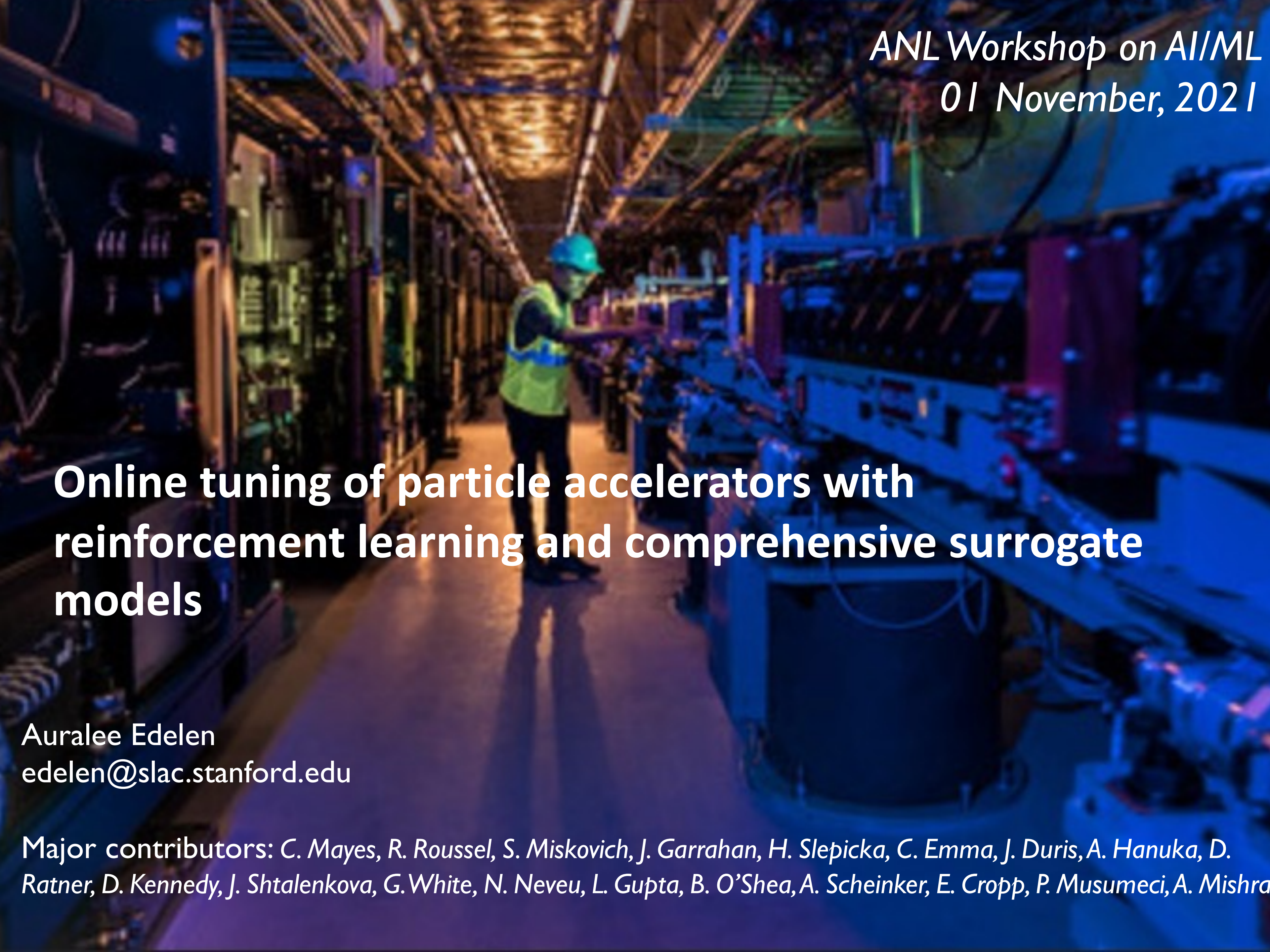
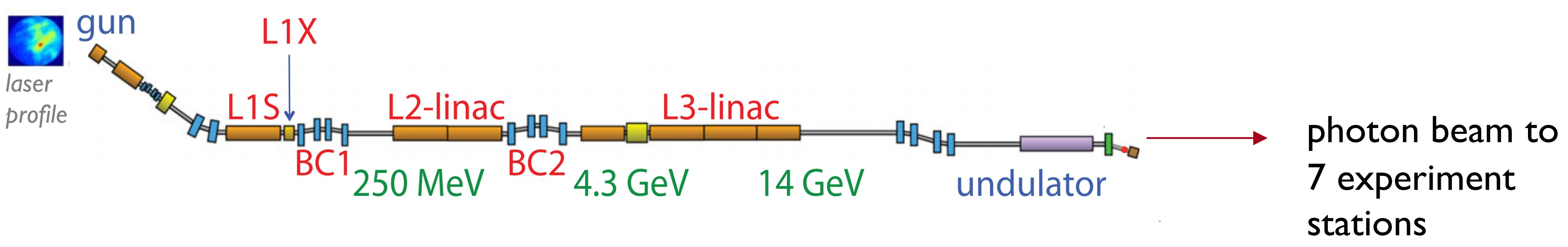


ANL Workshop on AI/ML
01 November, 2021

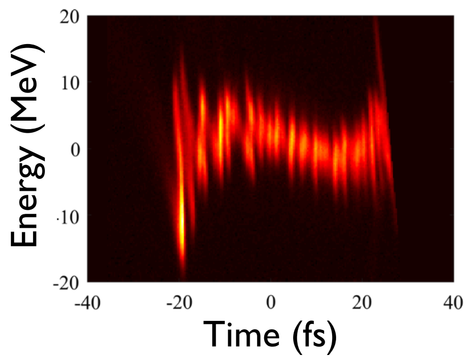
Online tuning of particle accelerators with reinforcement learning and comprehensive surrogate models

Auralee Edelen
edelen@slac.stanford.edu

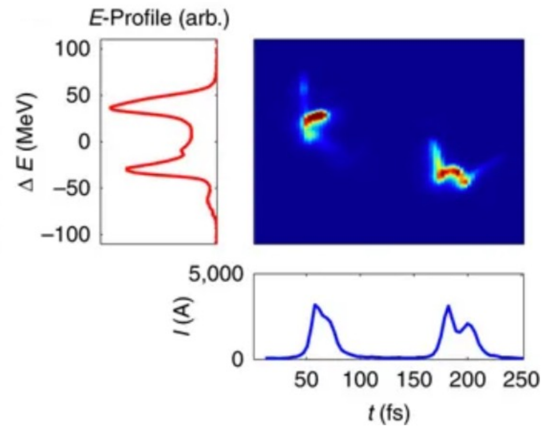
Major contributors: *C. Mayes, R. Roussel, S. Miskovich, J. Garrahan, H. Slepicka, C. Emma, J. Duris, A. Hanuka, D. Ratner, D. Kennedy, J. Shtalenkova, G. White, N. Neveu, L. Gupta, B. O'Shea, A. Scheinker, E. Cropp, P. Musumeci, A. Mishra*



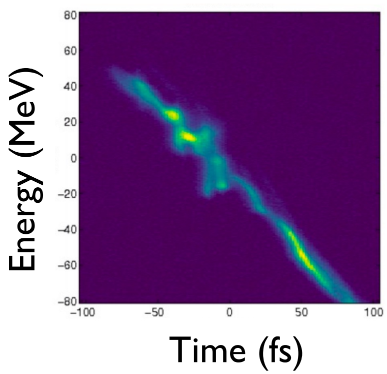
J. Qiang et al, PRAB (2017)



A. Marinelli, et al., Nat. Commun. 6, 6369 (2015)



A. Marinelli, IPAC'18

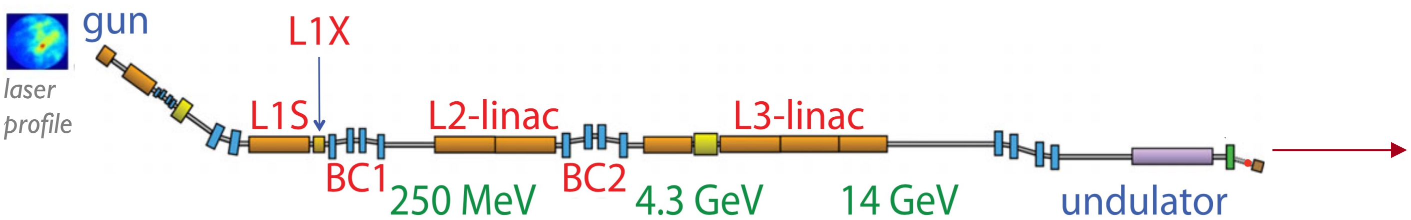


Approximate Annual Budget: $145 million

Approximate hours of experiment delivery per year: 5000

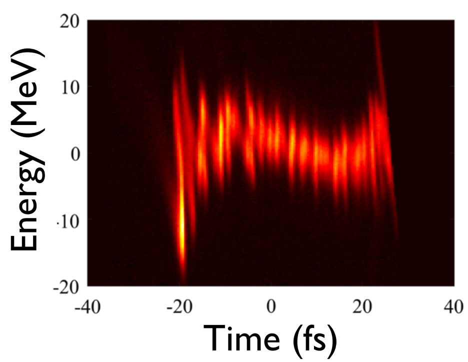
About $30k per experiment hour to run!

400 hours hand-tuning in a year → $12 million value ~10 additional experiments

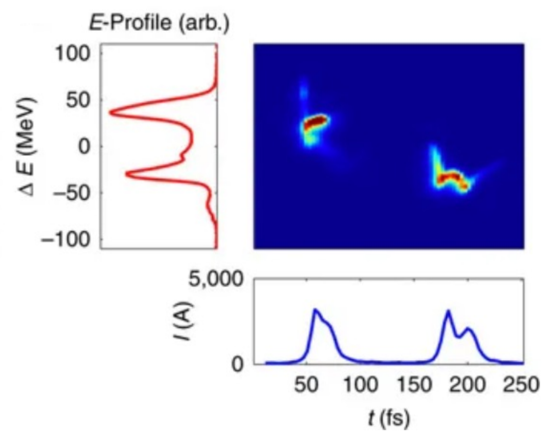


photon beam to 7 experiment stations

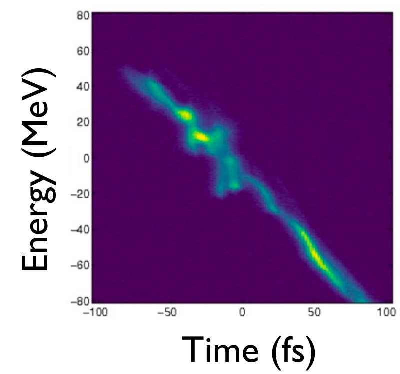
J. Qiang et al, PRAB (2017)



A. Marinelli, et al., Nat. Commun. **6**, 6369 (2015)



A. Marinelli, IPAC'18

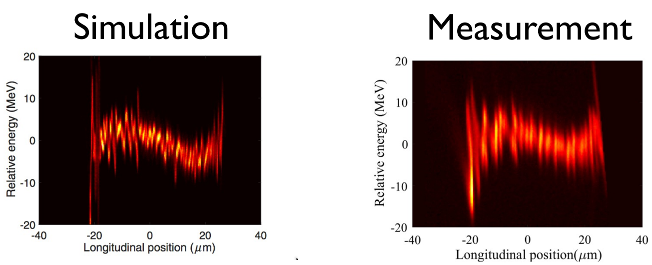


→ *efficient tuning matters to maximize* ***science-per-dollar*** *spent*

→ ***new machine configurations*** *enable new science (e.g. attosecond pulses), but are difficult to bring to operation initially*

Many difficulties…

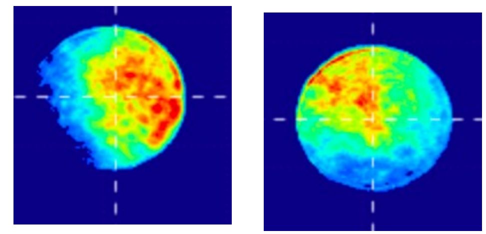
computationally expensive simulations

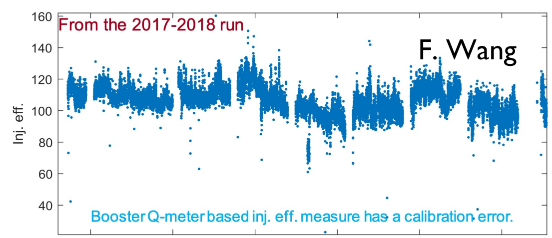


"10 hours on thousands of cores at the NERSC"

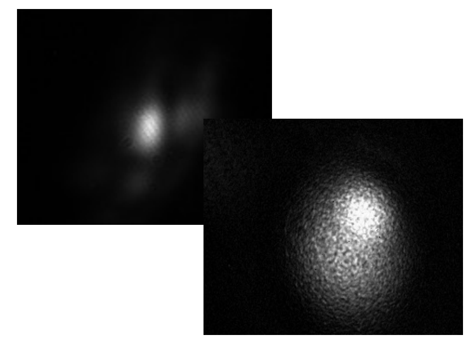
J. Qiang, et al., PRSTAB30, 054402, 2017

fluctuations/noise
(e.g. laser spot)

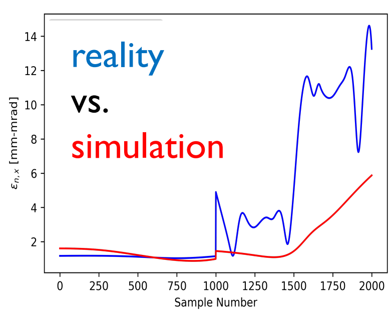




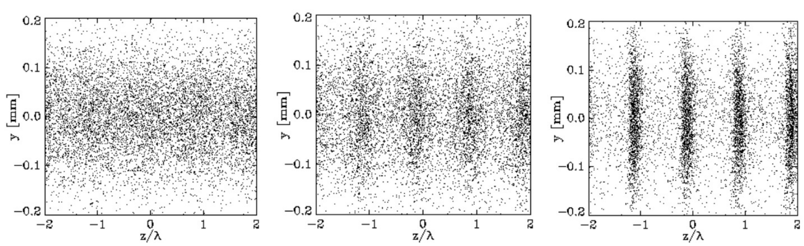
hidden variables / sensitivities



drift over time



many small, compounding sources of uncertainty

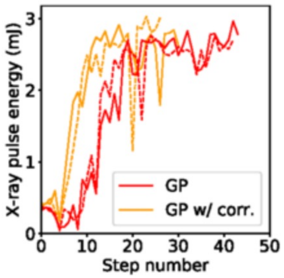


nonlinear effects / instabilities

Many examples where BO and RL have been used in accelerators …

Bayesian Optimization

FEL optimization (Duris et al. 2020, Kirschner et al. PMLR 2019)



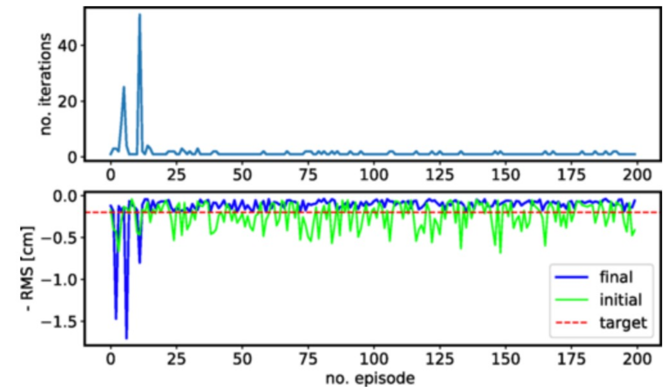
Emittance optimization at SPEAR3 (Hanuka et al. 2021)

Laser plasma accelerators (Jalas et al, PRL 2021, Shaloo et al Nature 2020)

Injection efficiency

Reinforcement Learning

Trajectory control (Kain et al., PRAB 2020)



FEL optimization (O'Shea, et al., 2020)

Magnet power supply (St. John et al., PRAB 2021)

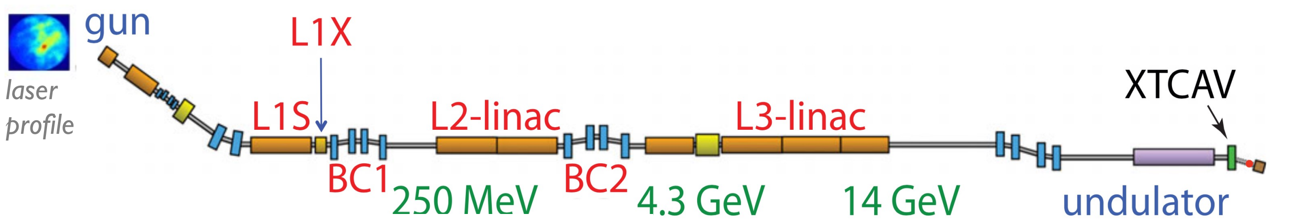
Fast switching between FEL pulse energies (Edelen et al., NeurIPS 2017)

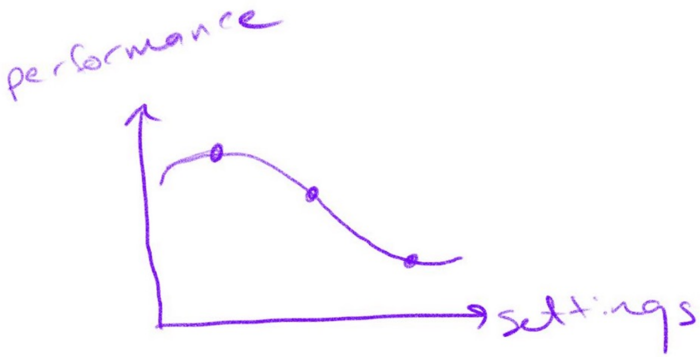
Model Predictive Control

RF resonant frequency (Edelen, et al. TNS 2016)
Ion source control (NIMA 2016)

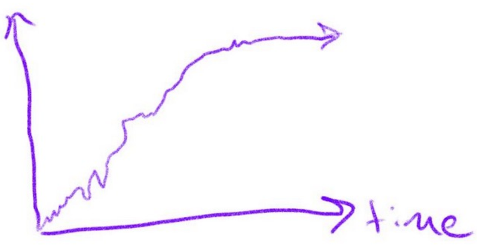
… but when to choose which approach?

Can treat many high-level accelerator tuning problems as either time-dependent or time-independent…





"search for optimal settings"



"game to take actions that maximize performance over time"

as machine drifts over time → reoptimize, or keep playing

Some problems need to be treated as time-dependent...

RF electron gun at the Fermilab Accelerator Science and Technology (FAST) facility

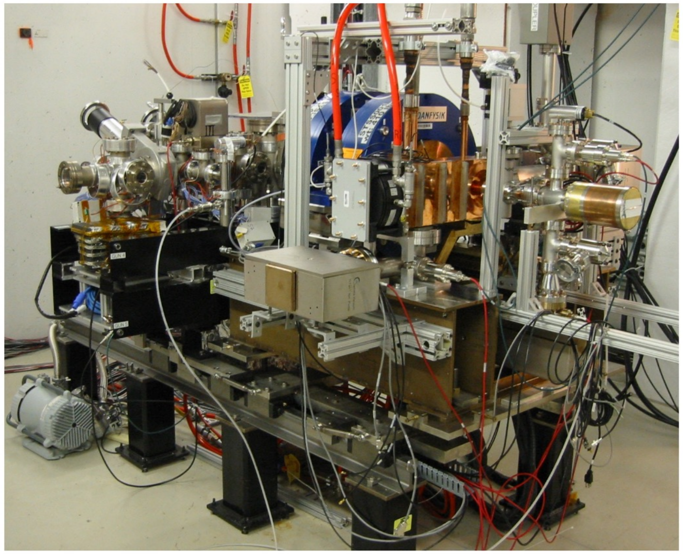
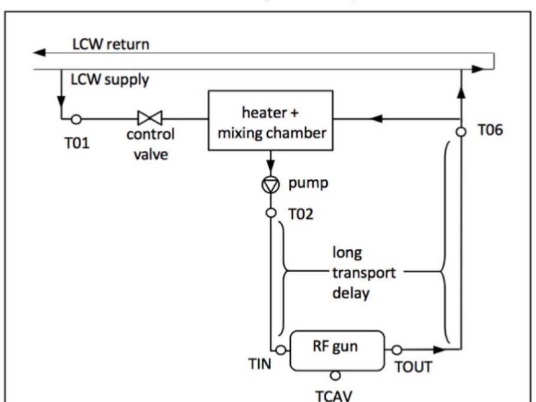


Photo: P. Stabile



Radio frequency quadrupole (RFQ) for the PIP-II Injector Test

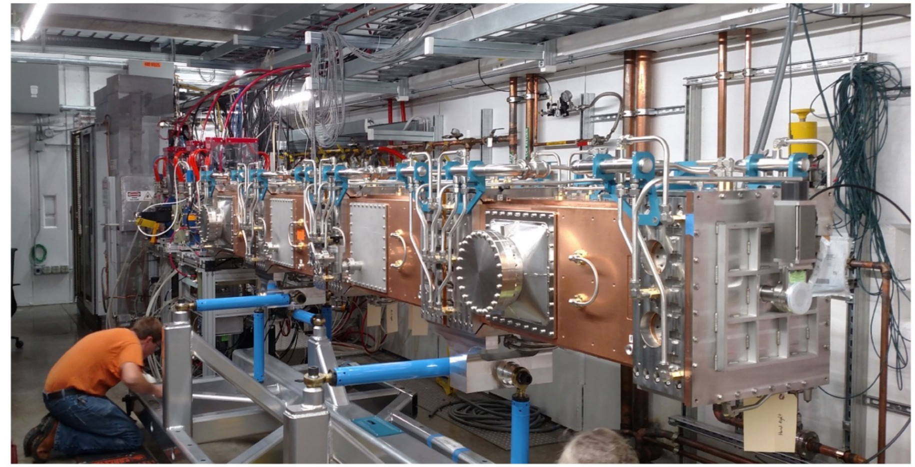
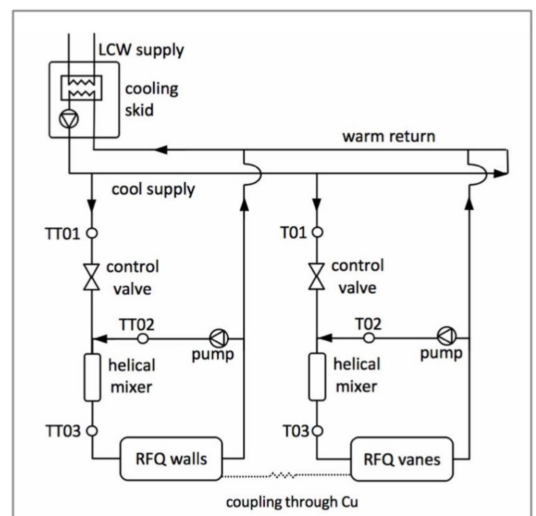
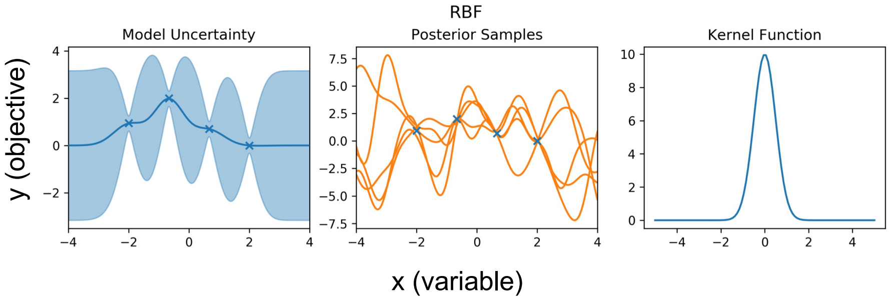
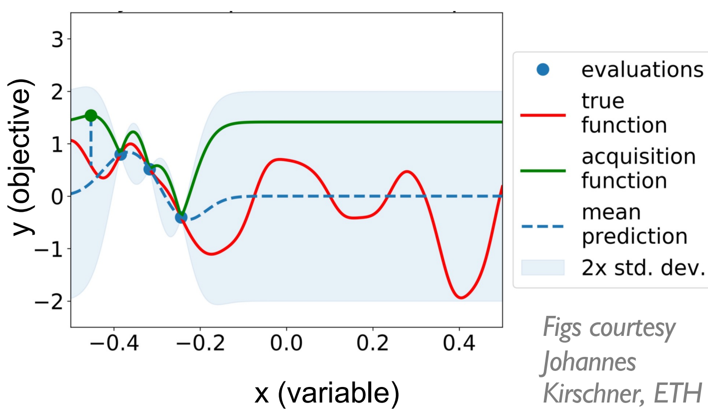


Photo: J. Steimel



Bayesian Optimization

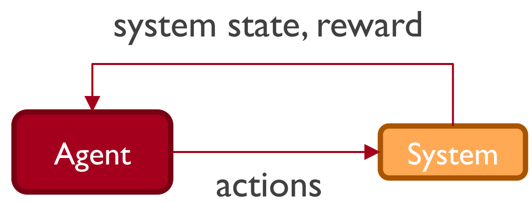


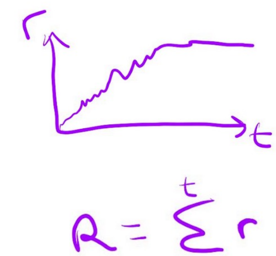


Figs courtesy Johannes Kirschner, ETH

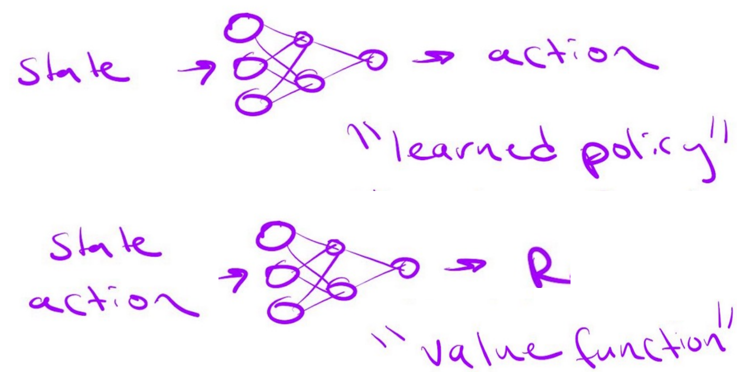
Select sample x → observe objective → refit surrogate model → use model predictions and uncertainty to choose next point according to an acquisition functions

Reinforcement Learning





Many ways to construct agent that learns from reward:

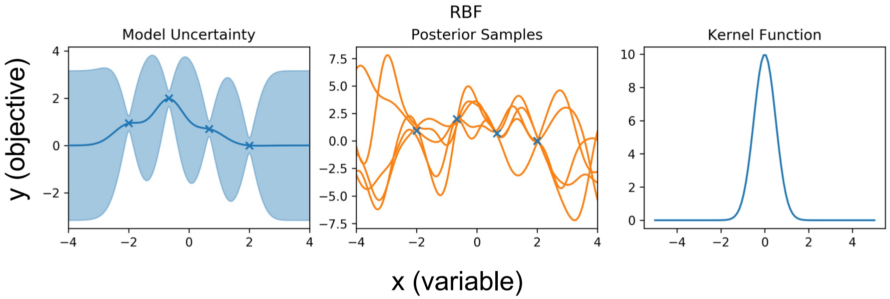


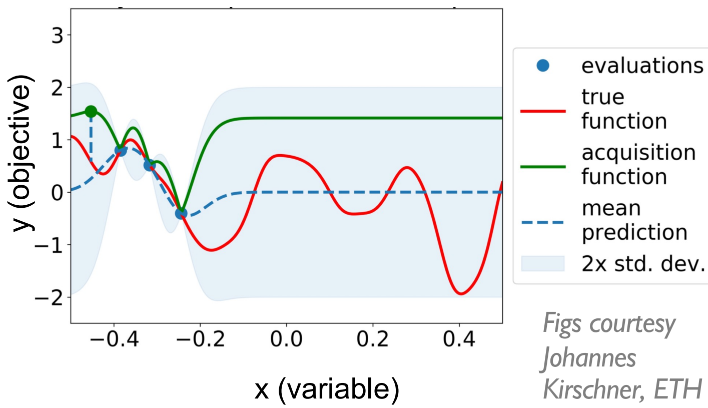
Observe state → take action according to a control policy → observe reward → update policy or value function

Analogous concepts, different terminology and usually different setting:

objective → reward
surrogate model → value function
acquisition function → policy
acquire new sample → take an action

Bayesian Optimization

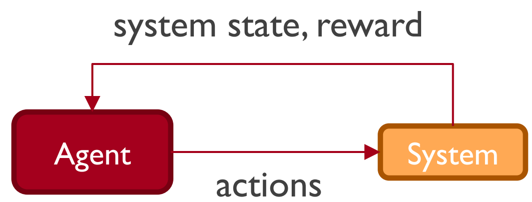


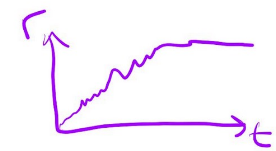


Figs courtesy Johannes Kirschner, ETH

Select sample x → observe objective → refit surrogate model → use model predictions and uncertainty to choose next point according to an acquisition functions

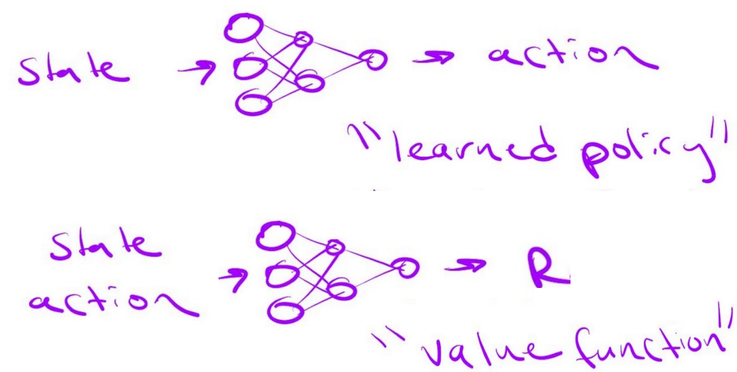
Reinforcement Learning





$R = \sum^{t} r$

Many ways to construct agent that learns from reward:



Observe state → take action according to a control policy → observe reward → update policy or value function

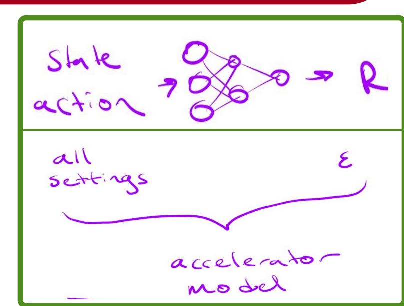
Analogous concepts, different terminology and usually different setting:

objective → reward

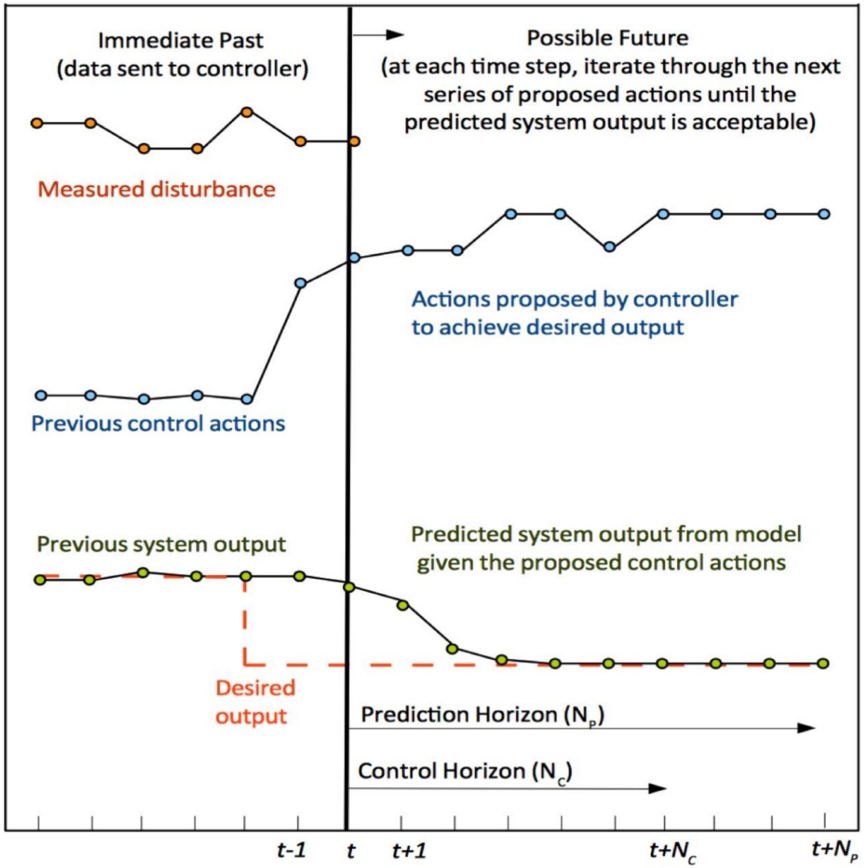
surrogate model → value function

acquisition function → policy

acquire new sample → take an action

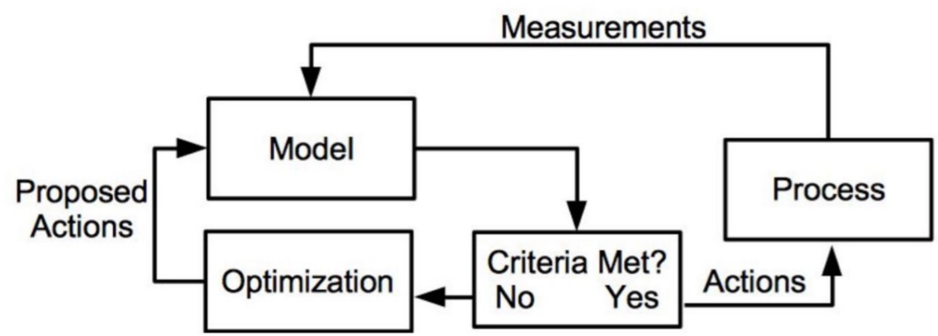


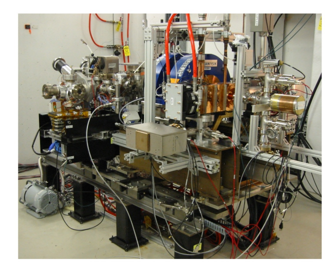
Model Predictive Control



Basic concept:

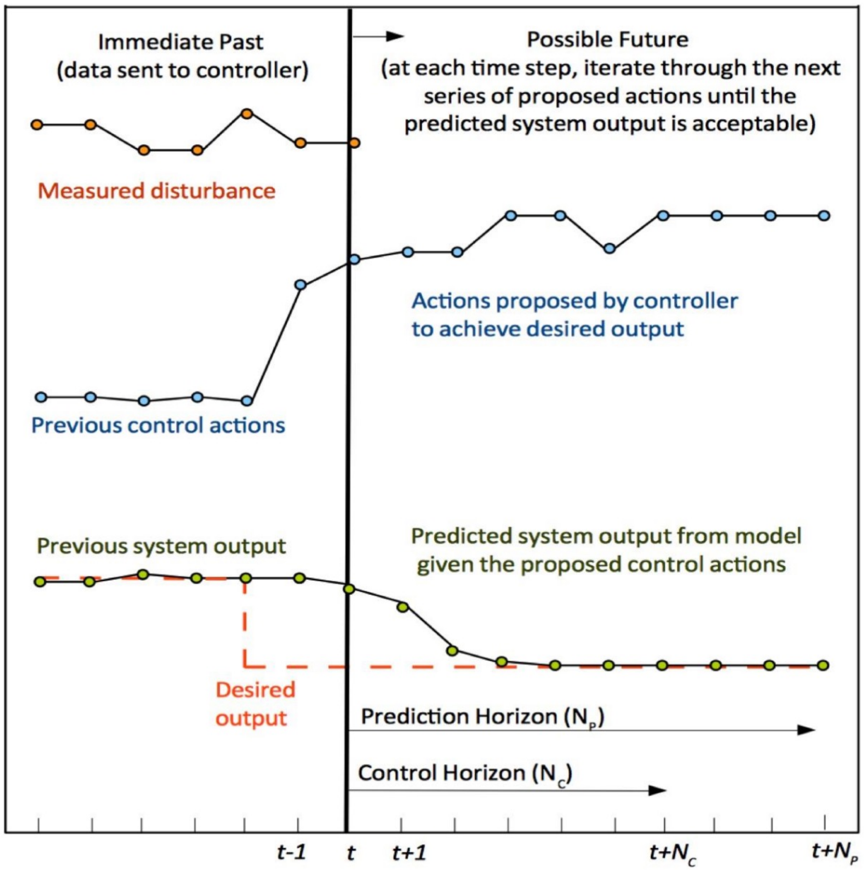
1. Use a predictive model to assess the outcome of possible future actions
2. Choose the best series of actions
3. Execute the first action
4. Gather next time step of data
5. Repeat





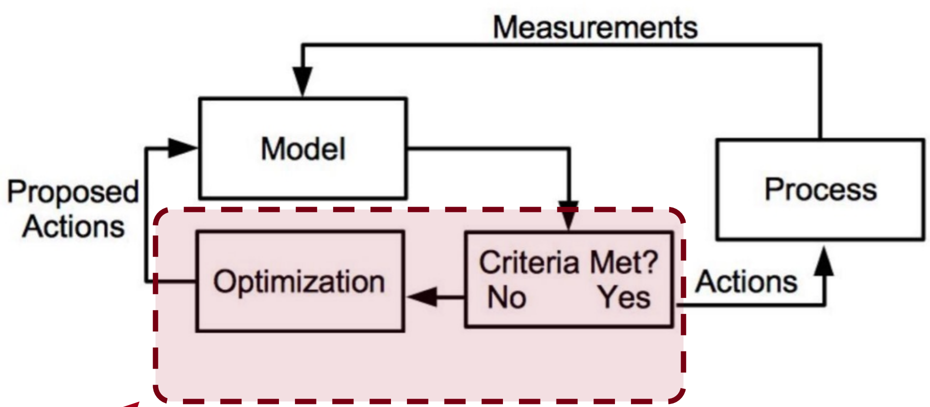
Edelen et al., TNS 2016 https://ieeexplore.ieee.org/document/7454846

Model Predictive Control



Basic concept:

1. Use a predictive model to assess the outcome of possible future actions
2. Choose the best series of actions
3. Execute the first action
4. Gather next time step of data
5. Repeat



- **RL can be thought of as trying to learn the step for optimization over future time horizon** *(choose optimal action at time t to maximize reward / minimize cost over future)*
- **Without time-dependence, becomes optimization over an online system model** *(as we often use in accelerators)*

assumed knowledge of machine

less ← → more

Model-Free Optimization

Observe performance change after a setting adjustment

→ estimate direction toward improvement

Model-guided Optimization

Update a model during each search step

→ use model to help select the next point

Global Modeling + Feedforward Corrections

Make fast / accurate system model

→ provide guess for good settings
→ make predictions about machine

gradient descent
Simplex
Extremum Seeking

← Bayesian optimization

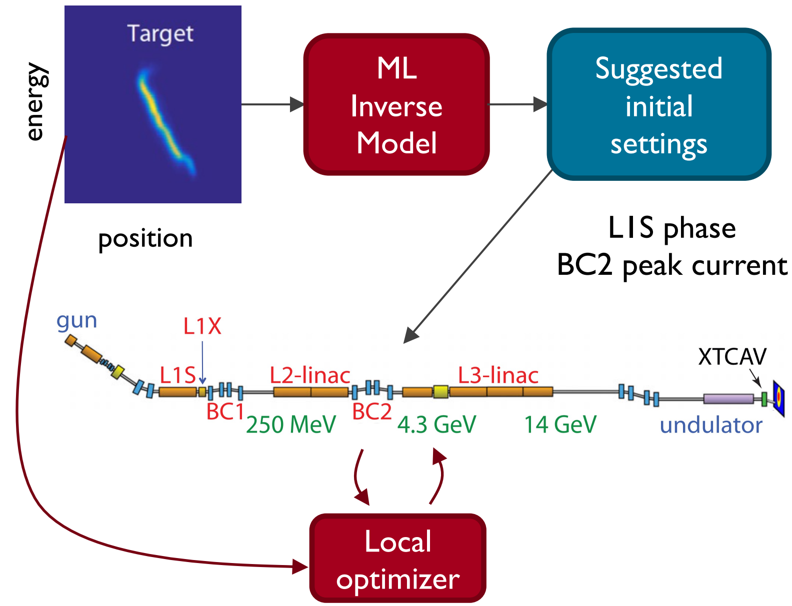
Reinforcement learning →

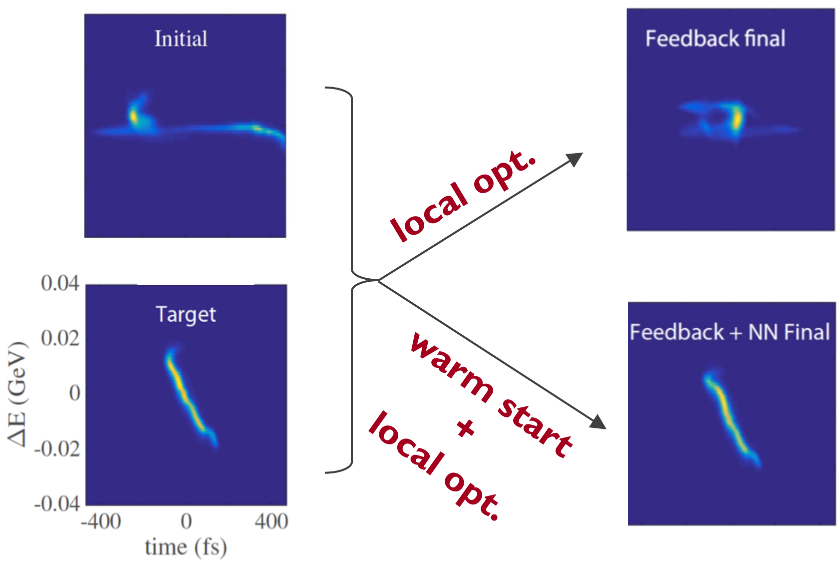
ML system models + inverse models

Inverse models: example from LCLS

- Use global inverse model to give rough suggested settings
  → then fine-tune with local optimizer

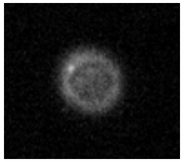
- Preliminary study at LCLS:
  - Two settings scanned *(L1S phase, BC2 peak current)*
  - Compared optimization algorithm with/without warm start

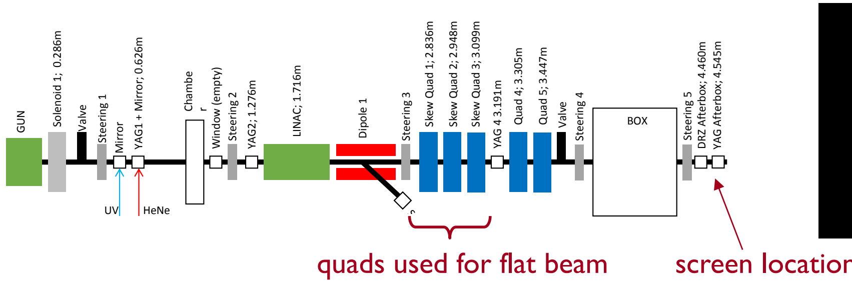


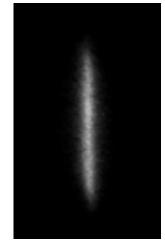


Local optimizer alone was unable to converge → **able to converge after initial settings from neural network**

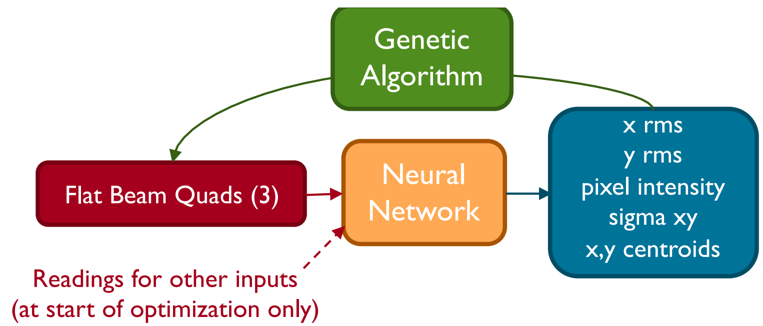
Another way: run optimizer on a learned online model

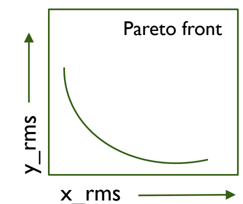




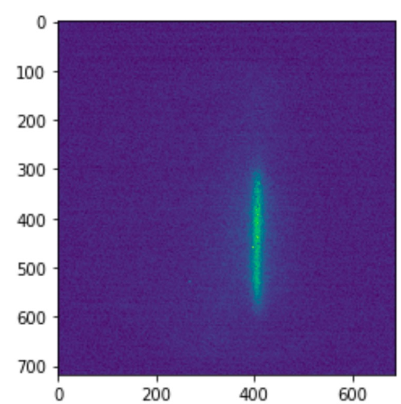


Expert hand-tuning:
10 – 20 minutes



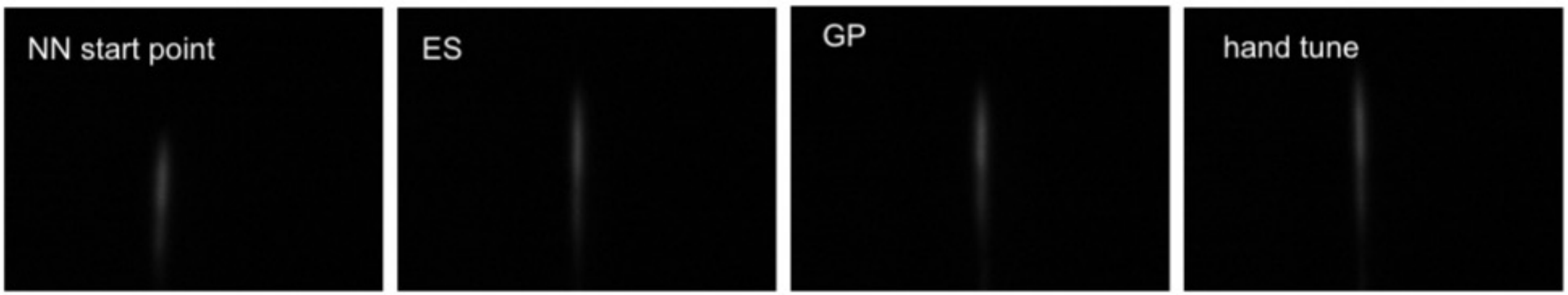


- Round to flat beam transforms are challenging to optimize
- Took measured scan data at Pegasus (UCLA)
- Trained neural network model to predict fits to beam image
- Tested online multi-objective optimization over model (3 quad settings) given present readings of other inputs



Results are for one full day after last training data

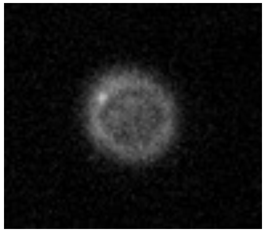
Can use neural network to provide first guess at solution, then fine tune with other methods…

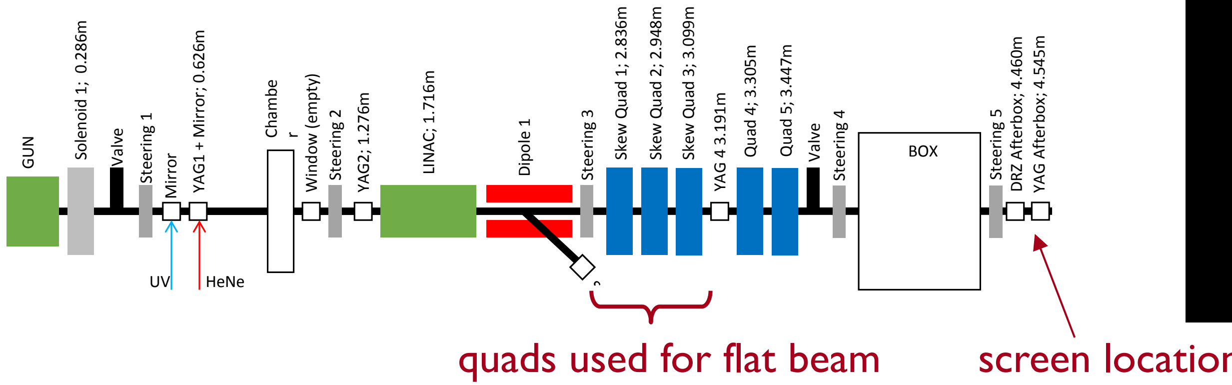


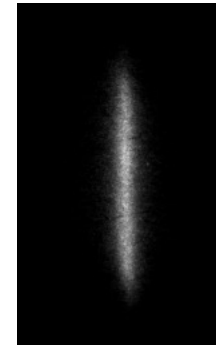
Hand-tuning in seconds vs. tens of minutes

Significant boost in convergence speed for other algorithms

E. Cropp et al., in preparation

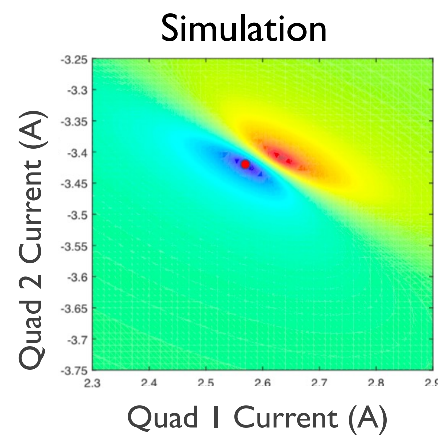


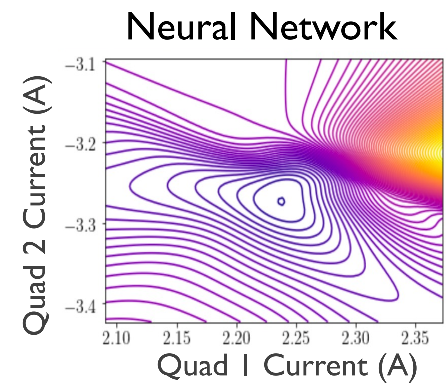


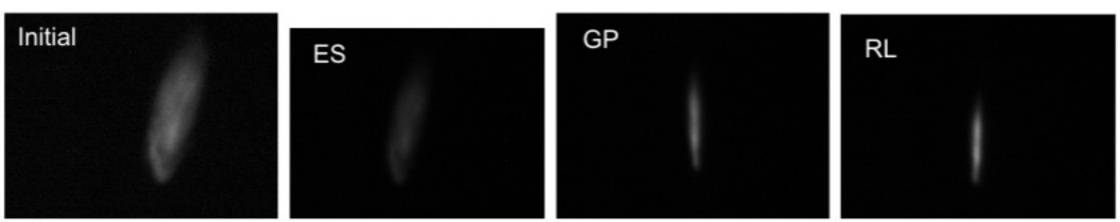


RL on the round-to-flat beam transform:

- Trained DDPG offline using learned model
- Transferred to machine for retraining
- *Once trained, RL had fastest convergence compared with other methods*



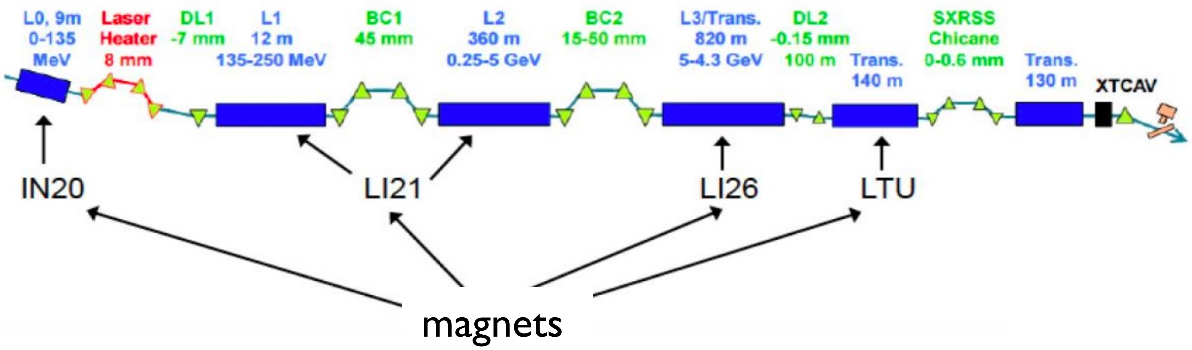


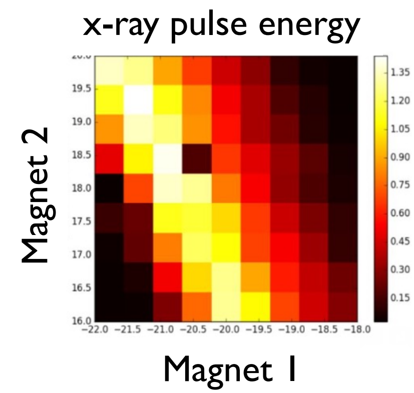


Model-informed Bayesian optimization

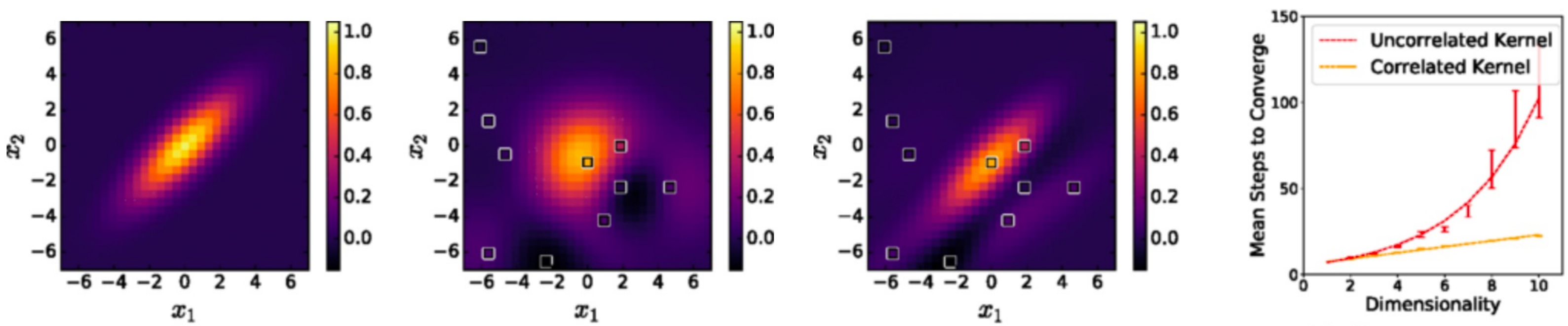
J. Duris et al., PRL, 2020

→ can design GP kernel based on expected physics





Goal: adjust focusing magnets to maximize x-ray pulse energy



(a) Ground truth (b) Isotropic kernel (c) Correlated kernel (d) Convergence tests

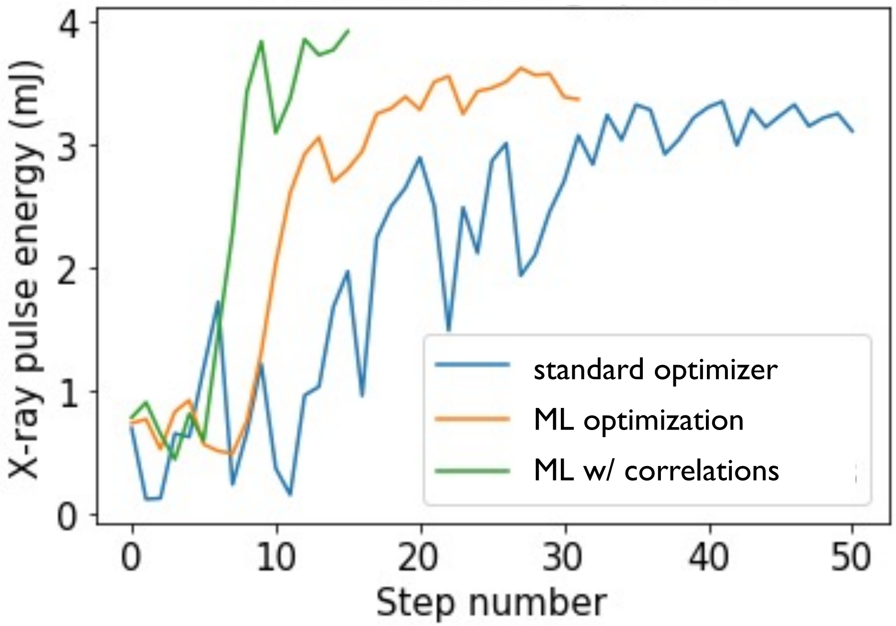
Including expected correlation improves ability to model the data with fewer samples → faster optimization

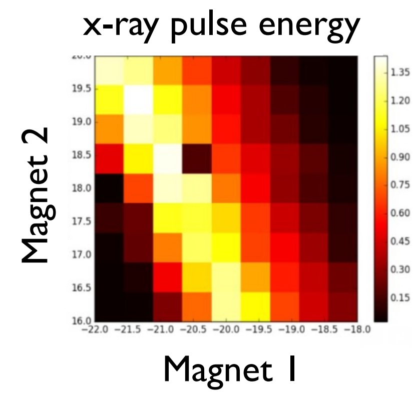
https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.124.12480

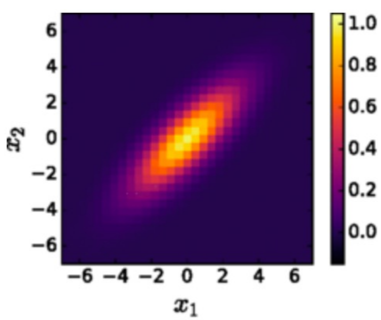
Model-informed Bayesian optimization

J. Duris et al., PRL, 2020

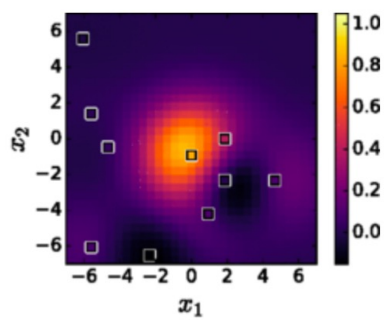
→ can design GP kernel based on expected physics



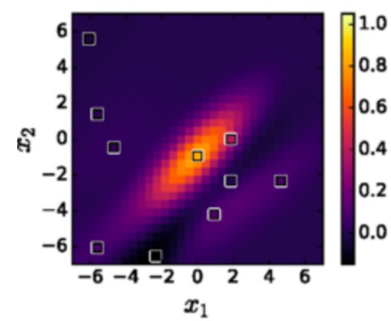




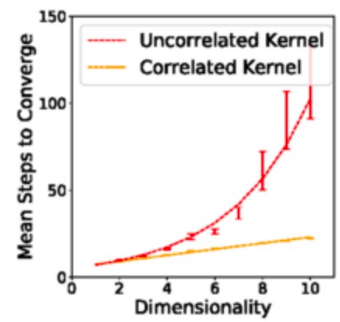
(a) Ground truth



(b) Isotropic kernel



(c) Correlated kernel

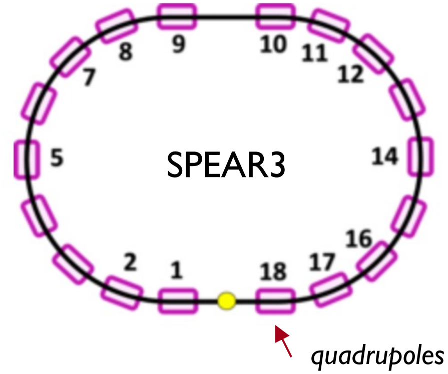


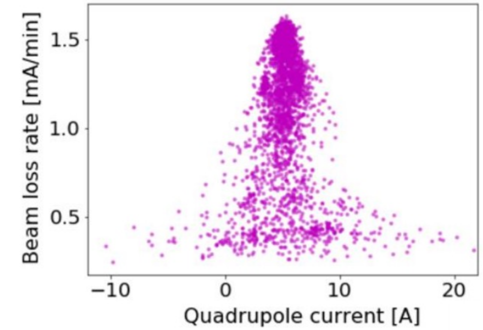
(d) Convergence tests

Including expected correlation improves ability to model the data with fewer samples → faster optimization

https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.124.12480

Model-informed Bayesian optimization

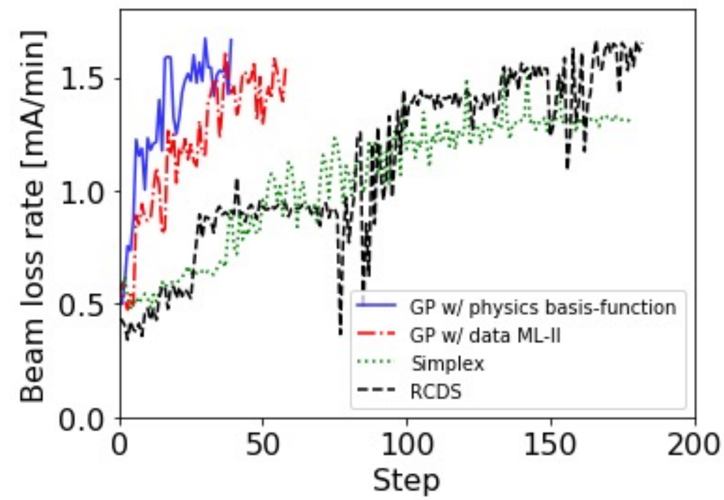




An easier way to get the correlations:

Take the Hessian of a model at the expected optimum → use those correlations in the GP kernel

As long as qualitative behavior is correct, should result in faster convergence



Was demonstrated at SPEAR3 for minimizing the vertical emittance (beam loss rate)

→ **No measured data needed, just a simulation**

A. Hanuka et al., NeurIPS 2019
A. Hanuka et al., PRAB 2021

Qualitative physics models can be easily incorporated into Bayesian optimization for fast tuning

https://journals.aps.org/prab/abstract/10.1103/PhysRevAccelBeams.24.072802 assumed knowledge of machine

less ← → more

Model-Free Optimization

Observe performance change after a setting adjustment

→ estimate direction toward improvement

Model-guided Optimization

Update a model during each search step

→ use model to help select the next point

Global Modeling + Feedforward Corrections

Make fast / accurate system model

→ provide guess for good settings
→ make predictions about machine

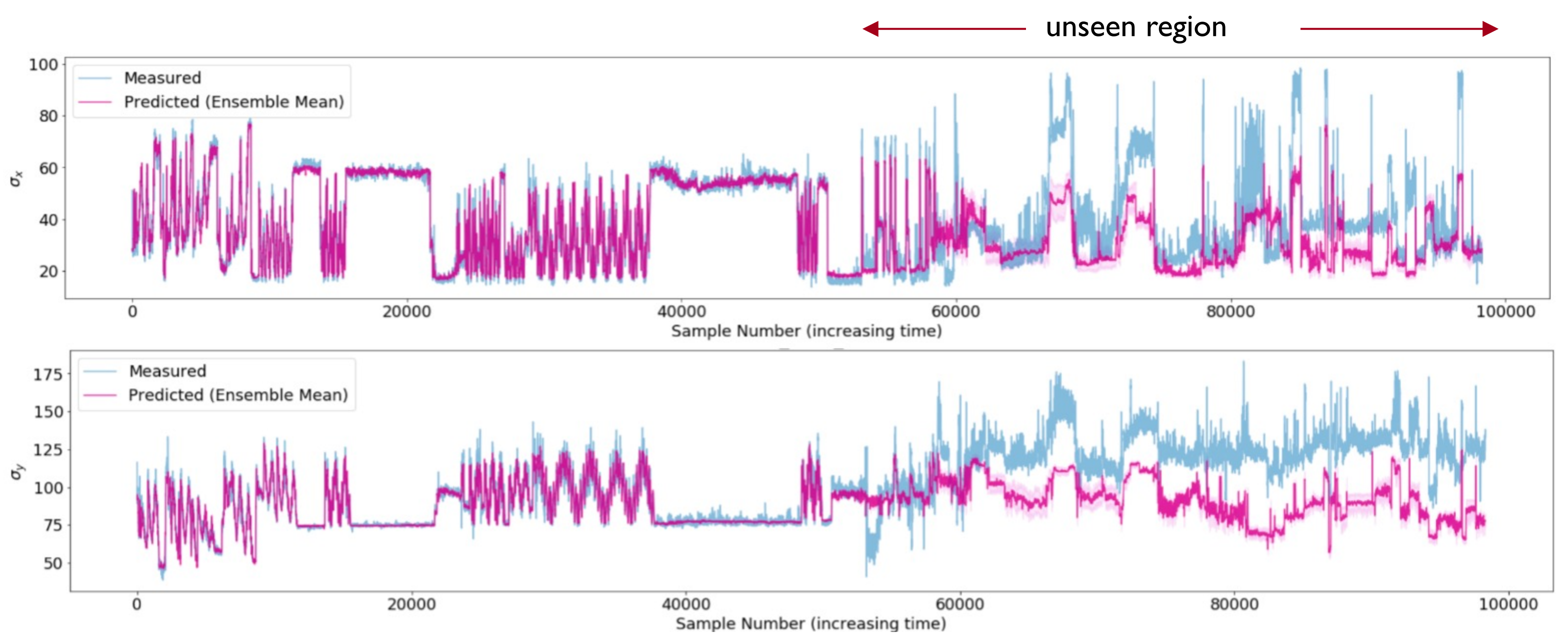
gradient descent
Simplex
Extremum Seeking

← Bayesian optimization

Reinforcement learning →

ML system models +
inverse models

Example of prediction under large drift in inputs (and possibly hidden variables):



Uncertainty estimate from neural network ensemble does not accurately cover the OOD prediction error, but it is relatively higher than for in-distribution data

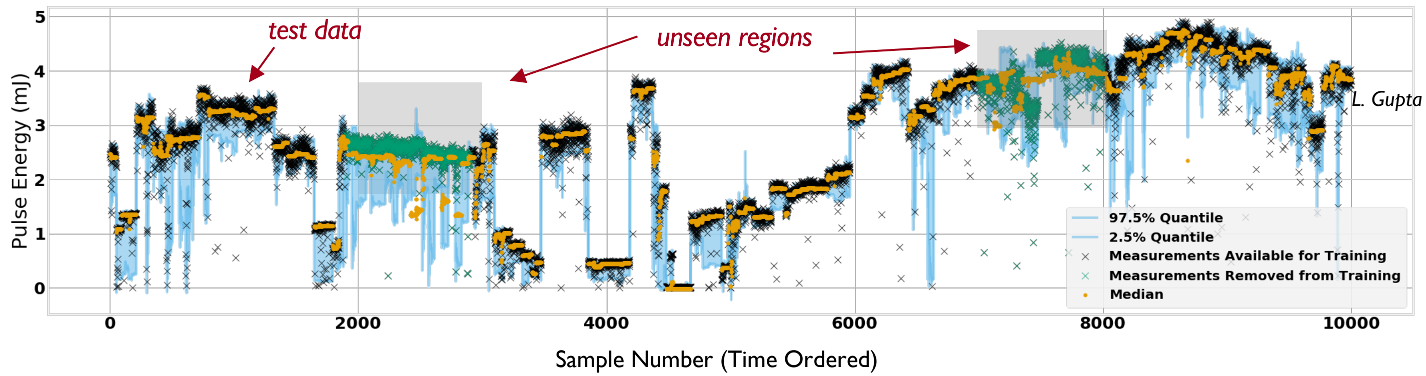
→ *Uncertainty estimates are not always accurate and do need to be validated/calibrated*

Uncertainty Quantification

Need prediction uncertainties to use model reliably in prediction and control
→ *standard neural network models are unaware of what they do not know*

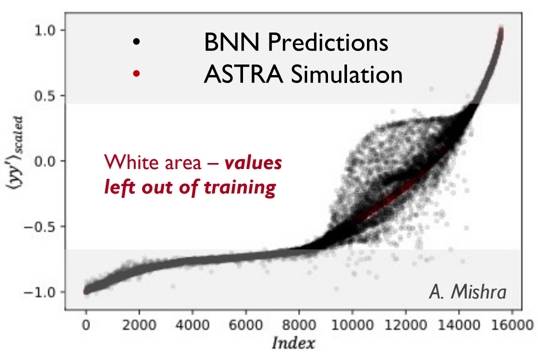
Want to know when one is out of the training distribution (OOD) making predictions less valid (e.g. something on the machine has changed, new region of parameter space is entered)

Current approaches
- Ensembles
- Gaussian Processes
- Bayesian NNs
- Quantile Regression



Neural network with quantile regression predicting FEL pulse energy at LCLS

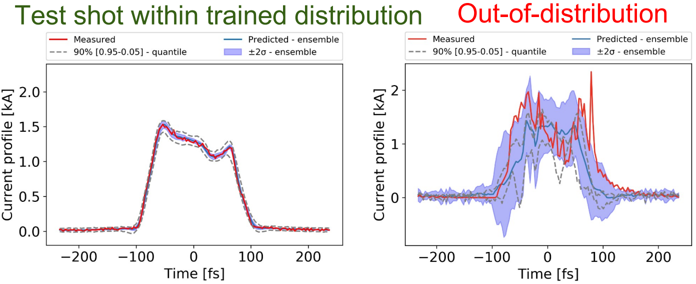
https://github.com/lipiqupta/FEL-UQ/blob/main/notebooks/QR--Interp-2.ipynb



Bayesian neural network predicting scalar parameters for the LCLS-II injector

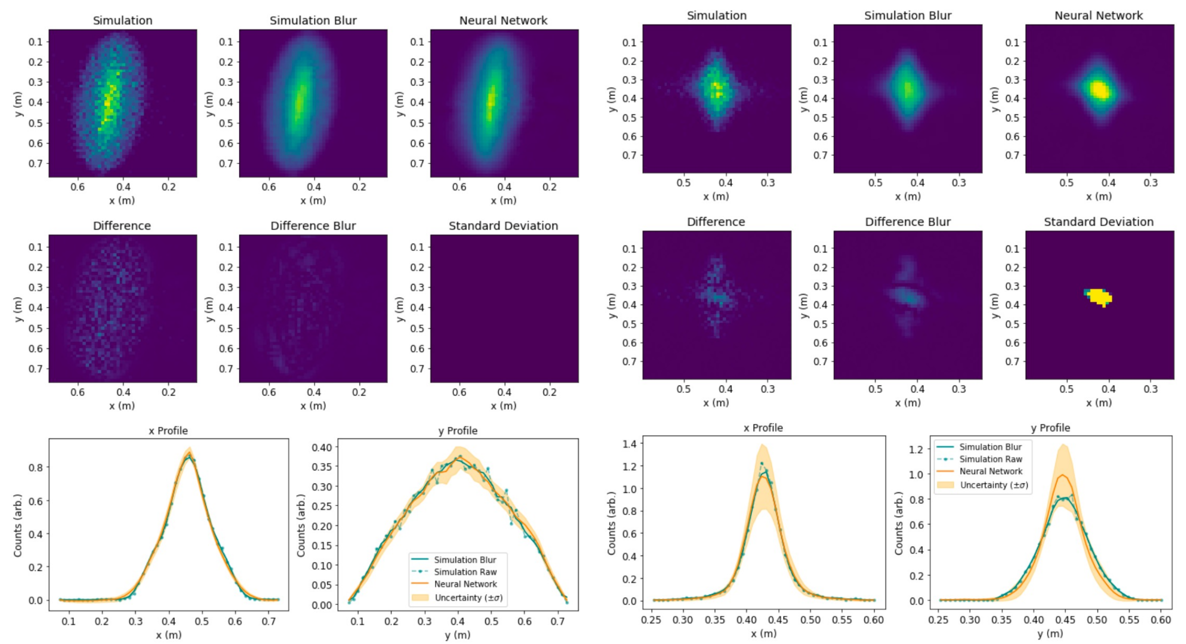
Test shot within trained distribution

Out-of-distribution



Longitudinal phase space beam profiles

O. Convery, PRAB, 2021

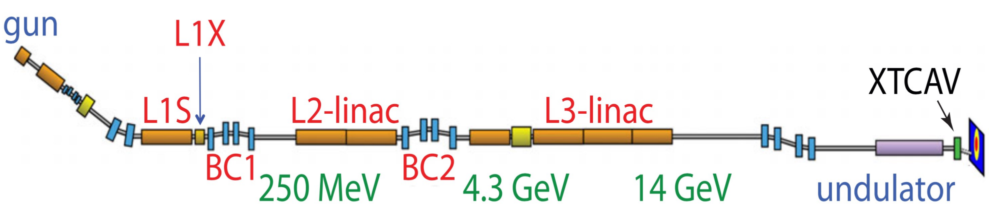


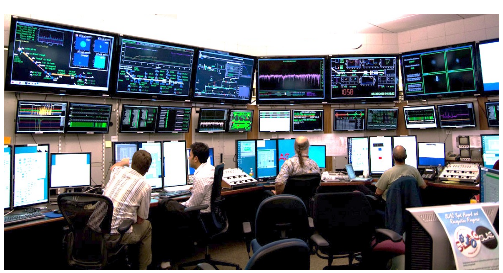
LCLS injector transverse distributions on out-of-training distribution shots, neural network ensemble

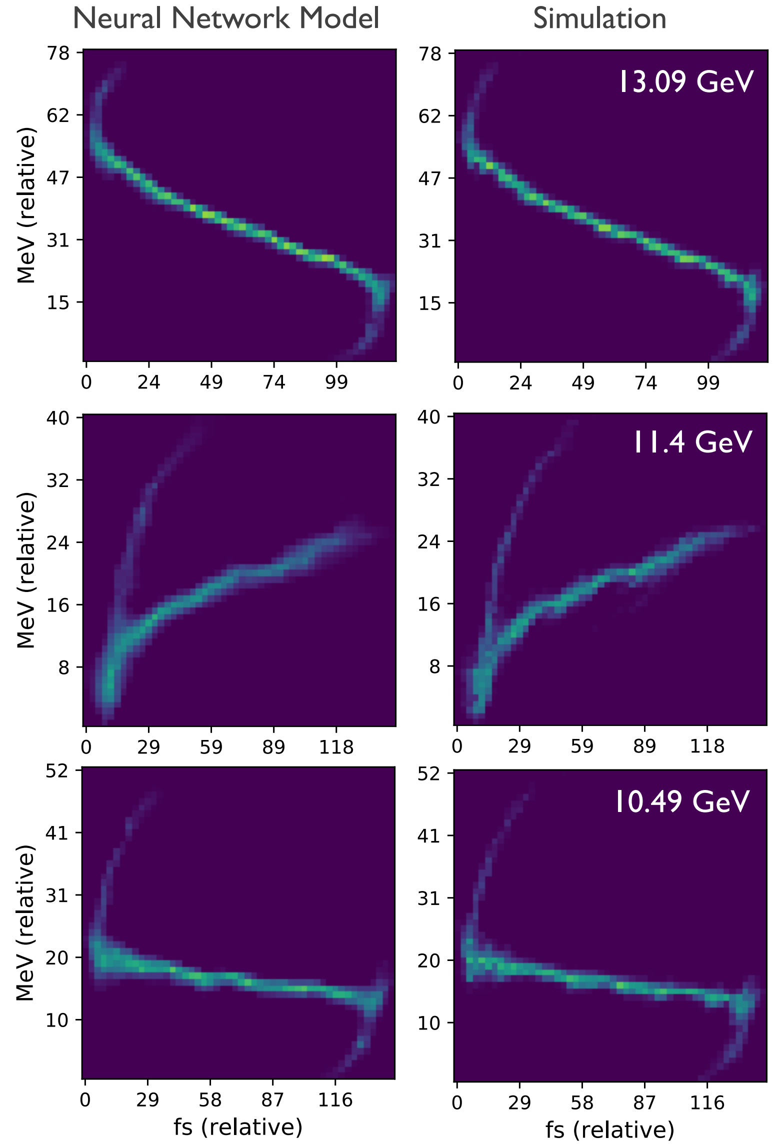
A. L. Edelen, et al., NeurIPS 2019

Trained neural network on simulation data

→ ~ million times faster execution





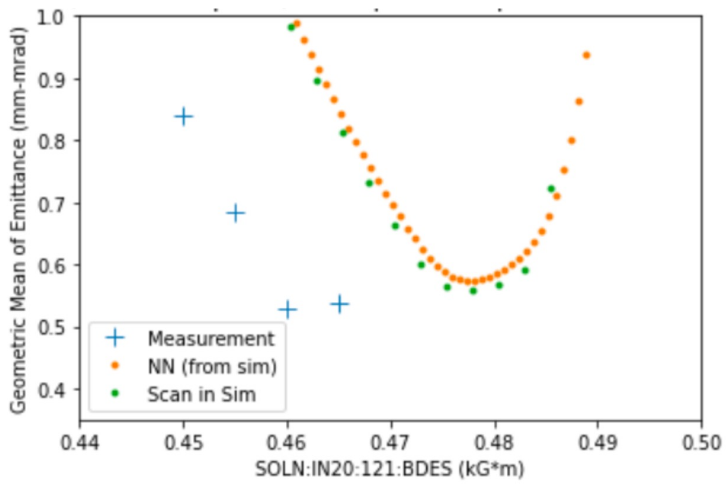


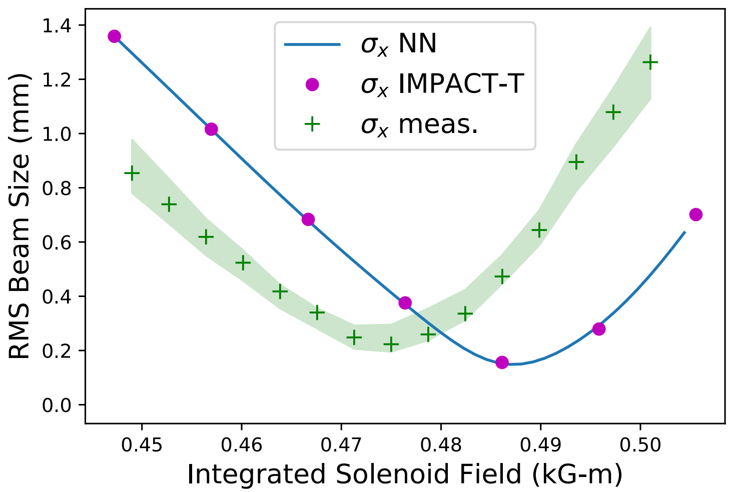
NN predicts 25 scalar outputs ($\sigma_{x,y,z}$ $\varepsilon_{x,y}$ $\sigma_{x',y'}$ σ_E *etc...*) and phase space at the undulator entrance

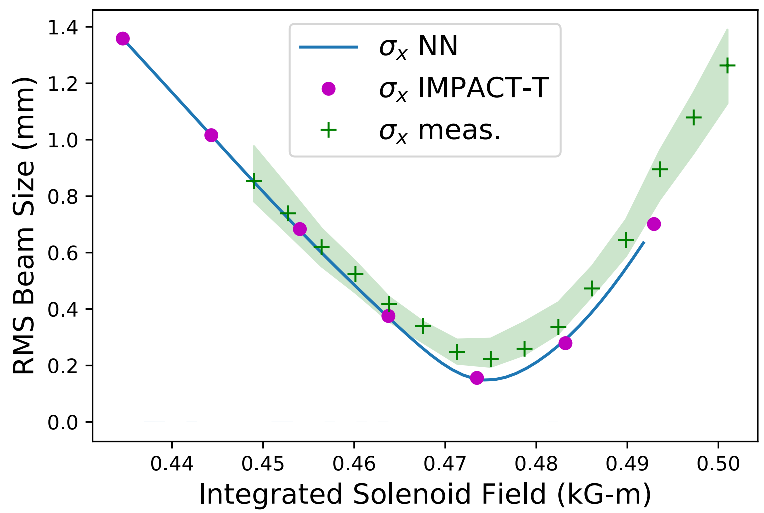
Finding Sources of Error Between Simulations and Measurement

Real accelerator can have many non-idealities and miscalibrations not included in physics simulations

→ *Neural network model allows fast / automatic exploration of possible error sources*

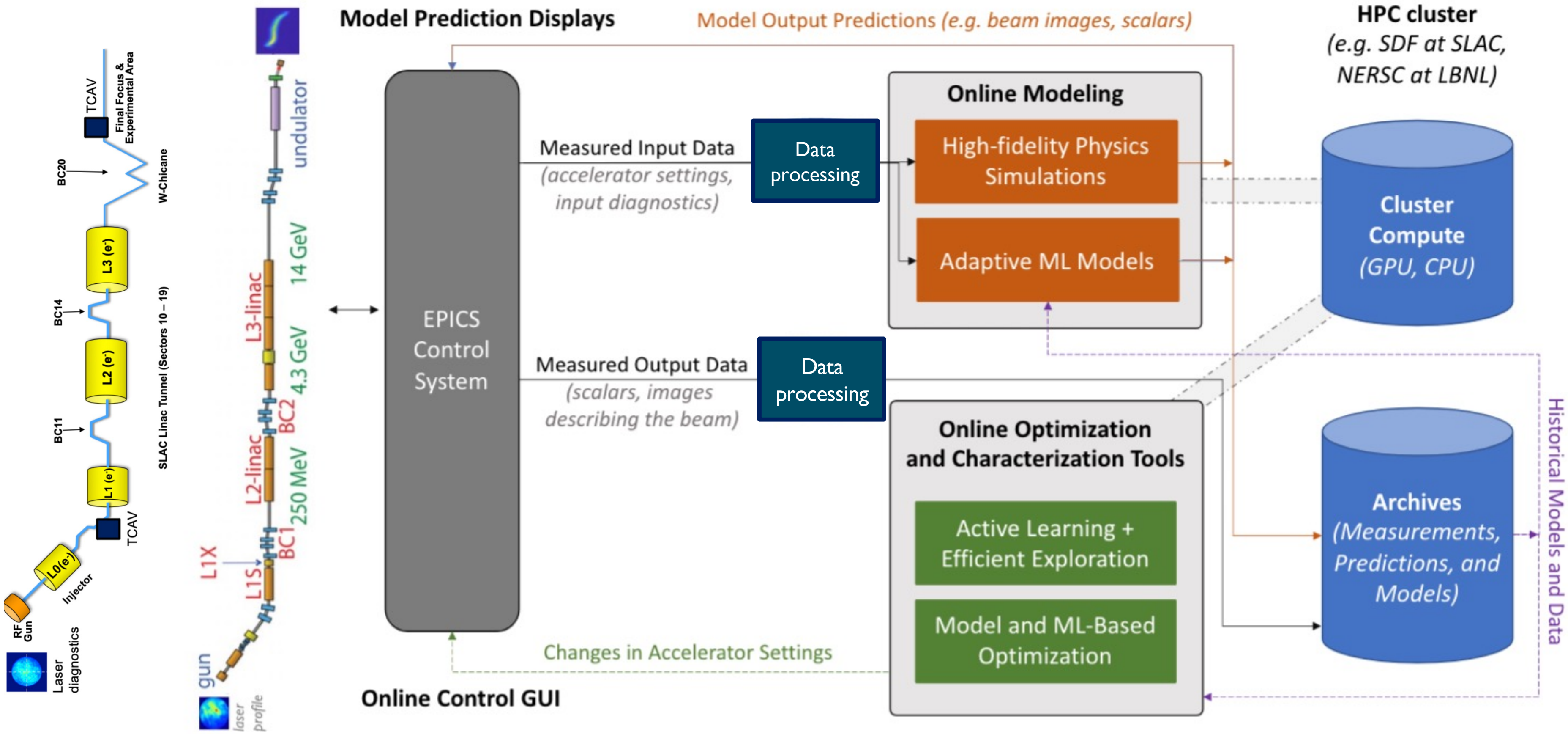






Here: calibration offset in solenoid strength found automatically with neural network model (trained first in simulation, then calibrated to machine)

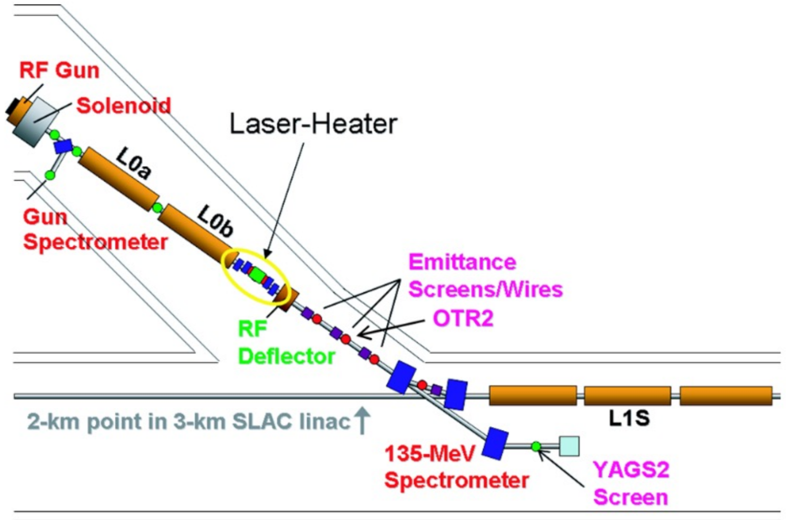
Tying it all together: integration with HPC and continuous online learning



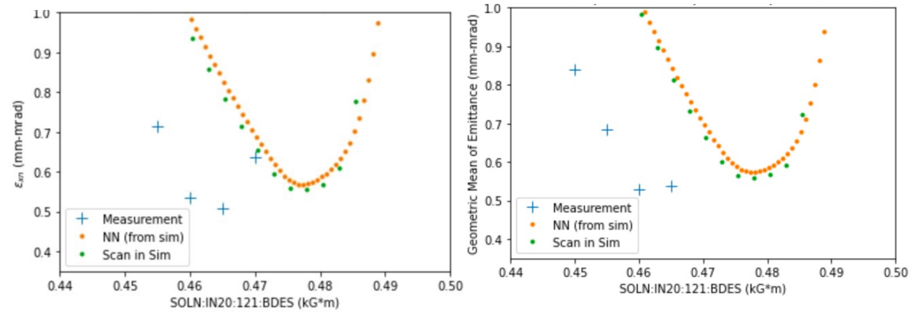
FACET-II LCLS

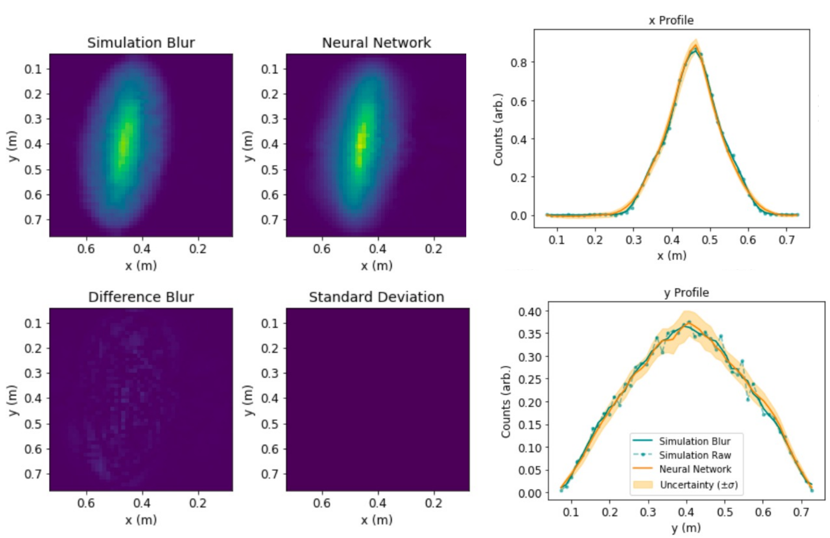
LCLS Injector Surrogate Model

- **Many versions (predict phase space, evolution along z etc); including one with scalar outputs of interest at OTR2**
    - **Inputs:** *laser length + spot size, L0A/B phases, Solenoid, SQ quad, CQ quad, 6matching quads*
    - **Outputs:** *emittances, bunch length, spot sizes, covariances (for Twiss calc), energy*
- Neural network trained on IMPACT-T sims
- Set up to take machine inputs in PV units
- Focused on interpolation to sim vs. exact match to measurements
- Using in tuning algorithm + code testing

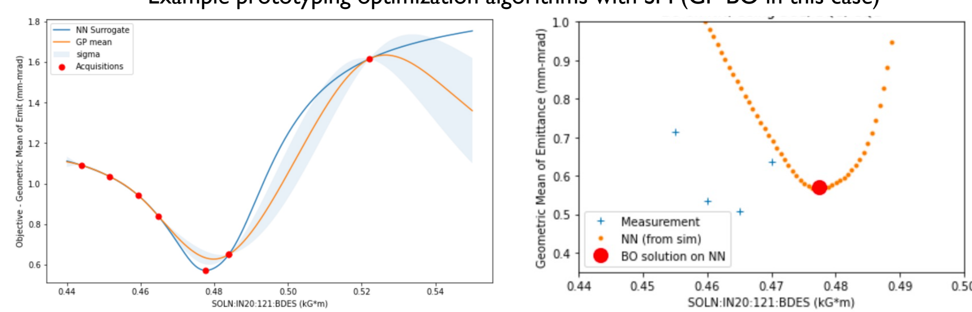


IMPACT-T and SM trained on it deviate from measurements, but similar qualitatively



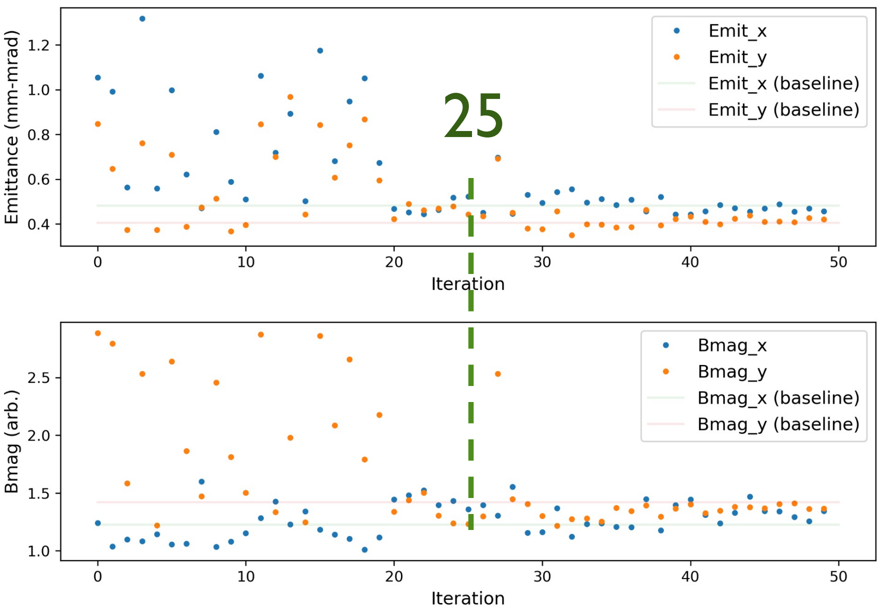


Example prototyping optimization algorithms with SM (GP-BO in this case)

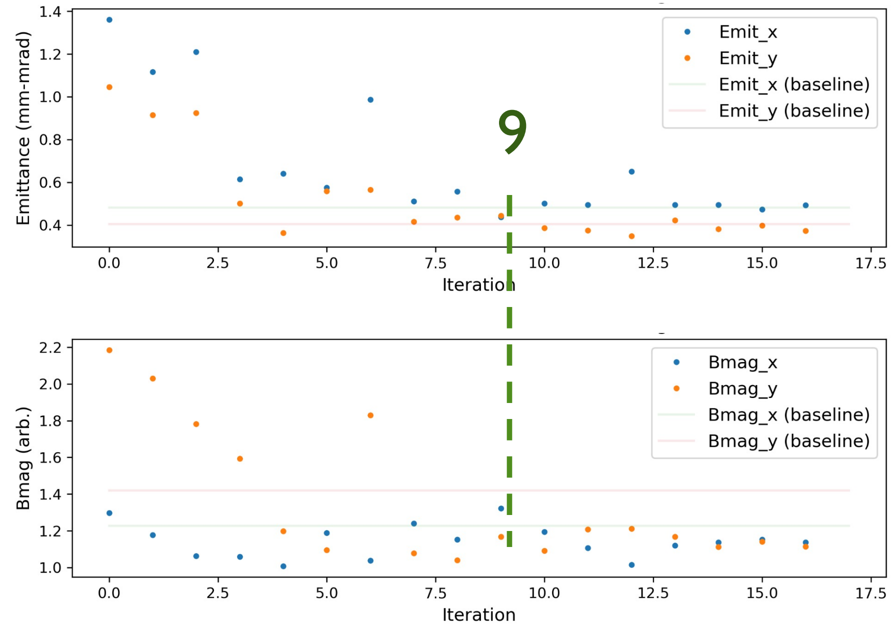


Using Injector Model for Bayesian Optimization

Standard RBF Kernel



Kernel from Hessian of Surrogate Model
(trained on IMPACT-T sims)



- Both start from randomly-sampling within the bounds
- "Baseline" is tuning solution that ops was using that day
- Emittance measurement takes 3-4 minutes

	Nominal	**Kernel from Hessian**	**Standard RBF**
emit_x	0.4317	0.428	0.488
emit_y	0.424	0.373	0.420
bmag_x	1.368	1.137	1.128
bmag_y	1.422	1.113	1.233

Using simulation surrogate model to inform Bayesian optimization allows rapid tuning to human-level quality without any previous data

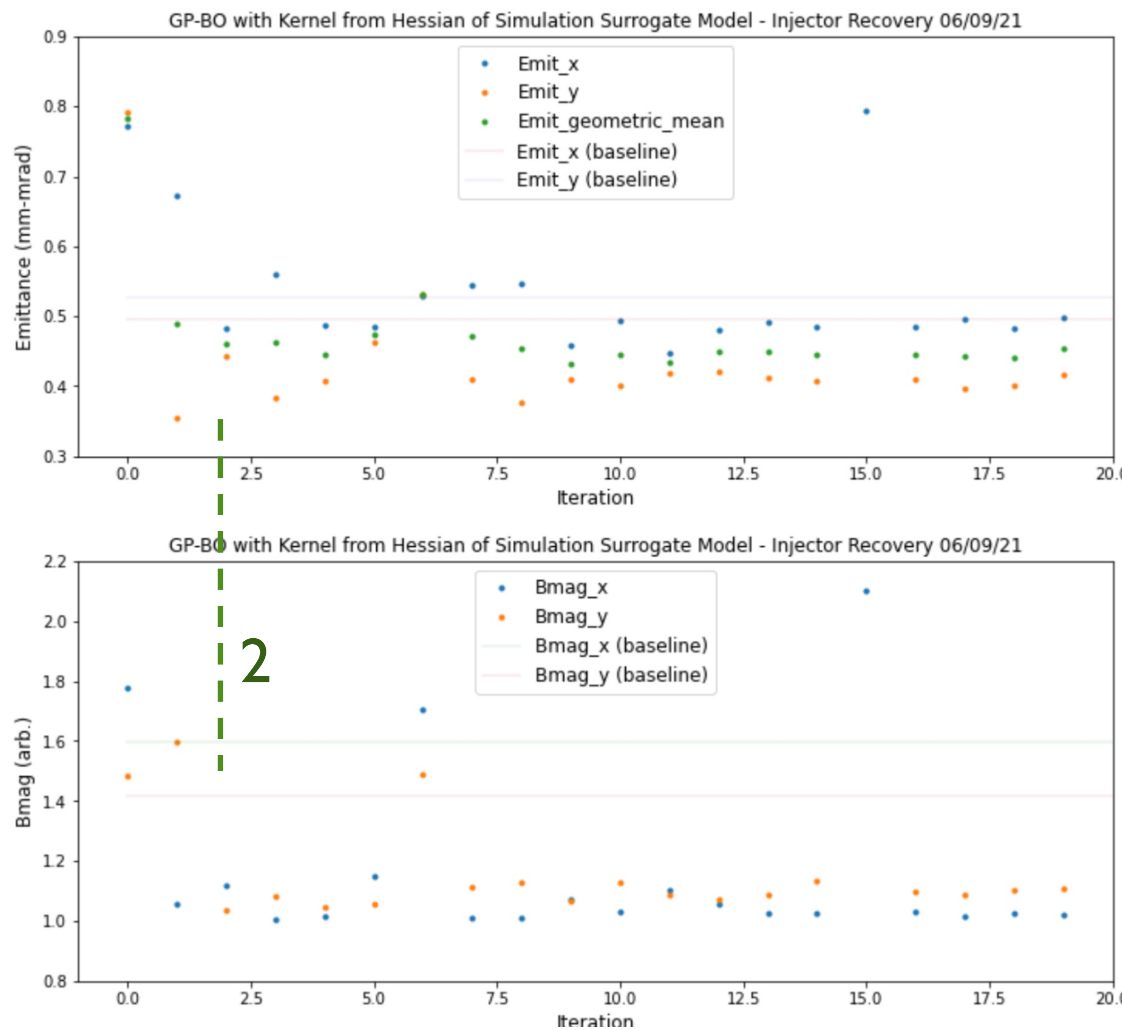
Re-using learned information: injector recovery after a brief shutdown

Seeded with 5 random training points from the previous run *(may help or hurt convergence depending on how much has changed)*

"Baseline" is the solution from before the shutdown

By iteration 2 already had a decent solution

→ Suggests this is viable for use in regular injector tuning



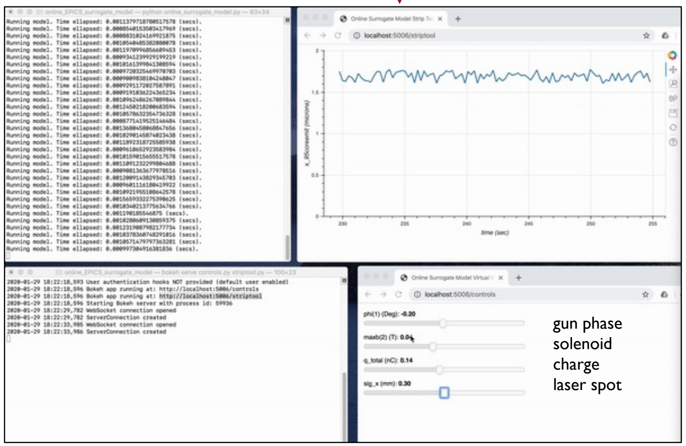
Biggest impediment right now is the robustness of the emittance measurement itself (quad scan)

Control Room Integration

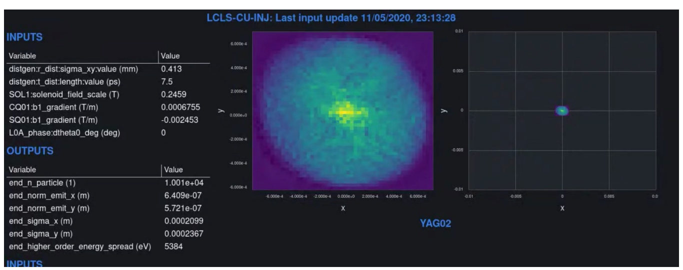
Live read-backs and user controls

EPICS server running surrogate model

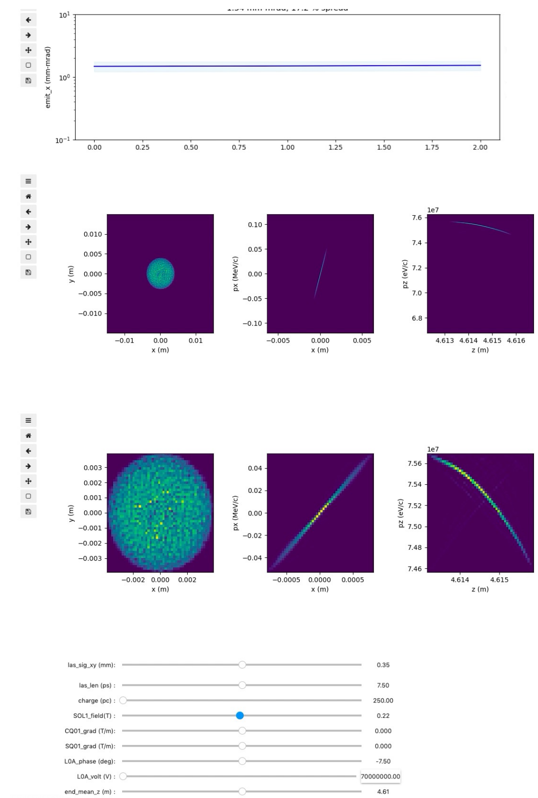
Launch controls and readback GUI



This case: ASTRA sim with 3D space charge evaluates in milliseconds (vs. 5-6 minutes)



live EPICS prediction from surrogate model, streamed to control room



interactive model widget

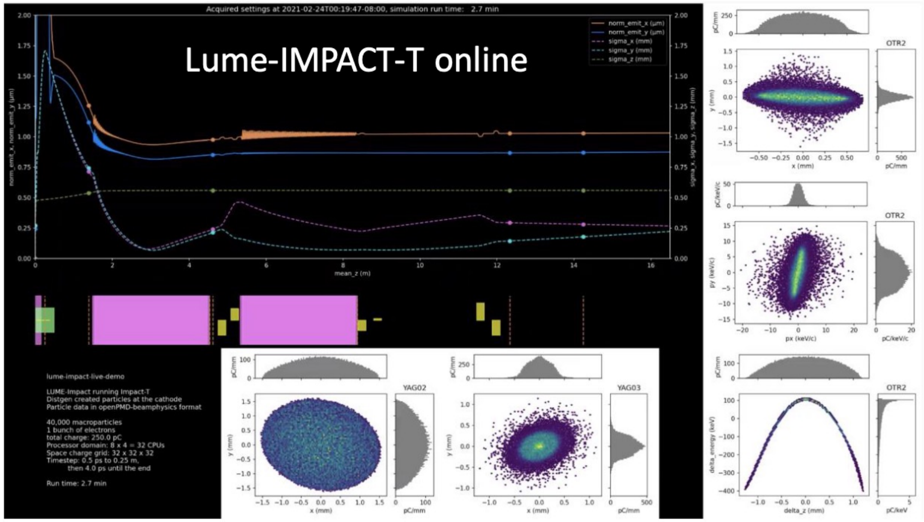
Using + developing lume-model and lume-epics (https://www.lume.science/)

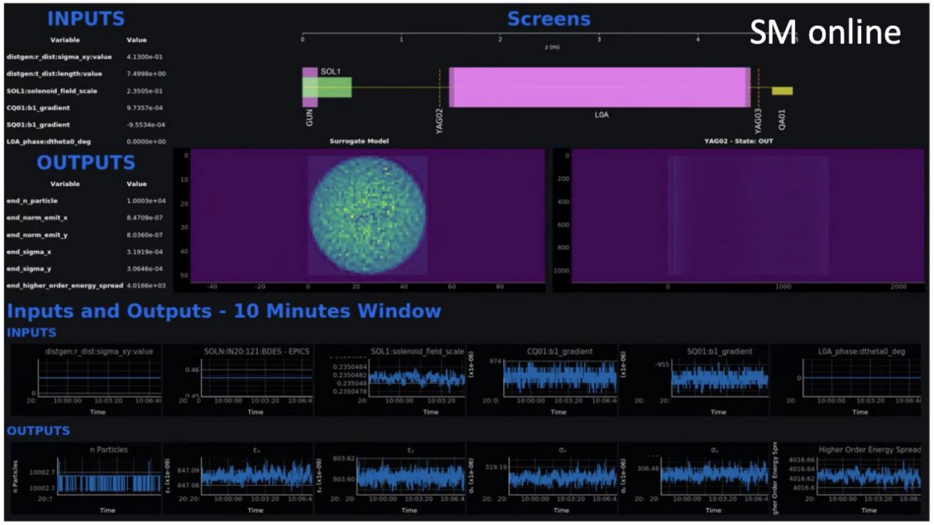
→ *Demo in Binder: https://github.com/jacquelinegarrahan/lume-model-server-demo*

Running LUME-IMPACT-T and Neural Network Model of LCLS Injector Online

- Lume-IMPACT-T online
  - Read EPICS PVs as input
  - Displays phase space predictions at OTR2 + line plots
  - Updates every 2 minutes (length of time for one IMPACT-T run)
  - https://www.youtube.com/watch?v=P6HYfpV6xXM

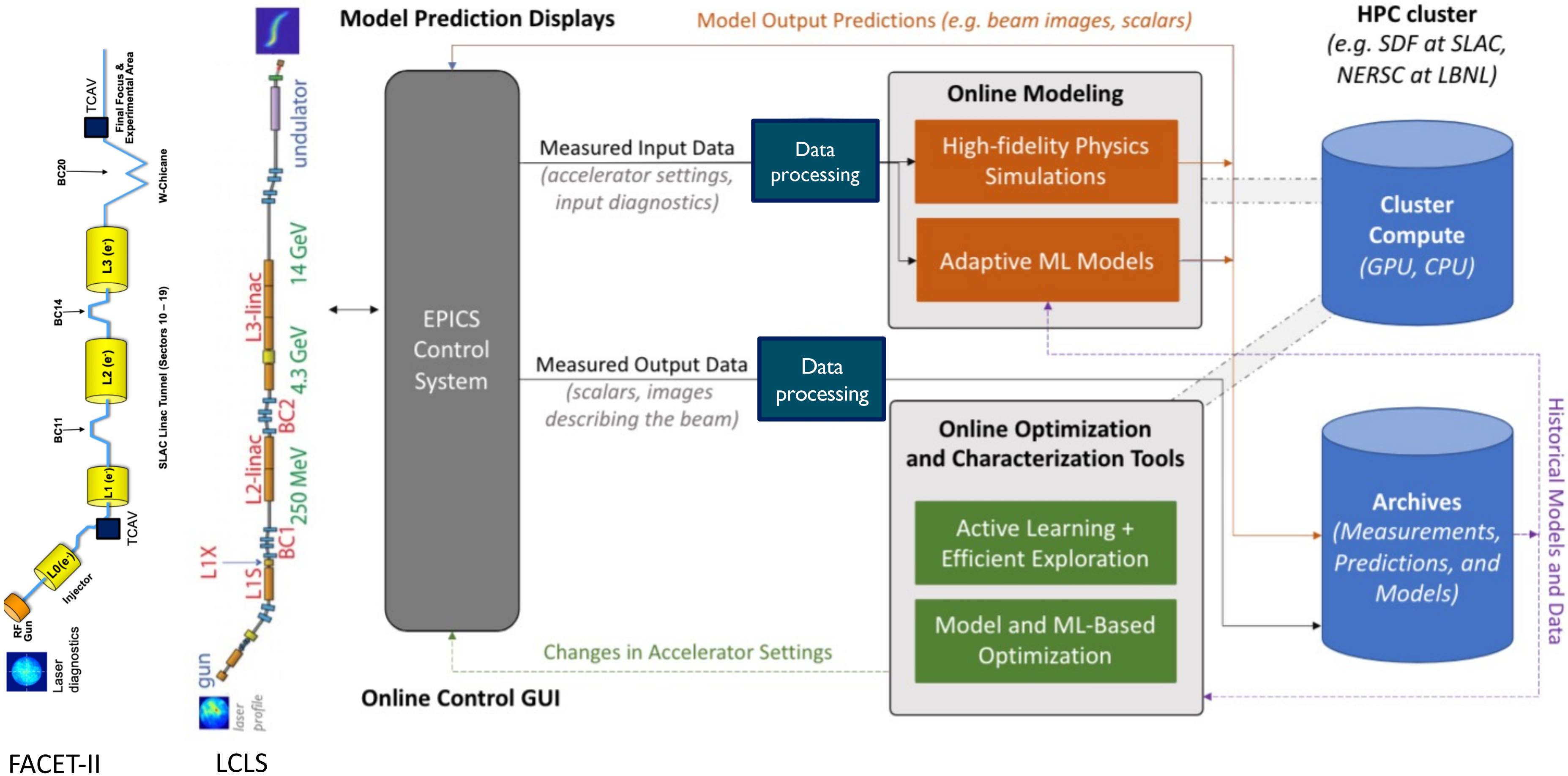
- SM at YAG02
  - Continuously updates
  - Serves output PVs
  - Will update to include OTR2, line plots soon
  - https://www.youtube.com/watch?v=FZny98PGcmU&feature=youtu.be





LUME tools are available and open source: https://www.lume.science/

Tying it all together: integration with HPC and continuous online learning



Summary

Bayesian optimization and reinforcement learning both of utility for high-level tuning and control

- *Grew out of different communities and time dependent vs. time independent setting, but share fundamental commonalities*

BO and RL excel in different regimes

- *BO: exploratory + low data regime, optimization of new setups, slow measurements*
- *RL: high data regime, continuous control*

Both can benefit substantially from better system models

- Warm starts from system models
- Model-informed kernel for BO
- Pre-training RL agents using fast-executing system models

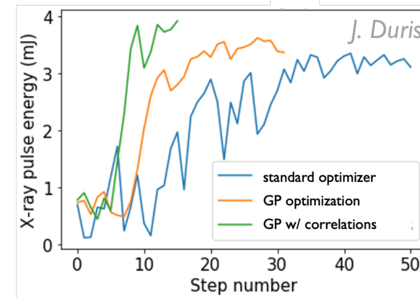
→ Tying together strengths of different approaches

→ Improve system modeling (speed + accuracy), use model-informed BO for exploring new setups, use pre-trained RL policies for fast switching between setups + continuous control

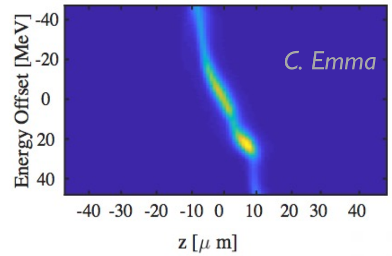
Backups

Future: tying together and scaling these to higher dimension + more extreme beams

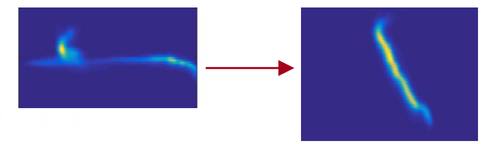
automated control
+ optimization

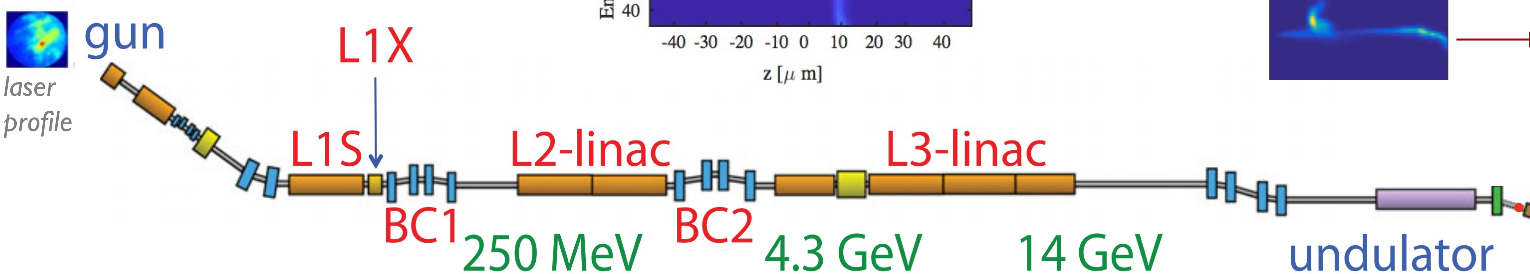


advanced diagnostics
(reconstruct / analyze beam)

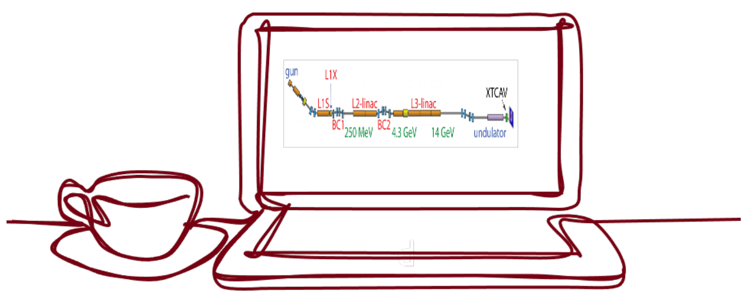


anomaly detection
failure prediction





incorporate
physics
information



extract unexpected
relationships
(feed into control / design)

digital twins + online modeling
(fast sims, autodiff sims, model calibration)

+ need UQ for all