Graph Learning for Efficient and Explainable Operation of Particle Accelerators

Chris Tennant, Song Wang, Jundong Li

November 20, 2025

DOE NP PI Exchange Meeting

Jefferson Lab
U.S. Department of ENERGY
University of Virginia

Outline

- Introduction
- Motivation
- Method
- Explainability
- Example: Data Exploration
- Summary

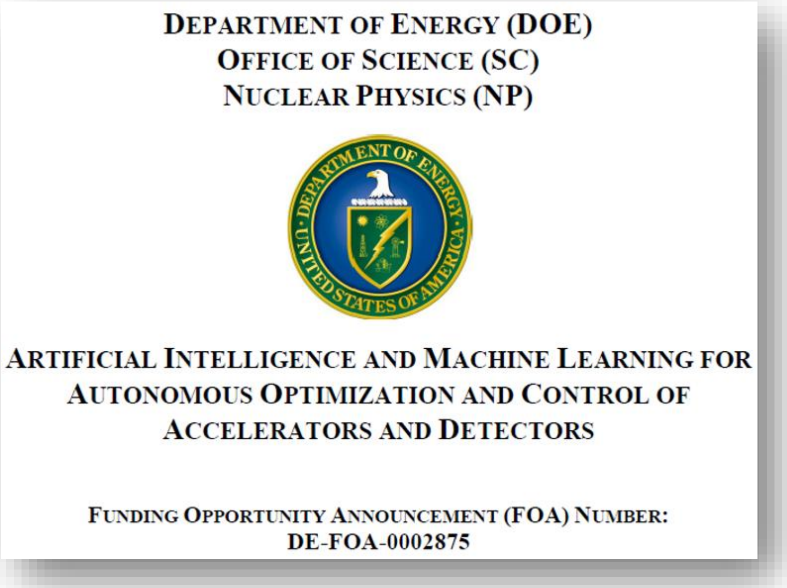
leverage deep learning on graph representations of a beamline for greater insights into complex systems

“Graph Learning for… Operation of Particle Accelerators”

“The fundamental idea is to apply deep learning over graph representations of the CEBAF injector beamline in order to extract information-rich, low-dimensional embeddings. With access to embeddings that capture the complex relationships of a many-dimensional beamline over time, several unique applications will be developed…”

leverage deep learning on graph representations of a beamline for greater insights into complex systems

- this project represents the intersection of:
  - ✓ graph analysis
  - ✓ deep learning
  - ✓ self-supervised learning
  - ✓ dimensionality reduction
  - ✓ visualization
  - ✓ explainable AI



DEPARTMENT OF ENERGY (DOE)
OFFICE OF SCIENCE (SC)
NUCLEAR PHYSICS (NP)

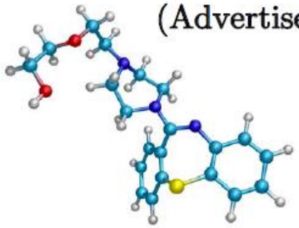
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FOR AUTONOMOUS OPTIMIZATION AND CONTROL OF ACCELERATORS AND DETECTORS

FUNDING OPPORTUNITY ANNOUNCEMENT (FOA) NUMBER:
DE-FOA-0002875

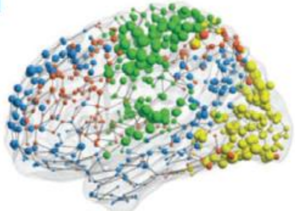
Introduction: What is a Graph?

- graphs are a general language for describing and analyzing entities with relations

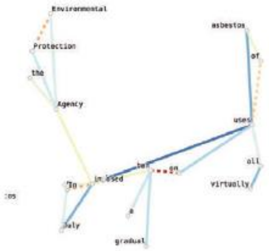
Social networks (Advertisement)



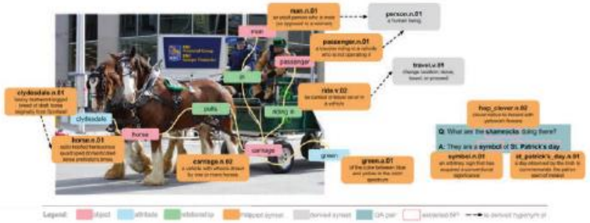
Drug/Material molecules (Chemistry)



Brain connectivity (Neuroscience)

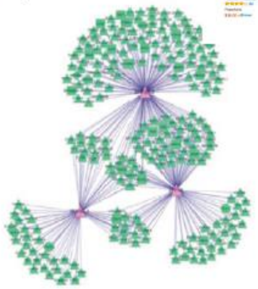


Words relationships (NLP)

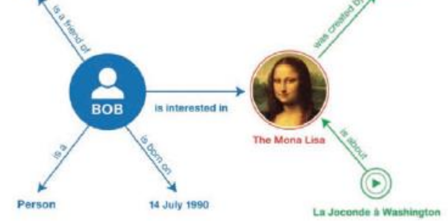


Scene understanding

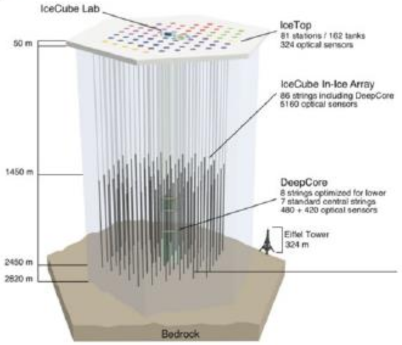
Recommender systems (Amazon, Netflix)



Gene Regulatory Network

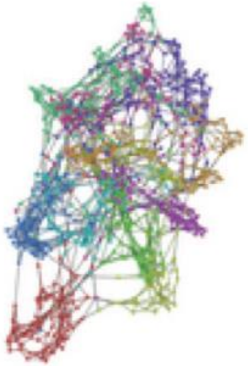


Knowledge graphs

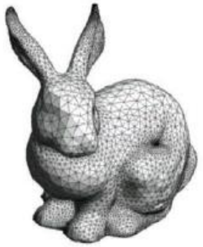


Neutrino detection (High-energy Physics)

=



Graph



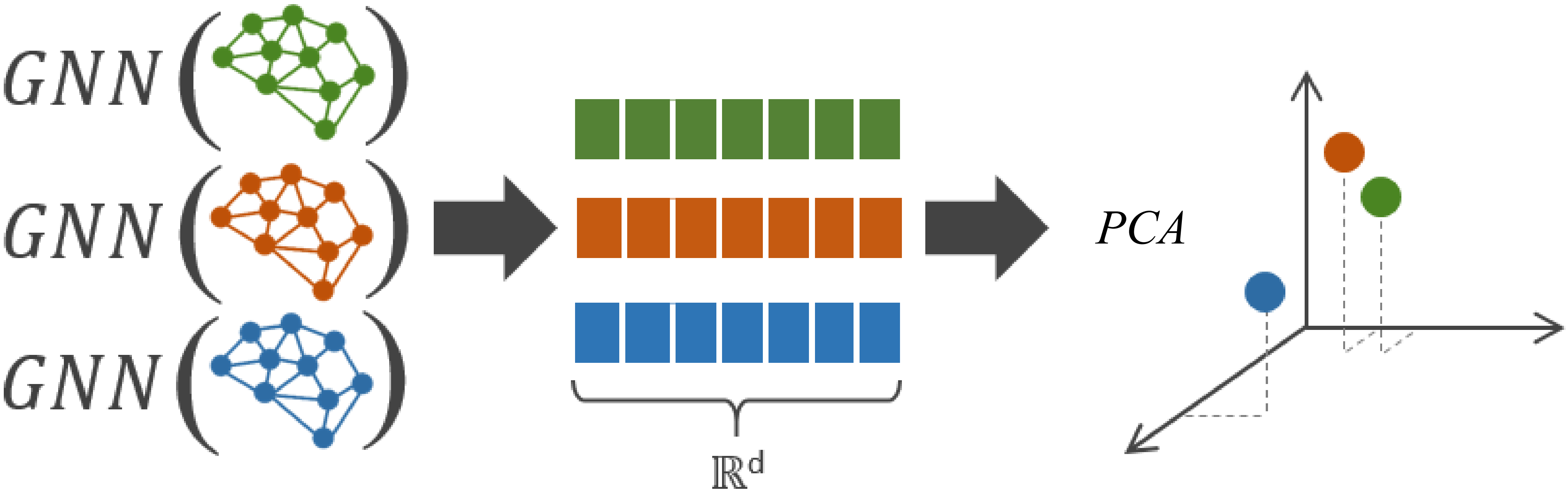
3D Meshes (Computer Graphics)

Transportation networks

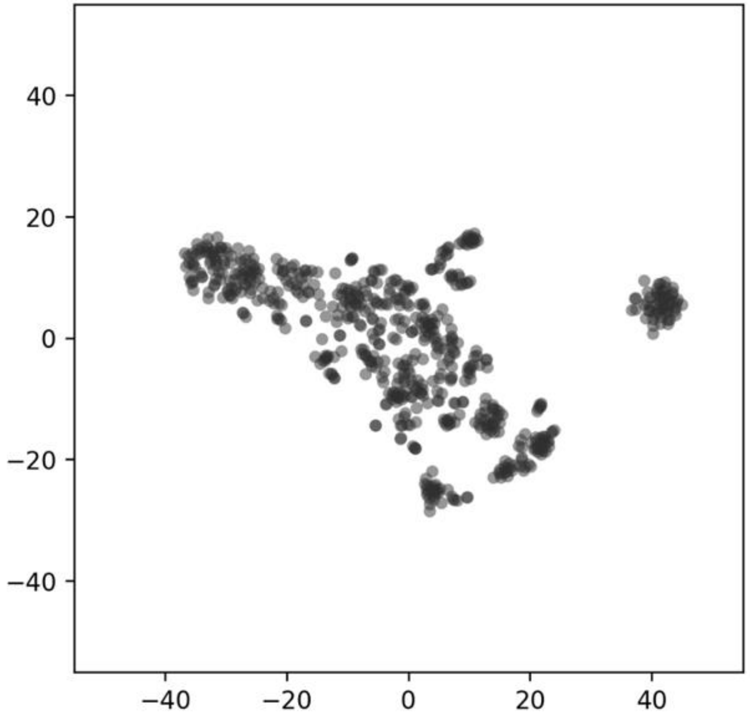
(courtesy X. Bresson)

Introduction: Whole-Graph Embedding

- represent the state of a beamline at a specific date and time as a graph
- embed graph into 2- or 3-dimensions using a graph neural network (GNN)
  - ✓ a GNN provides a framework for defining a deep neural network on arbitrary graph data

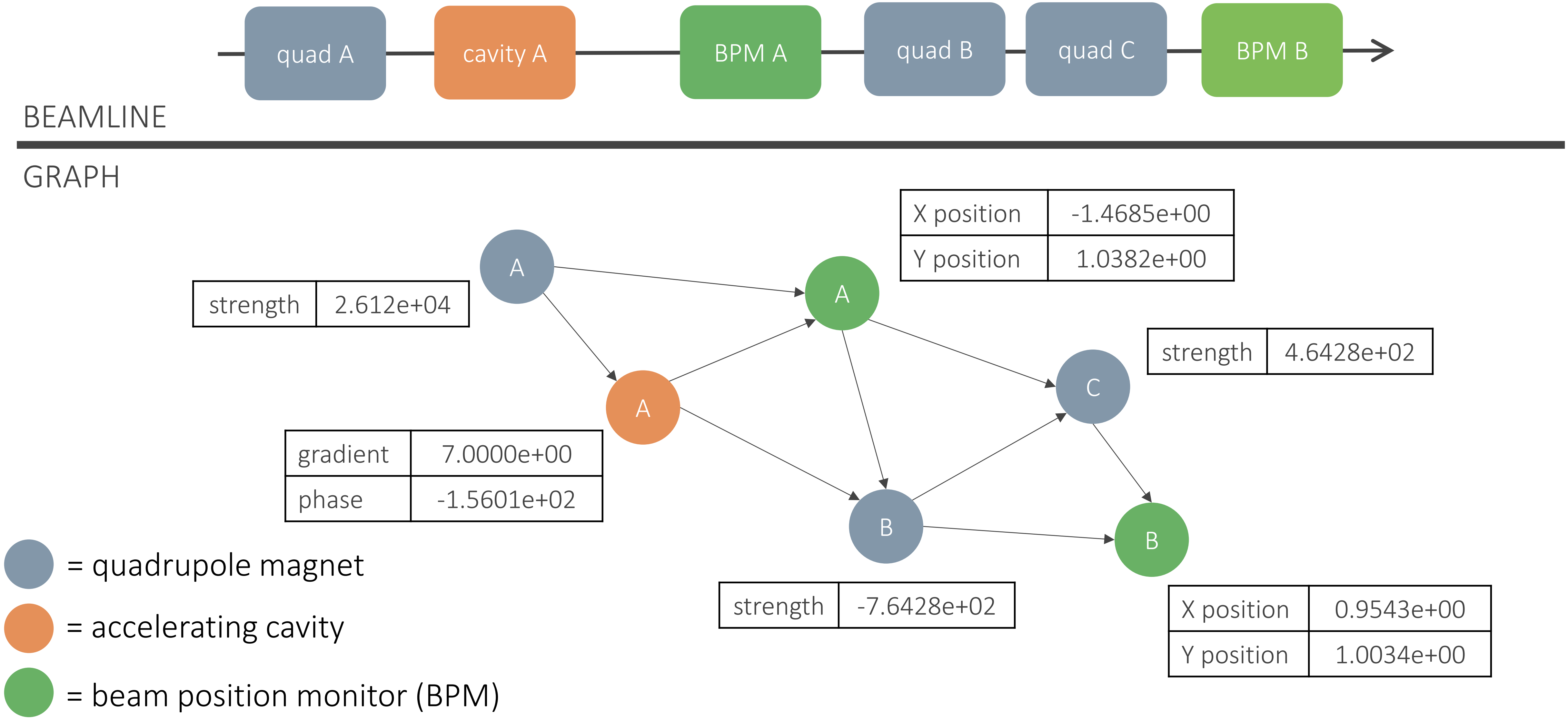


Motivation: Application for Beam Operations



- visualize the underlying distribution/pattern of beamline states
- use cluster analysis to identify regions of parameter space
- monitor changes in the machine and observe trajectory in latent space
  - ✓ more quickly converge to known optimal beamline configurations

Method: Beamline-to-Graph



BEAMLINE

GRAPH

strength	2.612e+04

X position	-1.4685e+00
Y position	1.0382e+00

strength	4.6428e+02

gradient	7.0000e+00
phase	-1.5601e+02

strength	-7.6428e+02

X position	0.9543e+00
Y position	1.0034e+00

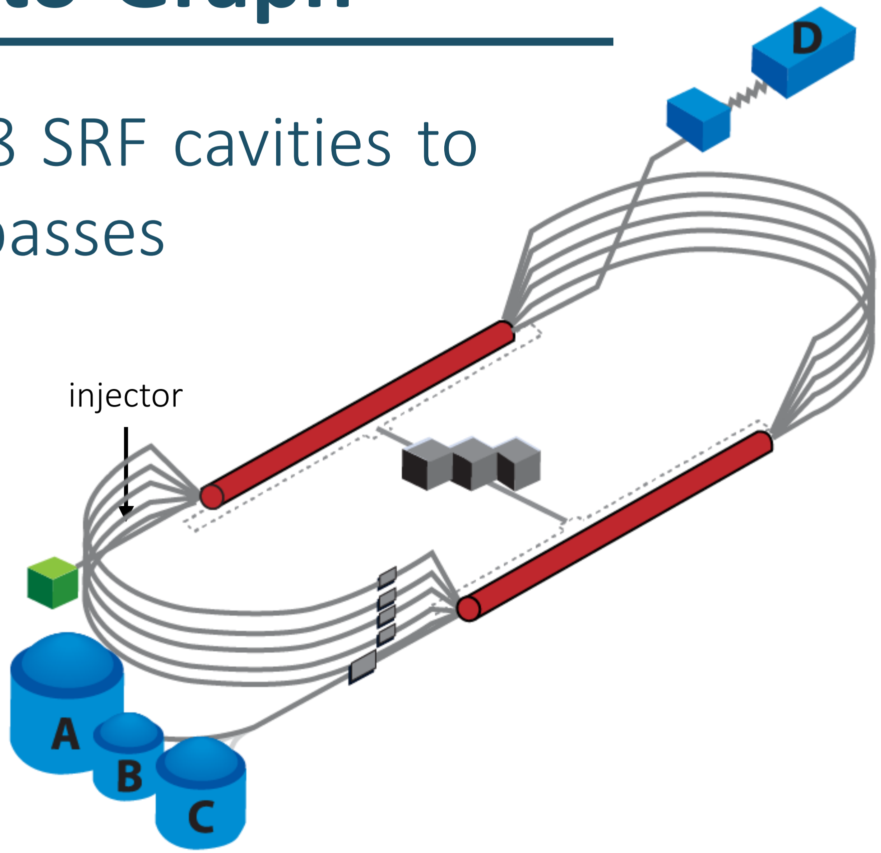
● = quadrupole magnet

● = accelerating cavity

● = beam position monitor (BPM)

Method: Beamline-to-Graph

- CEBAF is a CW recirculating linac utilizing 418 SRF cavities to accelerate electrons up to 12 GeV through 5-passes
- consider a 95 m portion of the injector
- software developed so that:
  - ✓ given a date and time stamp, a beamline is defined from the CEBAF Element Database
  - ✓ node features are populated by process variables stored in the operational archiver
  - ✓ each graph: 11 node types, 206 nodes, 296 node features, and 409 edges
- have generated over 100K graphs from the last few years of operational data for model training and analysis



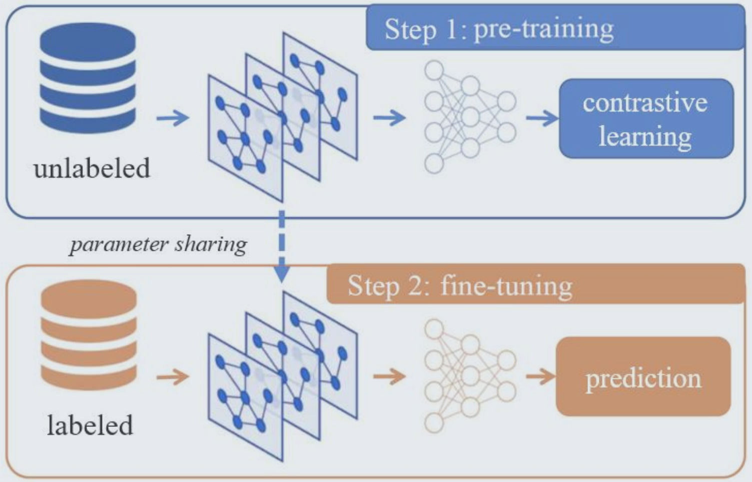
`info.dat`
defines the node types

`meta.dat`
lists the number of each node type in the graph

`node.dat`
[*node ID, node name, node type, node attributes*]

`link.dat`
[*start node ID, end node ID, edge type, edge weight*]

Method: GNN Model Training

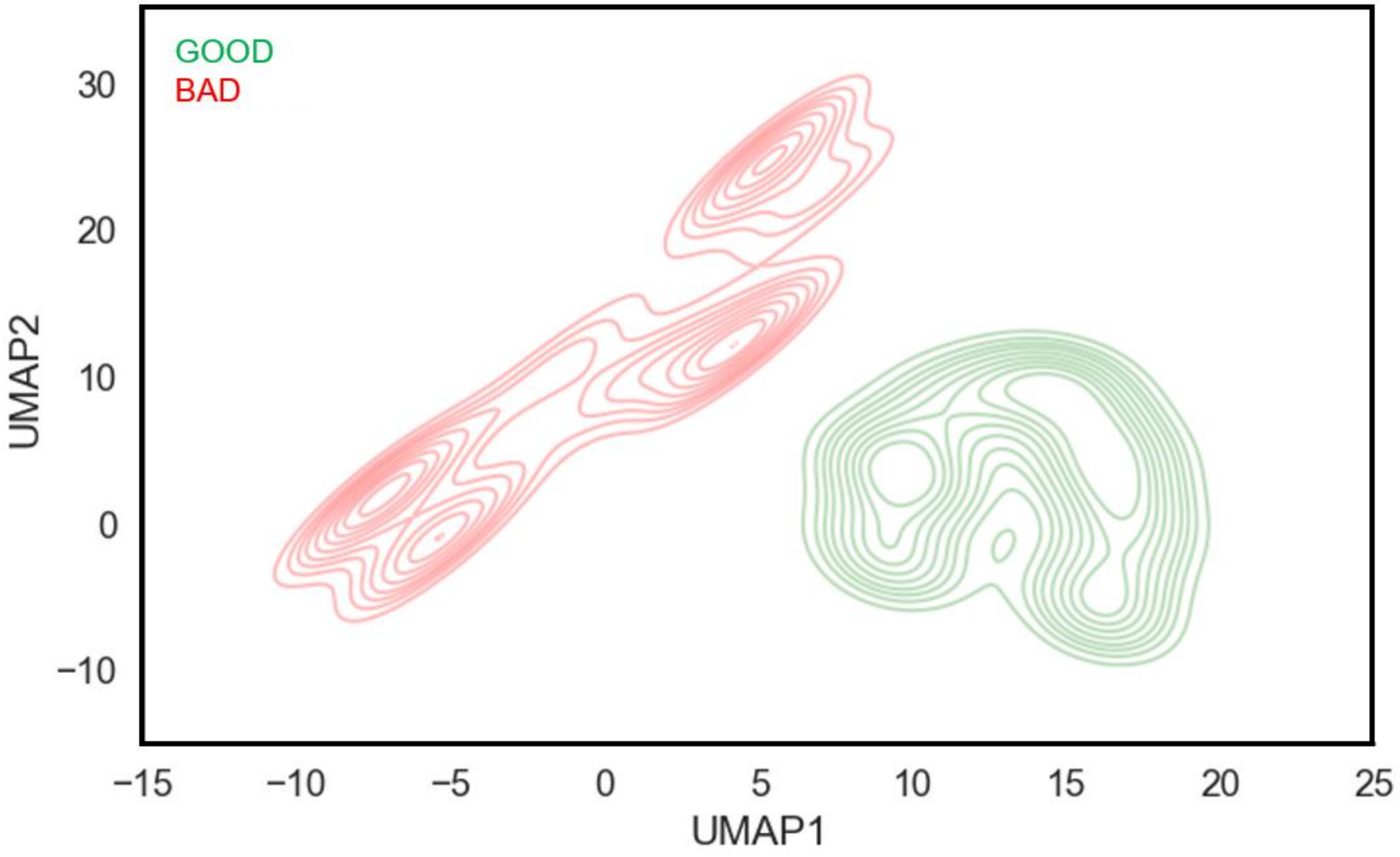


1. pre-training the model on a large set of *unlabeled* data using self-supervised learning
   - ✓ tries to learn as much as possible from the data alone
   - ✓ take advantage of years of operational data stored in the archiver without having to do the expensive task of hand labeling the data
   - ✓ a special class of loss function, known as contrastive loss, maximizes agreement between latent representations from similar graph pairs while minimizing agreement from unlike pairs
2. fine-tune the model on the downstream task of classifying good and bad setups using a smaller *labeled* dataset
3. a dimensionality reduction to visualize the results in 2D

this same basic workflow is being leveraged to produce state-of-the-art results in natural language processing and vision tasks

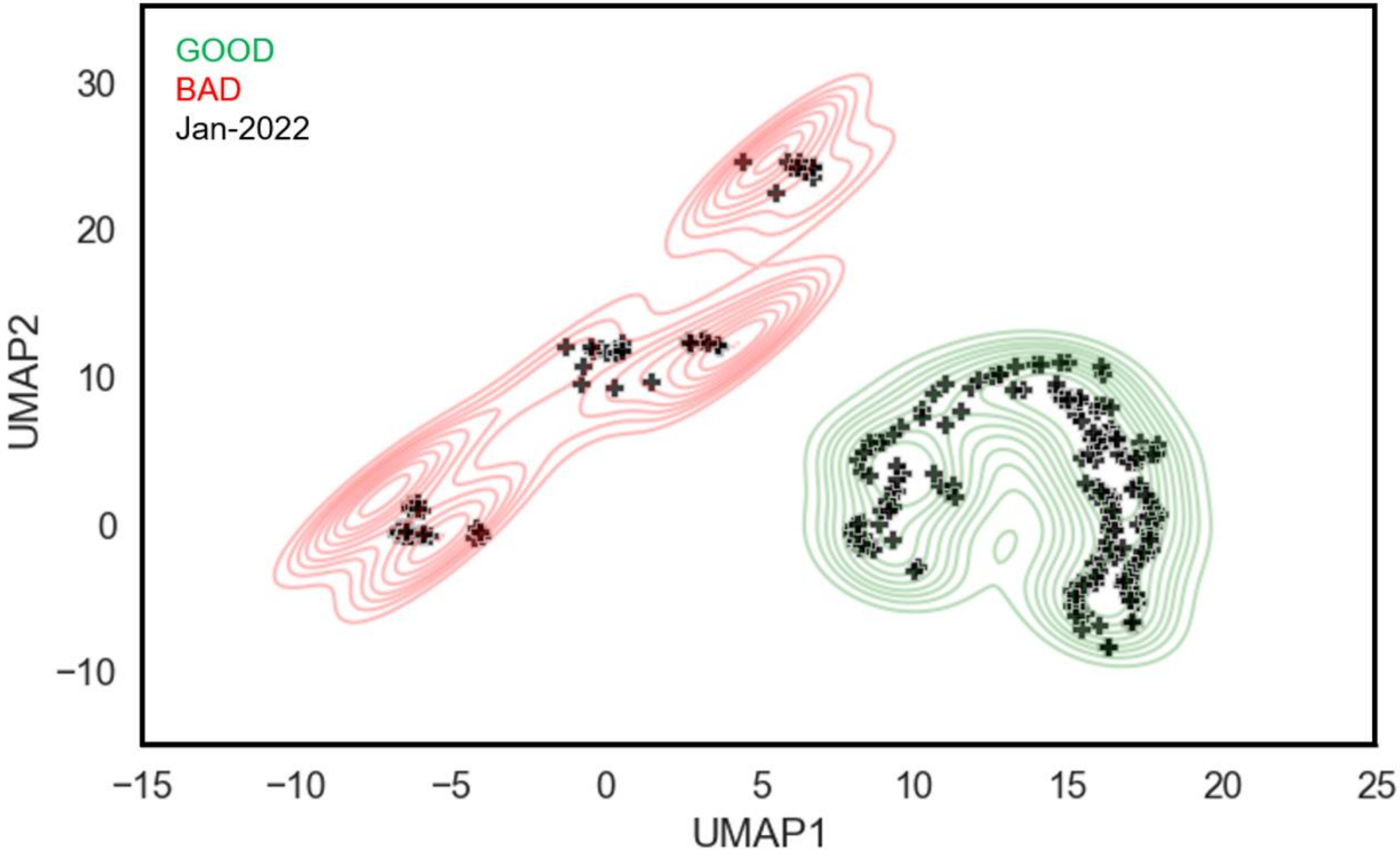
Identifying Regions of Parameter Space

- a GNN encoder pre-trained on the 2,129 unlabeled graphs from 2021
- fine-tune model on the smaller set of labeled (354 good, 324 bad) graphs



Identifying Regions of Parameter Space

- a set of 352 *unlabeled* graphs representing operations in January 2022 were input to the model and their resulting latent representations plotted



S. Wang, et al., Front. Big Data, Sec. Data Mining and Management, vol 7 (2024).

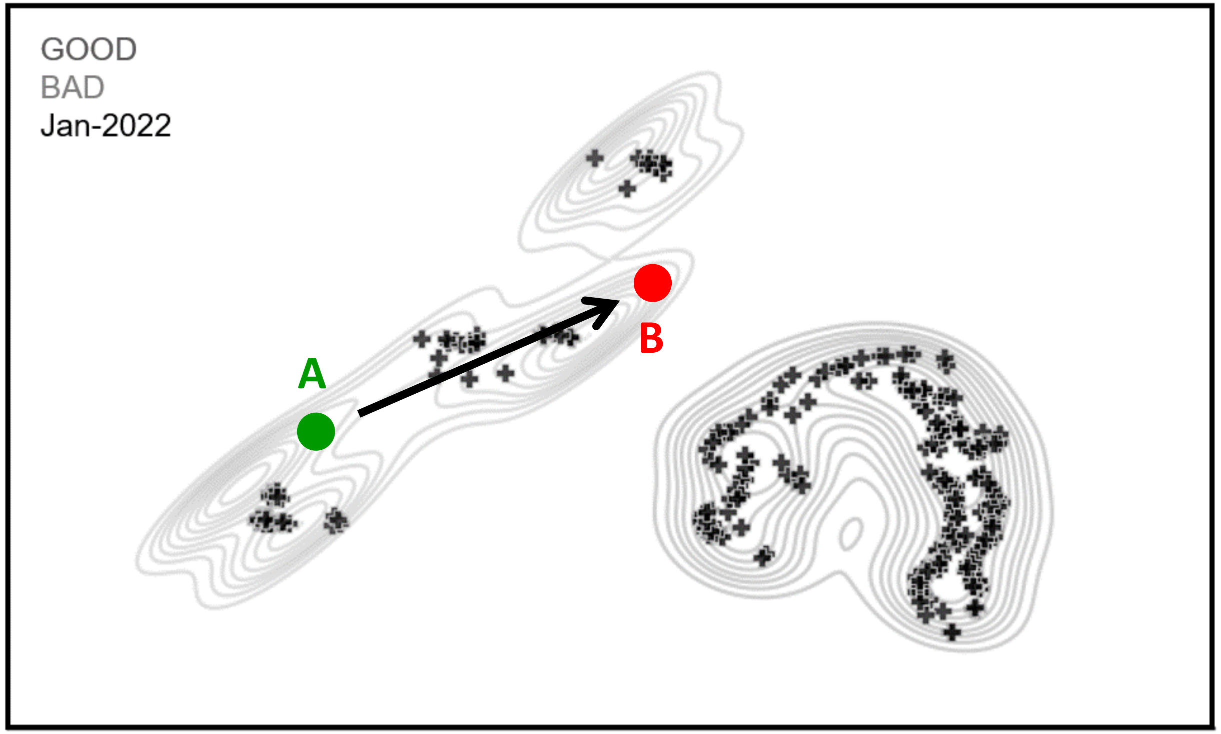
Explainability

- **GOAL:** what setting node is most important (i.e. best *explains*) the move from point A to B in latent space?
- the downstream impact of changing an element depends on the:

  ✓ element type
    - *a quadrupole affects the beam differently than a corrector than an RF phase*

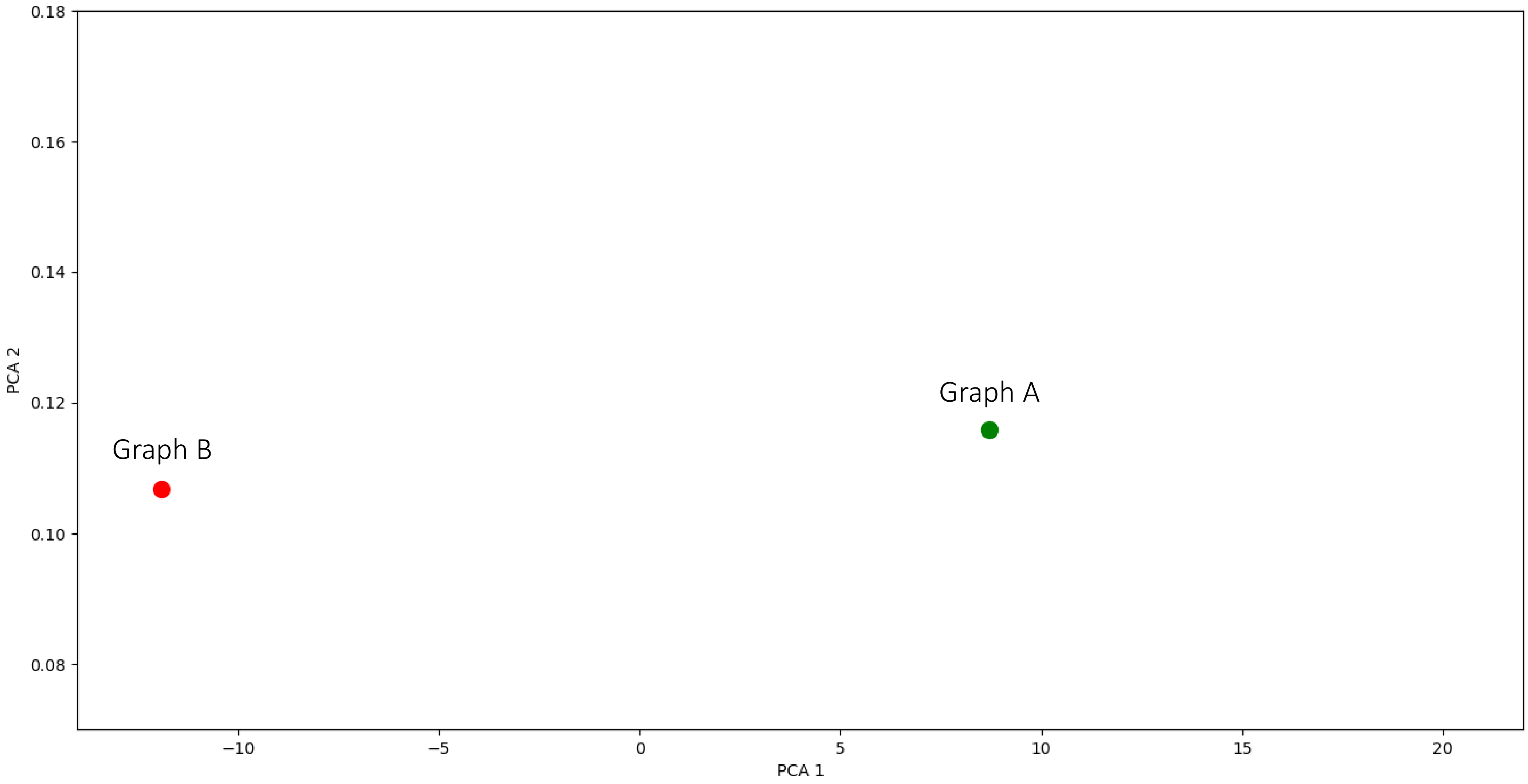
  ✓ element location
    - *a change further upstream provides a bigger lever arm, but it also depends on beam properties at location of change*

  ✓ magnitude of the change
    - *all other things being equal, a larger magnitude will have a bigger affect, however, even if we compare two of the same type of element, location matters*



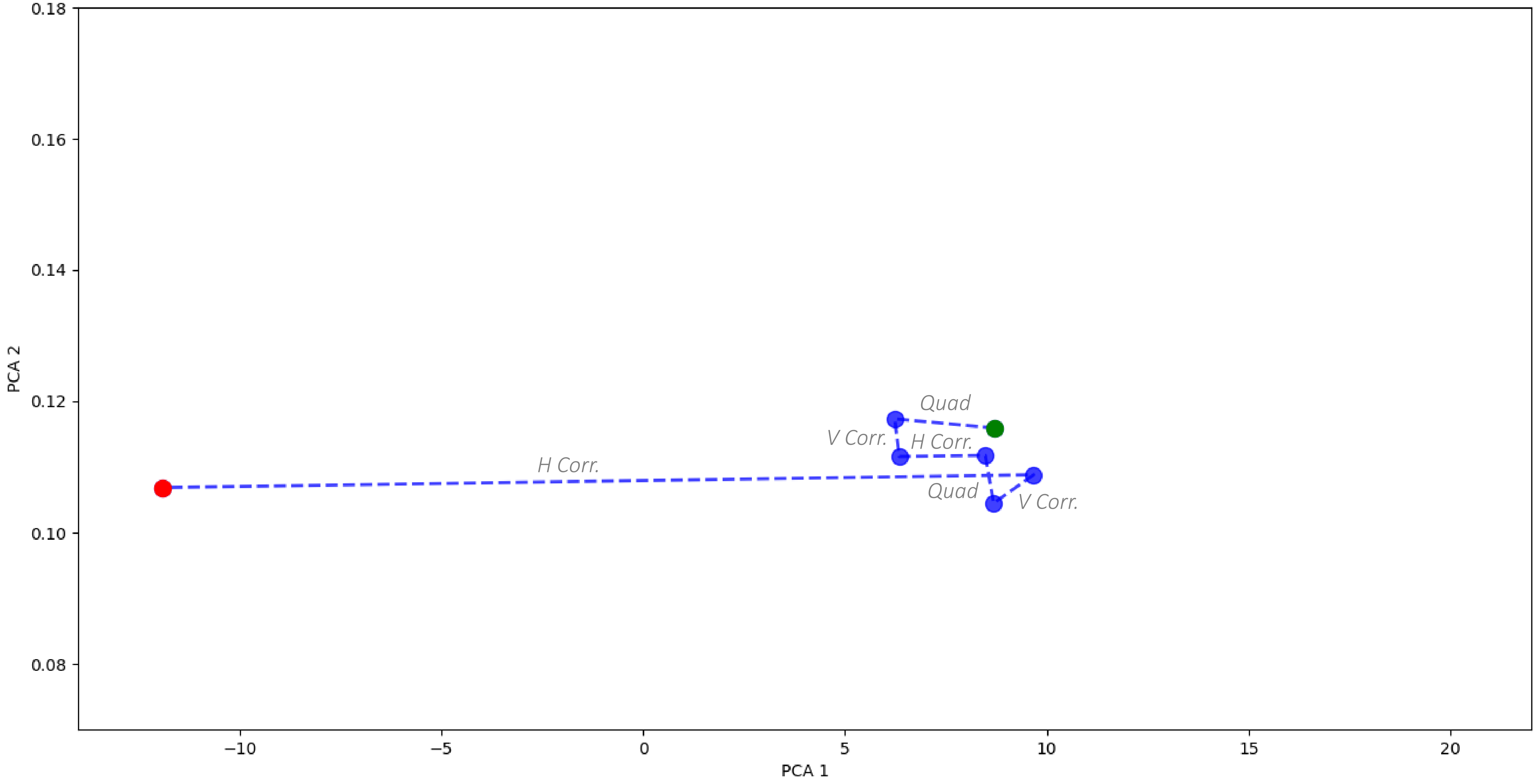
Explainability Example: Graph A → Graph B

- beam-based experiment transitioning from one injector configuration to another



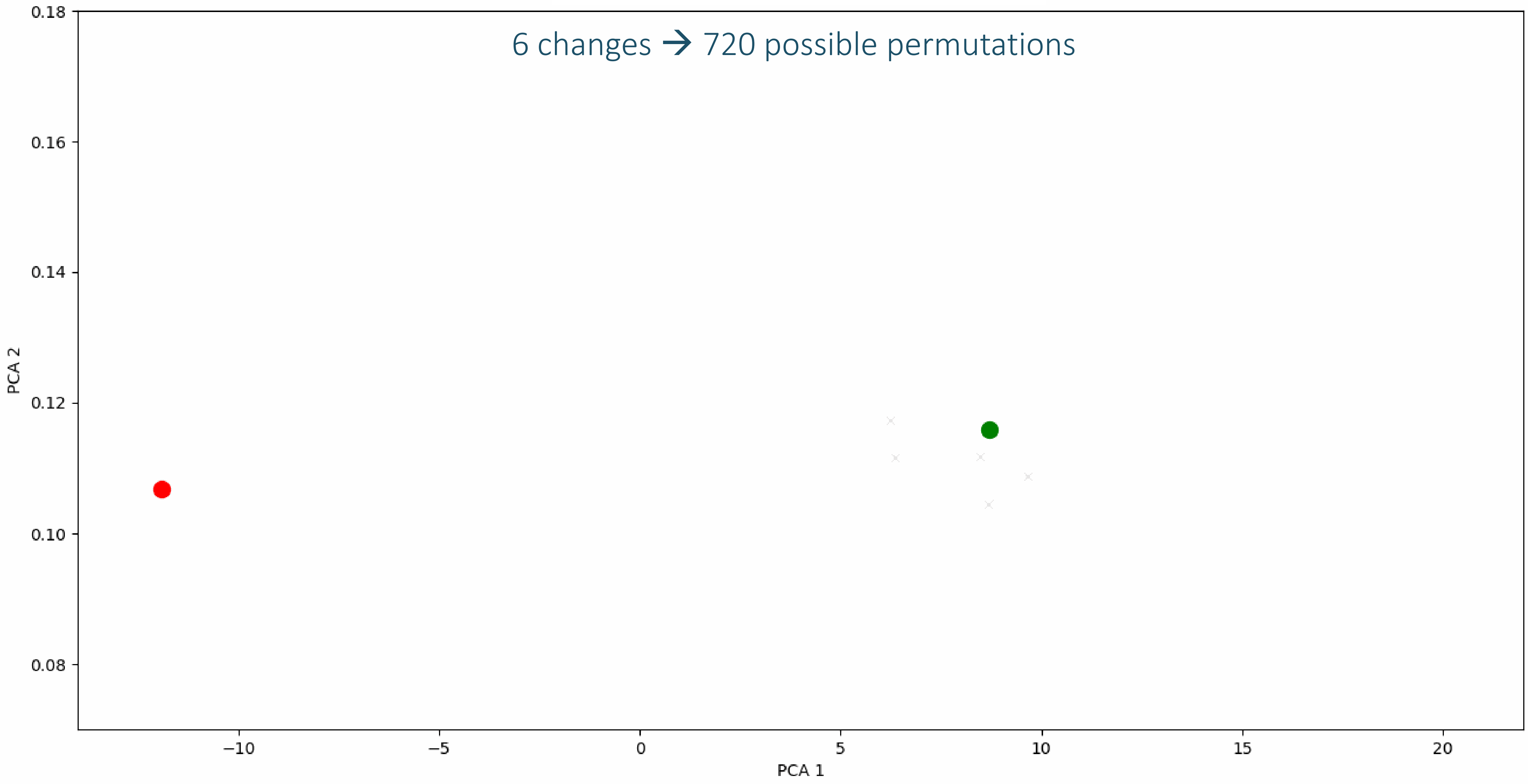
Explainability Example: Graph A → Graph B

- make methodical, incremental changes to go from one state to the other



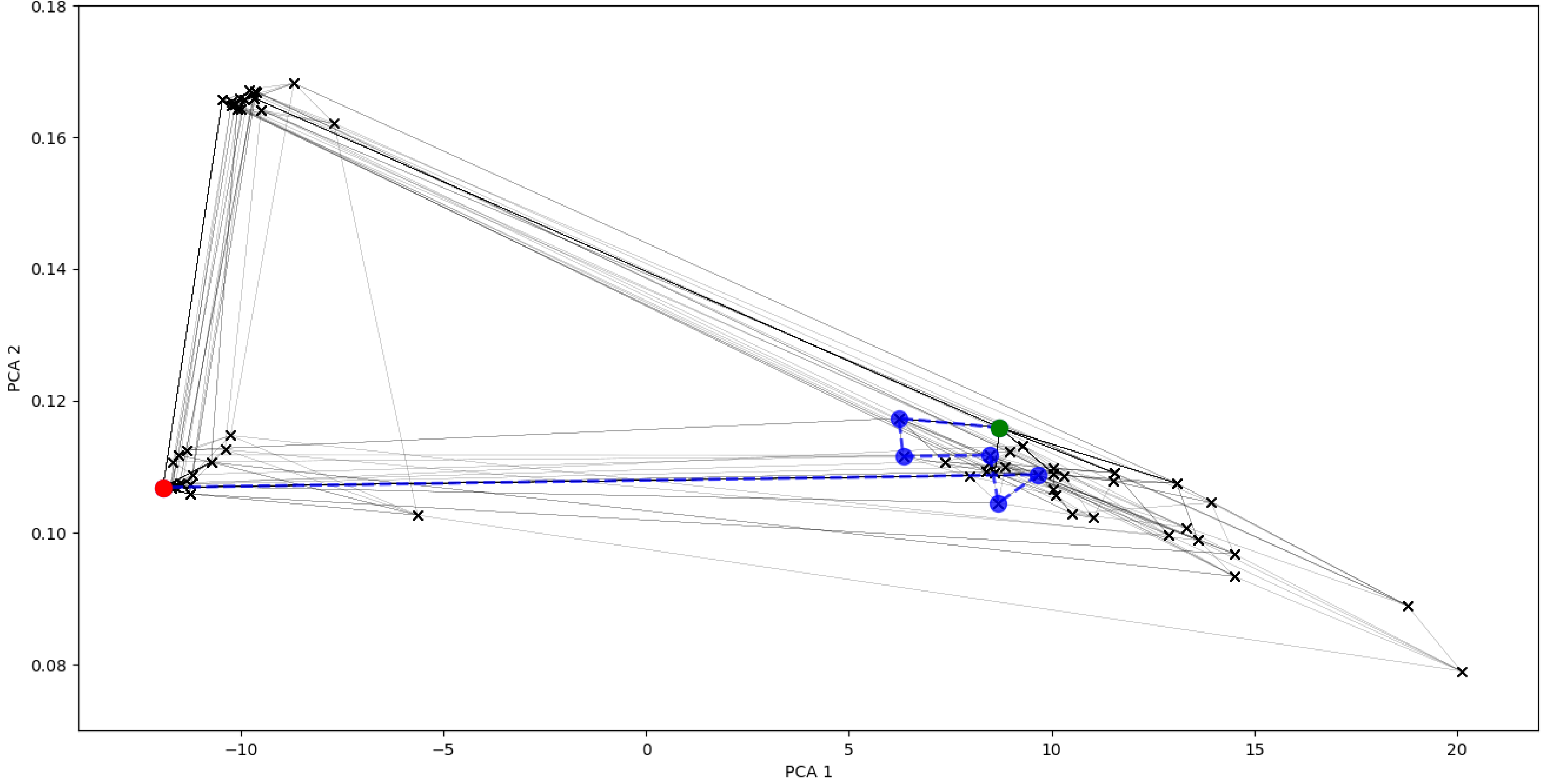
Explainability Example: Graph A → Graph B

- the *order* in which you make changes can impact how you interpret importance



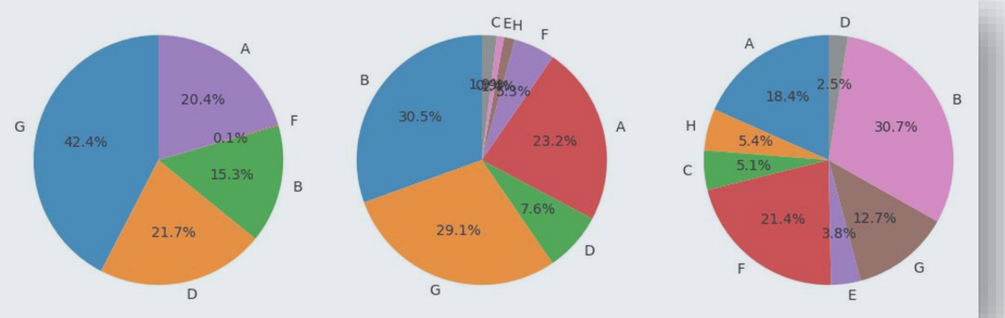
Explainability Example: Graph A → Graph B

- cannot assume the node that causes the largest displacement in latent space corresponds to the most important node



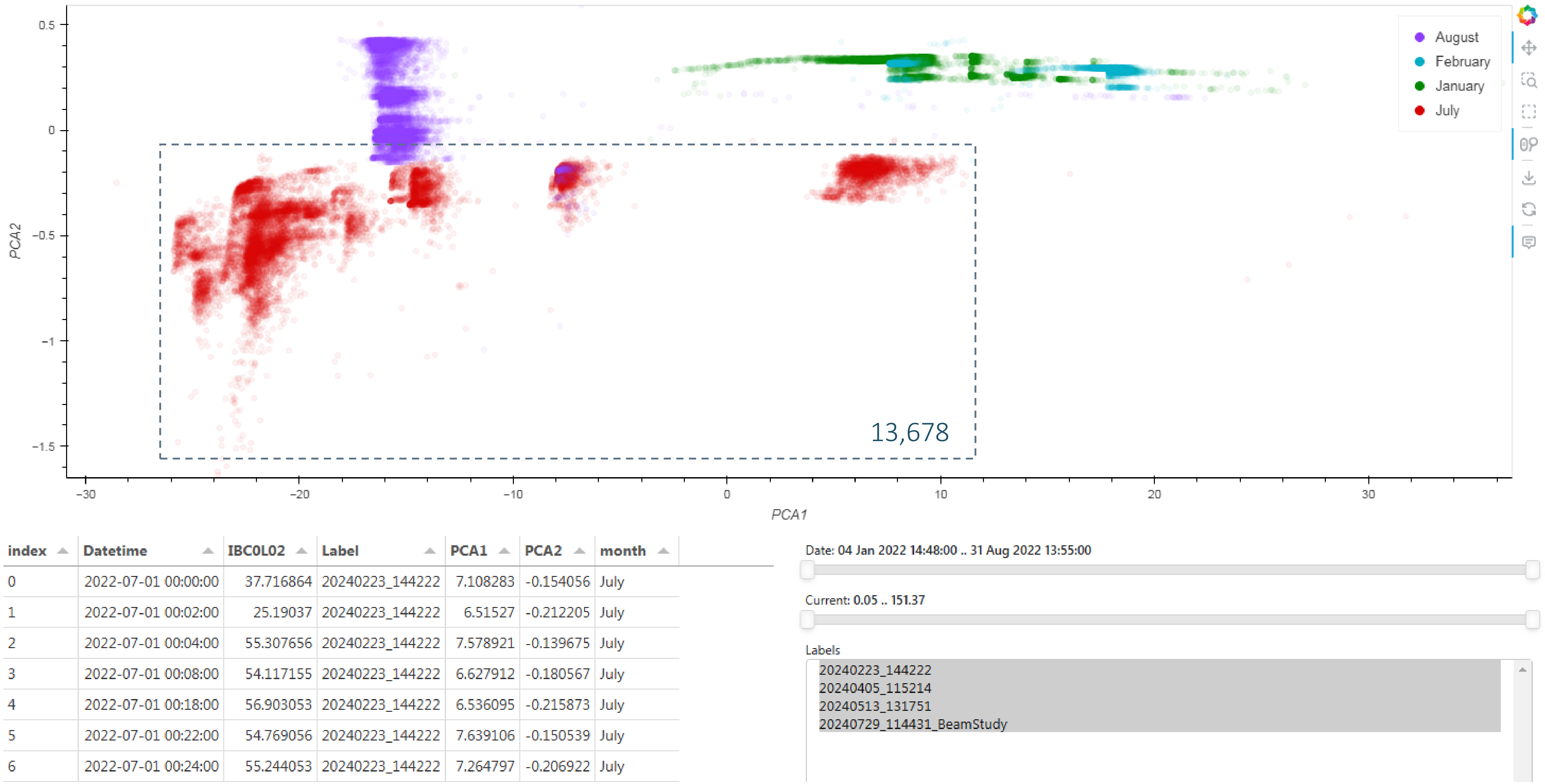
Explainability Framework

- we define the most important node as the one which makes Graph A look most like Graph B
- quantitatively, we define the most important node as the one which most lowers the MAE between the readings of Graph A and B
- we train a prediction model that given setting node features, predicts reading node features
- ideally we need to do this for *every permutation*
  - ✓ this can get computationally expensive
  - ✓ not uncommon to have several dozen setting nodes change between embeddings



- a good proxy is to evaluate effect of each setting node change *independently* and rank the nodes according to how much they lower the MAE

Spring 2022 Run: 51,781 Embeddings



13,678

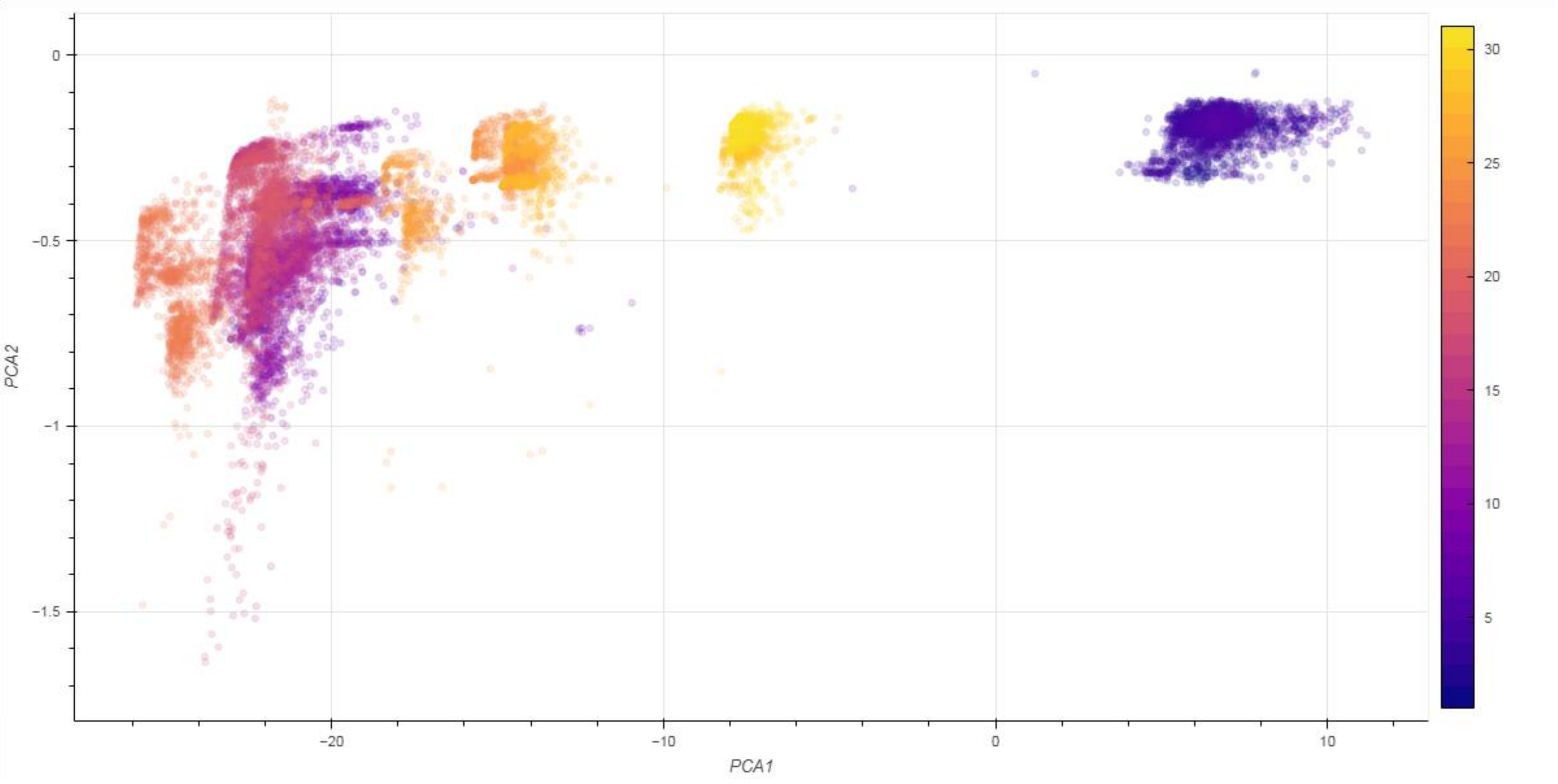
index	Datetime	IBC0L02	Label	PCA1	PCA2	month
0	2022-07-01 00:00:00	37.716864	20240223_144222	7.108283	-0.154056	July
1	2022-07-01 00:02:00	25.19037	20240223_144222	6.51527	-0.212205	July
2	2022-07-01 00:04:00	55.307656	20240223_144222	7.578921	-0.139675	July
3	2022-07-01 00:08:00	54.117155	20240223_144222	6.627912	-0.180567	July
4	2022-07-01 00:18:00	56.903053	20240223_144222	6.536095	-0.215873	July
5	2022-07-01 00:22:00	54.769056	20240223_144222	7.639106	-0.150539	July
6	2022-07-01 00:24:00	55.244053	20240223_144222	7.264797	-0.206922	July

Date: 04 Jan 2022 14:48:00 .. 31 Aug 2022 13:55:00

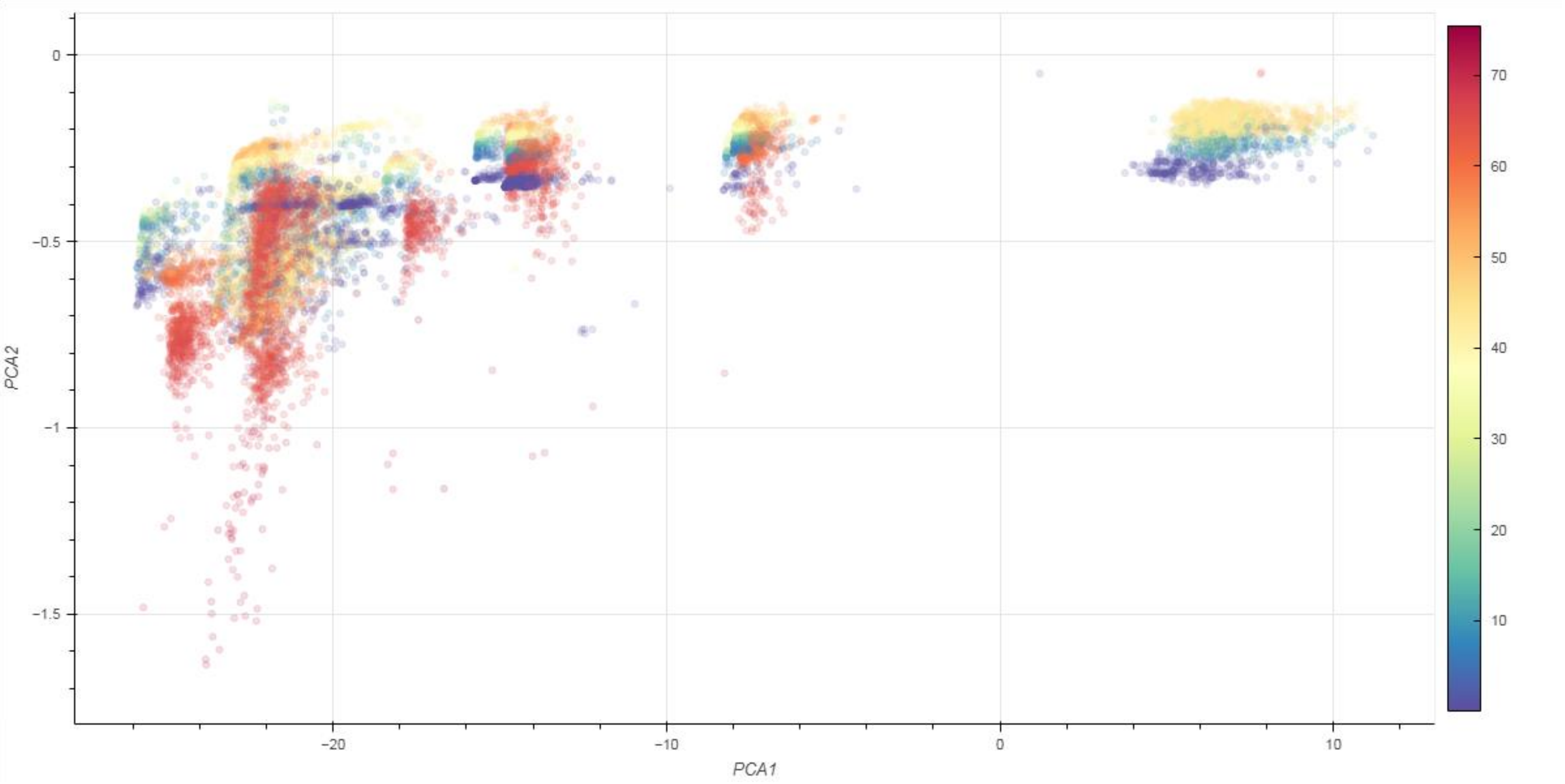
Current: 0.05 .. 151.37

Labels
20240223_144222
20240405_115214
20240513_131751
20240729_114431_BeamStudy

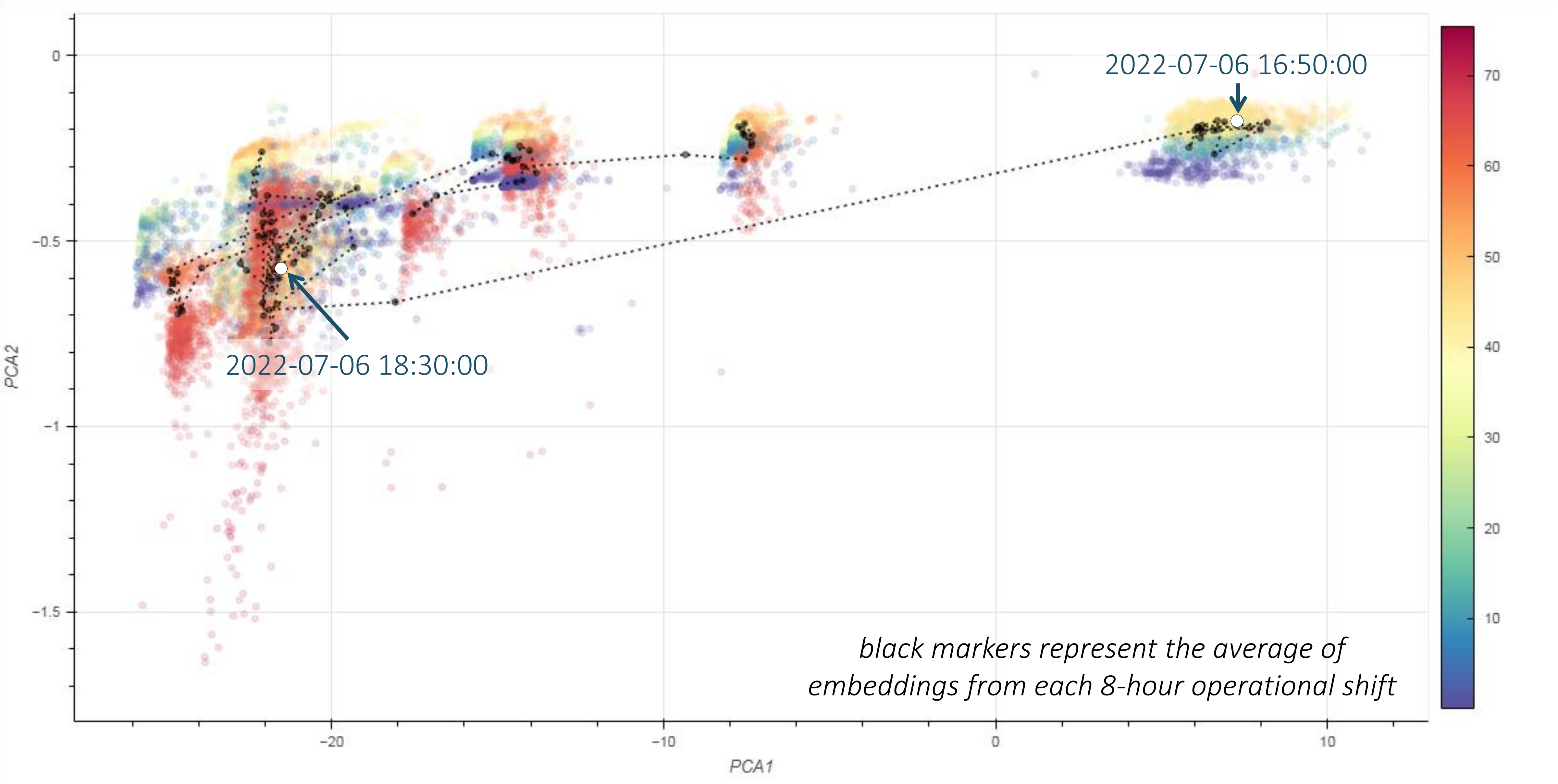
Embeddings: by Day



Embeddings: by Current



Embeddings: Current + Changes by Shift



2022-07-06 16:50:00

2022-07-06 18:30:00

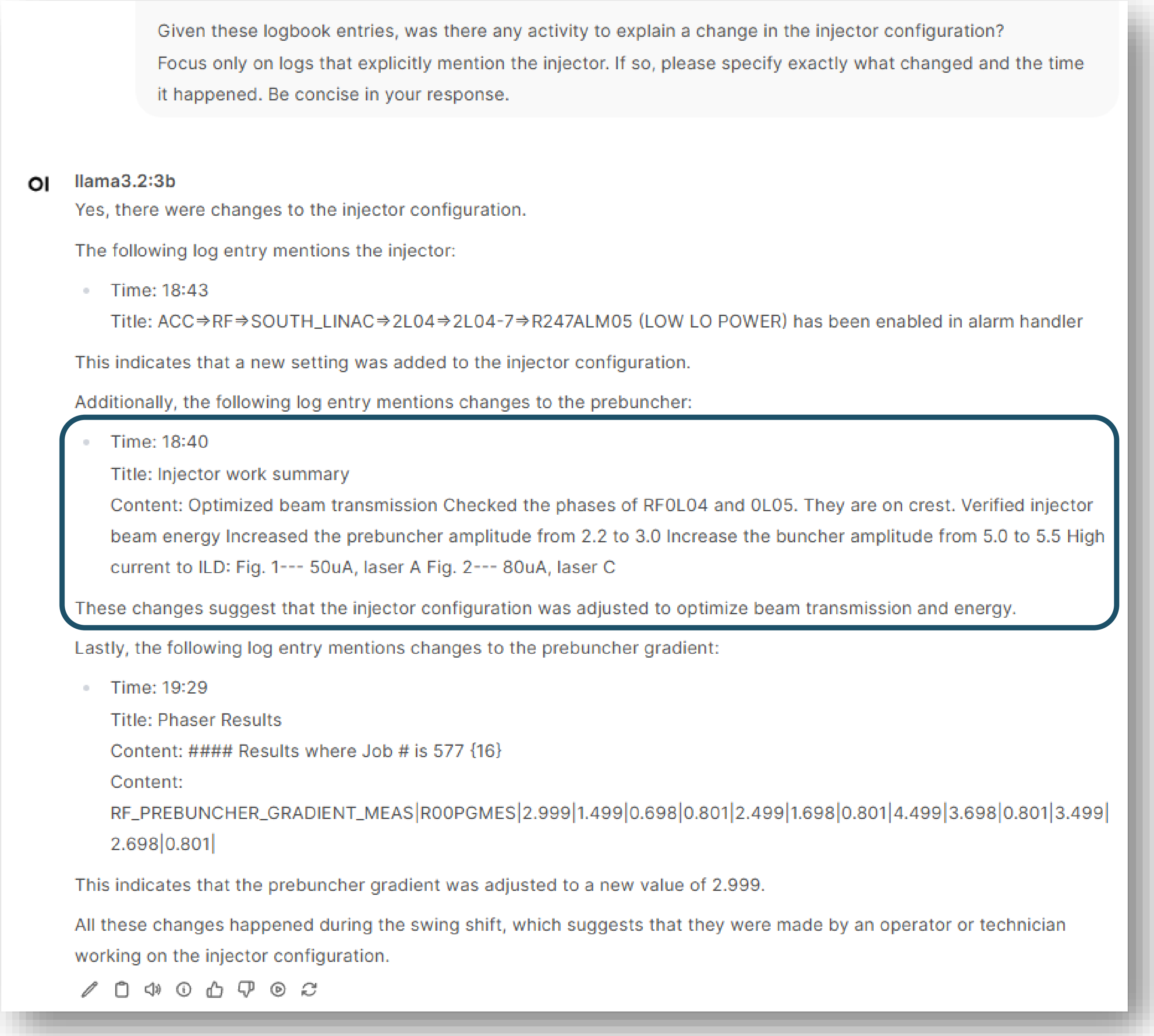
black markers represent the average of embeddings from each 8-hour operational shift

Embeddings: Explainability

- explainability results
    1. MDB0L06H
    2. MBH0I06V
    3. MDJ0L07V
    4. MHB0L02AH
    5. MAT0R03H

importance

- simple example of LLMs providing additional context



Given these logbook entries, was there any activity to explain a change in the injector configuration? Focus only on logs that explicitly mention the injector. If so, please specify exactly what changed and the time it happened. Be concise in your response.

llama3.2:3b

Yes, there were changes to the injector configuration.

The following log entry mentions the injector:

- Time: 18:43
  Title: ACC→RF→SOUTH_LINAC→2L04→2L04-7→R247ALM05 (LOW LO POWER) has been enabled in alarm handler

This indicates that a new setting was added to the injector configuration.

Additionally, the following log entry mentions changes to the prebuncher:

- Time: 18:40
  Title: Injector work summary
  Content: Optimized beam transmission Checked the phases of RF0L04 and 0L05. They are on crest. Verified injector beam energy Increased the prebuncher amplitude from 2.2 to 3.0 Increase the buncher amplitude from 5.0 to 5.5 High current to ILD: Fig. 1--- 50uA, laser A Fig. 2--- 80uA, laser C

These changes suggest that the injector configuration was adjusted to optimize beam transmission and energy.

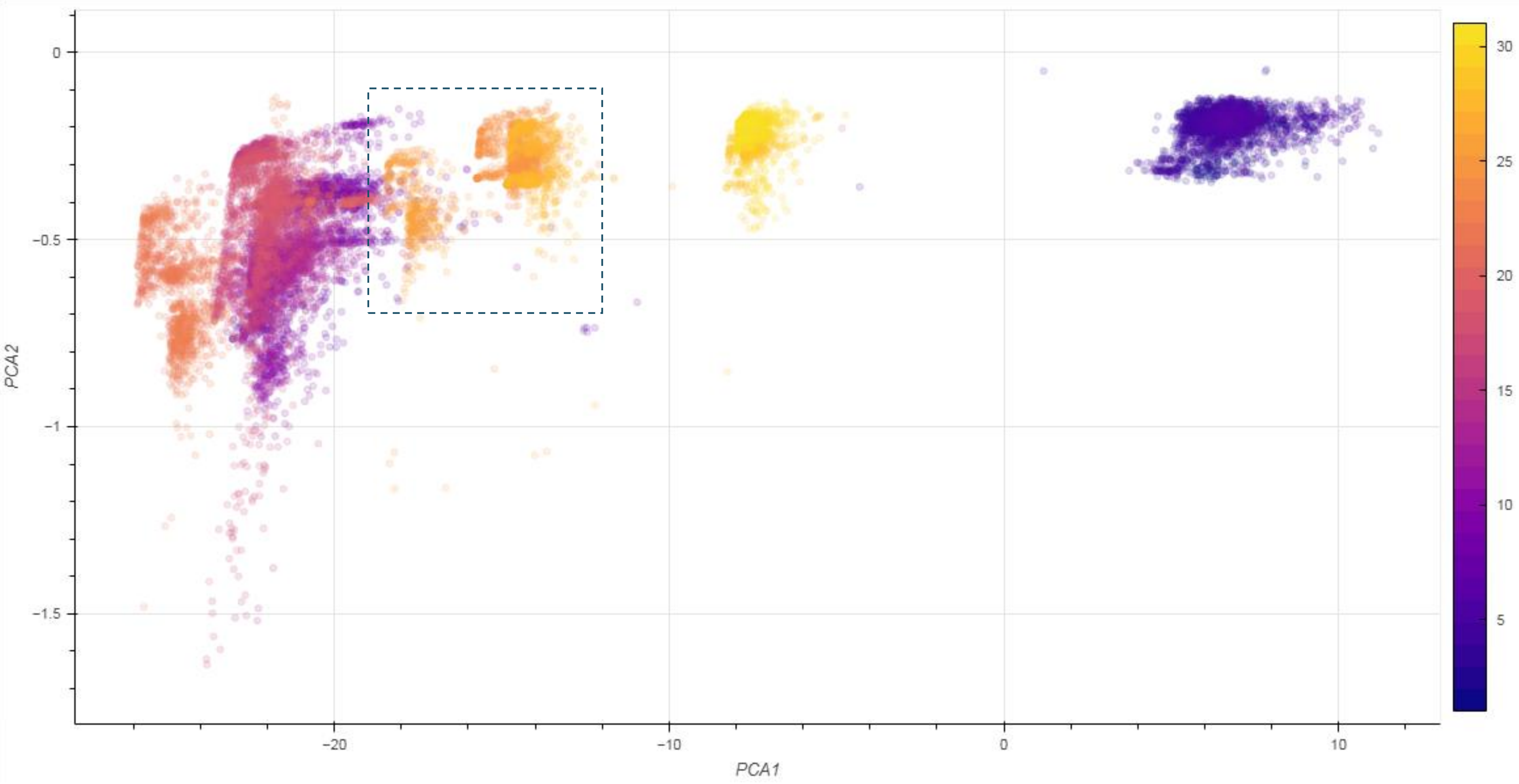
Lastly, the following log entry mentions changes to the prebuncher gradient:

- Time: 19:29
  Title: Phaser Results
  Content: #### Results where Job # is 577 {16}
  Content:
  RF_PREBUNCHER_GRADIENT_MEAS|R00PGMES|2.999|1.499|0.698|0.801|2.499|1.698|0.801|4.499|3.698|0.801|3.499|2.698|0.801|

This indicates that the prebuncher gradient was adjusted to a new value of 2.999.

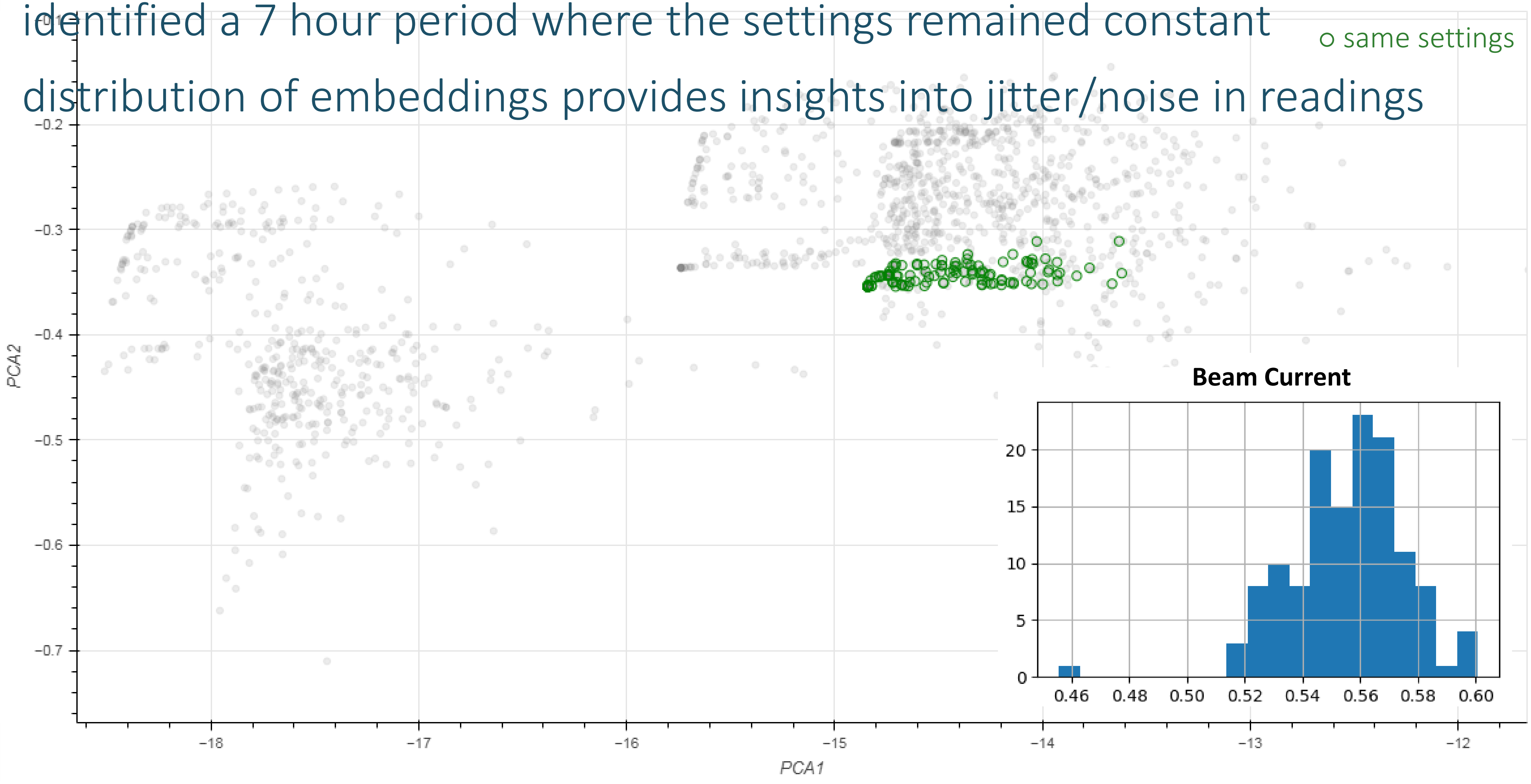
All these changes happened during the swing shift, which suggests that they were made by an operator or technician working on the injector configuration.

Embeddings: Data Exploration

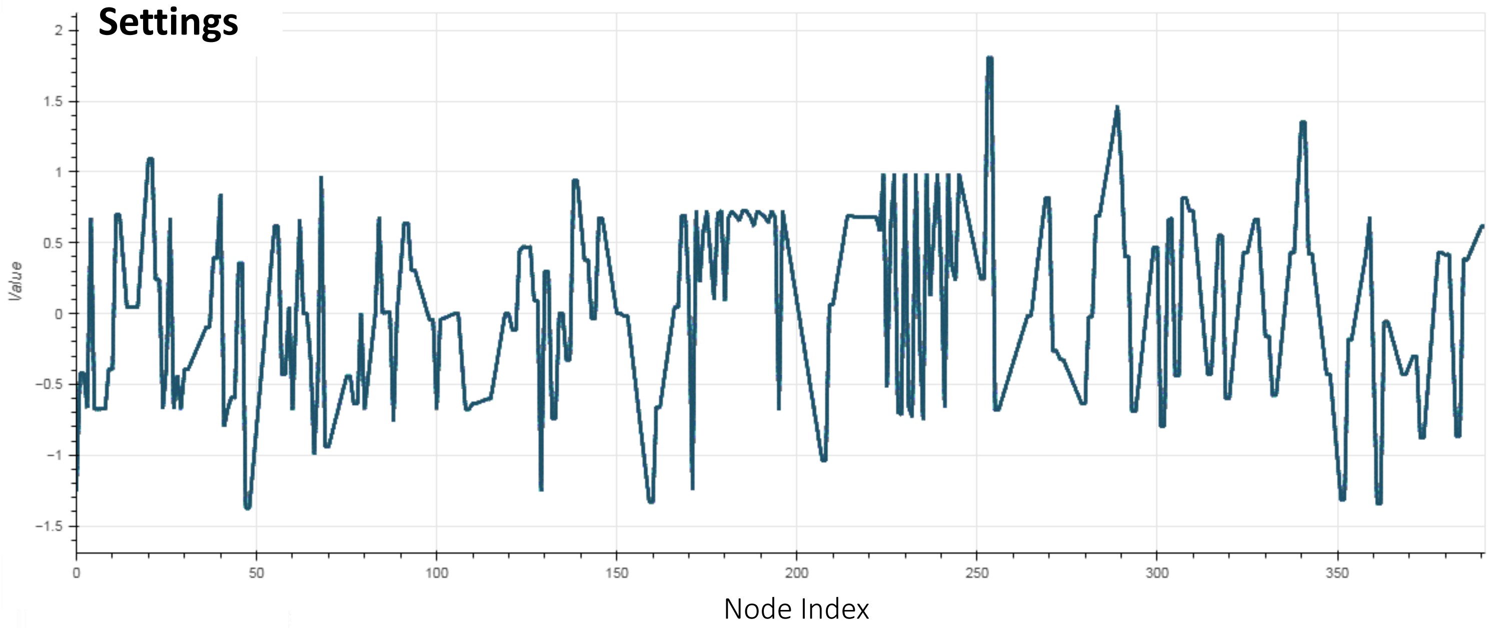


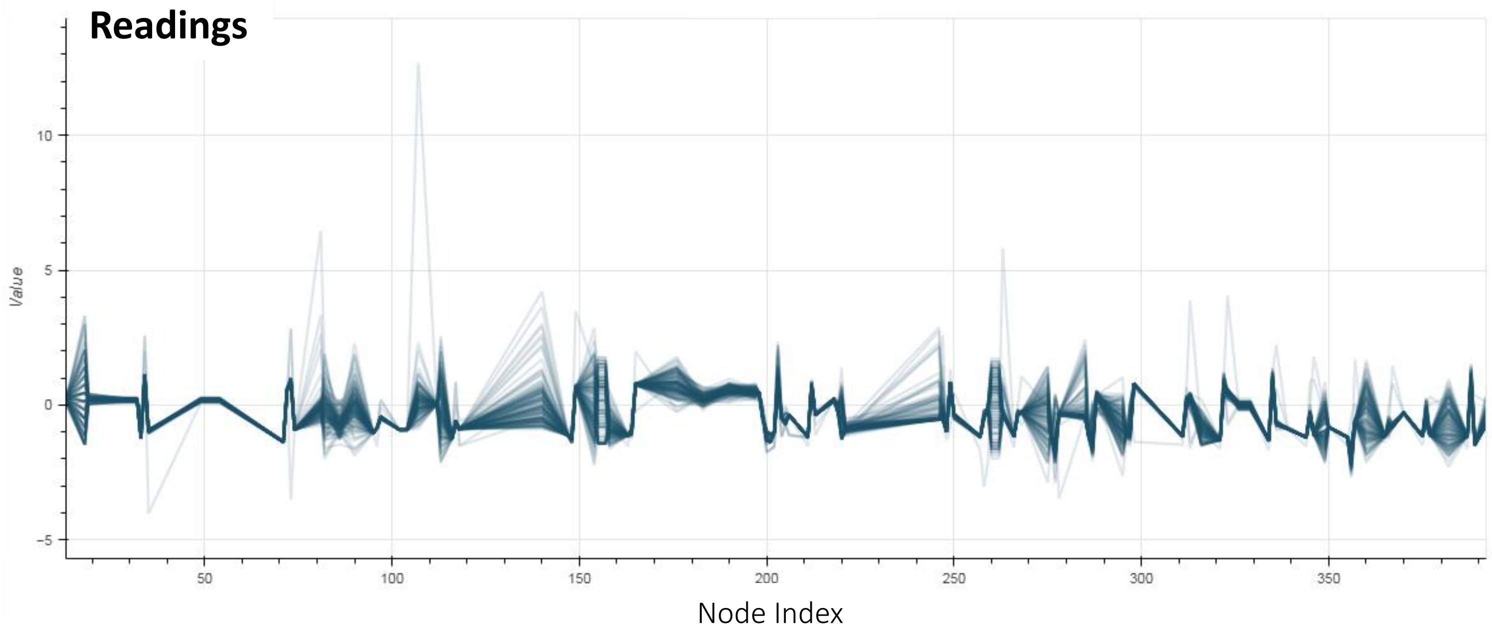
Embeddings: System Jitter/Noise

- identified a 7 hour period where the settings remained constant
- distribution of embeddings provides insights into jitter/noise in readings



Embeddings: System Jitter/Noise

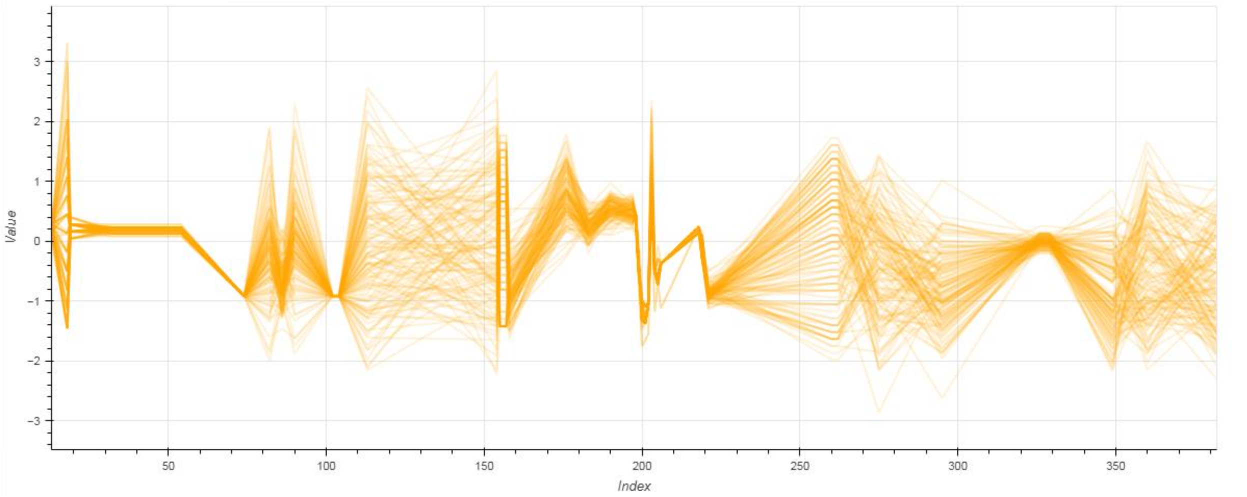




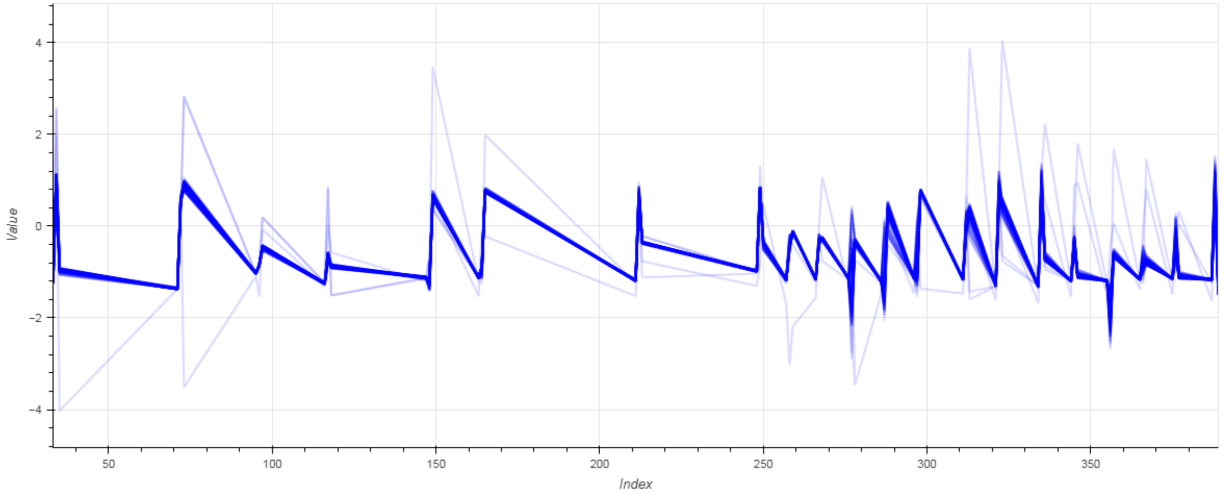
Embeddings: System Jitter/Noise

- inspect contributions from individual beamline components (node types)

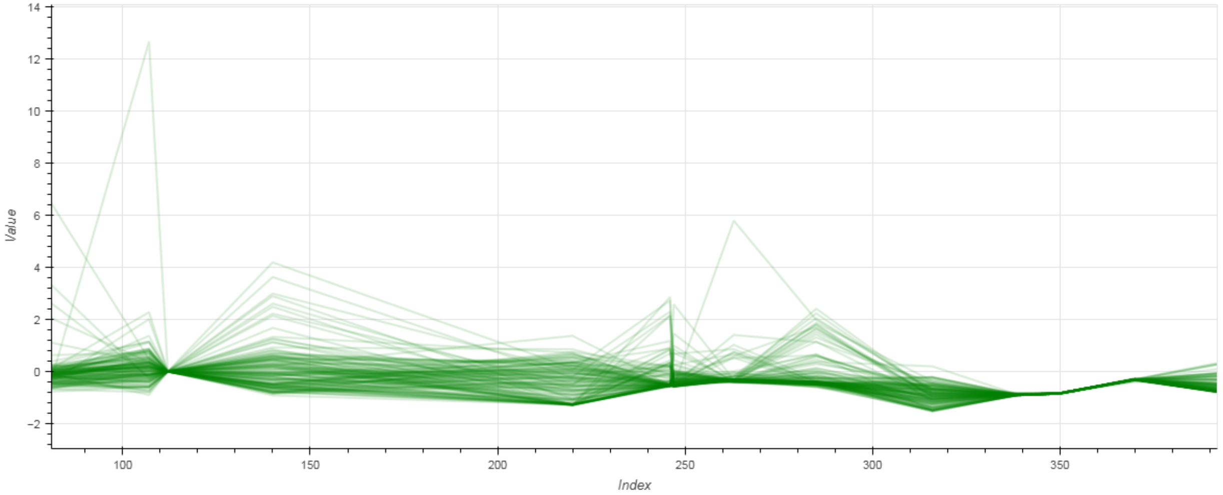
Vacuum



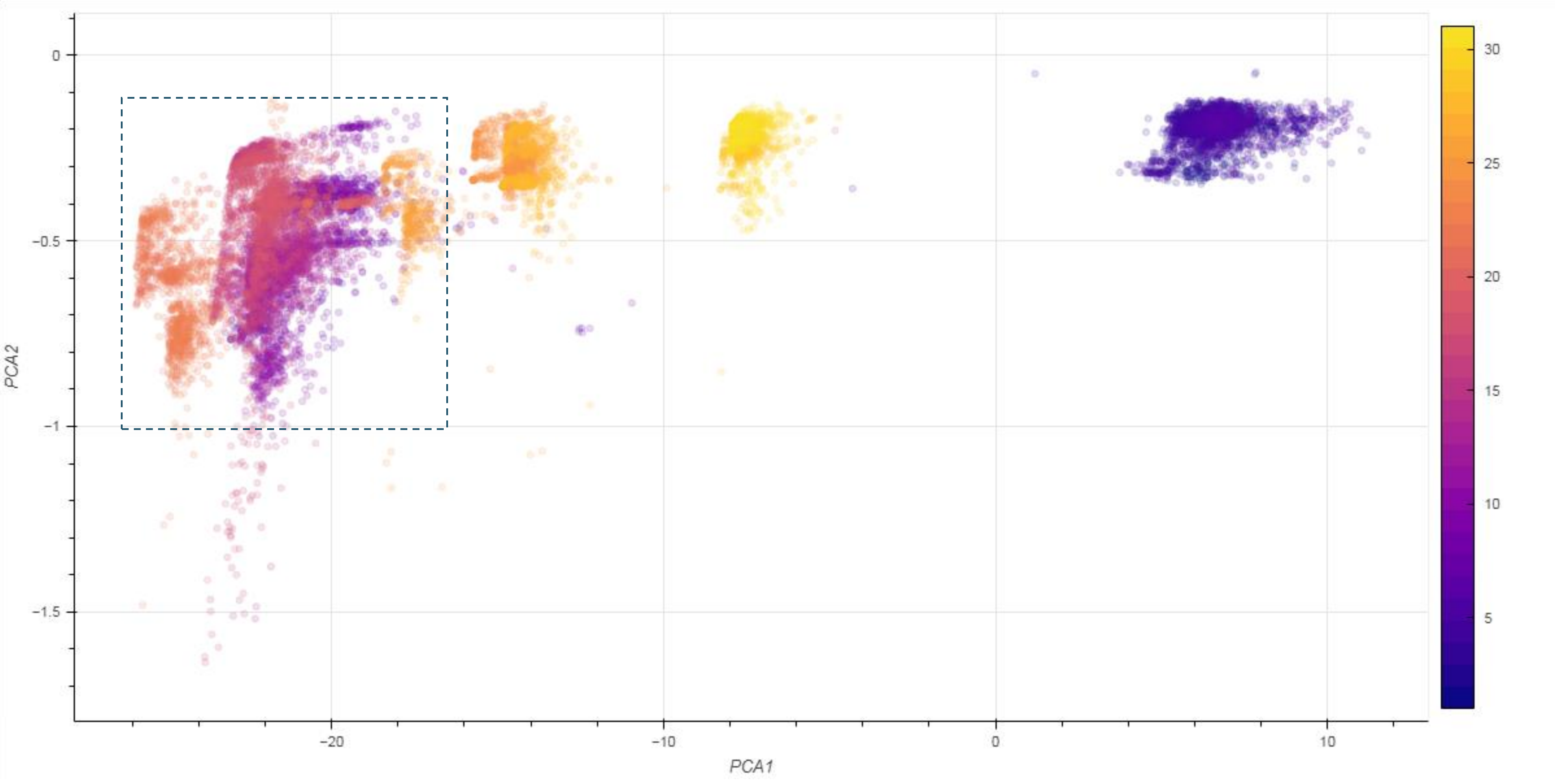
Beam Position Monitors



Beam Loss Monitors

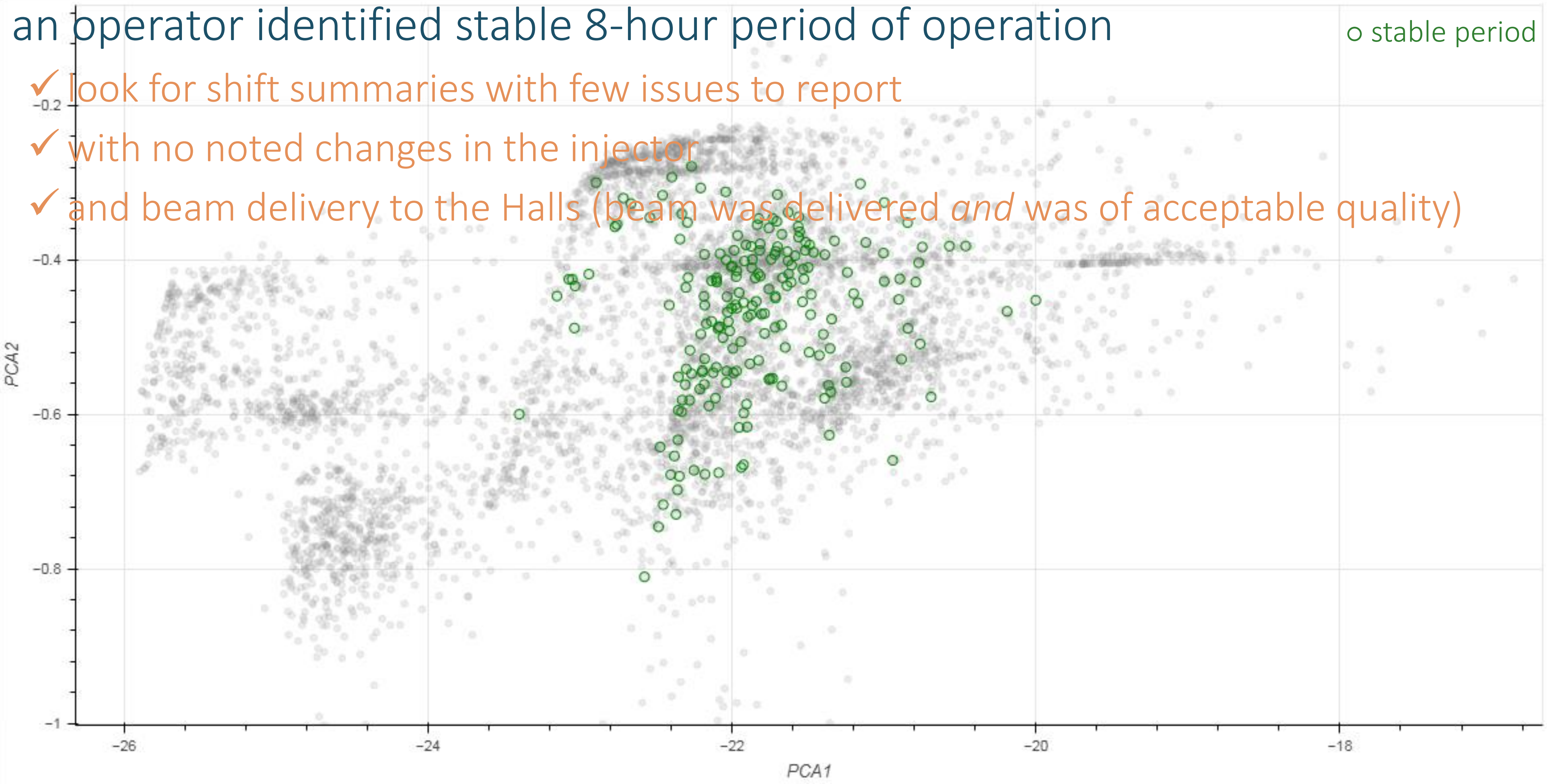


Embeddings: Data Exploration



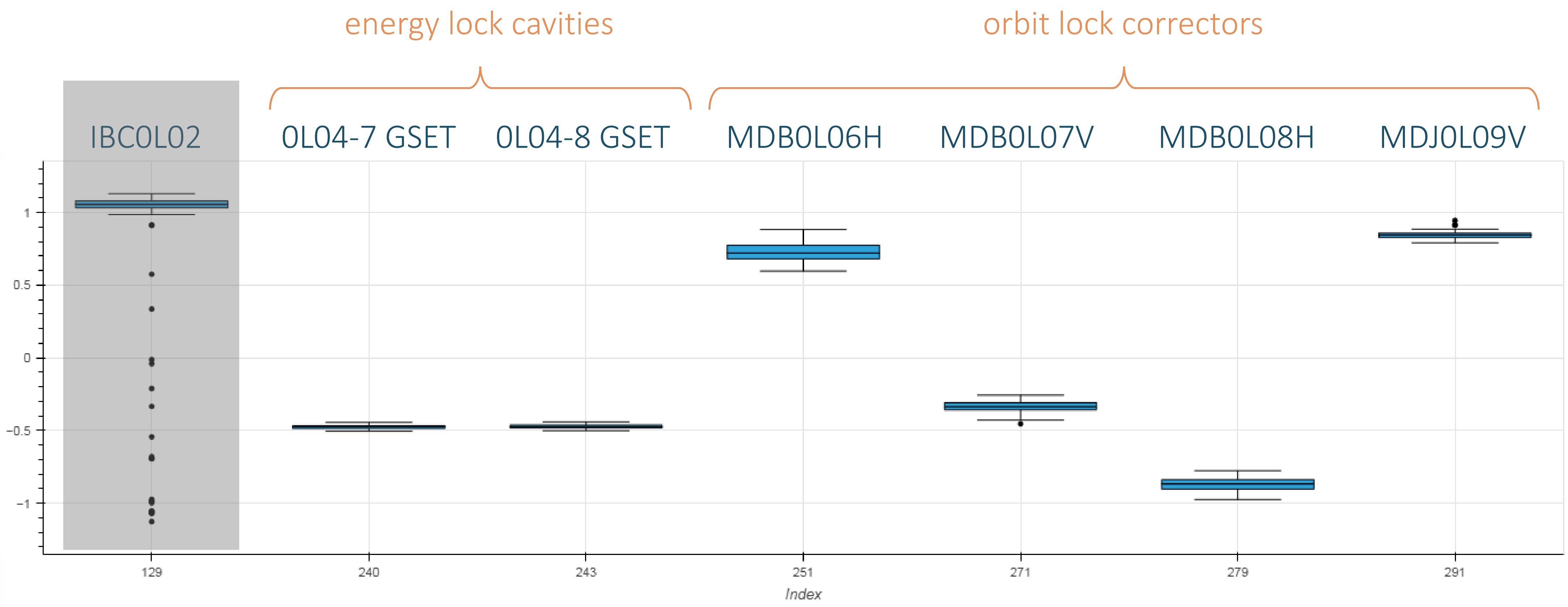
Embeddings: System Stability

- an operator identified stable 8-hour period of operation
  - ✓ look for shift summaries with few issues to report
  - ✓ with no noted changes in the injector
  - ✓ and beam delivery to the Halls (beam was delivered *and* was of acceptable quality)



Embeddings: System Stability

- beamline components showing the most variation during this time



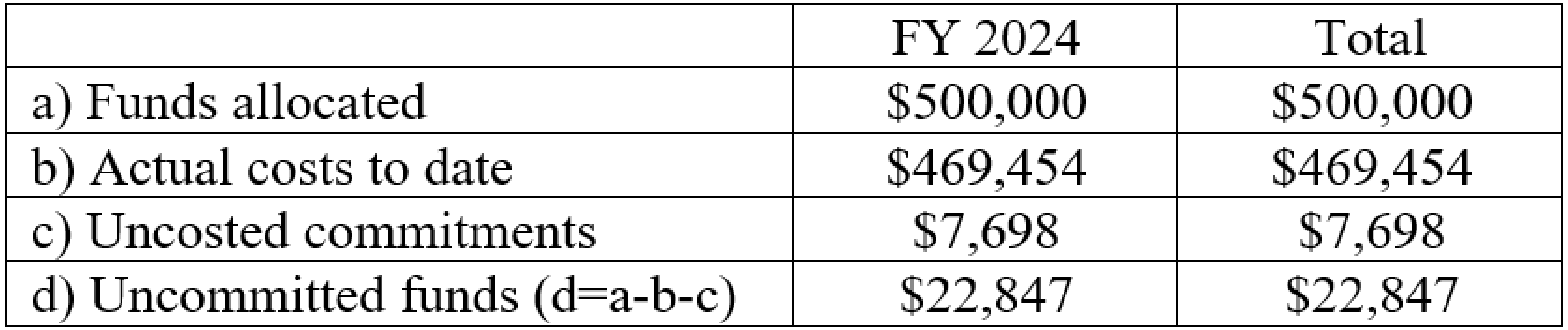
Summary

- all the analysis presented is *looking back* (post mortem)
- ideally this would be incorporated for *real-time* analysis
  - ✓ monitor stability
  - ✓ track and provide explanations for changes
  - ✓ provide visual feedback for beam tuning tasks
- the project has
  - ✓ demonstrated how deep learning over graph representations of a beamline can extract information-rich embeddings
  - ✓ exercised the archiver database in a new ways (i.e., collecting hundreds of PVs every 2 minutes over many months)
  - ✓ generated a peer-reviewed paper, with another currently under review
  - ✓ generated a US Patent

Project Summary: Major Deliverables and Schedule

Project	Deliverable	Date
Graph Learning for Efficient and Explainable Operation of Particle Accelerators	Analyzing attention weights for identifying important nodes	12/2025
	Project End	04/2026

Project Summary: Annual Budget



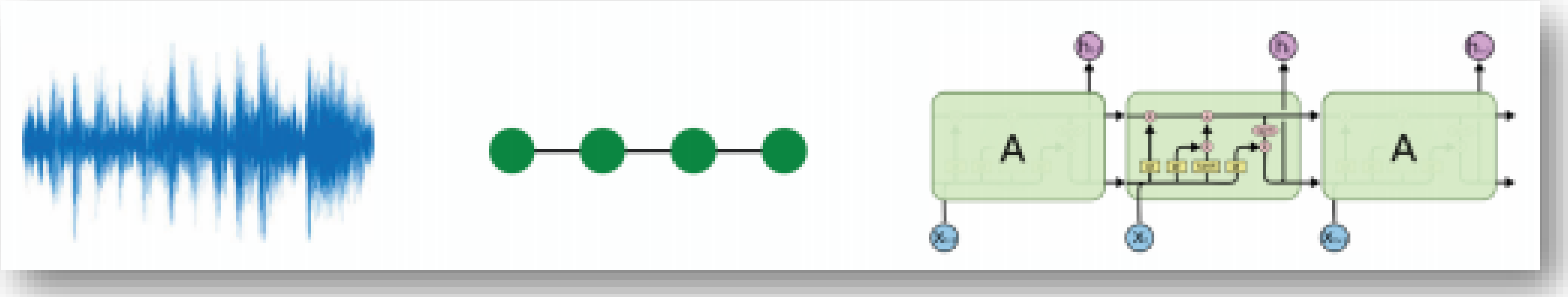
	FY 2024	Total
a) Funds allocated	$500,000	$500,000
b) Actual costs to date	$469,454	$469,454
c) Uncosted commitments	$7,698	$7,698
d) Uncommitted funds (d=a-b-c)	$22,847	$22,847

Thank you.

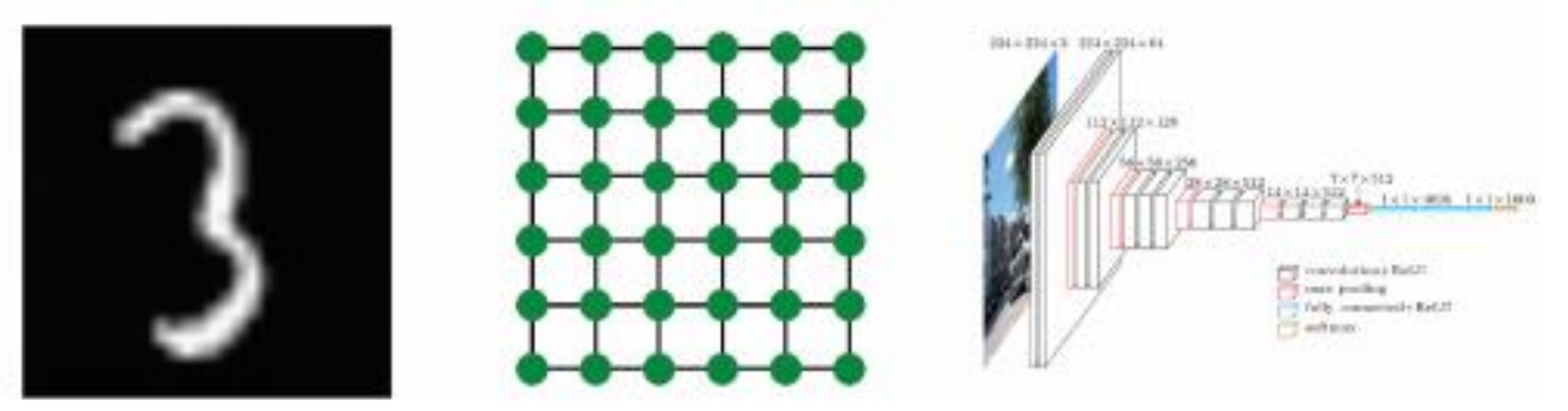
This work is supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under Contract No. DE-AC05-06OR23177.

Successes of Deep Learning

- recurrent neural networks (RNN) for sequences (e.g. text)



- convolutional neural networks (CNN) for fixed-size grids (e.g. images)

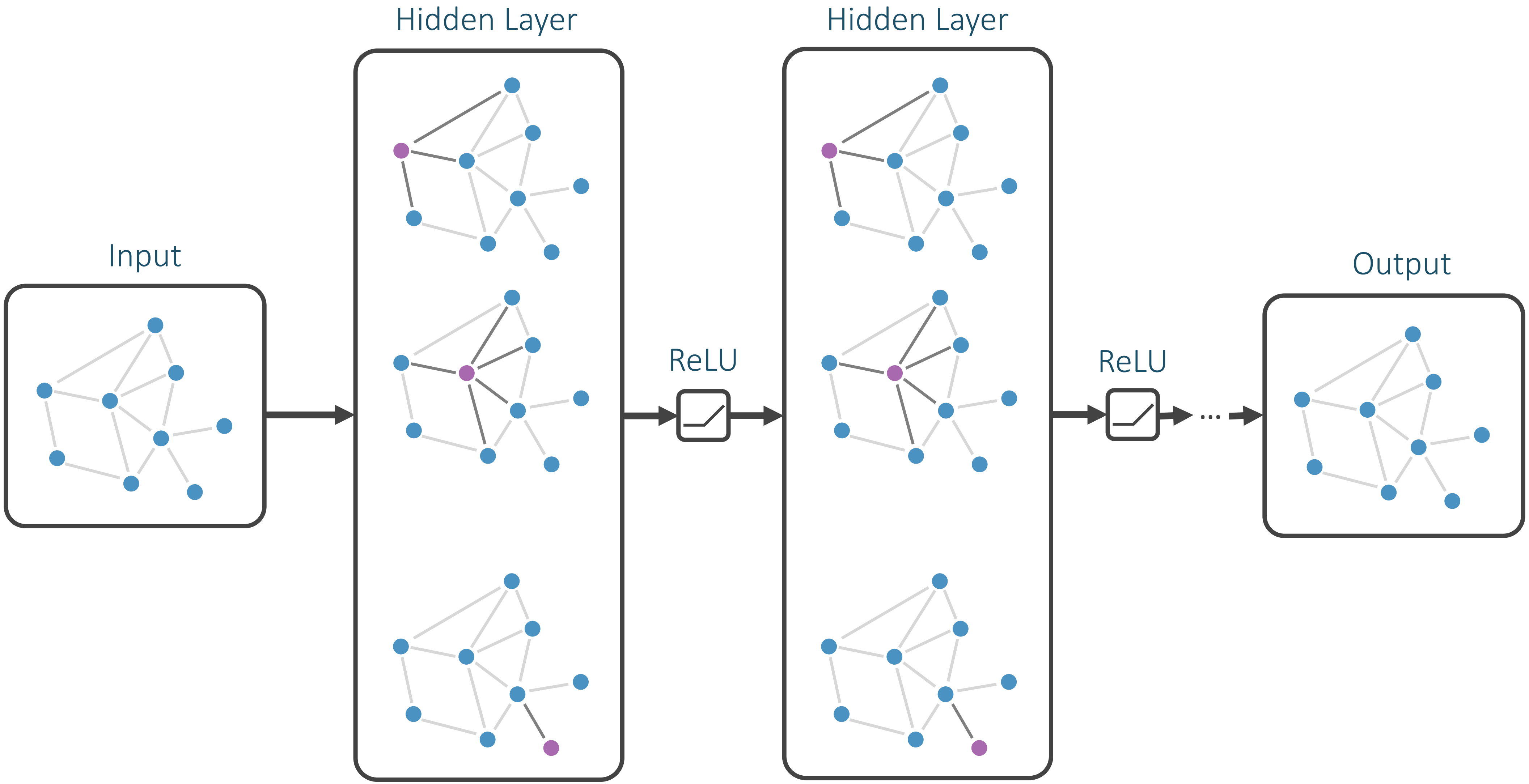


- modern deep learning toolbox is designed for simple sequences and grids

Graph Properties

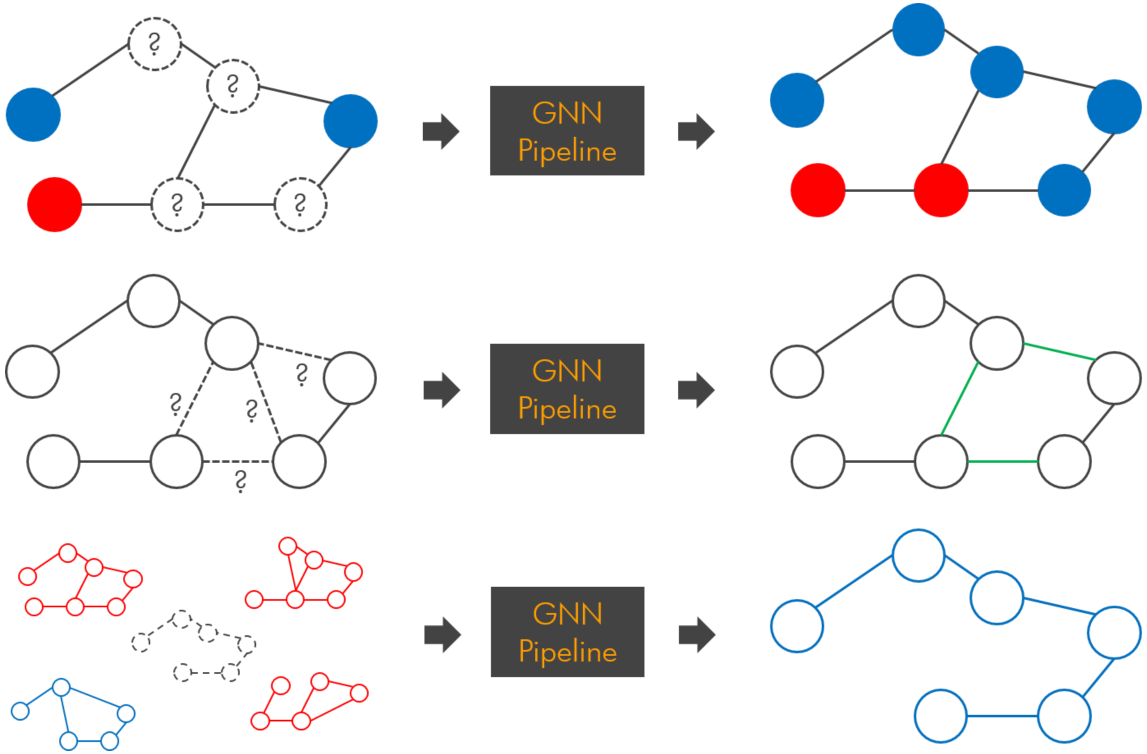
- heterogeneous, directed, weighted graph
  - ✓ heterogeneous → node types represent different elements
    - *elements have unique features (often of different dimensions)*
  - ✓ directed edges → a downstream element cannot influence an upstream element
  - ✓ weighted edges → edge weights are the inverse of the distance between elements
    - *use the inverse of the distance because elements in closer proximity (smaller distance) should have great influence/weight*
- a user-set “window” size (in this example 2) defines graph edges
  - ✓ e.g. an element is connected to the 2 immediate downstream elements
    - *depending on the downstream task and the size of the beamline being modeled, this value may need to change*
- generate a subgraph by filtering on a specified set of node types
  - ✓ e.g. use just the quads → homogenous graph

Deep Learning on Graphs



(courtesy T. Kipf)

GNN Downstream Tasks

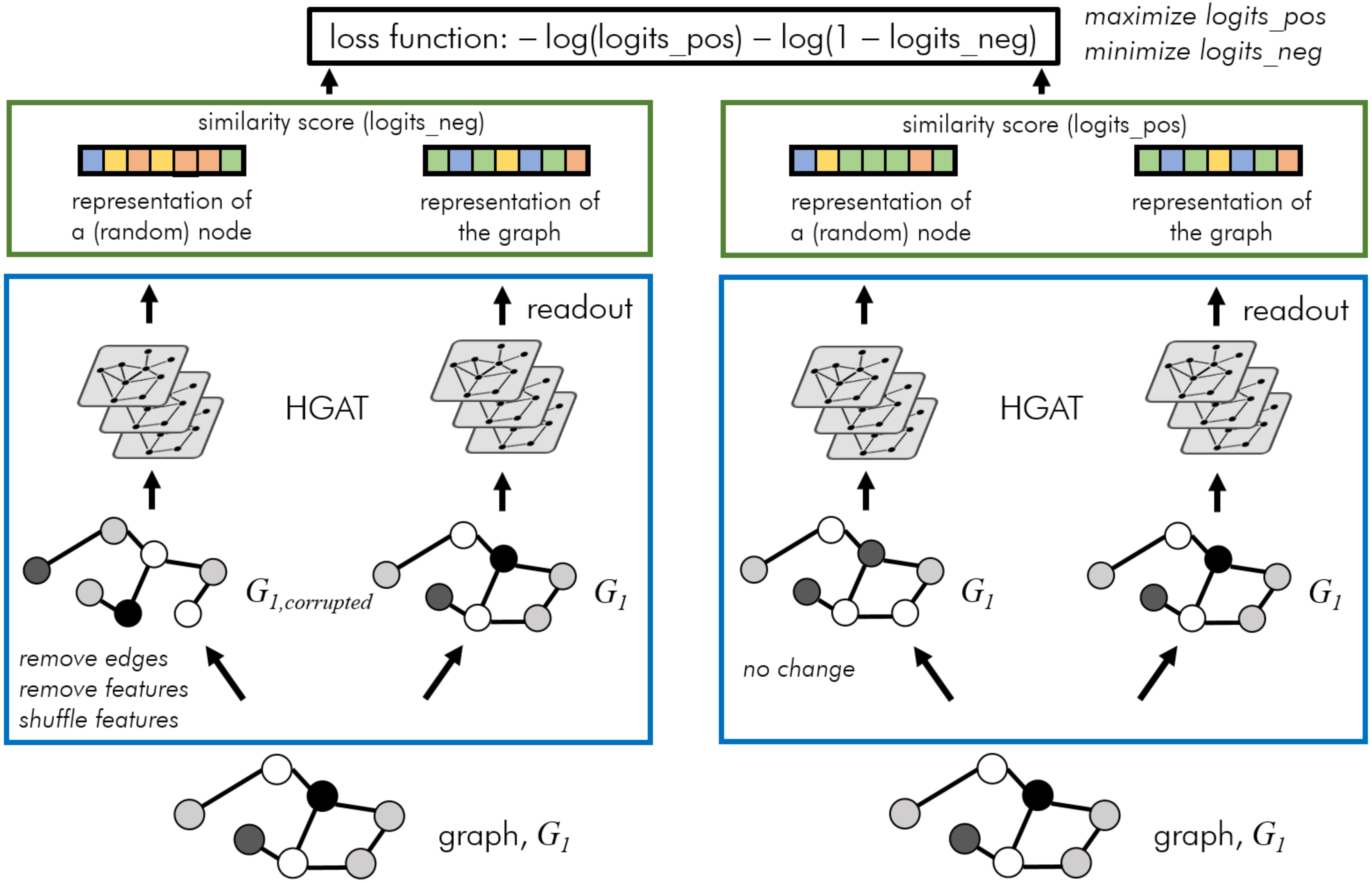


Node-level Prediction: A data-driven solution for adjusting quadrupole strengths. Setting the correct quadrupole strength is often done using conventional simulation tools for initial guidance, and then fine-tuning via (time-consuming) trial and error.

Edge-level Prediction: Edges were weighted more if the nodes they connect are in close proximity with one another. An alternate approach is to let the GNN model predict particular edge weights. Rather than impose a particular bias, this represents a data-driven approach to understanding the relative importance of elements to one another.

Graph-level Prediction: An entire graph is reduced to a single vector representation for visualizing and/or classifying a machine state.

Self-Supervised Learning

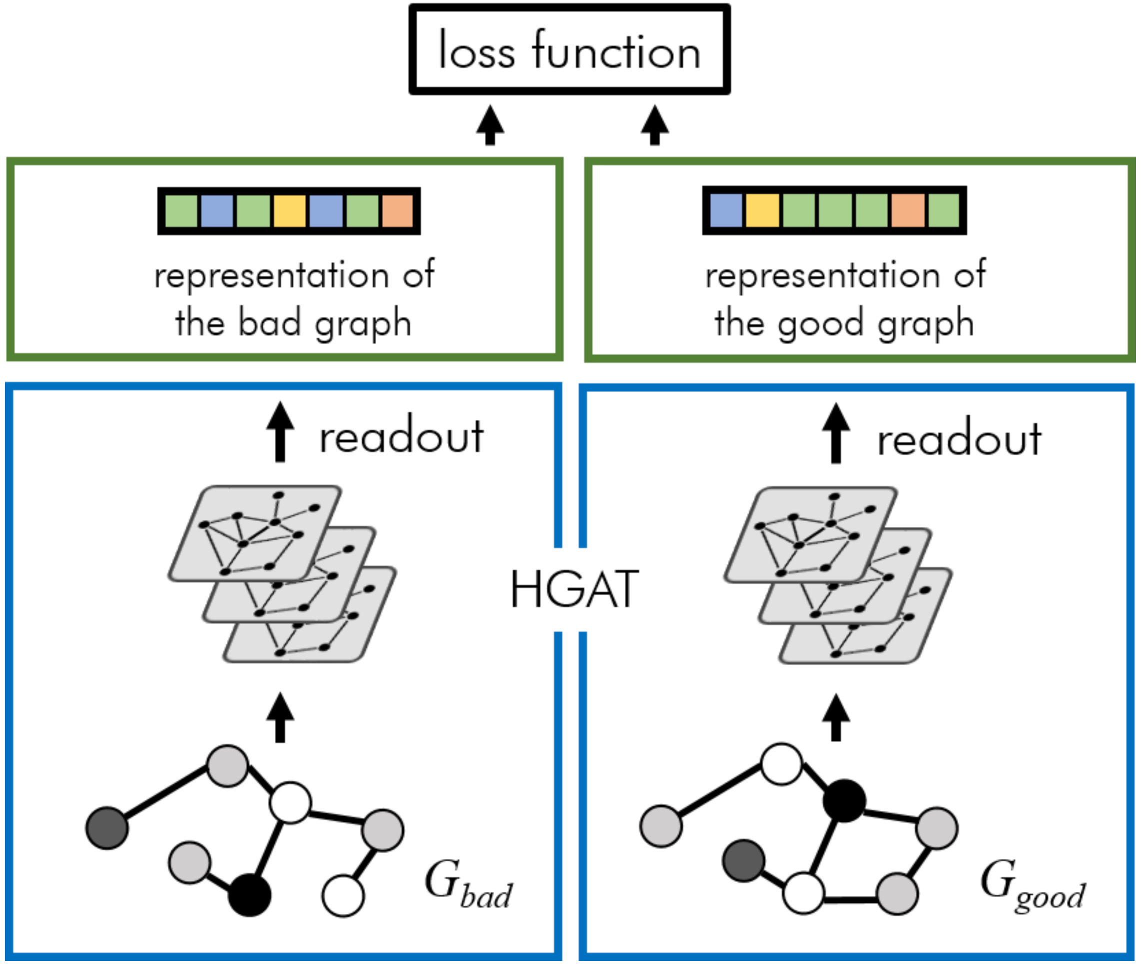


loss function: $-\log(\text{logits\_pos}) - \log(1 - \text{logits\_neg})$

maximize logits_pos
minimize logits_neg

similarity score (logits_neg)
representation of a (random) node
representation of the graph
readout
HGAT
$G_{1,corrupted}$
G_1
remove edges
remove features
shuffle features
graph, G_1
similarity score (logits_pos)
representation of a (random) node
representation of the graph
readout
HGAT
G_1
G_1
no change
graph, G_1

Supervised Learning


loss function
representation of the bad graph
representation of the good graph
readout
readout
HGAT
G_{bad}
G_{good}