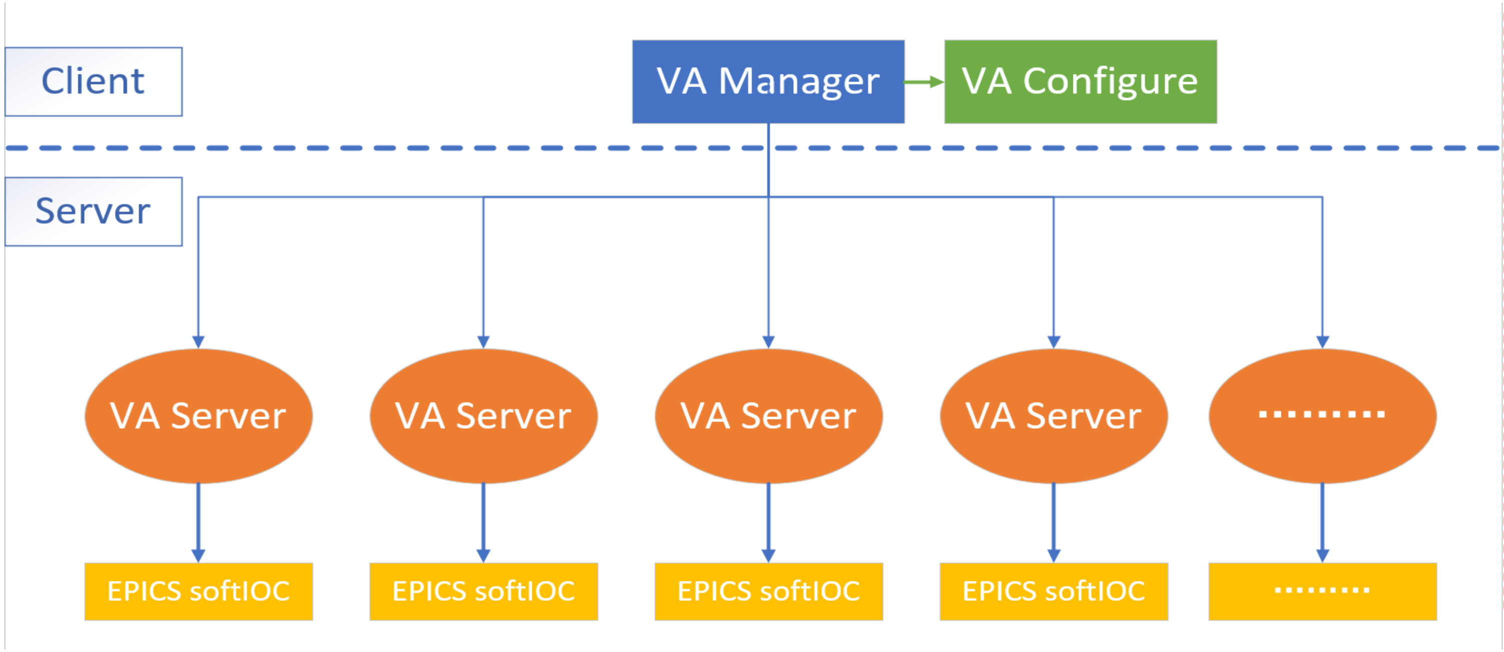
HLA Development for HEPS

LINAC	Transfer Line	Booster	Storage Ring
➢ Linac Virtual Accelerator ✓	➢ LB Virtual Accelerator ✓	➢ Lattice Preparation ✓	➢ Lattice Preparation ✓
➢ Transimission Efficiency ✓	➢ Emittance Measurement ✓	➢ Booster Virtual Accelerator ✓	➢ Booster Virtual Accelerator ✓
➢ Emittance Measurement ✓	➢ Orbit Correction ✓	➢ First Turn Measurement ✓	➢ First Turn Measurement ✓
➢ Orbit Correction ✓	➢ Beam-based Alignment ✓	➢ Dispersion Measurement ✓	➢ Dispersion Measurement ✓
➢ Phase Scan ✓	➢ Local Orbit Correction ✓	➢ Chromaticity Measurement ✓	➢ Chromaticity Measurement ✓
➢ Energy Spread Measurement ✓		➢ Tune adjustor ✓	➢ Global Orbit Correction ✓
➢ Beam-based Alignment ✓		➢ Global Orbit Correction ✓	➢ Local Orbit Correction ✓
➢ Linac Control ✓		➢ Local Orbit Correction ✓	➢ Beam Lifetime ✓
		➢ Beam Lifetime ✓	➢ Orbit Display ✓
			➢ Slow Orbit Feedback ✓
			➢ K-Value adjustor ✓
			➢ Fast BBA ✓

All HLAs are developed based on *Pyapas*
More than 40

Design of A Multi-User Virtual Accelerator

➢ Create virtual hardware to simulate real machines, providing corresponding PV channels and beam information



C/S Architecture

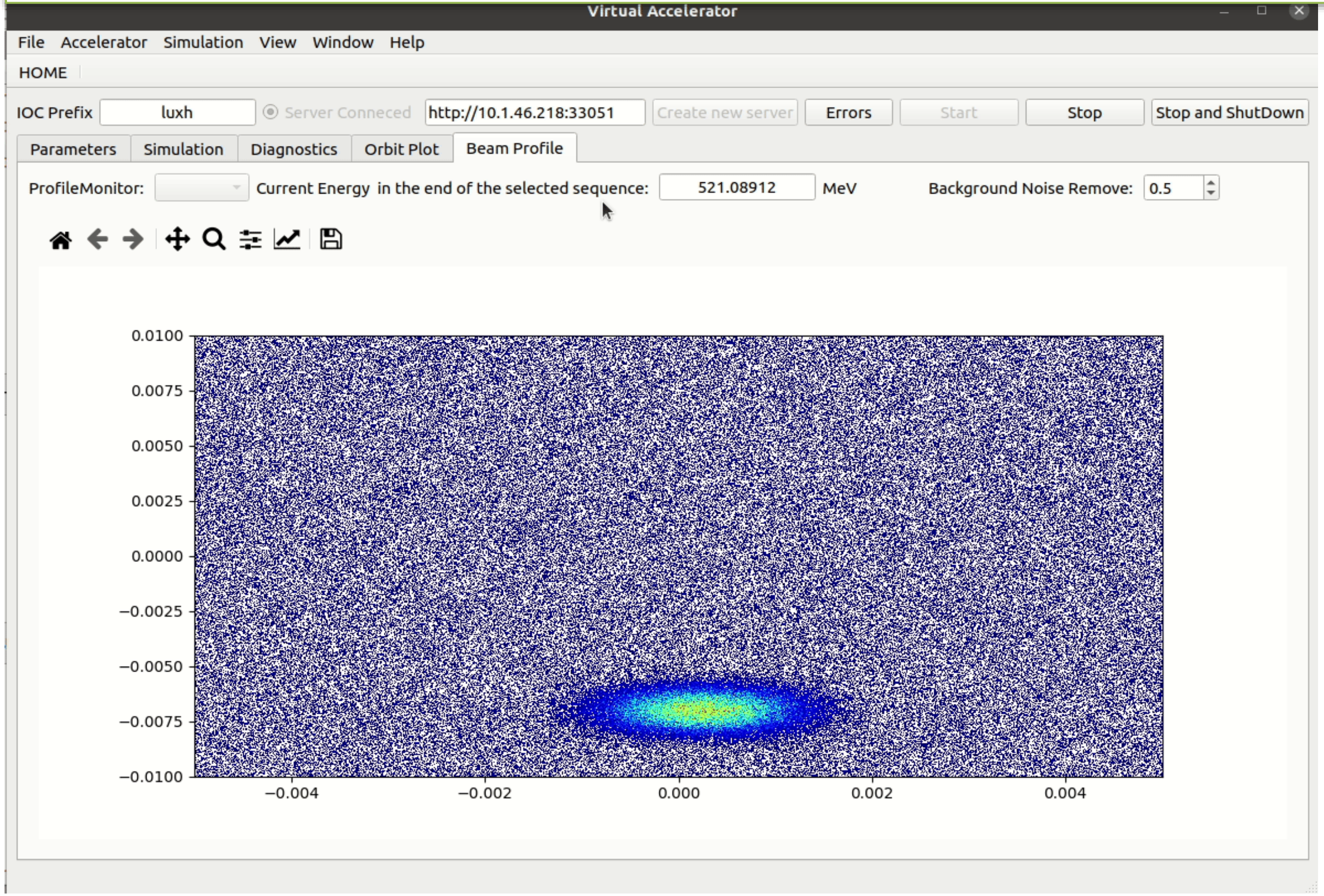
- ✓ **Multiple tasks running concurrently**
- ✓ **Multiple users operating simultaneously**
- ✓ **Unified management with one-click start and stop**

Within the same network environment or on the same server, any number of virtual accelerators can be launched without conflicting with each other

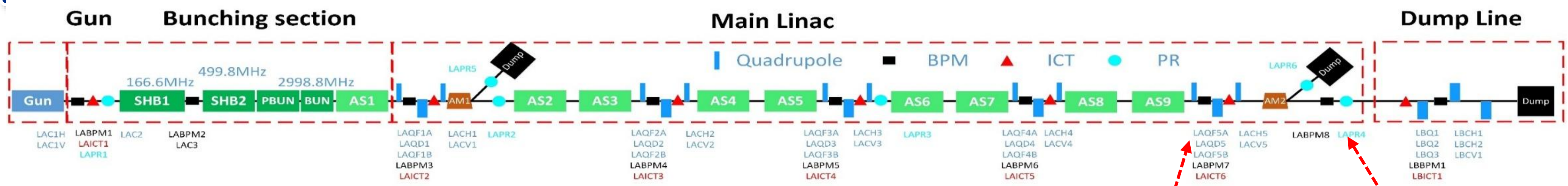
Virtual Accelerator of HEPS

Generate 'real-time' data for testing HLAs

- Single Particle Tracking
- Multi-Particle Tracking
- Vritual PR
- Full feature IOC
- Errors included



Example of a Linac Emittance Measurement Application



$$\begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ \vdots & \vdots & \vdots \\ R_{n1} & R_{n2} & R_{n3} \end{pmatrix} \begin{pmatrix} \beta\varepsilon \\ \alpha\varepsilon \\ \gamma\varepsilon \end{pmatrix}_0$$

$$R_{n1} = M[1,1] * M[1,1]|_{step=n}$$
$$R_{n2} = -2 * M[1,1] * M[1,2]|_{step=n}$$
$$R_{n3} = M[1,2] * M[1,2]M[1,2] * M[1,2]|_{step=n}$$

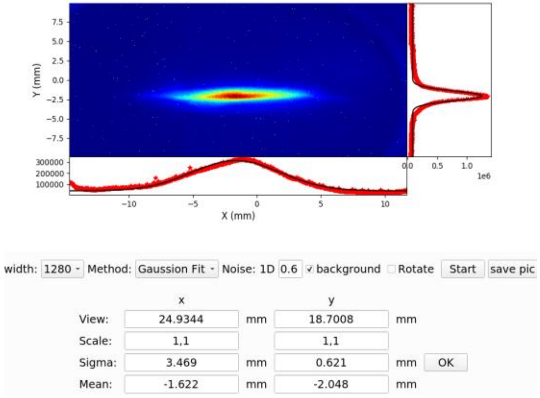
Measurement point

Profile Monitor

Fitting online

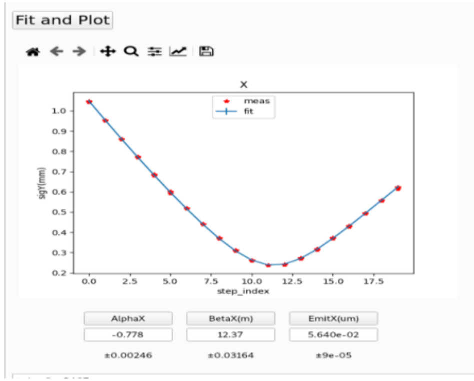
Real-time online calculation

Measured Result




```
calculator = TwissCalculator()
init_parameter = TwissParam()
accelerator_manager = AcceleratorManager(lattice_path)
accelerator = Accelerator_manager.get_accelerator()
sequence = Accelerator.get_sequence('Linac')
load_init_params(sequence, init_parameter)
sequence.sync()  # get the live value from real machine

# call physical model
physics_cell = write_cell(sequence, 'ocelot')
results = calculator.calc(physics_cell, init_parameter)
```

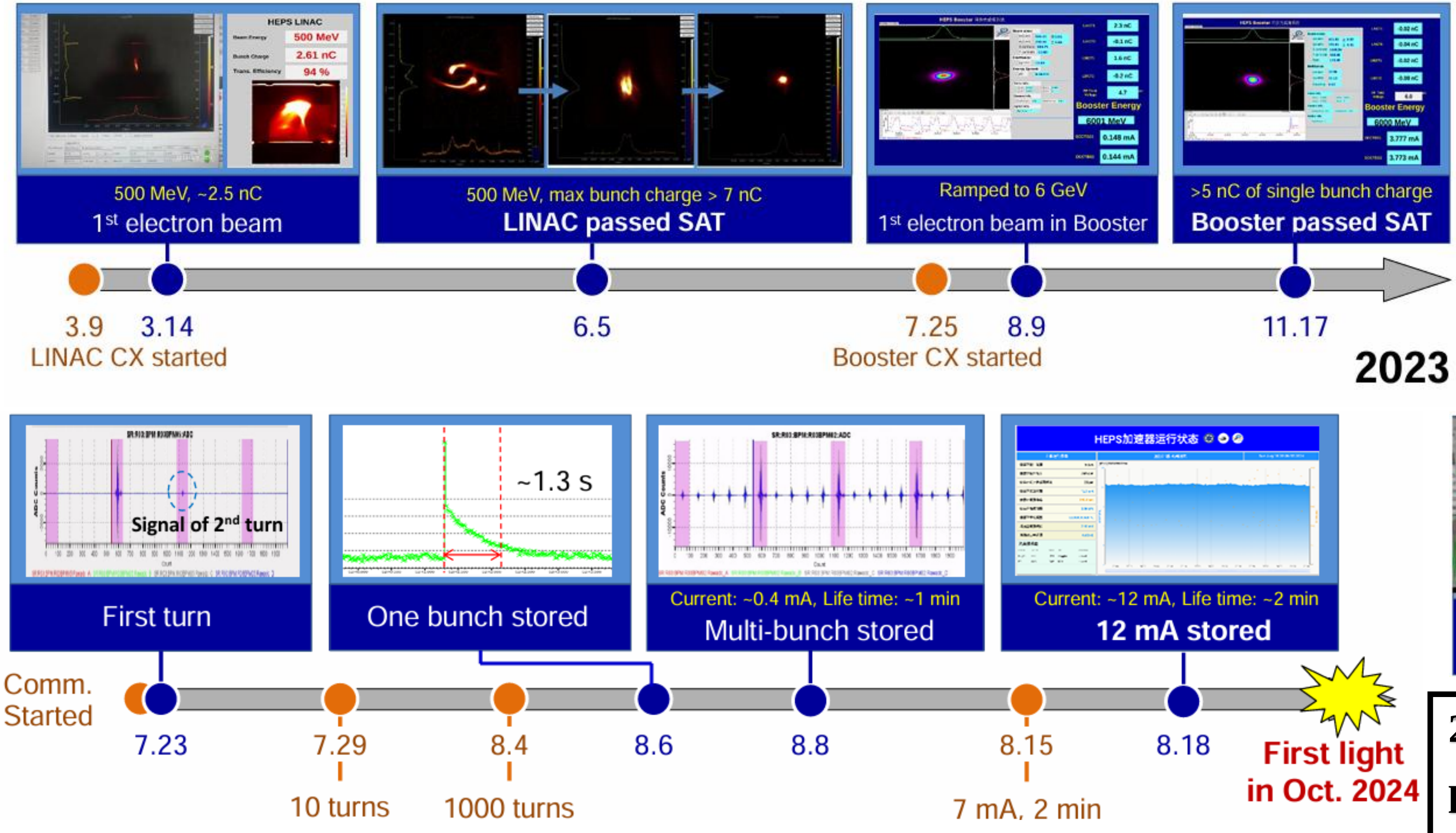


Pyapas-based HLAs for HEPS

- All the HLAs are developed based on Pyapas
- All the HLAs will be tested on the VA before deployment

Y.L. Zhao et al., Journal of Instrumentation, 2023; Y.M. Peng et al., Journal of Instrumentation, 2023;

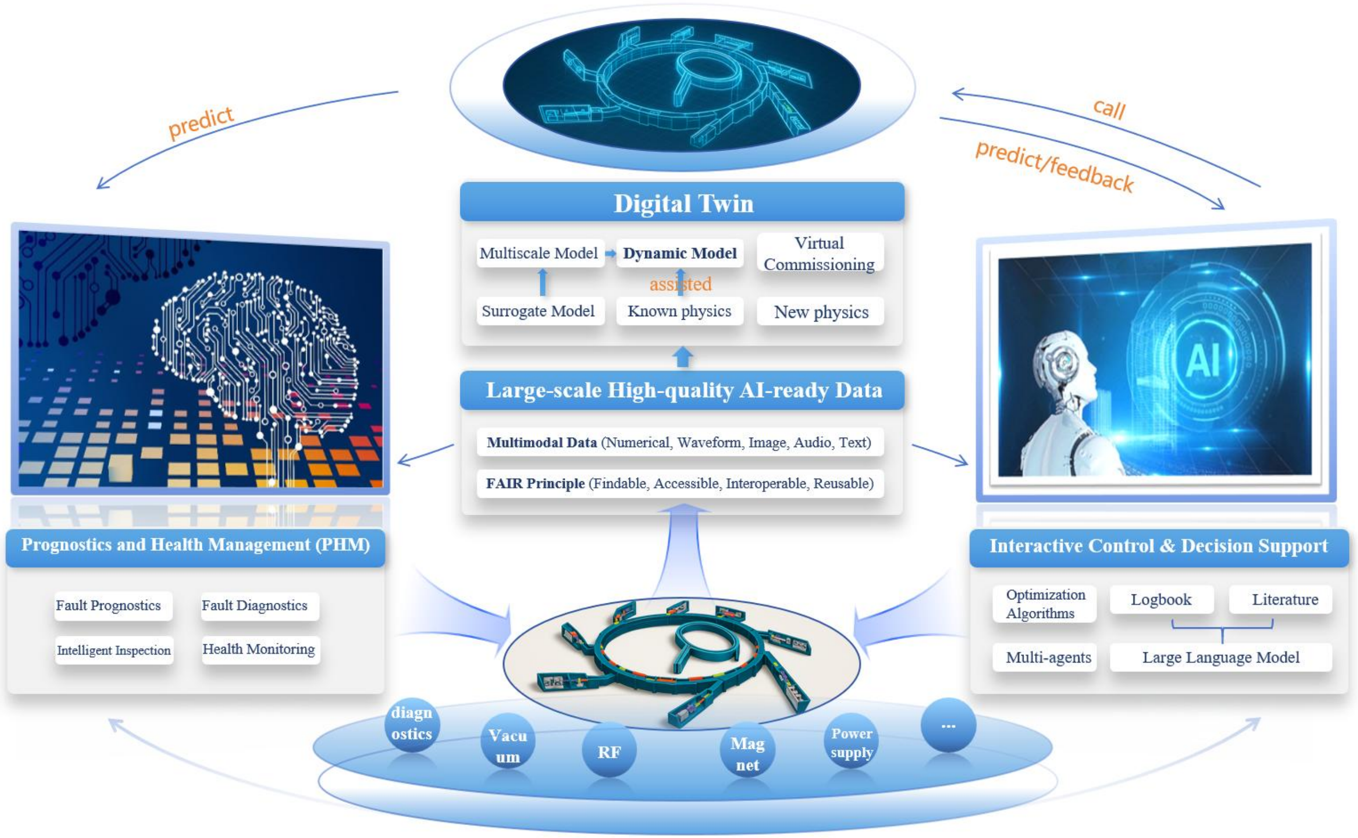
Beam Commissioning Milestones of HEPS



2025.10
passed the process qualification and acceptance

- HLA development requirements
- *Pyapas* for HLA development
- HLAs development for HEPS based on *Pyapas*
- **Machine Learning Application at HEPS**
- Conclusion

AI for Facilities: A Blueprint for Intelligent Accelerators



AI-Ready Data
Standardized, reusable, traceable

Digital Twin
Virtual–real mapping, closed-loop optimization

Fault Prediction
Early detection, early warning, early intervention

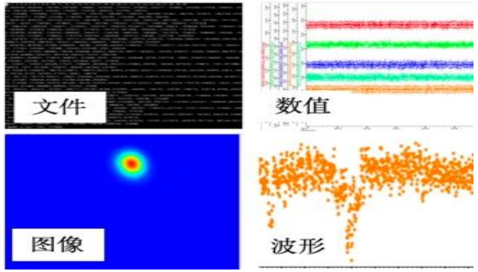
Intelligent Control
Automatic optimization, decision support,
human–machine collaboration

Technical Support

- **High-Performance Computing Platform** – Simulation and Training Acceleration
- **Data Platform and Storage** – Time-Series / Relational / File Data
- **AI Algorithms and Model Library** – Surrogate Models and Intelligent Algorithms
- **Open Interfaces and Software Ecosystem** – APIs / SDKs / Toolchains
- **Network and Security Assurance** – Access Control / Isolation / Auditing

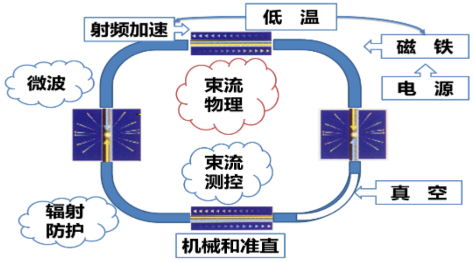
AI Development in Particle Accelerators: Data as the Foundation

Core Challenge: Large volumes of data already exist, yet a clear gap remains before they can directly support AI modeling, inference, and closed-loop control.



Diverse Data Types

Distributed and Heterogeneous Data Sources



Strong System Coupling

The key question is not whether data exist, but whether the data are standardized, time-aligned, and quality-controlled.

1. Heterogeneous Data Sources
2. Inconsistent Standards
3. Insufficient Sample Coverage
4. Difficulties in Correlation Modeling

Key Priorities for Building AI-Ready Datasets

- Unified Data Acquisition
- Time Synchronization
- Data Standardization
- Quality Labeling
- Traceability and Reusability

FARAD: Generating AI-Ready Datasets under FAIR Principles

Based on the FAIR principles, heterogeneous raw data from multiple sources are automatically governed, integrated, processed, and packaged to generate standardized, traceable, and reusable AI-Ready datasets.

Heterogeneous Multi-Source Raw Data

- Time-Series / PV
- Waveforms / Images
- Logs / Files
- Equipment / Conditions

FARAD

Data Governance and Generation Platform

- Task Definition
- Data Acquisition
- Data Cleaning
- Data Integration
- Feature Processing
- Dataset Packaging

AI-Ready Datasets

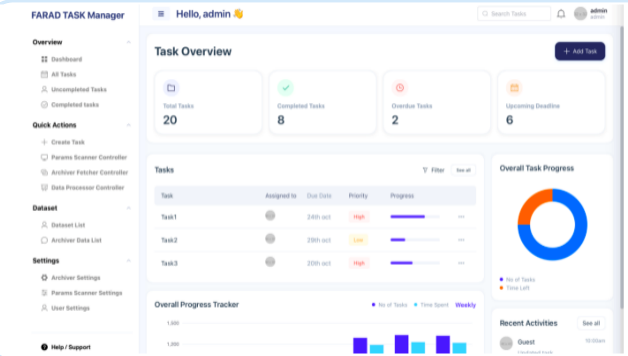
- Standardized
- Traceable
- Reusable
- Trainable

Core Capabilities

1. **Task-Driven**
   Generate datasets by scenario
2. **Multi-Source Access**
   Unified access to operational data
3. **Standardized Governance**
   Cleaning, sync, integration, labeling
4. **One-Click Dataset Generation**

Platform Prototype and Application Validation

FARAD Platform Homepage



Data Task Management and Generation Workflow

FAIR Principles

Findable
Unique IDs
Rich metadata

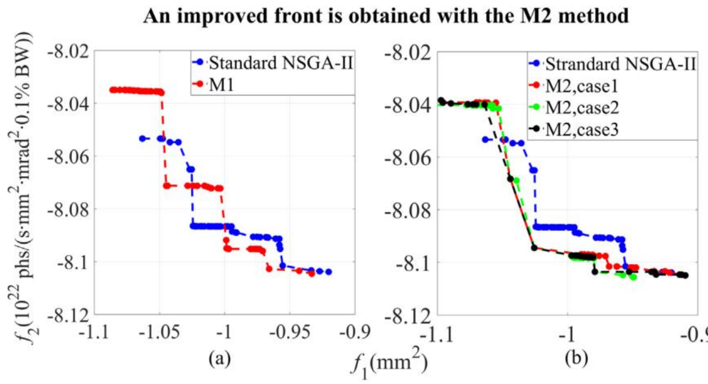
Accessible
Standard protocols

Interoperable
Common formats
Metadata schema

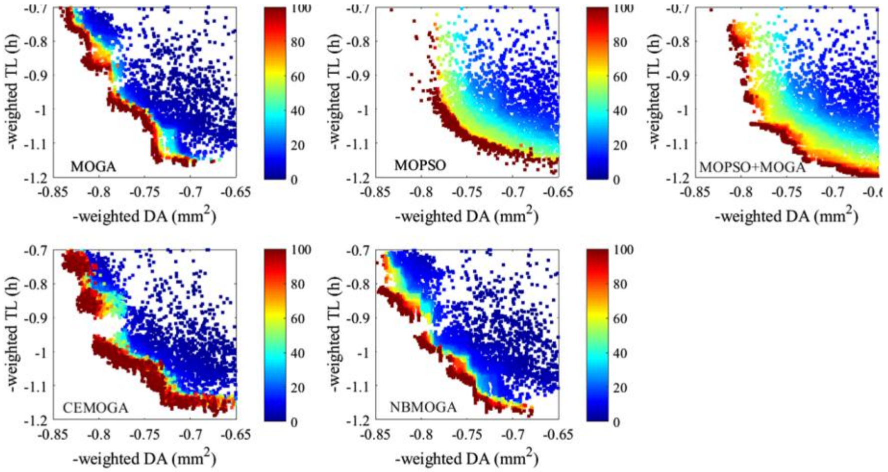
Reusable
Clear licenses
Data provenance

ML-assisted optimization of nonlinear beam dynamics

Two related works improve multi-objective optimization for HEPS lattice design.



ML-enhanced selection of repopulation pushes the brightness-DA front.



NBMOGA evolution for DA + TL optimization.

+0.2 mm

vertical DA improvement

+10%

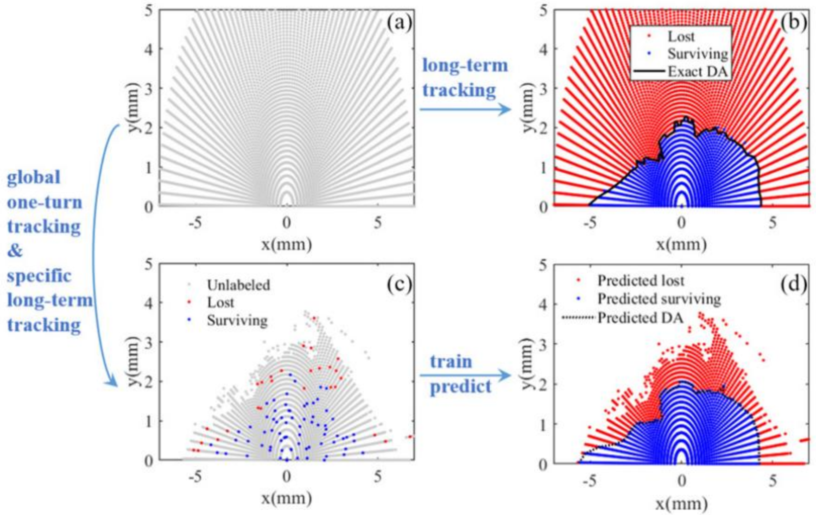
Touschek lifetime gain

ML is used to generate or screen candidate solutions, not to replace the final physics evaluation.

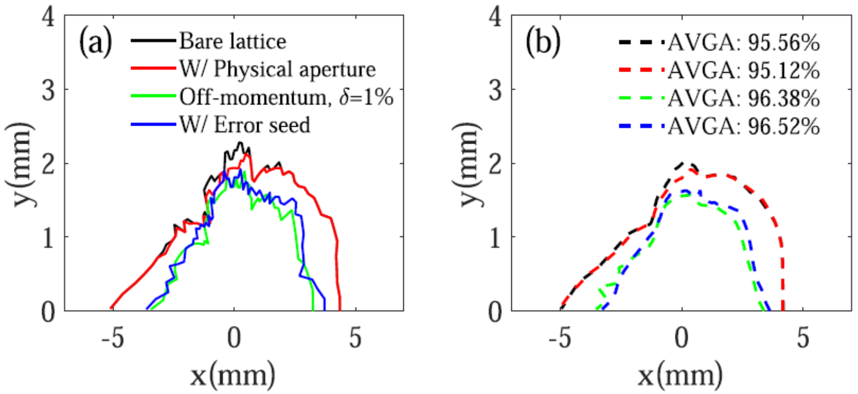
The optimization data accumulated during evolution becomes useful training data for the next generation.

Fast dynamic aperture evaluation

Instead of massive particle tracking, the method tracks a small labeled subset for long-term survival and uses short-turn trajectories to classify the rest.



Workflow: conventional long-term tracking vs ML-predicted DA boundary



Real DA vs. Predicted DA

5-10% long-term tracking

~95% prediction accuracy

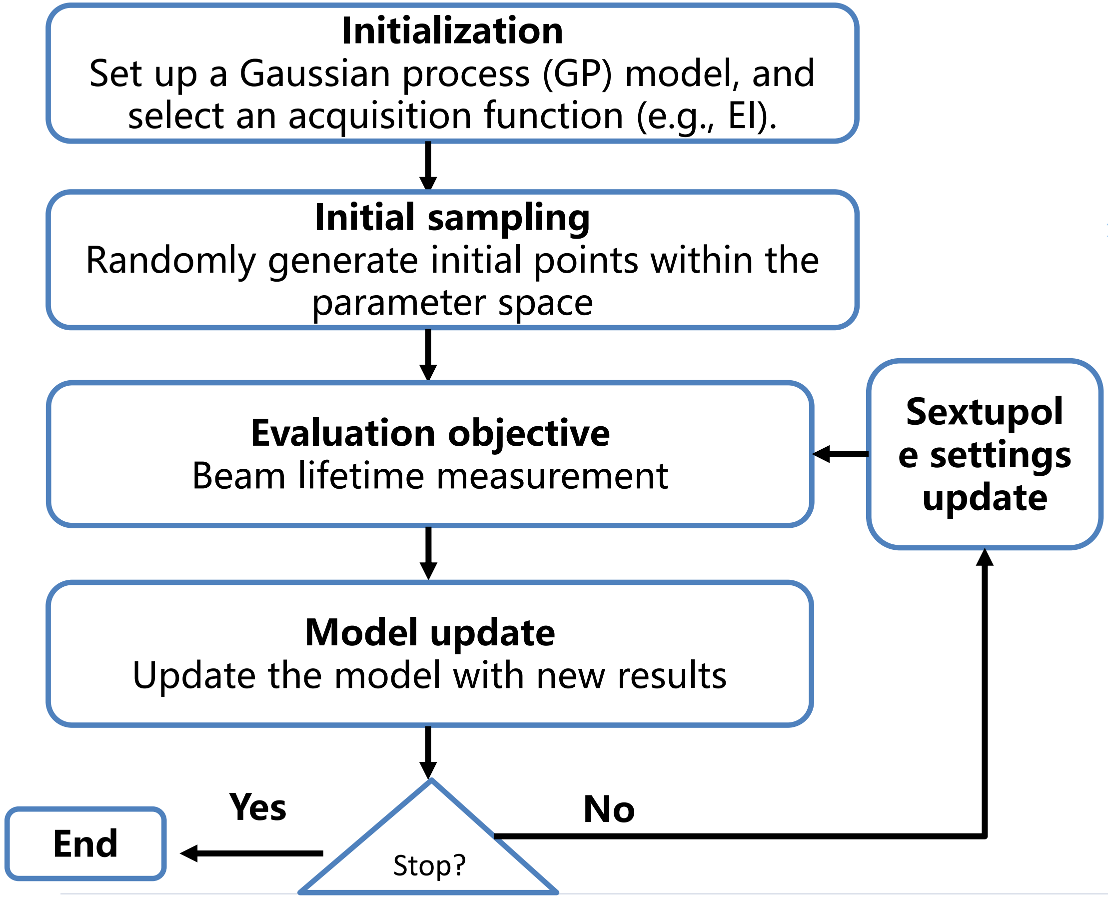
~10x DA evaluation acceleration

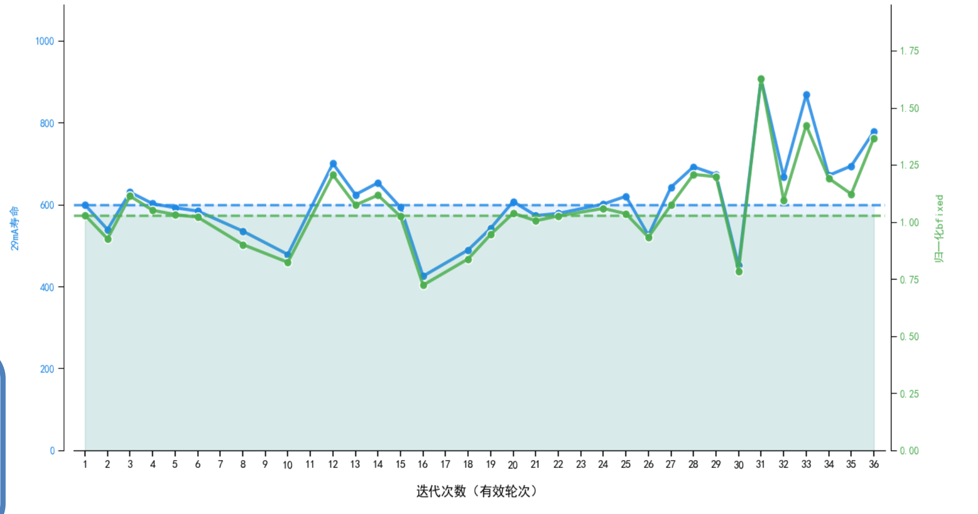
A similar work is implemented for fast local momentum acceptance (LMA) evaluation for the HEPS

Wan & Jiao, New Journal of Physics 24 (2022) 063030; Bao et al., Physica Scripta 100 (2025) 076018.

Bayesian optimization for beam lifetime

BO is implemented to optimize beam lifetime by tuning the sextupole strengths



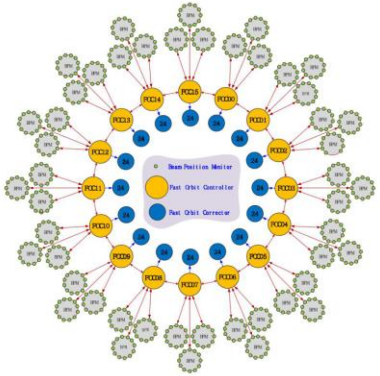


Beam lifetime is improved after ~30 iterations.

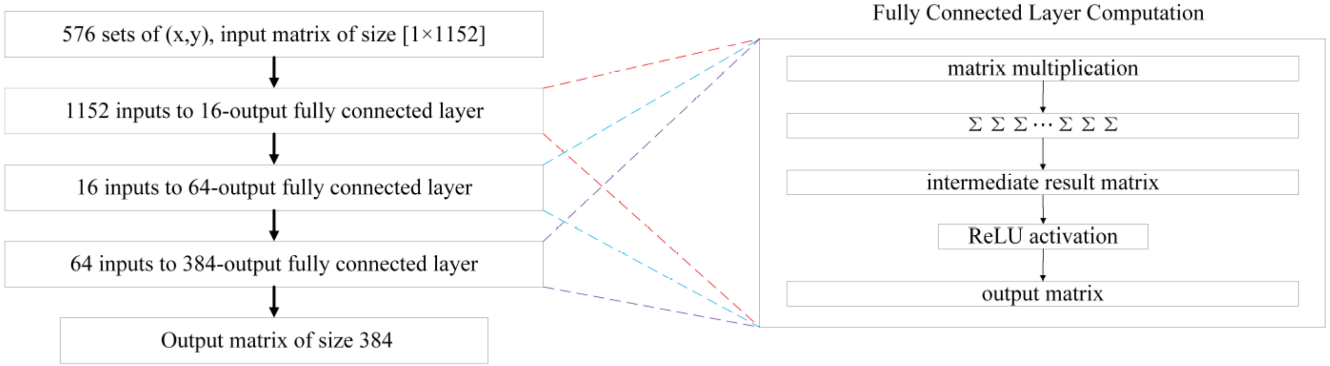
Currently, beam lifetime is determined by many factors. Simply changing sextupole settings may be insufficient for the optimization.

ML-based fast orbit feedback for HEPS

Use neural network to replace explicit response-matrix/PID dependence

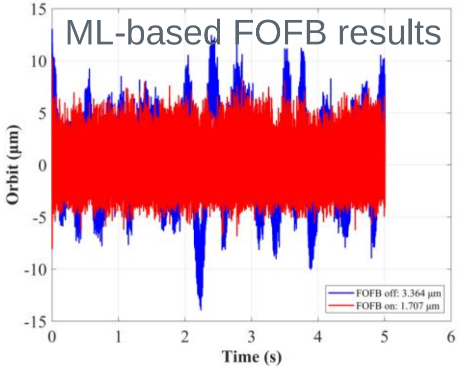


HEPS FOFB topology: 576 BPMs, 384 fast correctors.



MLP algorithm structure for FPGA deployment.

ML-based FOFB results

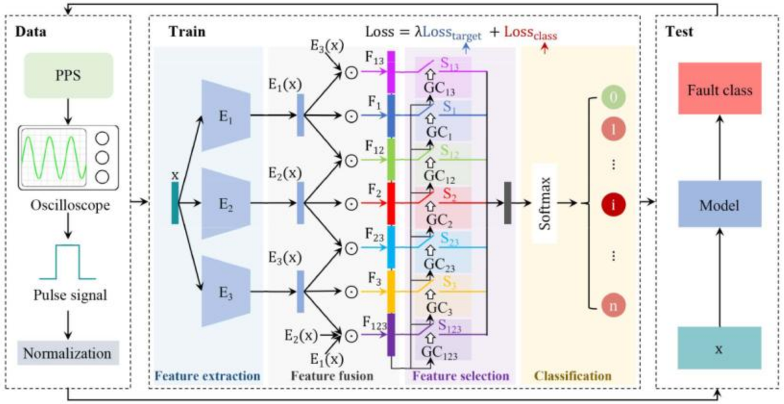


The current HEPS FOFB system relies entirely on a response-matrix-based approach. We are exploring how AI can be effectively integrated into orbit-feedback tasks.

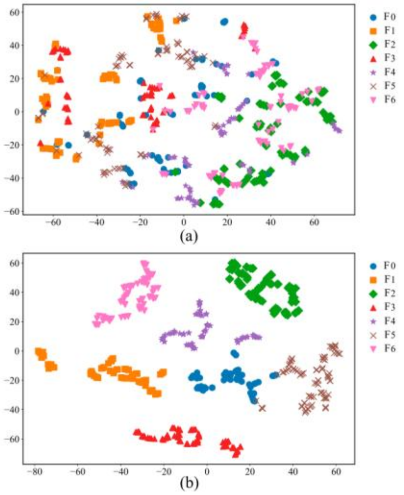
Li et al., Journal of Instrumentation 21 (2026) T03012.

Fault identification for HEPS pulse power supplies

A convolutional neural network (CNN) is designed to identify early parameter deviation and IGBT open-circuit faults from single pulse waveforms, under strict real-time requirements.



CNN structure: multiscale feature extraction, fusion, dynamic selection, classification.



Learned feature separate fault classes

Multiscale	Fusion	Dynamic	Feature	Model	ACC	Variants
✗	✗	✗	F_1	A	0.9680	0.0187
✗	✗	✗	F_2	B	0.9573	0.0294
✗	✗	✗	F_3	C	0.9760	0.0107
✓	✗	✗	$F_1 + F_2 + F_3$	D	0.9760	0.0107
✓	✗	✓	(F_1, F_2, F_3)	E	0.9787	0.0080
✓	✓	✗	F_{12}	F	0.9787	0.0080
✓	✓	✗	F_{13}	G	0.9760	0.0107
✓	✓	✗	F_{23}	H	0.9707	0.0160
✓	✓	✗	F_{123}	I	0.9813	0.0054
✓	✓	✗	$F_1 + \cdots + F_{123}$	J	0.9787	0.0080
✓	✓	✓	$(F_1, \ldots, F_{123})$	MSC-DA	**0.9867**	-

Predictive accuracy

It reached an accuracy of over 98% for early parameter deviation fault identification for electromagnetic emission current source and klystron modulator

- HLA development requirements
- *Pyapas* for HLA development
- HLAs development for HEPS based on *Pyapas*
- Machine Learning Application at HEPS
- **Conclusion**

Conclusion

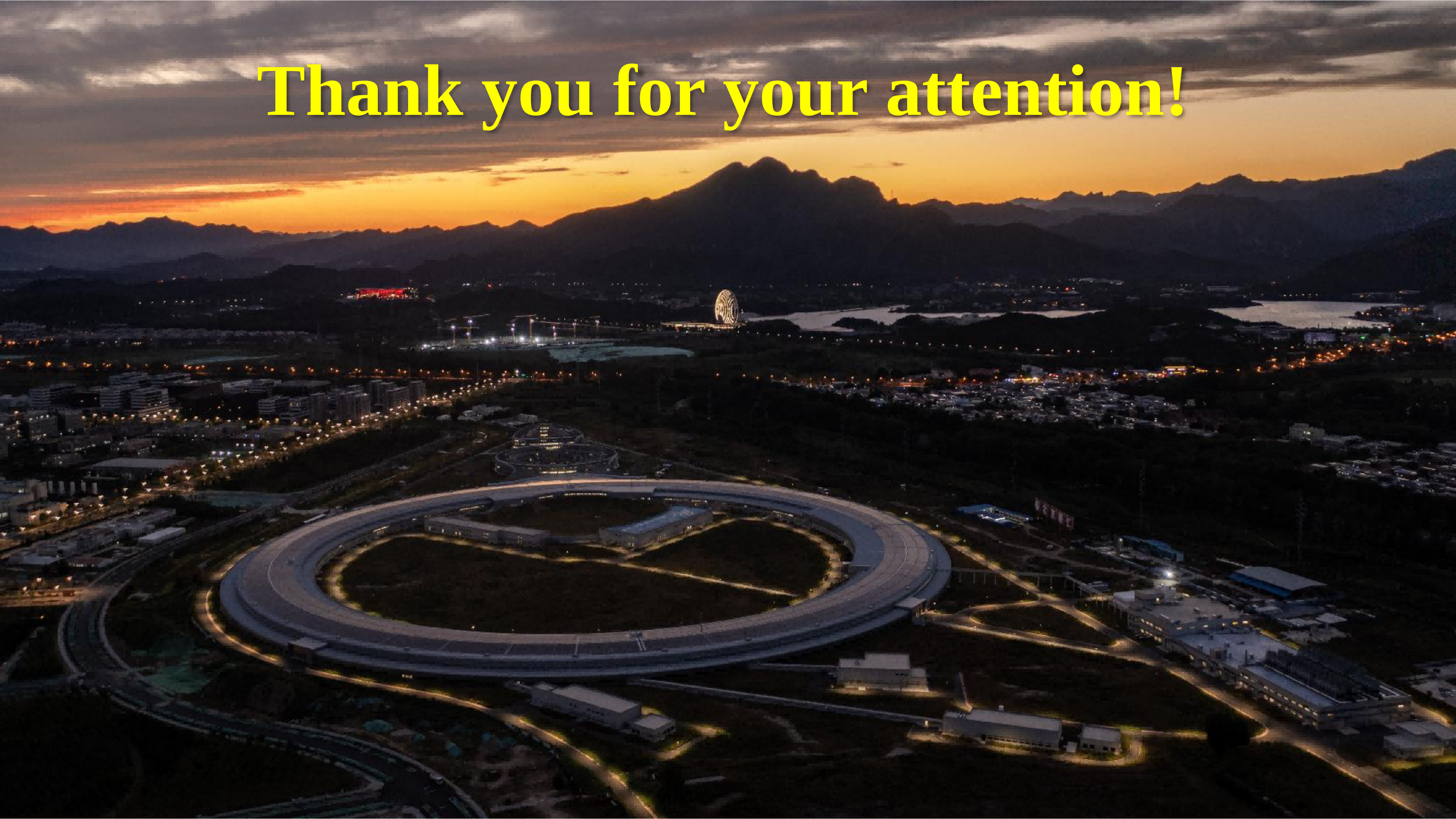
- A brand-new framework, *Pyapas*, was developed based on Python to enhance the efficiency of HLA development.

- *Pyapas* has been successfully applied to the development of HLAs for both HEPS and CSNS-II, greatly improving development efficiency.

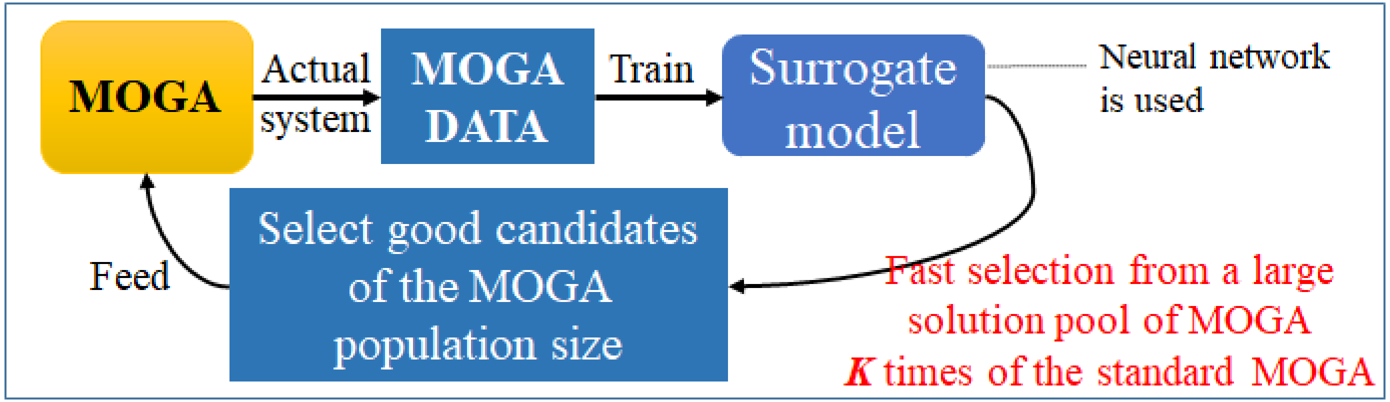
- *Pyapas* is one of the very few frameworks applied to HLAs development for **proton**, **electron**, and **pwfa**.

- Machine learning has been applied in the design and commissioning of HEPS, with promising initial results. Further research will continue in the future.

Thank you for your attention!



Neural network-based MOGA

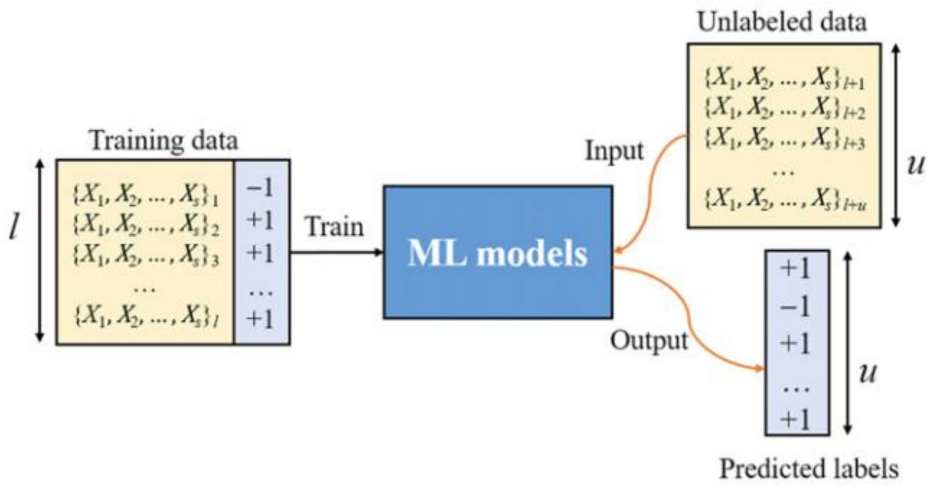


NBMOGA trains a neural network from early generations of MOGA data and uses it to estimate objectives for a much larger offspring pool.

Only a selected set of competitive candidates is evaluated with the real accelerator evaluator, reducing wasted tracking calls.

Compared with MOGA, MOPSO, MOPSO+MOGA, and CEMOGA, NBMOGA improves convergence and diversity; in the HEPS case, TL improves by about 10% at similar DA area.

DA evaluation details



Labels: +1 surviving, -1 lost after long-term tracking; features are short-term trajectories / period-by-period coordinates.

The paper demonstrates approximately 95% accuracy with only 5%–10% labeled long-tracking samples.

The method remains accurate when lattice model details change, because it learns correlations between trajectory features and stability rather than magnet-strength-to-DA regression.

FOFB model details

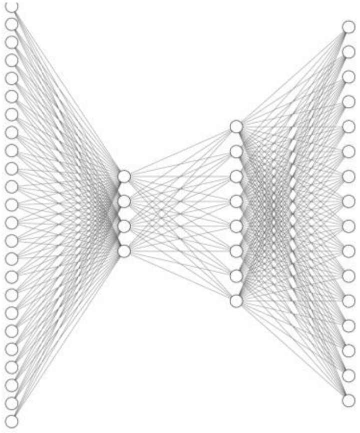
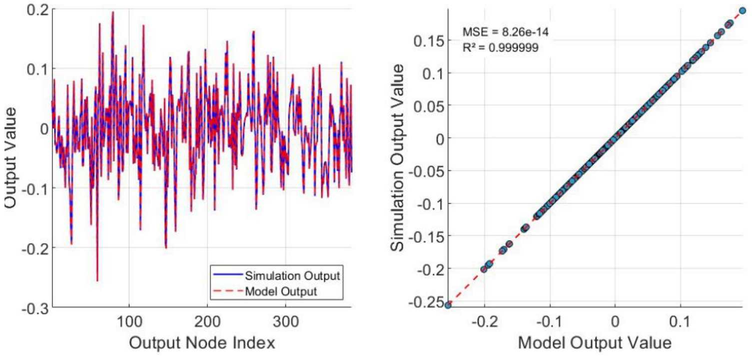


Figure 2. MLP neural network structure.

MLP structure: 1152 input nodes -> [16,64] -> 384 outputs.



10. Comparison between simulation and deployed model outputs (left: overlaid waveforms
ıcy scatter plot), showing excellent agreement with MSE = 0.000083 and R^2 = 0.999999.

Simulation vs deployed model consistency.

Inputs: 576 BPMs in two planes, represented as 1152 position features; outputs: 384 fast corrector values.

Dataset: 440,000 samples; 80% training, 20% testing; corrector values combine measured BPM data and simulation targets.

The selected [16,64] two-hidden-layer MLP balances prediction accuracy, MAC count, and FPGA resource constraints.