DEEP LEARNING FOR AMPLITUDE ANALYSIS
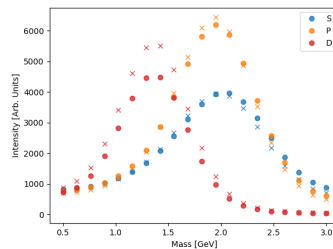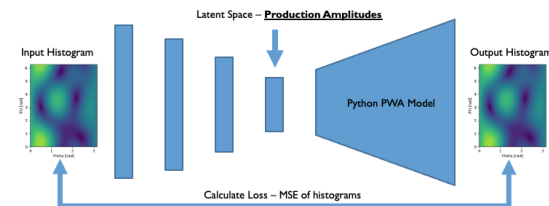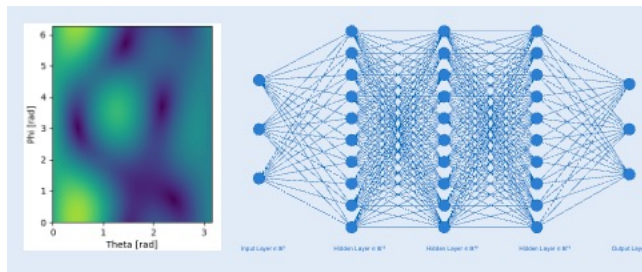
William Phelps

Christopher Newport University/Jefferson Lab

Roadmap

- Primer on PWA/PyPWA

- Deep Learning - Partial Wave Analysis (PyPWA)

  - Uncertainty Quantification

  - Wave Selection

  - Mass Dependent

t time

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.**

- In development since 2014 and has been significantly improved with each revision. Version 4.0 with PyTorch library has been released.

- Efficient amplitude analysis framework including multithreading, CUDA support, and PyTorch libraries

- Optimizers include Minuit, Nestle, MCMC (or add your own!)

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

Group Members
Carlos Salgado (NSU/Jlab)
Mark Jones (NSU)
Peter Hurck (Glasgow)
William Phelps (CNU/Jlab)
Andru Quiroga (CNU)
Nathan Kolling (CNU)
Ryan Brunk (CNU)
Rafael Diaz-Cruz (CNU)
Brian Rotich (NSU)

Former Group Members
Josh Pond
Stephanie Bramlett
Brandon DeMello
Michael Harris (NSU)
Bruna Goncalves (NSU)

NIM Article

March 2024

Contents lists available at ScienceDirect

Nuclear Inst. and Methods in Physics Research, A

journal homepage: www.elsevier.com/locate/nima

ELSEVIER

Full Length Article

PyPWA: A software toolkit for parameter optimization and amplitude analysis

Mark Jones [a], Peter Hurck [b,*], William Phelps [c,d], Carlos W. Salgado [a,d]

[a] *Norfolk State University, Norfolk, 23504, VA, USA*
[b] *University of Glasgow, Glasgow, G12 8QQ, United Kingdom*
[c] *Christopher Newport University, Newport News, 23606, VA, USA*
[d] *The Thomas Jefferson National Accelerator Facility, Newport News, 23606, VA, USA*
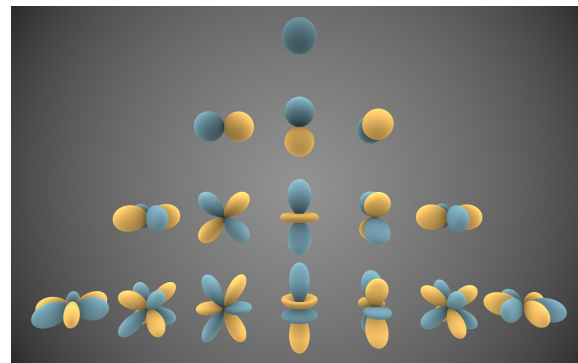
ARTICLE INFO

ABSTRACT

PyPWA is a toolkit designed to optimize parametric models describing data and generate simulated distributions according to a model. Its software has been written within the python ecosystem with the goal of performing Amplitude or Partial Wave Analysis (PWA) in nuclear and particle physics experiments. We briefly describe the general features of amplitude analysis and we provide a description of the PyPWA software design and usage. We also provide benchmarks of the scaling and an example of its application.
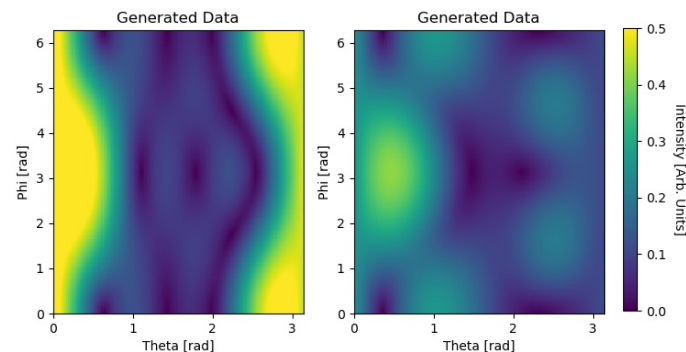
PWA using Neural Networks

- 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

Tools of the Trade

- Python 3.10 – Anaconda
  - Keras/TensorFlow – NN Libraries
  - Pandas/Numpy – Data Handling
  - Matplotlib – Visualization
  - Uproot – Native Python ROOT Library (J. Pivarski)
  - Optuna – Hyperparameter optimization library
- Institutional GPU nodes or those through Jefferson Lab
  - Either through Jupyterhub or interactively using slurm to request a node
  - Several institutions with Nvidia V100 and A100 Cards (NSU/JLAB)
  - Several machines with 4 NvidiaTitan RTX GPUs and some with 14 Nvidia T4 GPUs

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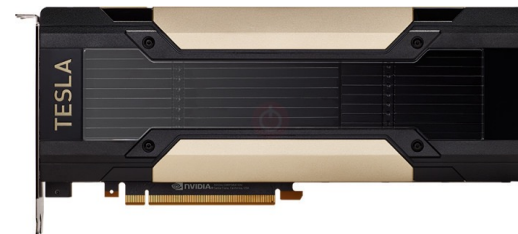
```python
test = pd.read_csv("TRAIN/TRAIN.csv")
labels = pd.read_csv("TRAIN/TRAIN_labels.csv")
activation = 'relu'

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)
```
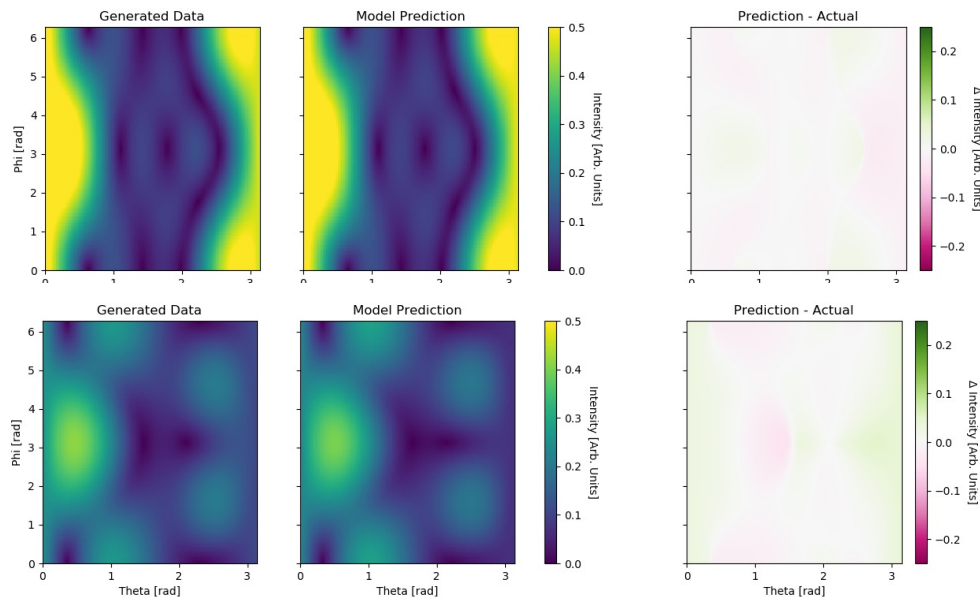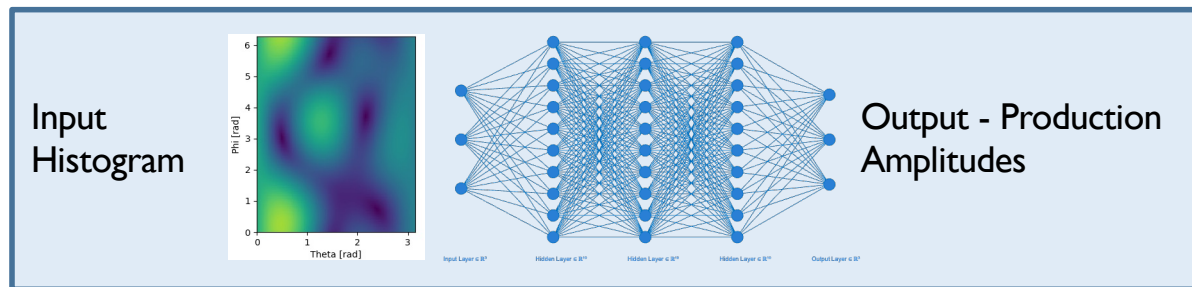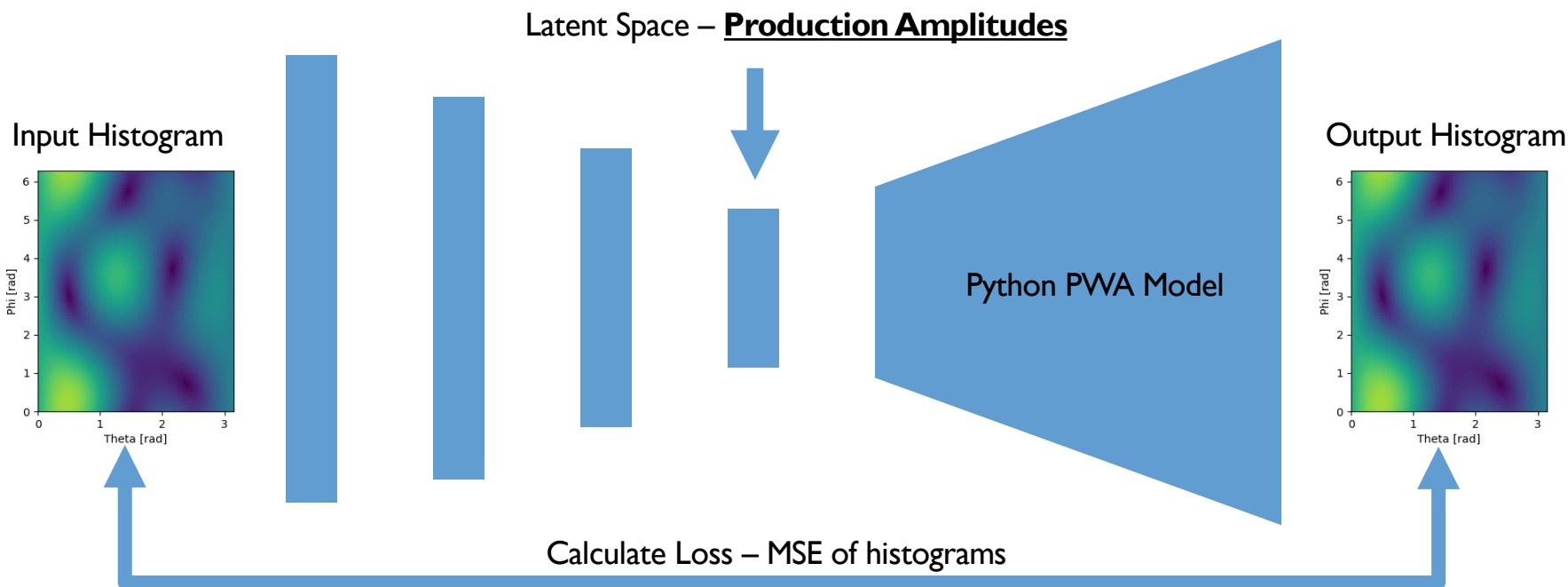
MLP Results

- 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

Useful Tools: Generators, Complex Valued Deep Learning

Autoencoder for PWA

Unsupervised learning!

Latent Space – **Production Amplitudes**

Input Histogram

Python PWA Model

Output Histogram
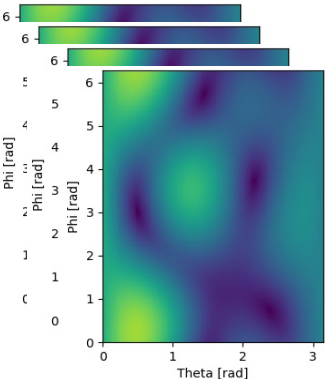
Calculate Loss – MSE of histograms

Autoencoders for PyPWA

- Encoder portion is a standard MLP, but without labels!

- Decoder is a PyPWA model that takes in production amplitudes and produces a histogram

- Autoencoders *dramatically* improved the accuracy!

- Even works well for noisy data
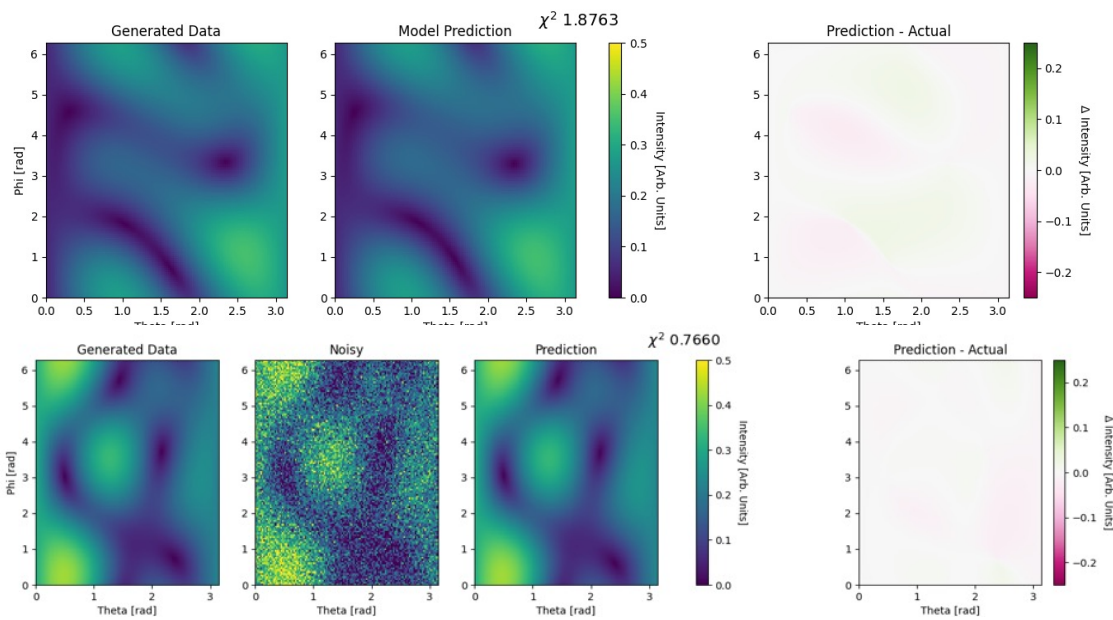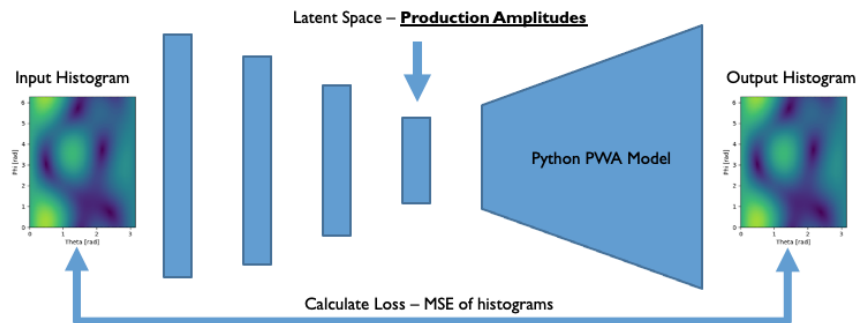
Mass Dependent Autoencoder work for PWA
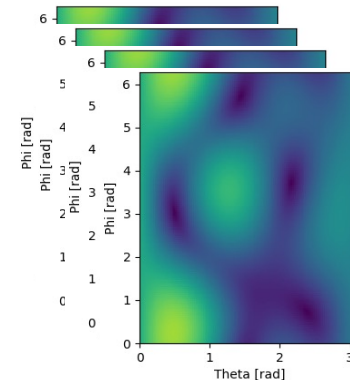

Input ~~Histogram~~ (20x) Histograms

Latent Space – **Production Amplitudes**

Output ~~Histogram~~ (20x) Histograms

Python PWA Model

Latent Space p.2 – Resonance Mass and Width

Calculate Loss – MSE of 20 histograms

Results

- With a CONV3D input to our autoencoder we see a good agreement with the generated data and inference from our neural networks

- Shown on the right are three different tests with randomly generated data/resonances

Crosses – Generated
Circles – Inference
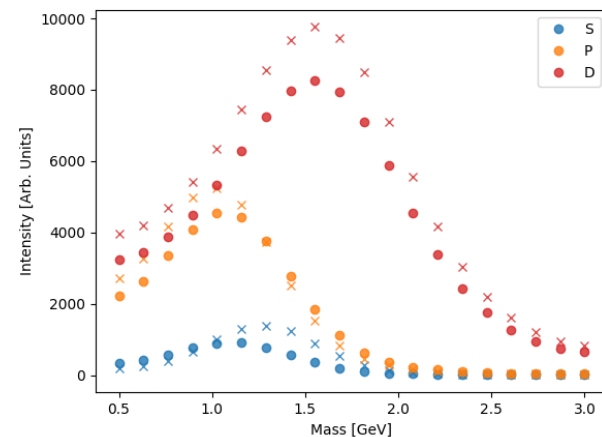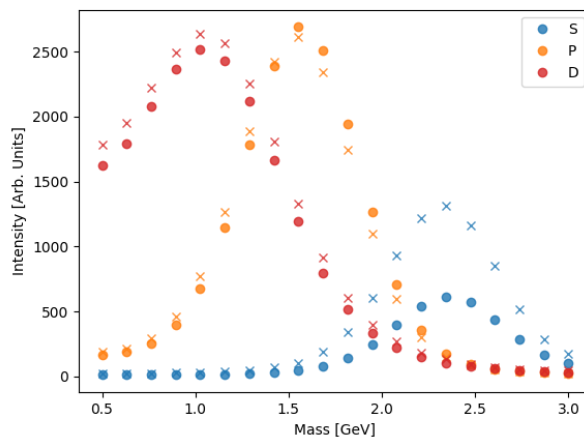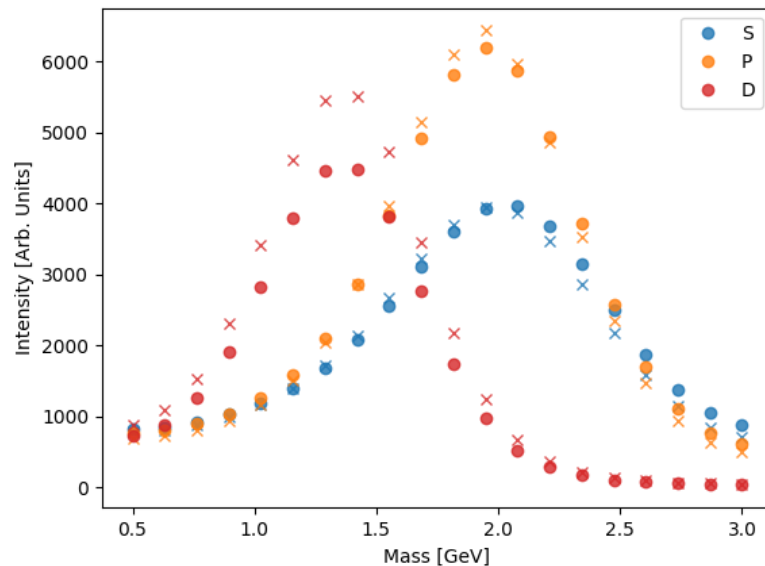
Uncertainty Quantification - VAE

- For uncertainty quantification we are using Variational Autoencoders (VAE) with some success

- Traditional (hybrid) autoencoder performs better for now

- Future work could involve some constraints to resolve ambiguities and allow better fits

Ambiguities appear as rotations/sign flips
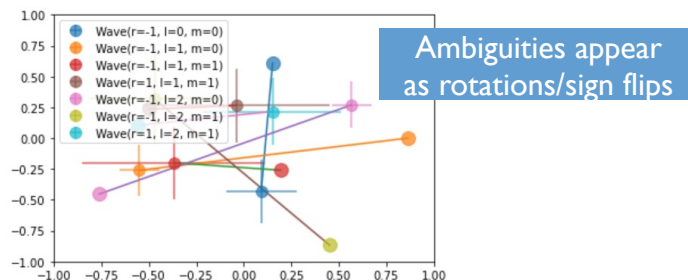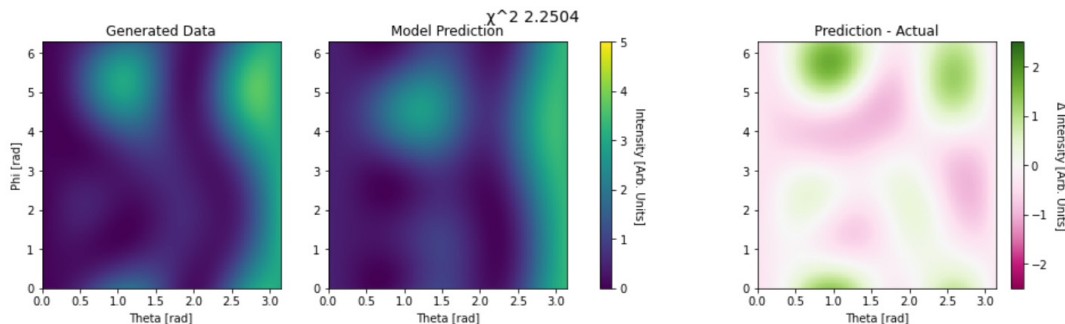
		X_0	X_1	X_2	X_3	X_4	X_5	X_6	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
True	0.15	0.86	0.19	-0.50	-0.76	0.45	-0.56	0.61	0.00	-0.26	0.24	-0.45	-0.87	0.10
Pred	0.09	-0.55	-0.37	-0.04	0.57	-0.47	0.15	-0.43	-0.26	-0.20	0.26	0.27	0.32	0.22

Wave Selection DNN

Output: Wave Selection

- One of the problems that is regularly seen in PWA is choosing the right waves to use in your fit
- We simplified the regression problem we have posed in earlier slides to create a tool that could be used to select which waves are present
- Multi-label classification
- May be used as a part of an ensemble

Input Histogram

FC Layer FC Layer FC Layer

1
0
1
·
·
·
1
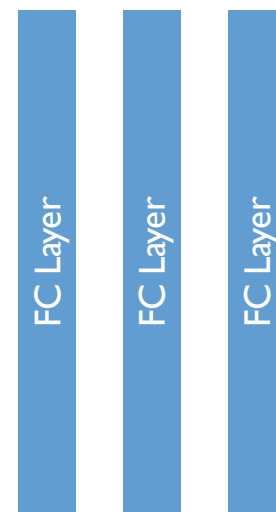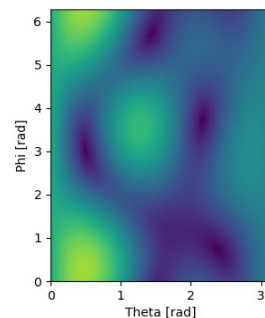
Preliminary results:
79% accuracy in selecting the right set of waves (Lmax=2)
96.3% wave/"digit"-wise accuracy

Summary

- We have been able perform PWA "fits" with neural networks

- Autoencoders dramatically improved the performance

- Variational autoencoders were tried with some degree of success for uncertainty quantification

- Future work includes continued work on hyperparameter optimization, uncertainty quantification, wave selection, and symbolic regression for PWA
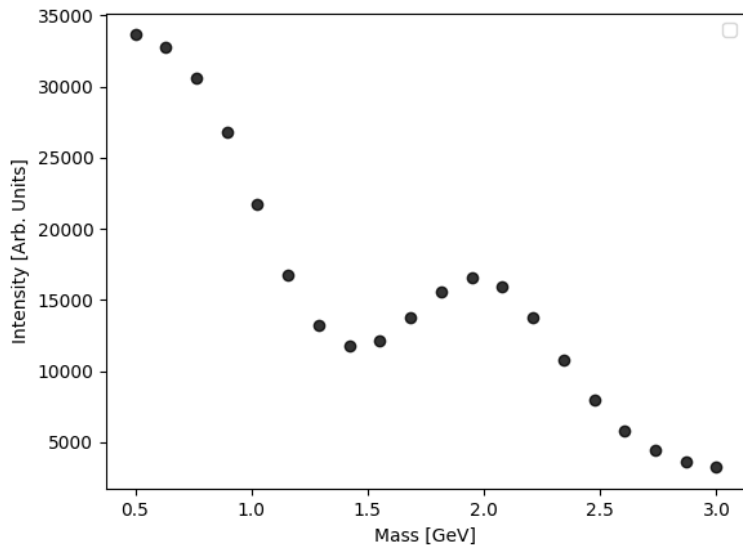
Future Work

London Underground (Kings Cross) advertisement for LLMs

Thanks!

Backup

The Mass-Dependent Generator

Randomly Generated Event
(Currently One Resonance per Wave)

Production Amplitudes
(Complex)

Breit-Wigner Coefficients
(Mean, Width)

Histogram
Generator

Set of
Histograms
Binned in Mass
(3D-Histogram)