Machine Learning and Tuning of the CLIC Final Focus System

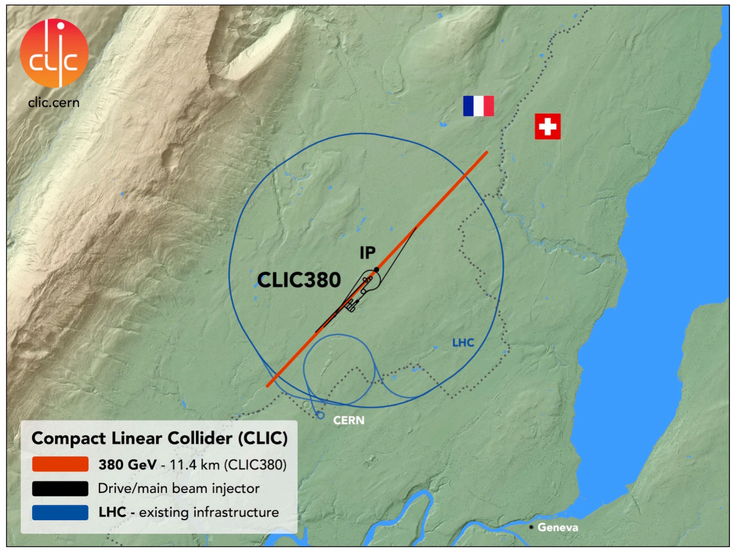
Jim Ögren

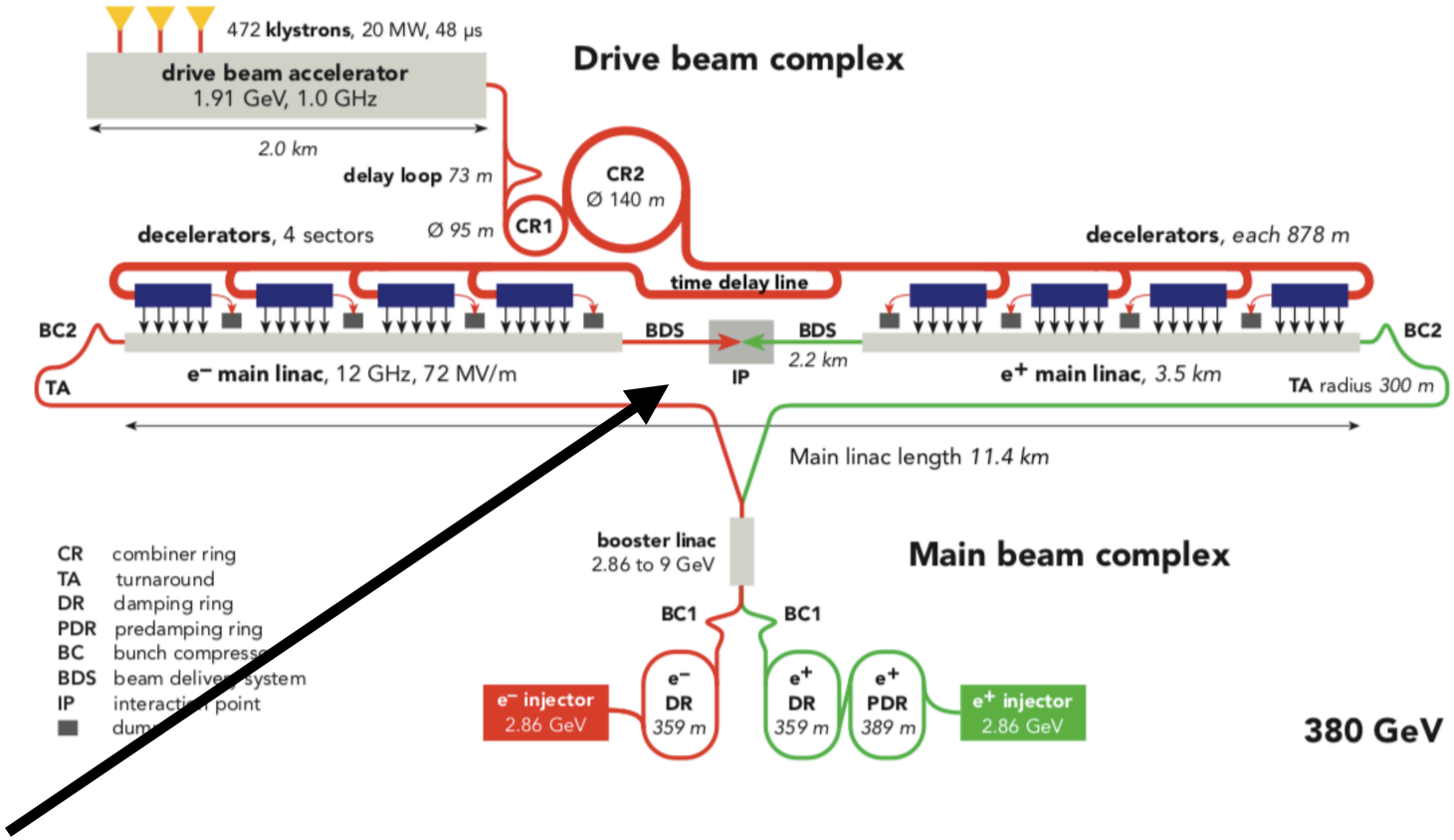
2nd ICFA Workshop on Machine Learning for Charged Particle Accelerators,
February 27, 2019, Villigen, PSI.

Acknowledgements: Chetan Gohil, Ryan Bodenstein,
Andrea Latina, Daniel Schulte

Compact Linear Collider (CLIC)

Footprint





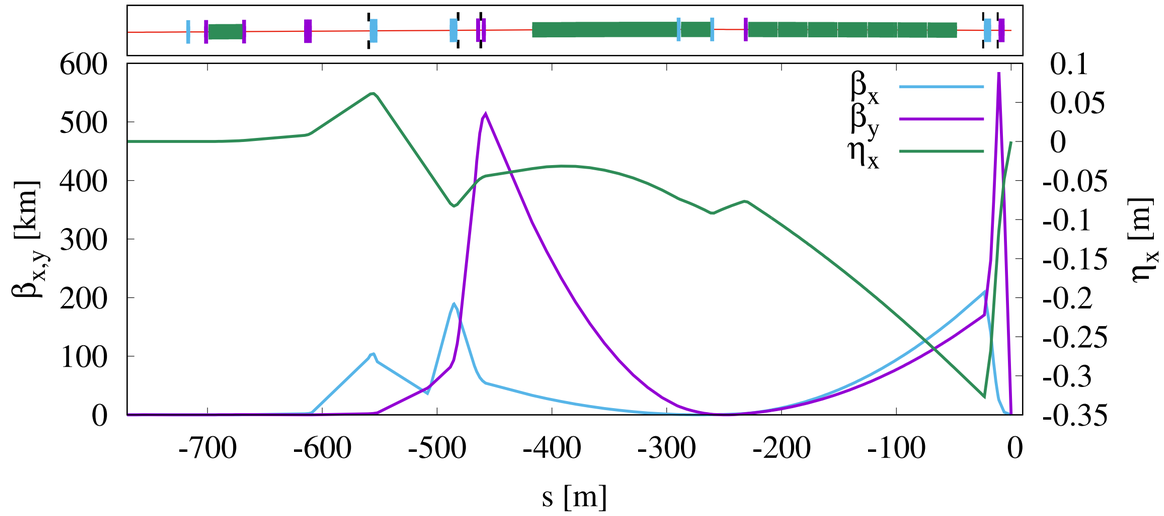
The beam delivery system - Last 2.2 km of the main linacs

- Collimation section
- Final-focus system

The 380 GeV CLIC FFS

The final-focus system (FFS):

- Local chromaticity scheme
- 780 m total length
- 20 quadrupole magnets
- 22 Beam-position monitors (BPMs)
- 6 sextupole magnets
- 2 octupole magnets



Norm. emittance (end of linac) $\gamma\epsilon_x/\gamma\epsilon_y$	[nm]	900 / 20
Norm. emittance (IP) $\gamma\epsilon_x/\gamma\epsilon_y$	[nm]	950 / 30
Beta function (IP) β_x^*/β_y^*	[mm]	8.2 / 0.1
Target IP beam size σ_x^*/σ_y^*	[nm]	149 / 2.9
Bunch length σ_z	[μm]	70
rms energy spread δ_p	[%]	0.35
Bunch population N_e	[10^9]	5.2
Number of bunches n_b		352
Repetition rate f_rep	[Hz]	50
Luminosity $\mathcal{L}_\mathrm{total}$	[$10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$]	1.5
Peak luminosity $\mathcal{L}_{1\%}$	[$10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$]	0.9

Luminosity calculation:

N = particles per bunch
n_b = number of bunches
f_r = repetition rate
σ = beam size at collision point
H = correction factor (hour glass effect, disruption)

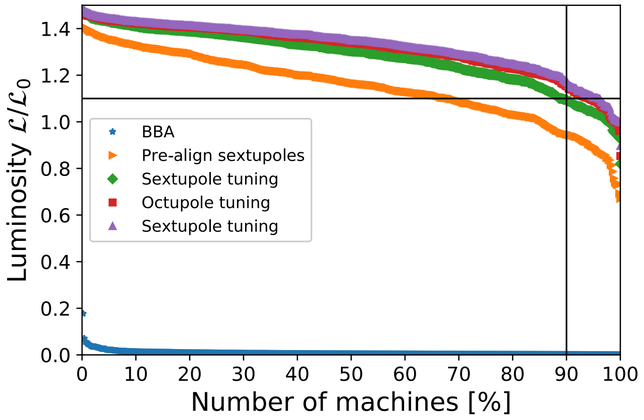
$$L = H\frac{N^2 n_b f_r}{\sigma_x \sigma_y}$$

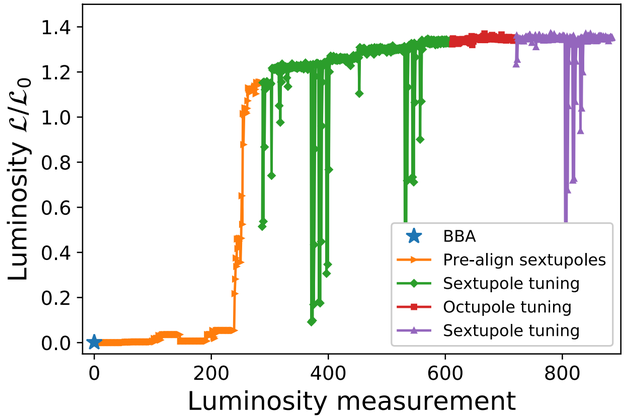
Tuning of the Final focus system

- Tuning = reaching performance for a machine with imperfections
  - Studied extensively for CLIC and other linear colliders
- The challenge
  - Complicated, interleaved system with many effects
  - Simulations are computationally heavy
- Our simulation setup
  - Tracking code: PLACET (has Python interface)
  - Beam-beam code: GUINEA-PIG for computing the luminosity
- First objective
  - A machine learning model for speeding up simulations

Single-beam static tuning simulations

- Single-beam: only half of FFS, beam mirrored at IP
- Static imperfections: transverse misalignments, rolls and magnetic strength errors
- Monte Carlo simulations. Tuning goal: 90% of machines successfully tuned
- Tuning time is essential for collider performance





J. Ogren, A. Latina, R. Tomas and D. Schulte, *Tuning of the CLIC 380 GeV Final-Focus System with Static Imperfections*, CERN-ACC-2018-0055, CLIC-Note-1141.

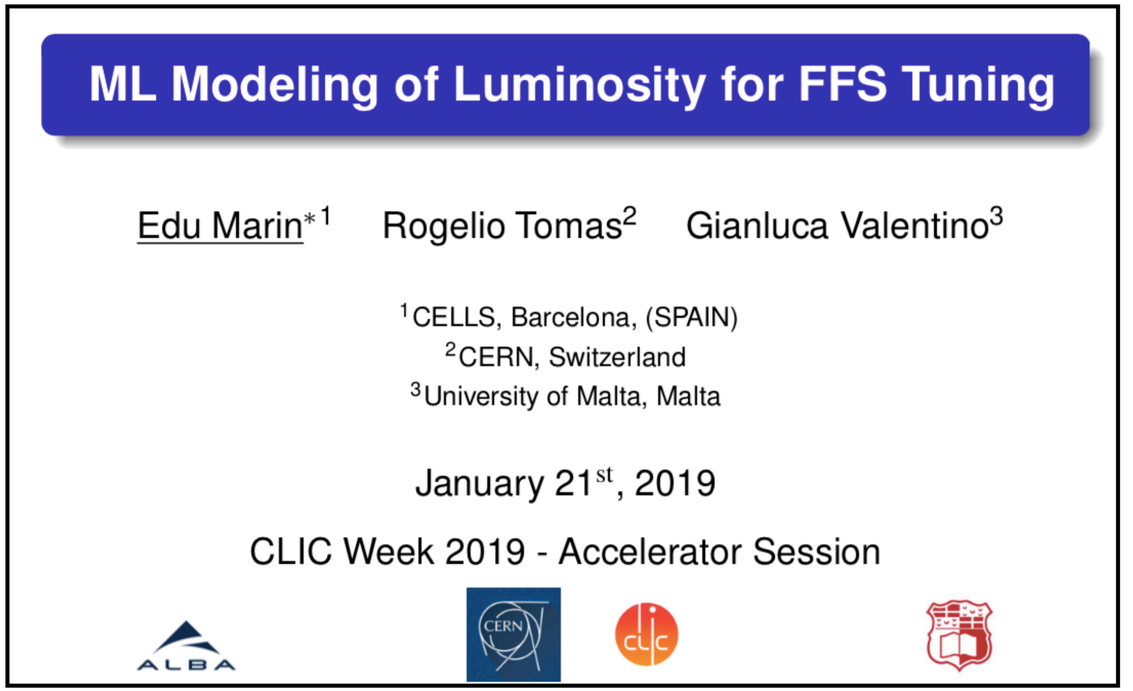
Static imperfections	Specified tolerance (rms error)	Elements
Resolution	20 nm	BPMs
Transverse misalignments	10 μm	BPMs, quadrupoles, multipoles
Roll errors	100 μrad	BPMs, quadrupoles, multipoles
Relative strength error	10^{-4}	Quadrupoles, multipoles

Previous work: ML and CLIC 3 TeV

Understanding tuned machines

- ML to categorize features of tuned machines
- Feature selection: try to determine which imperfections that are more relevant
- 5 categories: correctors, quadrupoles, multipoles, bends, quadupoles & multipoles
- Limited data set (100 cases)
- Ongoing work

For more information:



ML Modeling of Luminosity for FFS Tuning

Edu Marin*[1] Rogelio Tomas[2] Gianluca Valentino[3]

[1]CELLS, Barcelona, (SPAIN)
[2]CERN, Switzerland
[3]University of Malta, Malta

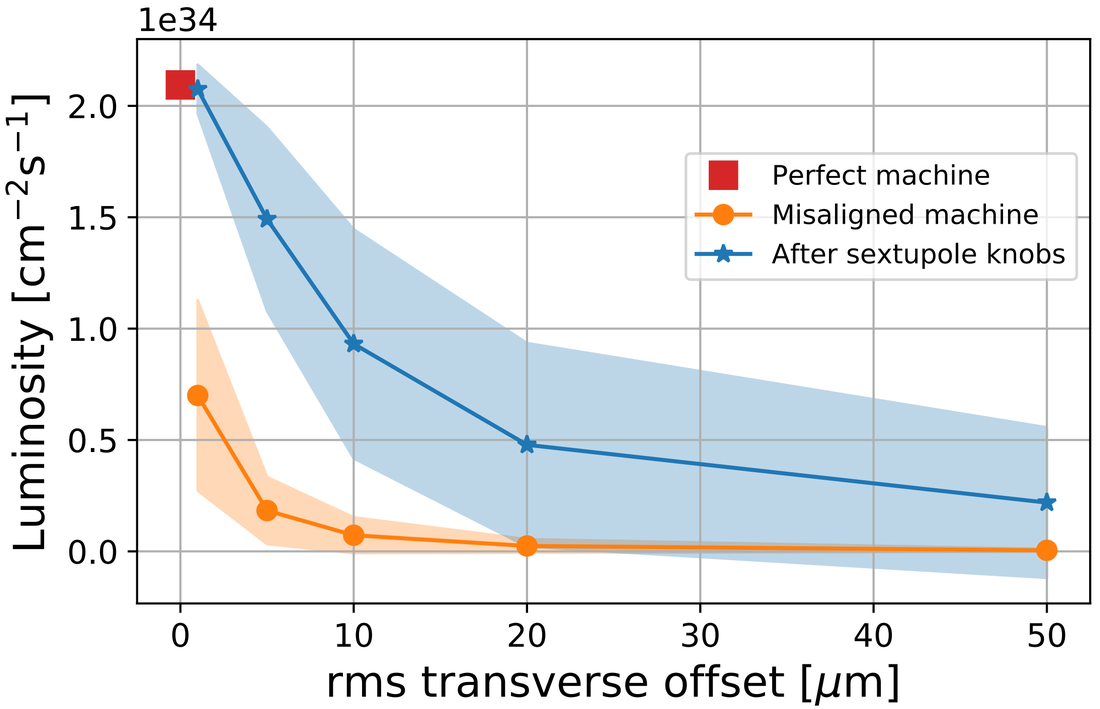
January 21st, 2019

CLIC Week 2019 - Accelerator Session

Different approach: focus on a subset of possible imperfections

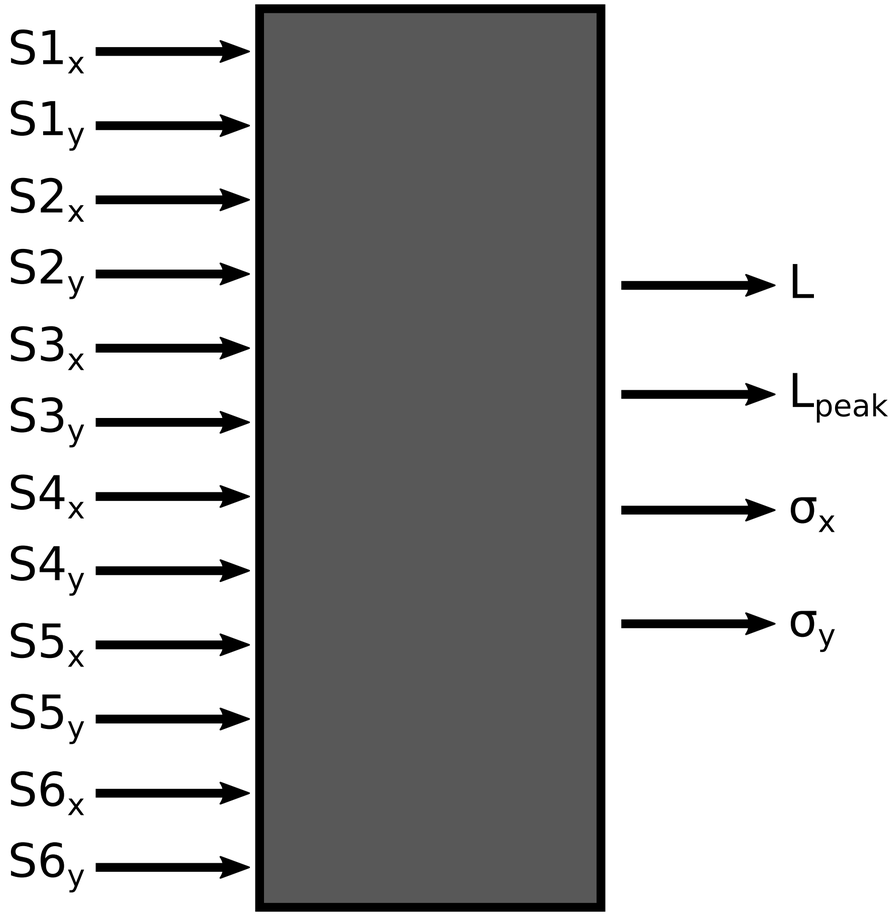
Sextupole offsets

- Sextupole transverse offsets
  - Big impact on luminosity
- Different approaches
  - Orthogonal knobs
  - Random walk
- Efficient tuning is crucial
- Make use of Machine Learning?



Sextupole surrogate model

- Train a model that can map sextupole transverse positions to Luminosity
- Goal: to have a fast estimator



Data generation

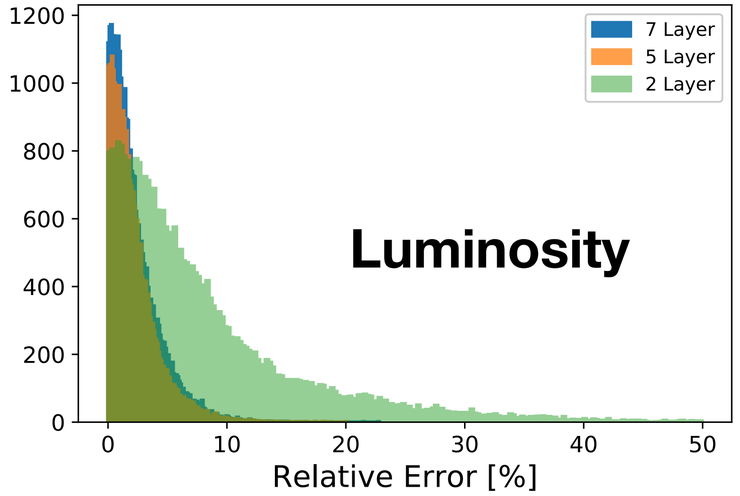
- Simulate perfect machine with sextupole transverse offsets (5, 10, 20 μm rms)
- 1 run = 10 random cases (less then 20 mins) 10,000 jobs at the time
- Generated about 650,000 data points

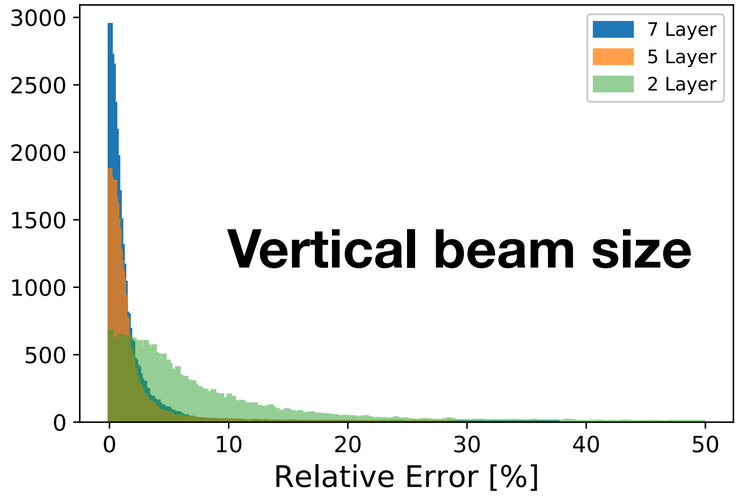
Machine Learning:

- Deep artificial neural networks
- TensorFlow with Python interface Keras

Model performance

	Luminosity			Vertical beam size		
	2 Layers	5 Layers	7 Layers	2 Layers	5 Layers	7 Layers
Mean(Rel_error) [%]	8.1	2.2	2.3	6.8	1.6	1.4
Max(Rel_error) [%]	120	20	23	115	29	38
90% less than [%]	18.9	4.7	5.0	15.1	2.5	3.3



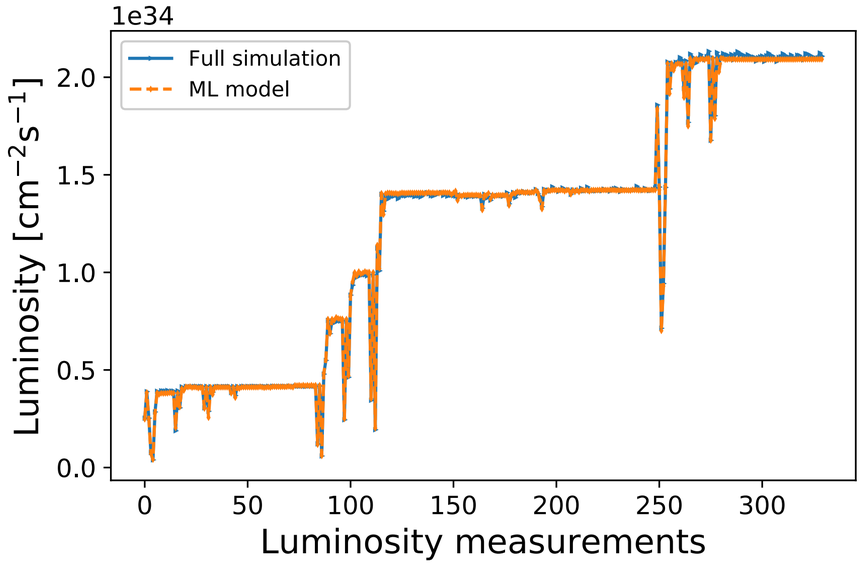


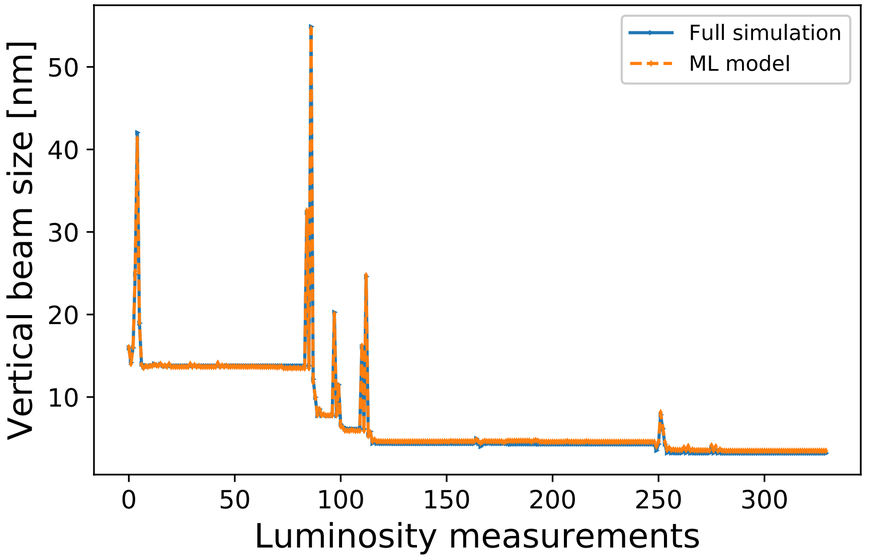
- It seems difficult to get the mean error below 2-3%
- Part of it comes from Luminosity uncertainty (~1%)
- Accuracy is not the most crucial aspect
- A model that correctly characterizes the behavior is very useful…

Compare with full simulation

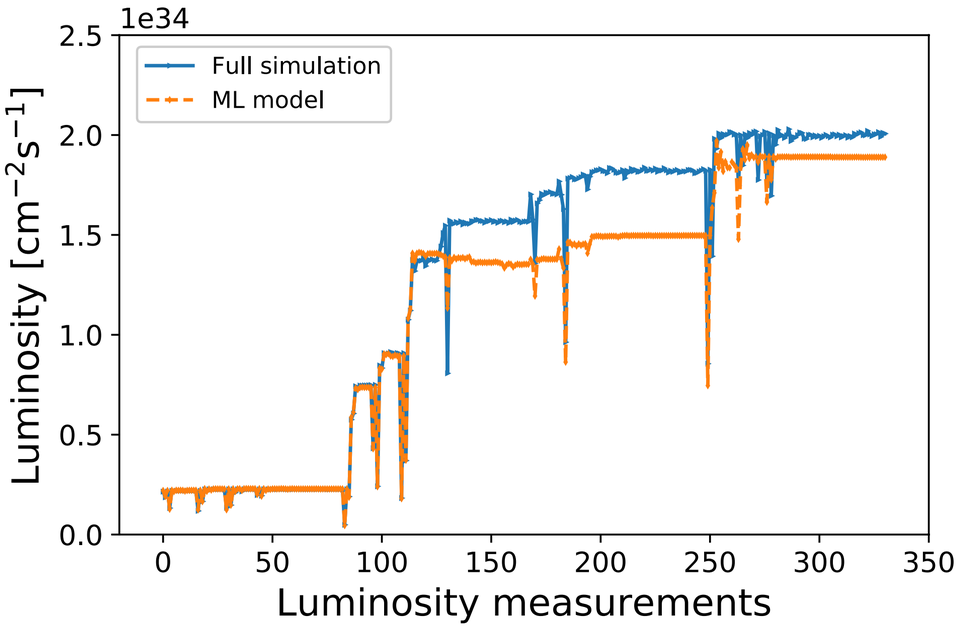
Simulation

- Tuning of the perfect machine with sextupole transverse offsets only
- Use the normal tuning knobs and full-scale tracking (100,000 macroparticles)
- At each step: evaluate luminosity from ML model as well and save to file
- Example: mean(rel_error) ~1%



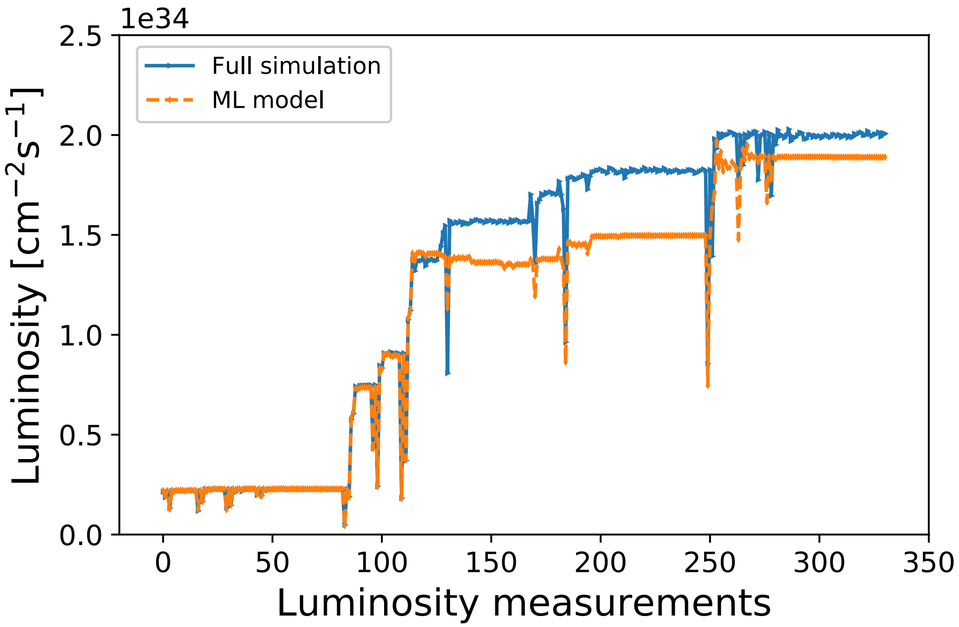


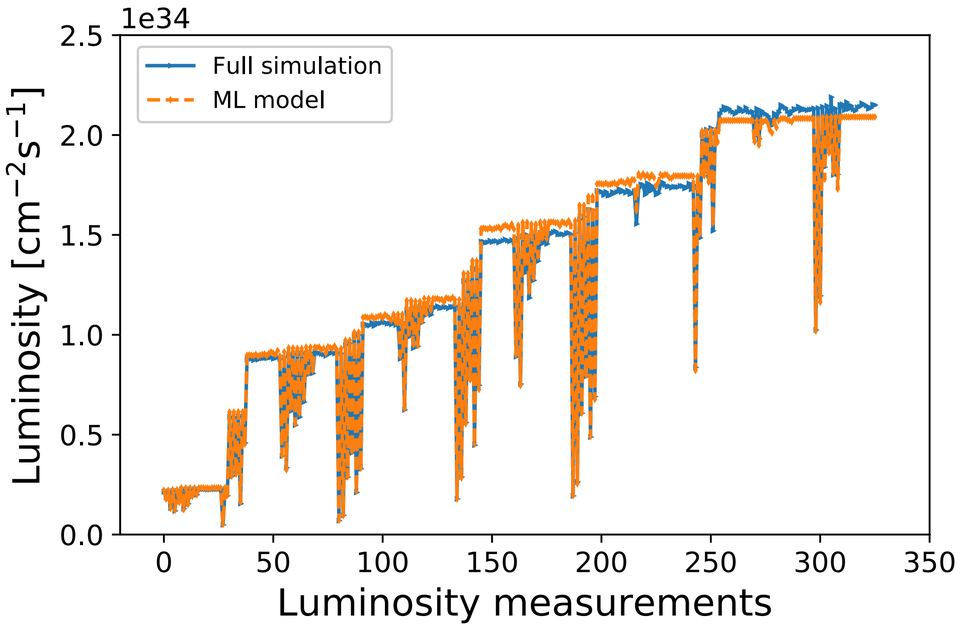
Compare with full simulation



- Not always good agreement
- Sometimes the sextupole knobs tune towards large offsets
- Outside of reliable range of ML model

Compare with full simulation



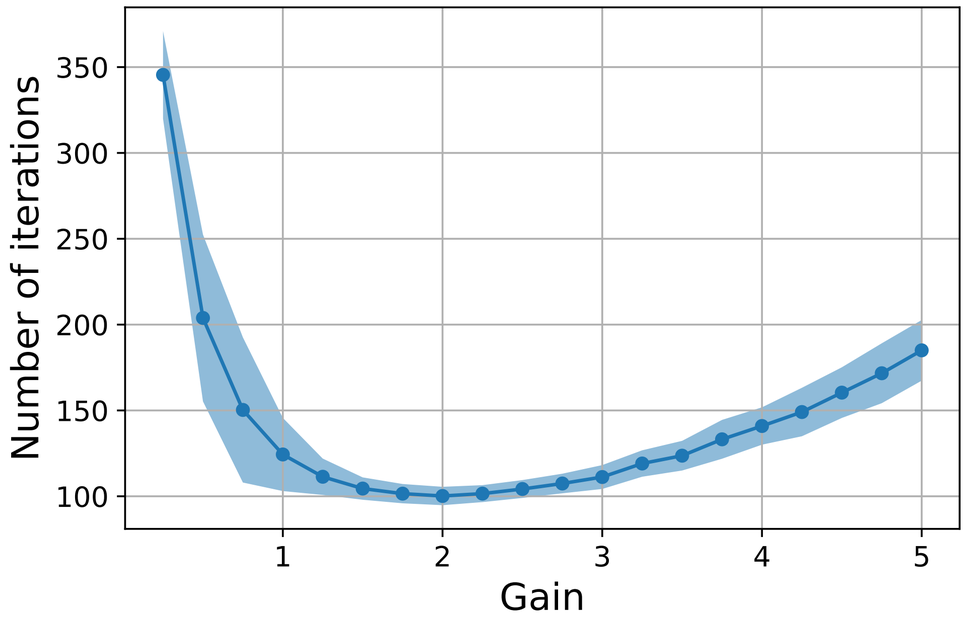


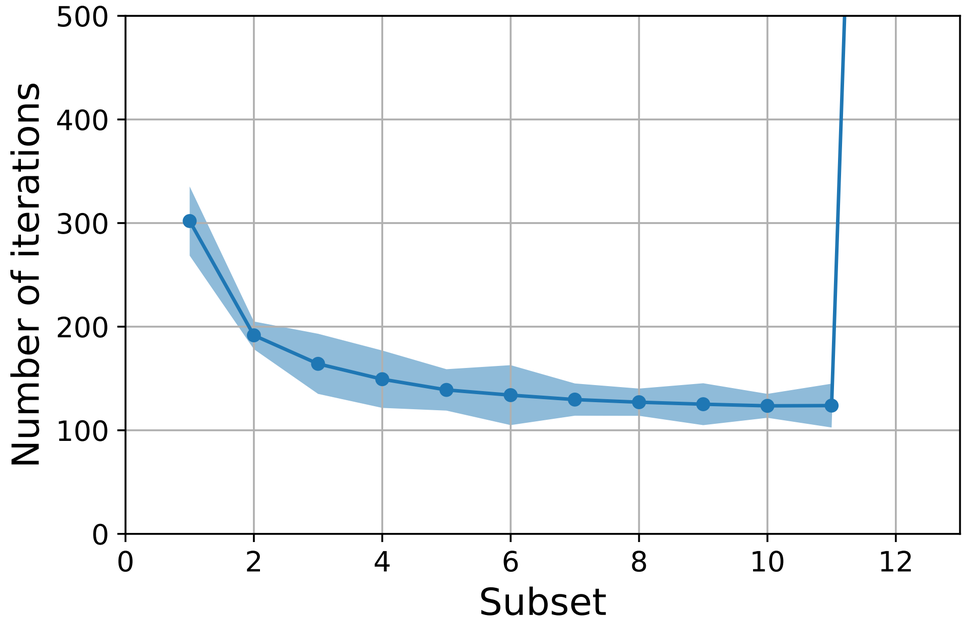
- Not always good agreement
- Sometimes the sextupole knobs tune towards large offsets
- Outside of reliable range of ML model

- Improvement of knobs
- Method designed using ML model
- Avoid large offsets
- Better agreement with ML model

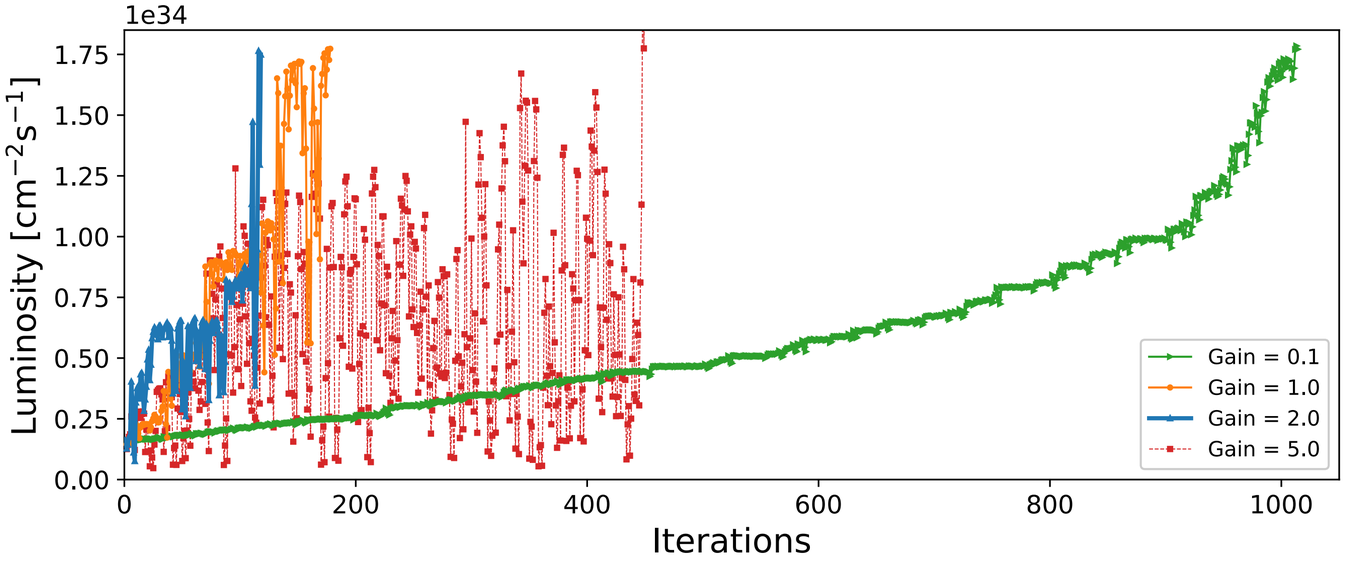
Random walk algorithm

- ML model: 1000 iteration random walk tuning takes a few seconds
- Full-scale simulation: 1000 iterations random walk tuning takes ~8h
- Use ML model to optimize algorithm
- For each setting: 100 different random seeds, each tuned 100 times
- Random walk:
  - Select a subset out of the 12 DOF
  - Make steps in random direction: gain*[-1, -0.5, 0.5, 1]

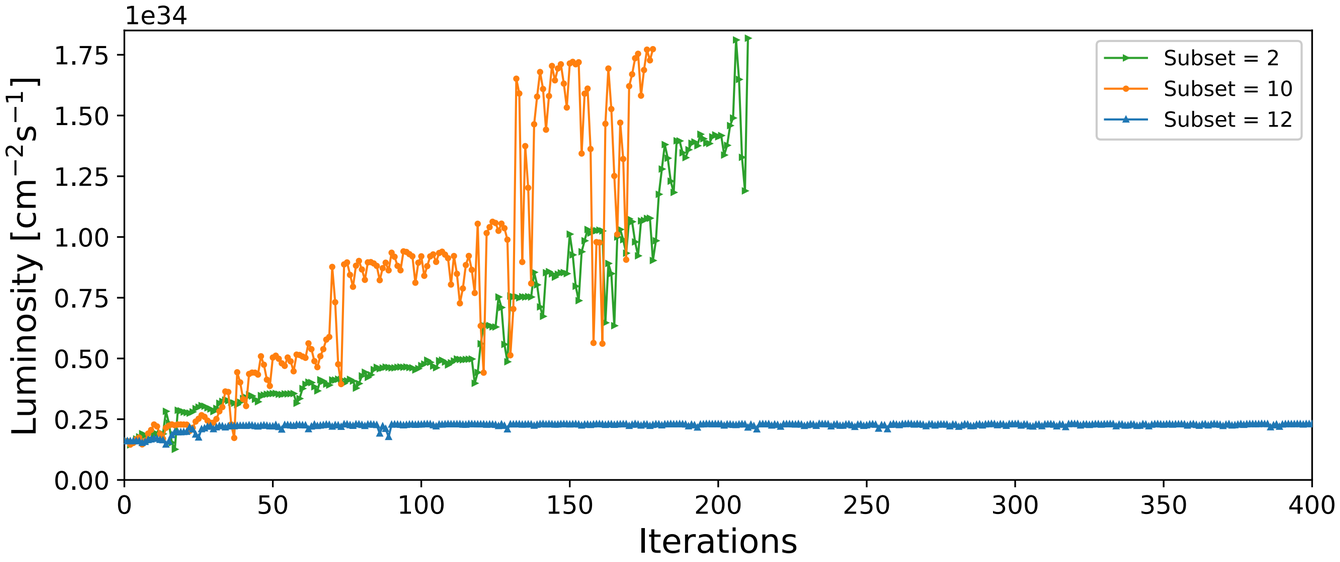




Random walk algorithm - full simulation



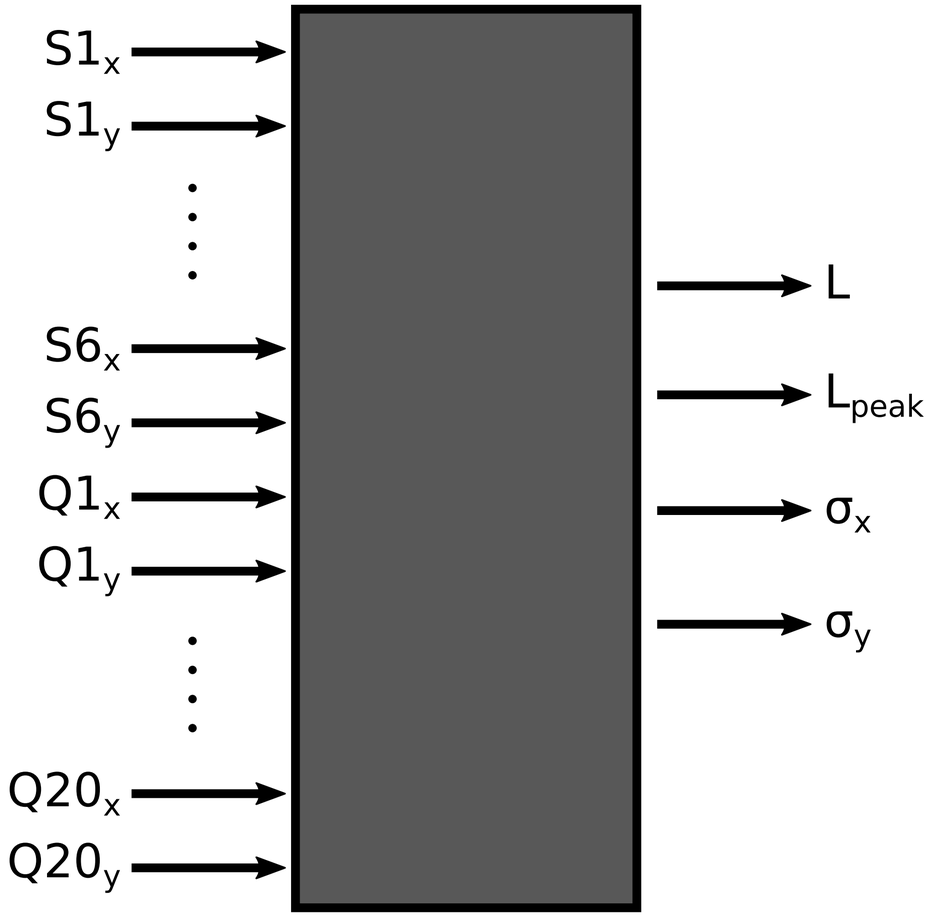
- Gain = 1 and 2 fastest



- Subset = 10 slightly better than subset = 2
- Subset = 12 did not converge in 2000 iterations

Sextupole and quadrupole offsets

- Include transverse offsets on sextupoles (12) and quadrupoles (20)
- Number of inputs: **52**



Data generation

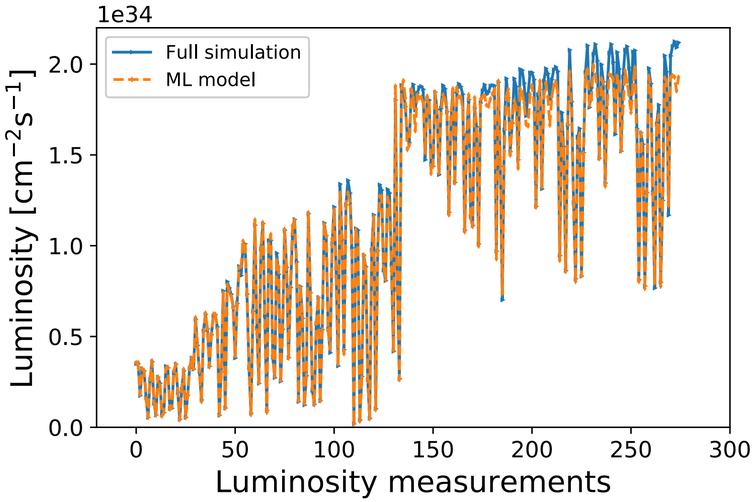
- Simulate perfect machine with sextupole and quadrupole transverse offsets
- Generated about 480,000 data points

Model

- Difficult to train model
- Cut data: $L < 10^{31}$ $cm^{-2}s^{-1}$
- Small range

Sextupole and quadrupole tuning?

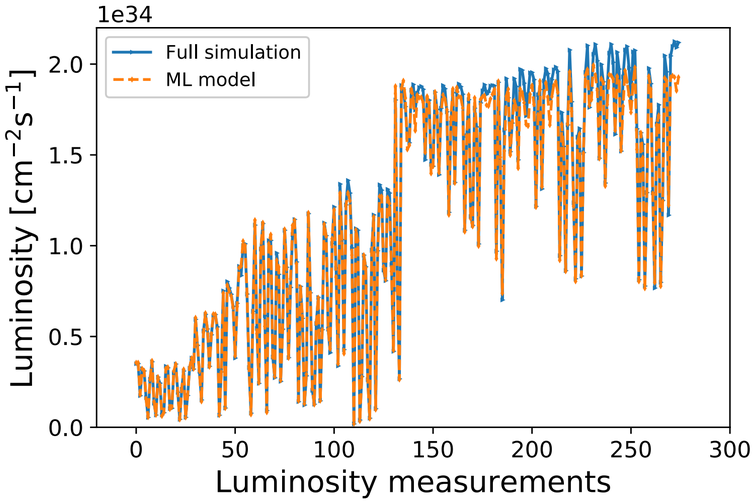
- First attempt of random walk tuning moving quadrupoles and sextupoles
- Previously: BBA separately from sextupole tuning

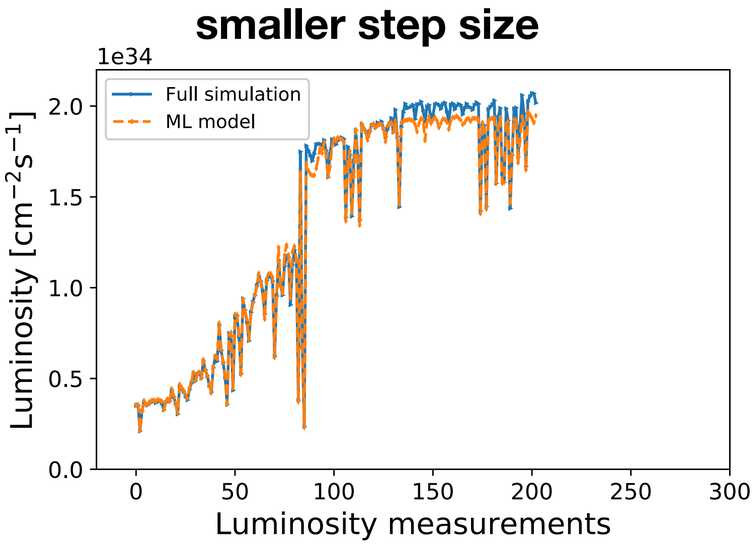


- Not so robust yet
- Fine-tune algorithm (adjust step size)

Sextupole and quadrupole tuning?

- First attempt of random walk tuning moving quadrupoles and sextupoles
- Previously: BBA separately from sextupole tuning

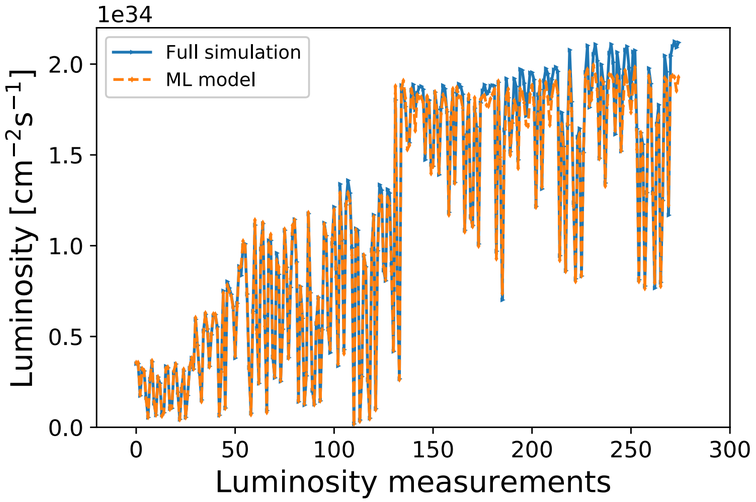




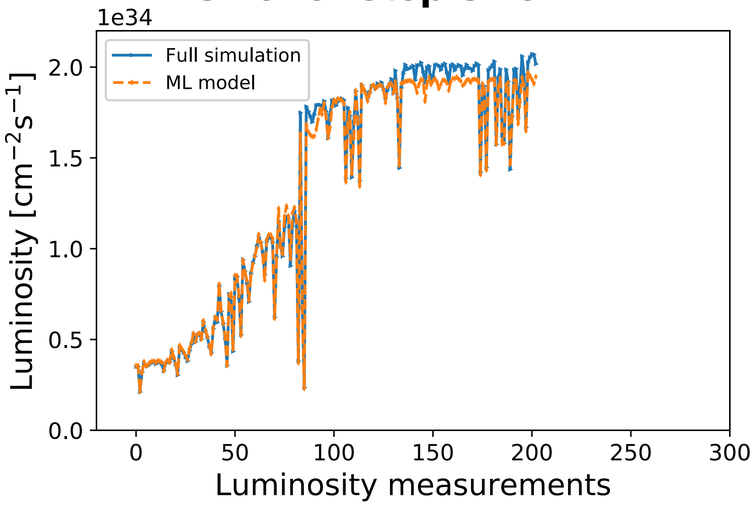
- Not so robust yet
- Fine-tune algorithm (adjust step size)

Sextupole and quadrupole tuning?

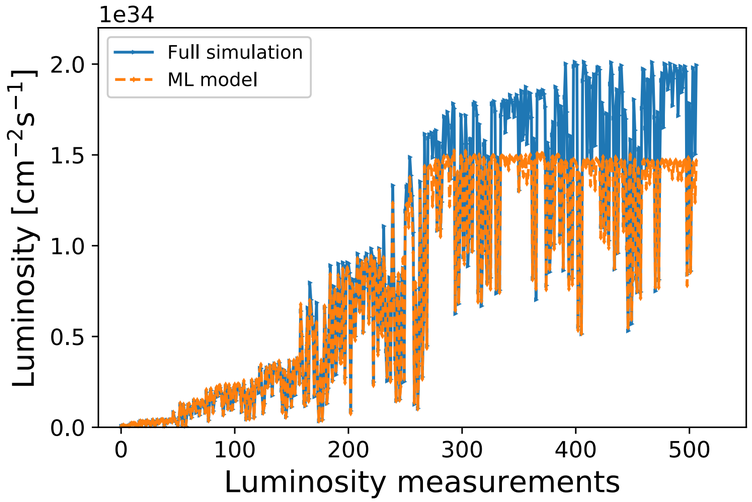
- First attempt of random walk tuning moving quadrupoles and sextupoles
- Previously: BBA separately from sextupole tuning



smaller step size



Different seed



ML model needs improvement

- Not so robust yet
- Fine-tune algorithm (adjust step size)

Summary

- Tuning is crucial for a linear collider final-focus system - interesting problem
- Neural networks to act as surrogate models for CLIC 380 GeV FFS
- Fast simulations - testing of algorithms. Ex: optimizing random walk
- Also implemented a model including quadrupole offsets

Summary

- Tuning is crucial for a linear collider final-focus system - interesting problem
- Neural networks to act as surrogate models for CLIC 380 GeV FFS
- Fast simulations - testing of algorithms. Ex: optimizing random walk
- Also implemented a model including quadrupole offsets

Outlook

- Extend model to include more imperfections
- Improve on models: do we need more data or more knowledge?
- Can we gain more knowledge from the models themselves?
- Solving the inverse problem?
- ML techniques for the tuning itself?
- Reinforcement learning? Other techniques?
- Related topics: luminosity estimate from other beam-beam signals?

Thank you!

Backup slides

Model architecture

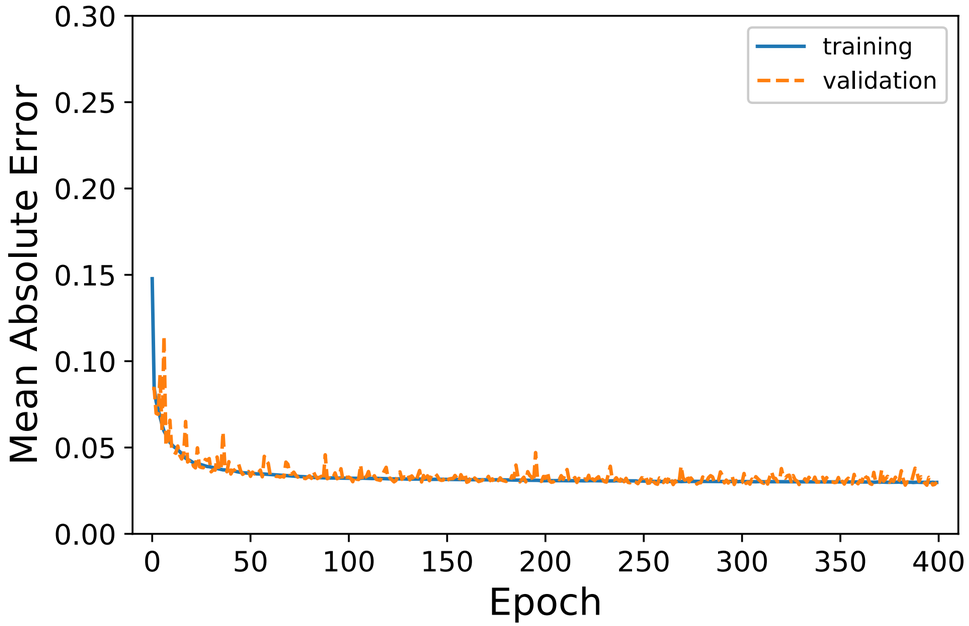
```
import tensorflow as tf
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(12, activation='relu', input_shape=(x.shape[1],)))
model.add(tf.keras.layers.Dense(12, activation='relu'))
model.add(tf.keras.layers.Dense(12, activation='relu'))
model.add(tf.keras.layers.Dense(12, activation='relu'))
model.add(tf.keras.layers.Dense(12, activation='relu'))
model.add(tf.keras.layers.Dense(4))
model.compile(optimizer='adam', loss='mse', metrics=['mae'])

history=model.fit(x, y, epochs=400, batch_size=50, validation_split=0.1)
```

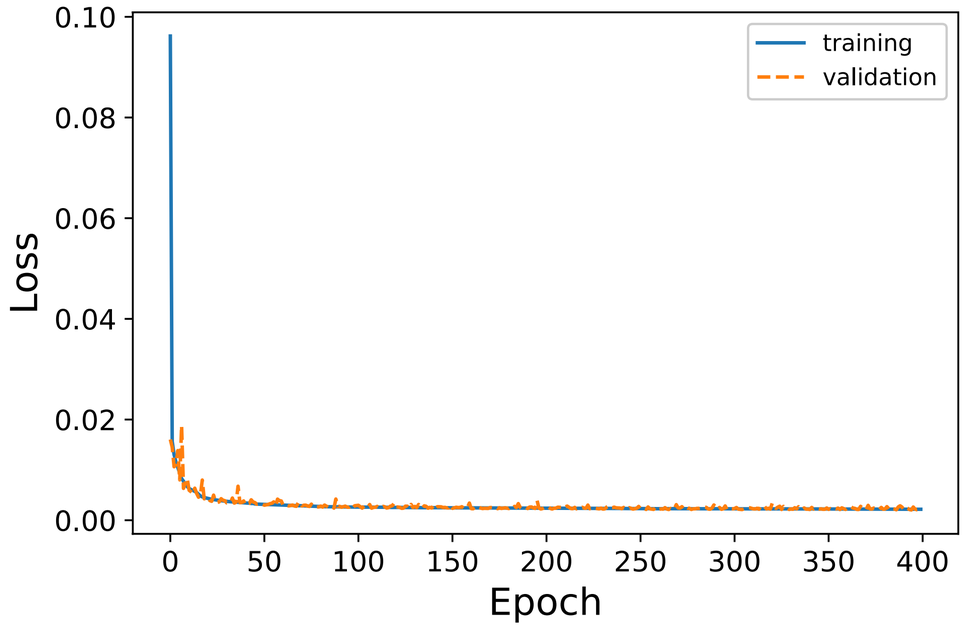
Sextupole model

```
import tensorflow as tf
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(52, activation='relu', input_shape=(x.shape[1],)))
model.add(tf.keras.layers.Dense(52, activation='relu'))
model.add(tf.keras.layers.Dense(52, activation='relu'))
model.add(tf.keras.layers.Dense(52, activation='relu'))
model.add(tf.keras.layers.Dense(52, activation='relu'))
model.add(tf.keras.layers.Dense(26, activation='relu'))
model.add(tf.keras.layers.Dense(26, activation='relu'))
model.add(tf.keras.layers.Dense(26, activation='relu'))
model.add(tf.keras.layers.Dense(1))

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

model.fit(x, y, epochs=1000, batch_size=50, validation_split=0.1)
```

Sextupole and quadrupole model

Training history - Sextupole model, 5 Layers





Sextupole knobs

- Transverse movement of sextupoles

- The response matrix of 2nd order moments
  - SVD to find orthogonal 'knobs' (vectors from matrix V)

$$U\lambda V^T = \begin{bmatrix} \frac{\partial \sigma_{xx}}{\partial X_1} & \cdots & \frac{\partial \sigma_{xx}}{\partial X_6} & \frac{\partial \sigma_{xx}}{\partial Y_1} & \cdots & \frac{\partial \sigma_{xx}}{\partial Y_6} \\ \frac{\partial \sigma_{xx'}}{\partial X_1} & \cdots & \frac{\partial \sigma_{xx'}}{\partial X_6} & \frac{\partial \sigma_{xx'}}{\partial Y_1} & \cdots & \frac{\partial \sigma_{xx'}}{\partial Y_6} \\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial \sigma_{zz}}{\partial X_1} & \cdots & \frac{\partial \sigma_{zz}}{\partial X_6} & \frac{\partial \sigma_{zz}}{\partial Y_1} & \cdots & \frac{\partial \sigma_{zz}}{\partial Y_6} \end{bmatrix}$$

- Each knob is scanned over some range and luminosity is maximized

- Similarly for the octupoles