ARTIFICIAL INTELLIGENCE, ROBOTICS, AND QUANTUM COMPUTING

William Phelps

Christopher Newport University/Jefferson Lab

CHRISTOPHER NEWPORT UNIVERSITY

Jefferson Lab
Thomas Jefferson National Accelerator Facility

clas

Introduction

- William Phelps (CNU/JLab)
  - Assistant Professor and Joint appointment with Hall B since Fall 2019
  - Department of Physics, Computer Science and Engineering
- Andru Quiroga (CNU)
  - Undergraduate Research Assistant
  - Major: Computer Engineering/Computer Science

William Phelps

Andru Quiroga

Research Interests: Artificial Intelligence, Quantum Information Science, Hadron Spectroscopy, and Novel Detector R&D

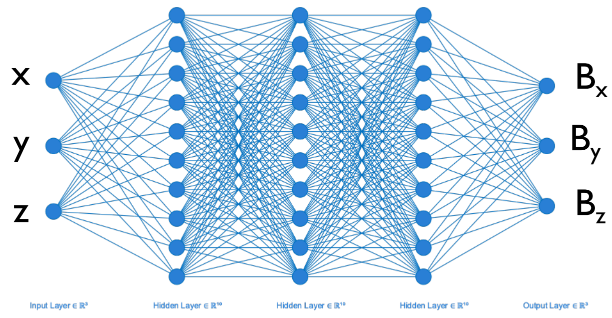
Quarterly AI Challenge

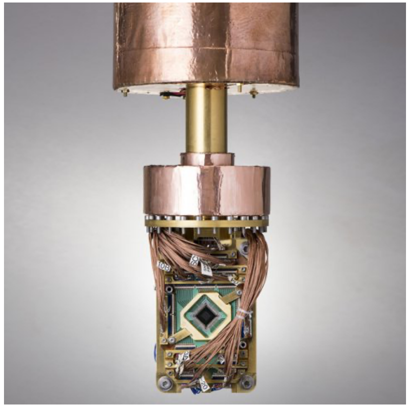
- The AI Lunch Group holds quarterly challenges
- Problem involved predicting the state vector of particles in the CDC in GlueX which involves training a network to map the magnetic field and energy loss
  - This led us to our first project!
- The AI Lunch Group is inspiring AI research at JLab and is helping to educate our future and current scientists!
  - If you are interested in getting involved with the AI community at JLab I will have slides at the end.

Andru's Award for 1st place

Roadmap

- Artificial Intelligence
  - Magnetic field map
  - Partial Wave Analysis
- Semi-autonomous robotic platform for accelerator diagnostics
- Quantum computing for reconstruction algorithms





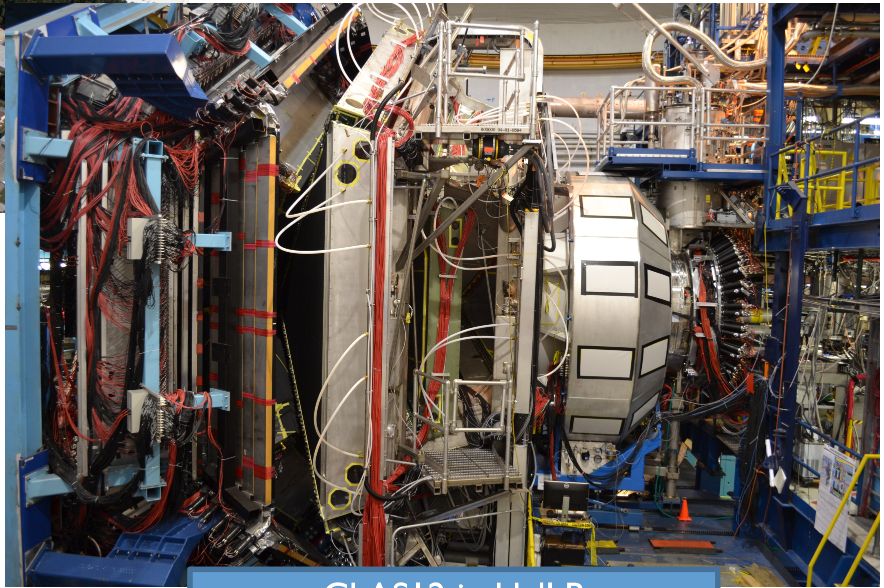
t time

CLAS12 Detector at Jefferson Lab

Jefferson Lab in 2014

Jefferson Lab
Thomas Jefferson National Accelerator Facility

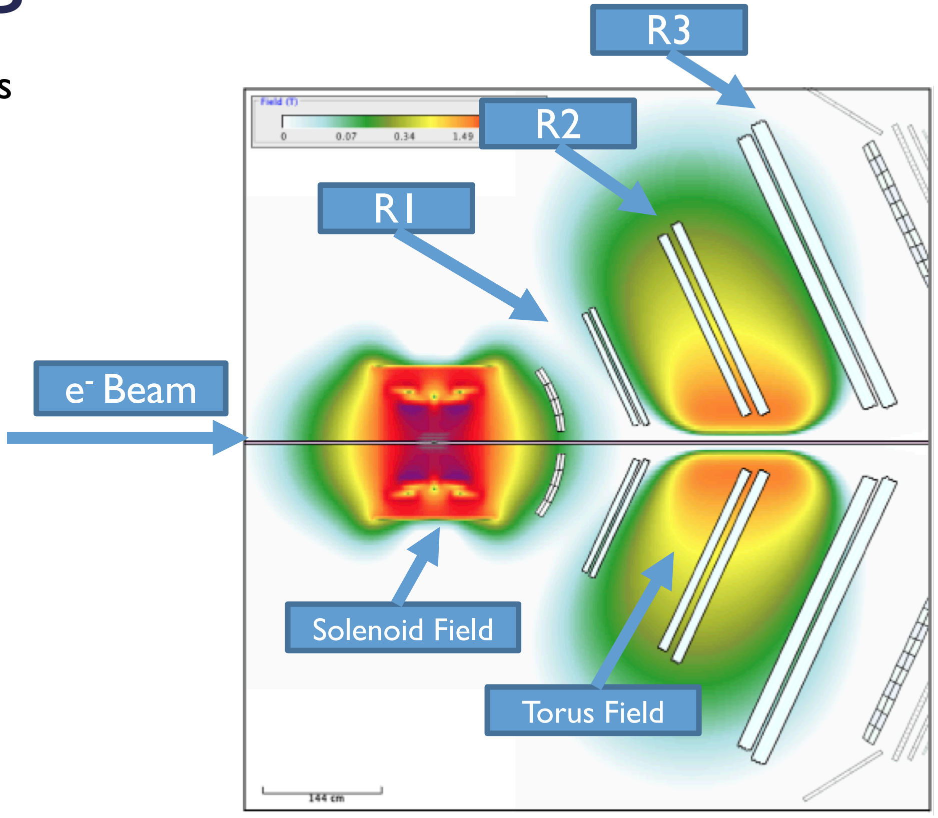
clas



CLAS12 in Hall B

Introduction to Magfield-AI

- After the AI Lunch quarterly problem it was clear that this would be a great real-world problem to tackle
- The production magnetic field is a non-negligible component of swimming/tracking and had a sizable memory footprint (up to 1GB before optimization)
- Can a neural network model be faster than the conventional model or provide other benefits where the tradeoff could be worth it?
- One problem - the current implementation of the magnetic field model is <u>*very*</u> fast (David Heddle – CNU)

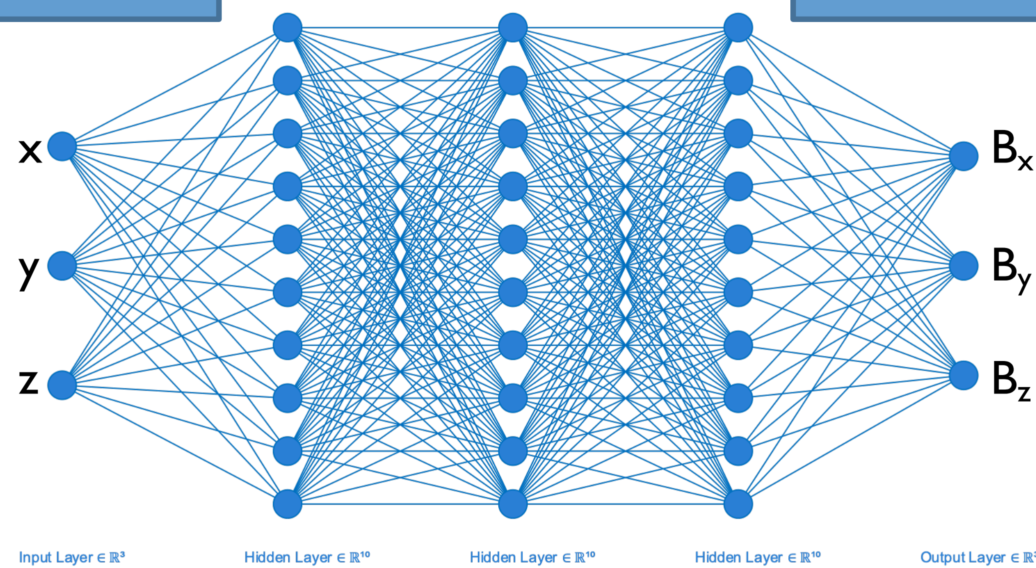


Approximating a Function with NN

- Our network architecture consists of 3 input and 3 outputs for the position and field vector respectively
- The number of layers and neurons per layer varied
- Different activation functions were also tested: sigmoid, tanh, **relu**.

Input: Position

Output: B-Field



Simplified Dense Layer Model

Tools of the Trade

TensorFlow

Keras

- Python 3.7 – Anaconda
  - Keras/TensorFlow - NN Libraries
  - Pandas/Numpy - Data Handling
  - Matplotlib - Visualization
- Three *excellent* machines that Scientific Computing provided which are accessible to jlab users!
  - 4 Titan RTX cards per node!

Sample Training Script

```
test = pd.read_csv("TRAIN/TRAIN.csv")
labels = pd.read_csv("TRAIN/TRAIN_labels.csv")
activation = 'relu'

model = Sequential()
model.add(Dense(units=1000, activation=activation, input_shape=(3600, )))
model.add(Dense(units=1000, activation=activation))
model.add(Dense(units=1000, activation=activation))
model.add(Dense(units=2))
model.compile(optimizer=adam(lr=.001), loss='mean_squared_error', metrics=['accuracy'])

model.fit(test, labels[labels.columns[1:]], epochs=300, batch_size=256, validation_split=0.2)
```

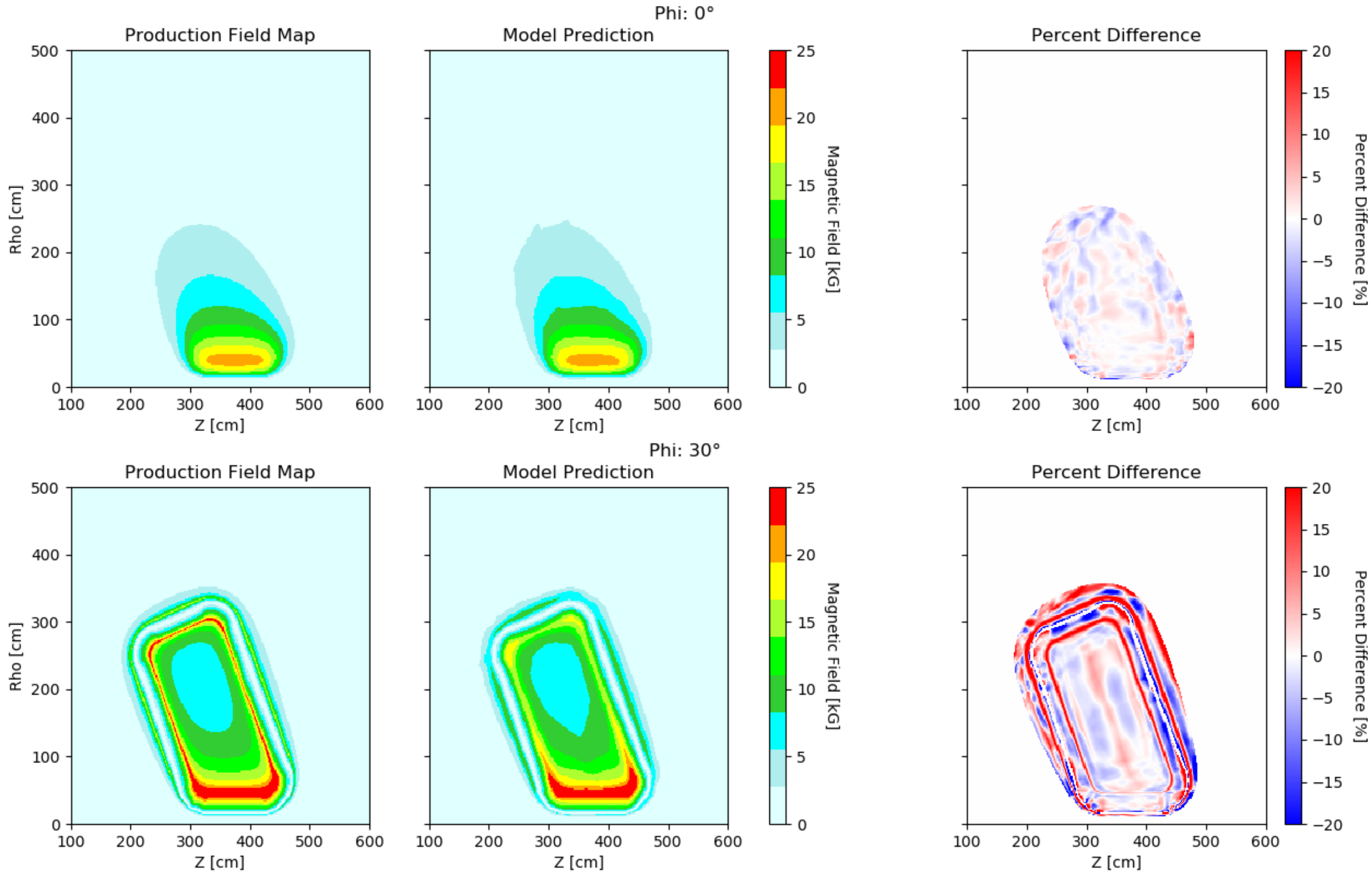
python

Magnetic Field Visualization

Architecture:
10 layers
100 Neurons/Layer

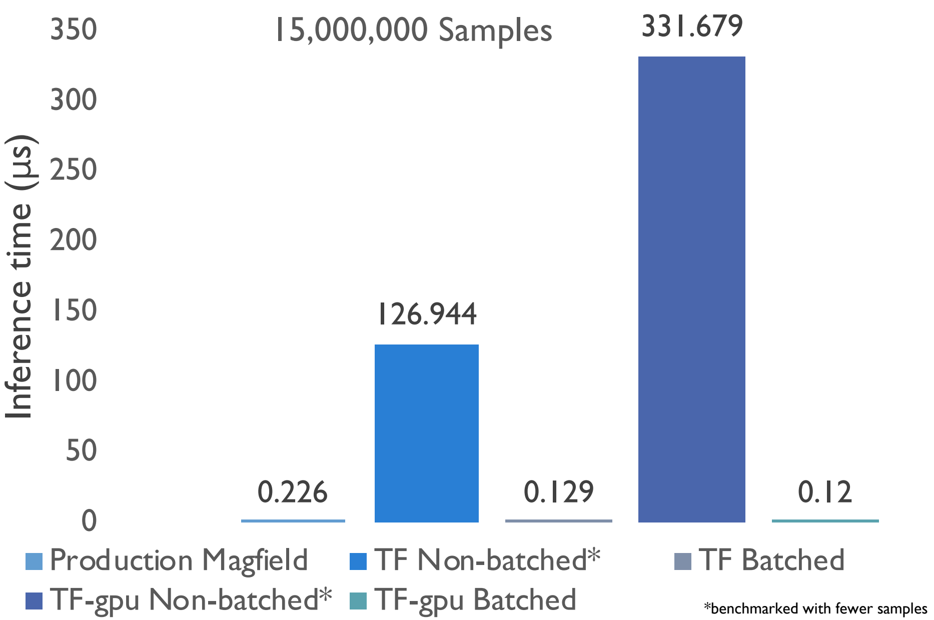
Sector Center

Coil Center



Performance Benchmarks

- Initial benchmarks on the CPU/GPU show that the inference time for a single position is extremely slow
- Batching refers to using TensorFlow to predict many values at one time which is not optimal for swimming/tracking.



Prediction without a NN Library

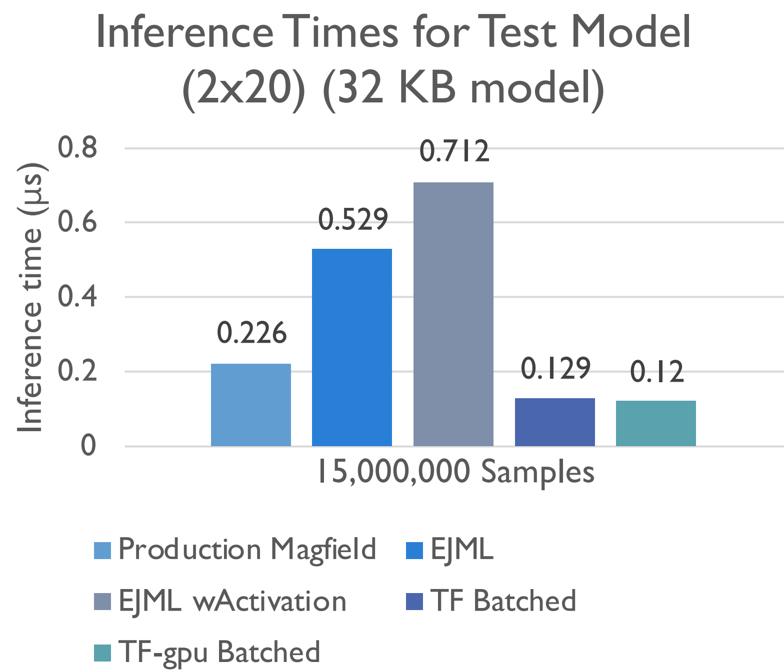
- DL4J/Keras/TensorFlow inference times are very fast – by industry standards
- In order to improve the inference time we explored multiple options
- One solution is to propagate values ourselves using the Efficient Java Matrix Library (EJML).

```
void feedForward(float[] input, float[] results) {
    SimpleMatrix matrix = new SimpleMatrix(new float[][]{input});
    for (int i = 0; i < LAYERS.length; i++) {
        matrix = matrix.mult(LAYERS[i]).plus(BIASES[i]);
    }
```

It will be more difficult for non-MLP models

Performance Benchmarks and Future Prospects

- With a simplified model the inference time is 3.2x slower the conventional algorithm and 2-300x faster than using Keras/TensorFlow
- We will continue testing to see if an improved model would be useful to put into production
- Could be useful for Open Science Grid transfers to save bandwidth and time
- It could also be used to initialize a "conventional" magnetic field in memory rather than reading in a file
  - Model files are ~1MB currently



The inference time (w/o batching) is off the charts at 100-300 µs

Partial Wave Analysis

PyPWA

- A python-based software framework designed to perform Partial Wave and Amplitude Analysis with the goal of extracting resonance information from multi-particle final states.
- Code base has been in development since 2014 and has been significantly improved with each revision - Version 3.0 just released!
- Efficient amplitude analysis framework including multithreading and CUDA support
- Optimizers include: Minuit, Nestle (or add your own!)

Group Members

Carlos Salgado (NSU/Jlab)
Mark Jones (NSU)
William Phelps (CNU/Jlab)
Michael Harris (NSU)
Andru Quiroga (CNU)

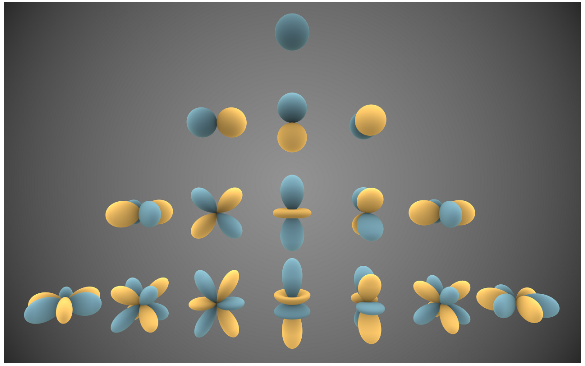
Former Group Members

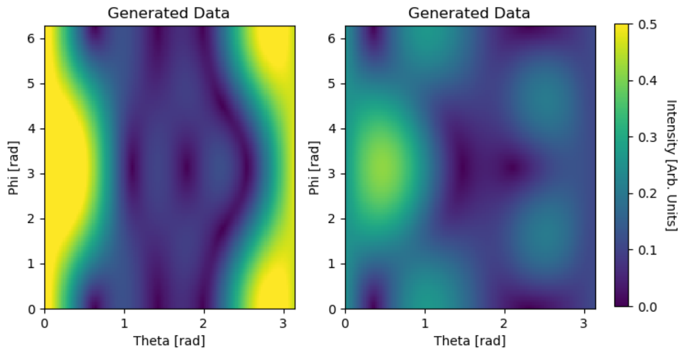
Josh Pond
Stephanie Bramlett
Brandon DeMello

Website: https://pypwa.jlab.org
GitHub: https://github.com/JeffersonLab/PyPWA

Preliminary Studies using Neural Networks

- Generate datasets using decay amplitudes (linear combination of spherical harmonics) with the following quantum numbers
  - L = 1,2,3
  - m = 0,1
  - ϵ_R= -1,+1
  - 9 total waves ("fit parameters")





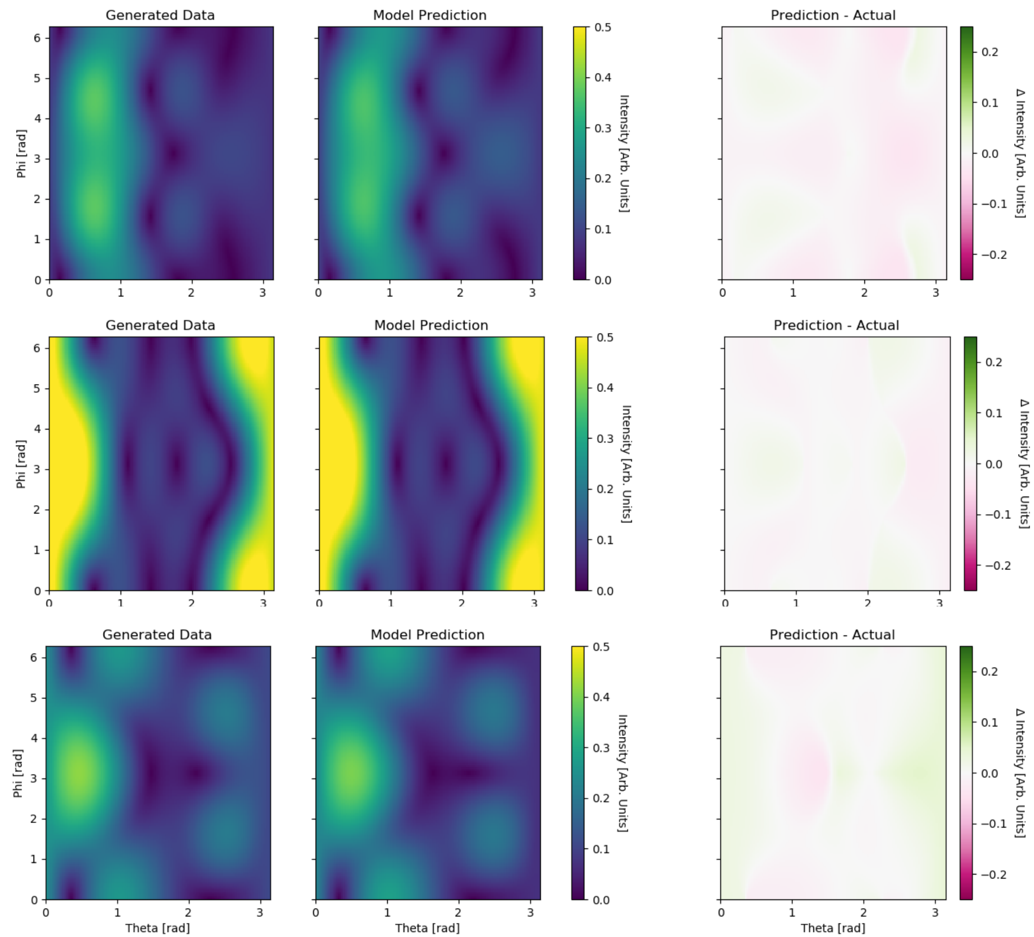
$$I(\Omega) = \sum_k \sum_{\epsilon_R} \sum_{l,|m|,l',|m'|} {}^{\epsilon_R}Y_l^{|m|}(\Omega)\ {}^{\epsilon_R}V_{l,|m|}^{k}\ {}^{\epsilon_R}V_{l',|m'|}^{k*}\ {}^{\epsilon_R}Y_{l'}^{|m'|*}(\Omega)$$

Production Amplitudes

Decay Amplitudes

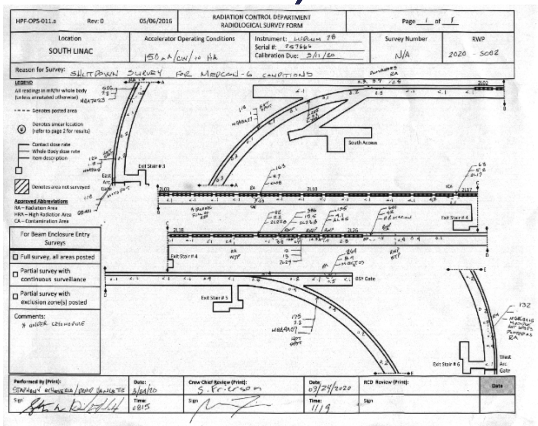
Results and Future

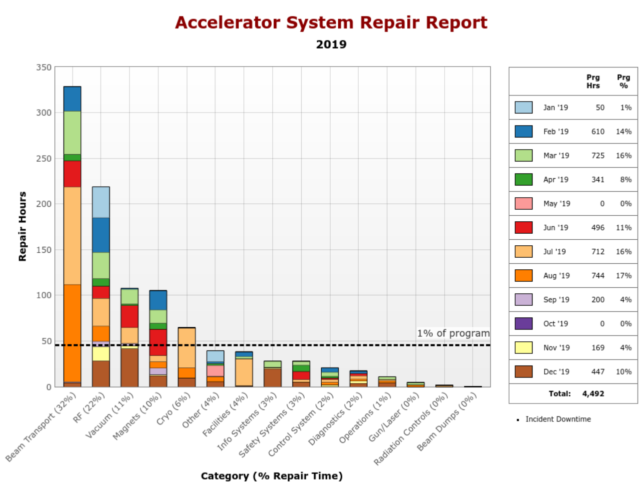
- We compare the intensity function and compare it to the model prediction
- Model Architecture:
  - 128x128 2D histogram as input
  - 9x128 Dense Layers – Relu activation
  - 9 production amplitudes as output
- In order to deal with the vast amounts of data we used generators to generate data for each epoch on the fly
- This is preliminary work and will continue to see where this may lead to



Collaborative Autonomous Sensor System for Intelligent Operation of Particle Accelerators (CASSIOPeiA)

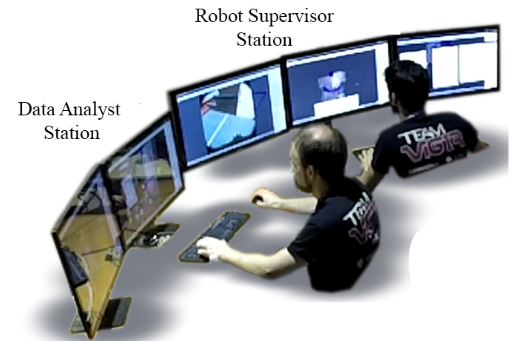
- A collaboratively autonomous mobile platform containing a sensor payload would be operated in the accelerator tunnel
- Mobile diagnostics can reduce accelerator downtime
- Planning to use AI with the new information-rich datasets which could help predict faults and when maintenance is required
- **Proposal submitted and is still under review**





David Conner PI (CNU), Chris Tennant Co-PI (JLab), William Phelps Co-PI (CNU/JLab), Nathan Lau (Va. Tech)

CASSIOPeiA



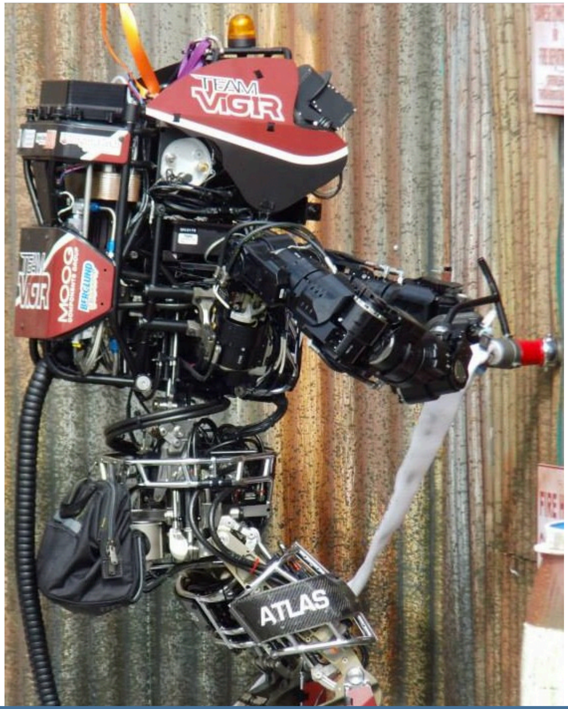
- Robot could consist of a wheeled platform and a modular arm with a sensor package mounted on the arm
- Sensor package could include: Radiation monitors, thermal cameras, optical cameras



Robot concept using Clearpath Husky base (left), HEBI Robotics modular arm assembly (middle), and incorporating an elevating platform (right).

What is Collaborative Autonomy?

- Collaborative Autonomy is when the robot and robot operators work together as a team
- Operators can Inject information
  - Intermediate goals
  - Templates for manipulation
- Operators can preempt behaviors
  - Prevent dangerous situations
  - Take advantage of superior perception
- Team ViGIR competed in a DARPA challenge simulating disaster relief scenarios where the wireless communications would be disrupted
  - Ideal for when the RF is on in the accelerator tunnel



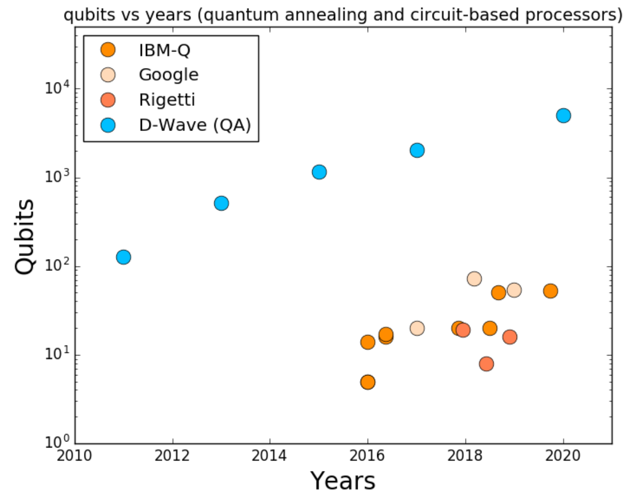
Boston Dynamics Atlas Humanoid Robot

CASSIOPeiA Summary

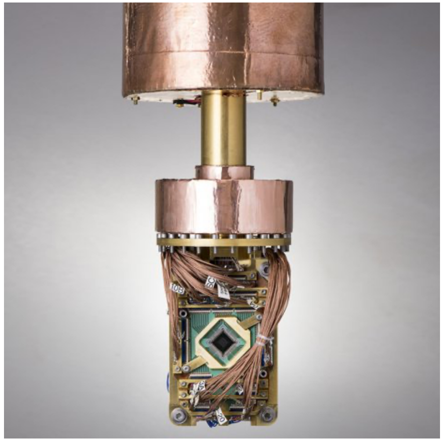
- The proposed robotic system would:
  - Improve the efficiency of standard tasks such as taking radiation surveys
  - Minimize personnel exposure to radiological environments promoting ALARA principles
  - Provide a platform for novel measurements and data acquisition
- Reducing unnecessary accelerator down time, automating repetitive and time-consuming tasks and improving the operational efficiency
- Preliminary estimates of the cost savings due to the increased efficiency total $850k/year

Quantum Computing

- Quantum computing is an emerging technology that is growing at an exponential rate
- In October 2019 a Sycamore QPU claimed to have quantum supremacy
  - Solving a problem that no conventional computer can feasibly solve
- In particular D-Wave has quantum annealers with >2k qubits and has allowed users to use their systems essentially as cloud QPUs



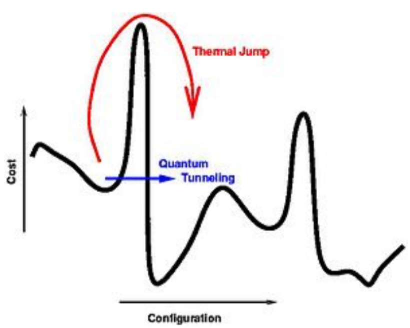
Number of qubits as a function of time for annealing and circuit-based processors

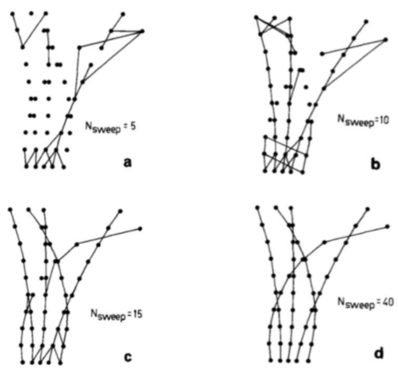


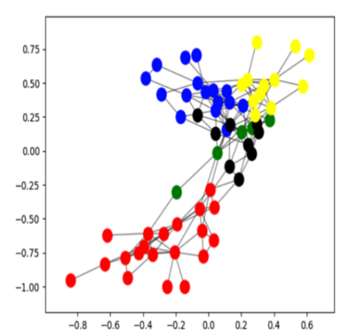
Reconstruction Algorithms with Quantum Annealers

- In the next few years it may be possible to harness quantum computing for nuclear physics applications
- We are currently developing a quantum computing based reconstruction algorithms
  - Could have the potential to revolutionize DAQ systems
  - Applications in detector alignment
- **Pending LDRD proposal has been submitted**

Group Members
Cristiano Fanelli (MIT/JLab)
Marco Battaglieri (JLab)
Evaristo Cisbani (INFN/ISS Roma1)
Alessio Del Dotto (INFN/LNF)
William Phelps (CNU/JLab)
Andru Quiroga (CNU)

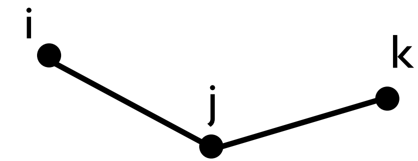






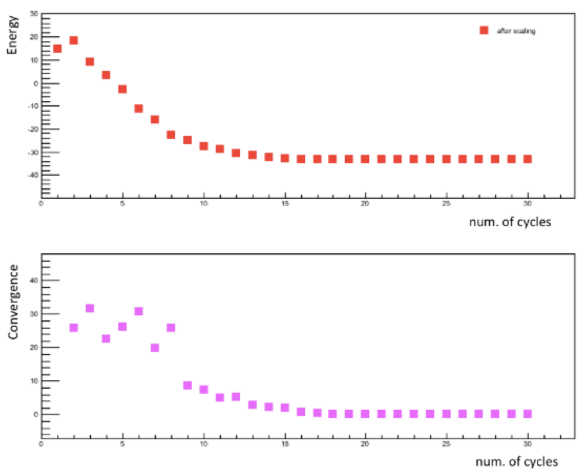
- Tracking
- Clustering
- Decision Trees
- …

Track Finding (Preliminary)



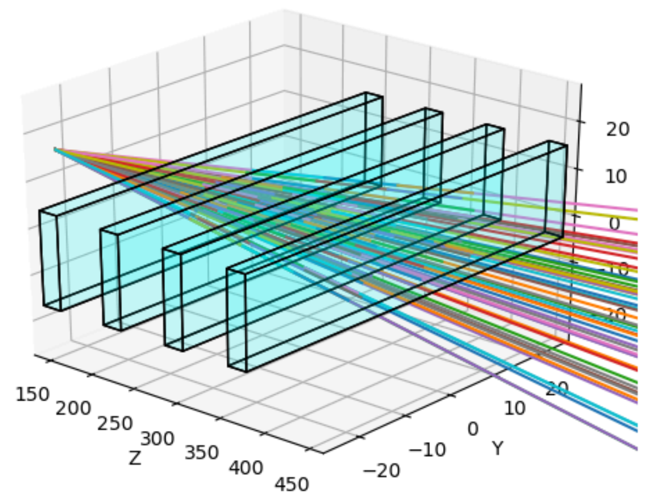
$$E = -\underbrace{\sum_{ijl} \frac{d_G \cos\theta_{ijl}}{r_{ij} + r_{jl}} \delta_{kj} V_{ij} V_{kl} + \sum_{ijl} \frac{d_G \cos\theta_{ijl} \delta_{li} V_{ij}}{r_{ij} + r_{jl}}}_{cost} + \underbrace{\alpha\Big(\sum_{ijl \neq j} V_{ij} V_{il} + \sum_{ijk \neq j} V_{ij} V_{kj}\Big)}_{constraint},$$

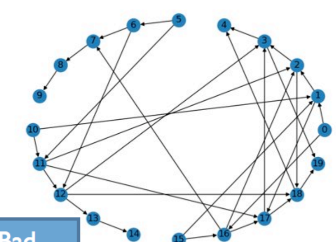
Classical Implementation



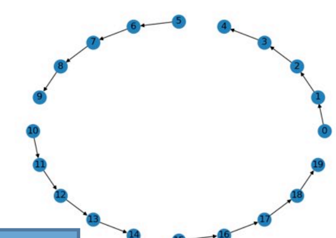
CF et al 2015, J. Phys. Conf. 608, 1, 012053

Implementation on Quantum Annealer





Bad



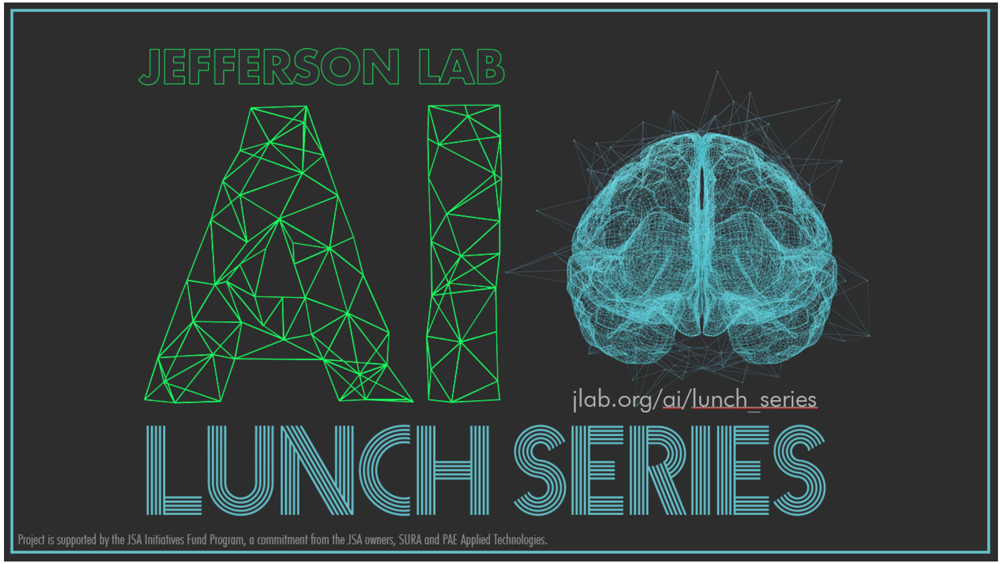
Good

Summary

- Artificial Intelligence
  - We have shown that it is possible to reproduce the Torus field for CLAS12 will continue benchmarking and tests
  - Initial work done with PyPWA looks promising but is also challenging
  - We are looking for future practical problems to utilize AI at Jefferson Lab
    - A possible future project is calorimeter response generation using NNs as it has already been done in other facilities and could save significant computational resources
- CASSIOPeiA is hopeful to bring state of the art robotics and AI in order to help improve accelerator reliability
- Quantum Computing is here and our preliminary reconstruction algorithms show promise

AI Growth and Outreach at Jefferson Lab

- Weekly, informal meeting
- Quarterly physics-inspired ML problems (i.e. mini-Kaggle)
- External speakers










Questions?

https://www.jlab.org/AI/lunch_series

https://www.jlab.org/AI/quarterly_problem

join #ml channel on slack!

https://jlab12gev.slack.com/messages/CFTBERJGK

Backup

Preliminary Results

Model Name:	Loss	Acc	Acc	TTE	Size	params
Full_torus_18Apr2018_Adam_100_360_data.h5	0.770534	0.3668Kg	0.4041|0.3697|0.3267	218ns	155Kb	10,803
Full_torus_18Apr2018_Adam_10x100.h5	0.005418	0.0306Kg	0.0320|0.0320|0.0277	406ns	1,170Kb	91,603
Full_torus_18Apr2018_Adam_1x1000.h5	1.569741	0.6642Kg	0.7442|0.7539|0.4946	312ns	105Kb	7,003
Full_torus_18Apr2018_Adam_2x1000.h5	0.199662	0.1816Kg	0.1843|0.1856|0.1748	750ns	12,122Kb	1,008,003
Full_torus_18Apr2018_Adam_2x20.h5	4.442259	0.9684Kg	1.1745|1.2241|0.5066	187ns	32Kb	563
Full_torus_18Apr2018_Adam_3x1000.h5	0.374696	0.2363Kg	0.2458|0.2355|0.2276	1187ns	24,140Kb	2,009,003
Full_torus_18Apr2018_Adam_5x100.h5	0.236885	0.2016Kg	0.2067|0.2002|0.1977	312ns	533Kb	41,103
Full_torus_18Apr2018_Adam_5x100_phi.h5	5.411231	0.8198Kg	0.9958|1.0124|0.4512	343ns	533Kb	41,103

To show a few, We tested some preliminary models for several criteria:

- **“Loss” (Fitness function):** Determines how precise the model performs
- **“Acc” (Accuracy):** Absolute error of the output, in kilogauss
- **“Acc” (Accuracy cont.):** Absolute error of the individual axes’ outputs, in kilogauss
- **“TTE” (Time to execute):** How long it takes to execute one prediction, in nanoseconds
- **“Size” (Size of model):** Size of model’s h5 file, in kilobytes
- **“Params” (Complexity):** Amount of changeable Parameters in the model