Faster, More Accurate Particle Accelerator Models using Machine Learning

Auralee Edelen, Daniel Ratner, Chris Mayes, Claudio Emma, Nicole Neveu, Nora Norvell, Xinyu Ren, Xiao Zhang, Dorian Bohler, Tim Maxwell, Yuantao Ding, Gabriel Marcus, Alberto Lutman, Alex Scheinker, Andreas Adelmann, Alan Heirich, Aashwin Mishra, Joe Duris

1st SLAC ML Workshop, 19 February, 2019

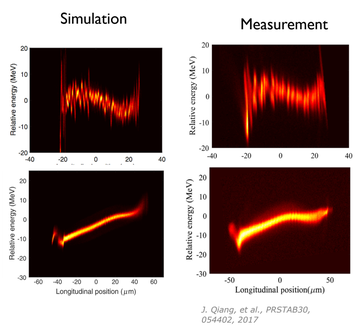
Accelerator simulations can be very computationally intensive and don't always match the machine well

Impedes use in offline start-to-end optimization and control development

Prohibits use as an online model (e.g. diagnostic + control applications)

Often takes much effort to replicate real machine behavior

Start-to-end LCLS simulation
10 hours on thousands of cores at the NERSC

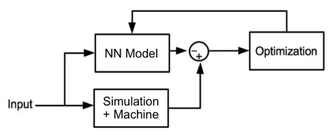


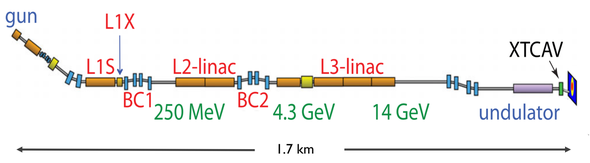
J. Qiang, et al., PRSTAB30, 054402, 2017

Complementary approach: ML model

Once trained, neural networks can execute quickly

Train on sparse sample from high-fidelity simulations
+
Train on measured data





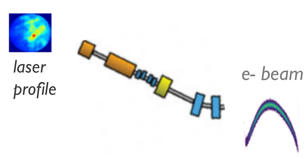
Nonlinear systems with large parameter spaces
Variety of diagnostics, *but limited in number and some not always available*
Time-varying/ non-stationary behavior
On-demand changes in operational state

Dedicated effort in AD to create fast-executing, accurate accelerator models, and use these in control + optimization

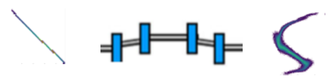
Three Main Approaches for ML Modeling Effort

Modules for slowest simulations

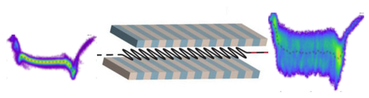
Injector



Bunch Compressors



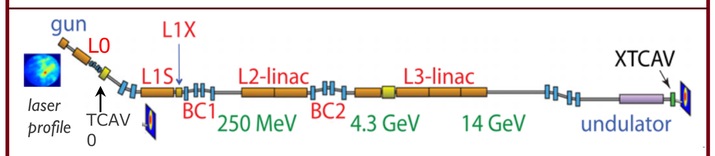
FEL Undulator



Controllable settings
Incoming beam distribution
→ ML Model → Outgoing beam distribution

Beam description:
- *images (beam projections/slices)*
- *6D point clouds (3 spatial, 3 features)*

Start-to-end accelerator models



Controllable settings
Input diagnostics
→ ML Model → Output diagnostics

e.g. beam images
x-ray spectra

e.g. laser spot image

Back-tracking and model inversion

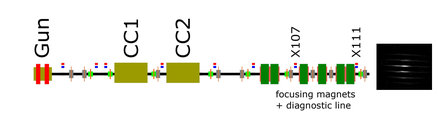
Desired beam output → ML Model → Incoming beam distribution **or** Suggested settings

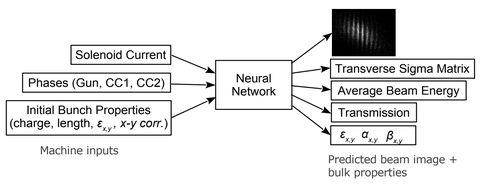
Can we bridge the gap between our simulations and empirical machine behavior?

Poor agreement between physics simulation and measured data
Can't do a full parameter scan on machine (cost / time)

Simulation Data (wide scan) → ML Model ← Measured Data (smaller scan + passive data)

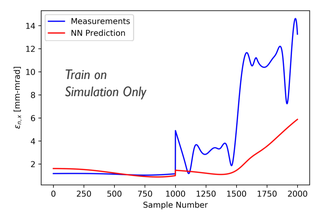
Can we pre-train in simulation and update with measured data?

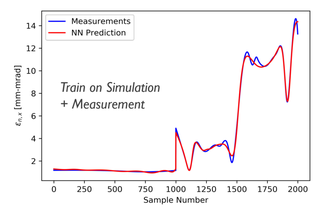




Initial results from study of injector systems look promising

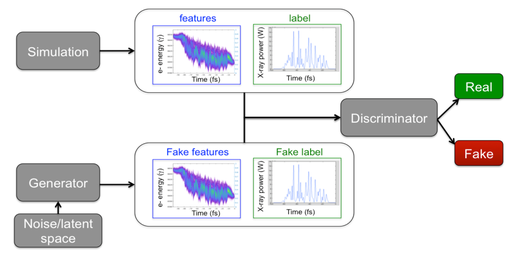
→ need to investigate strategies for doing this routinely and at larger scale

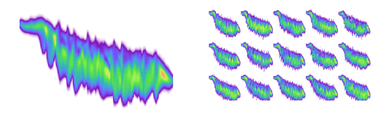




Can we handle statistical fluctuations realistically?

- FEL process has some inherent random behavior
- Want to generate many examples of FEL output with realistic statistical behavior quickly → *e.g. provide data sets for experiment planning*
- Train generative adversarial network (GAN) on simulation data





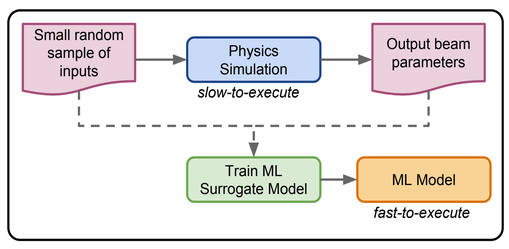
Genesis: ~1000 cpu-sec

GAN (neural net): ~0.001 gpu-sec

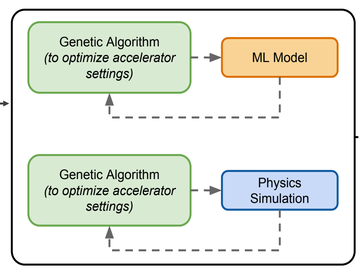
X. Ren

Can We Speed Up Multi-Objective Optimization?

Generate ML Model using Sparse Random Sample



Run Optimizer on ML Model and Physics Simulation

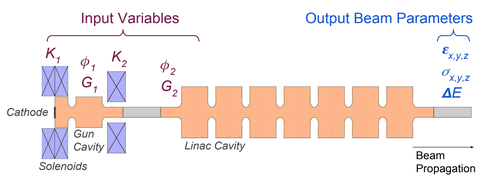


Test Case: Argonne Wakefield Accelerator Injector

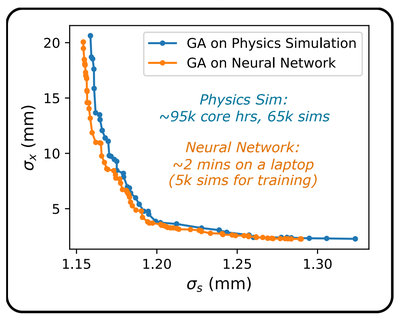
OPAL simulation (PIC) : 3D space charge 3D field maps 10k particles

NSGA-II for optimization: 200 generations ~350 individuals

5K random points for training



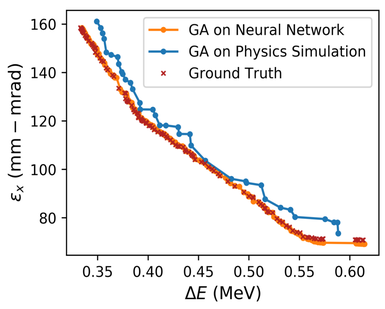
Compare Resulting Pareto Fronts

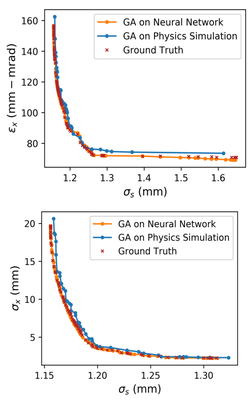


In some cases, optimization over simulation takes too long to converge

→ validate Pareto front from neural network more directly

Get Pareto points from Neural Net → Verify points through Simulation





Neural network gave a ***better* approximation of the true Pareto front** than the naïve GA on the simulation

Required **13x fewer simulations** and had **10^6 faster execution** in the optimization

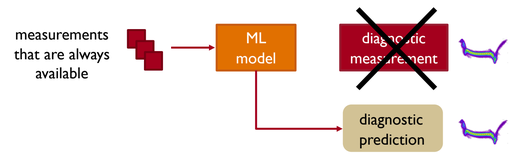
- Promising results on a common problem
- But, small parameter space (6 inputs, 7 outputs)
- Only deterministic processes (e.g. unlike FEL)

→ How will this scale to larger + more complex sections?

Can we use ML-based virtual diagnostics?

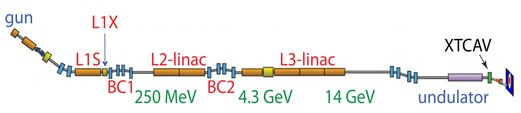
Real diagnostic not always available:
- *destructive, cannot use during user operations*
- *not sensitive in entire operating range*
- *slower update rate than desired*
- *moved to another location (e.g. cost constraints)*

→ *FACET-II XTCAV*
→ *LCLS-II XTCAV*



Still can have diagnostic prediction for user analysis and system control!

FACET-II / LCLS study

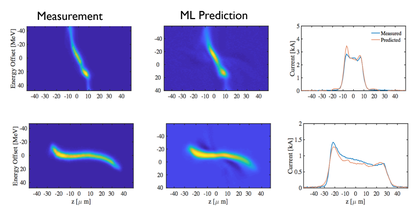


Two machine parameters scanned:

L1s phase: -21 to -27.8 deg

BC2 peak current: 1-7 kA

- 5 inputs to neural network
  - *L1S and L1X amplitude, L1S phase*
  - *BC1&2 peak current*
- Reasonable performance predicting profiles and XTCAV images
- Key is varied data set from scan, and data preparation (e.g. ROI selection)



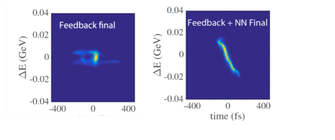
→ Long-term prediction accuracy? (e.g. drift) *→ How best to flag bad shots? (e.g. bad BC2 input)*

→ How best to handle region where detector not sensitive for FACET-II (fill in with simulation)?

Example inverse model: warm start for local optimizer

"ES" optimization algorithm can tune many parameters efficiently
- But can get stuck in local minima + can take awhile to converge *(e.g. hundreds of iterations)*
- *Given target beam parameters, a neural network can provide suggested initial settings*

Preliminary study at LCLS:
- Scanned L1S phase and BC2 peak current
- Trained network to map phase space to settings
- Compared ES with/without warm start for extreme change in phase space



→ ES alone unable to converge, but able to converge from settings suggested by NN

→ Need to extend to wider range of parameters, look at performance over time

Wish List and Challenges

Wish list
- Train on simulation → *orders of magnitude speedup + enable use as online model*
- Train on measurements → *more closely match to real accelerator behavior*
- Use as a virtual diagnostic → *predict what an unavailable diagnostic would show*
- Use for offline for experiment planning, design studies, and prototyping control algorithms
- Use online as an operational aid and in model-based control schemes

Challenges
- Flagging when to trust the model (e.g. bad shots)
- How best to retrain + strategies for deciding when to retrain
- How best to combine simulation and measured data
- Scaling to higher dimension + problem complexity
- Efficiently handling different data types: 6D point clouds, images, scalars