ARTIFICIAL INTELLIGENCE IN PWA

Carlos Salgado – NSU/JLab
Andru Quiroga – CNU
William Phelps – CNU/JLab

Partial Wave Analysis

- 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

Regression on (θ,φ) distributions: Pattern Recognition

Classic Approach

- Extended Maximum Likelihood - Minuit Minimization
- Intensity function is called (calculated) in every Minuit interaction (MIGRAD)

This (AI/ML) Approach

- Convolution Neural Network (CNN) using Tensorflow/Keras
- CNN trained on large generated sample using Model
- After NN has been trained – parameters of pattern recognition learn - *no further optimization is done (no calls to calculate amplitudes)*.
- A histogram of angular distributions is "recognized" through a *fast linear matrix multiplication.*

Model

Inspired by PWA amplitudes we use a two dimensional angular distribution given by :

$$I(\theta,\phi) = \left|\sum_{l,m} {}^{+}T_{l,|m|}\;{}^{+}Y_l^{|m|}(\theta,\phi)\right|^2 + \left|\sum_{l,m} {}^{-}T_{l,|m|}\;{}^{-}Y_l^{|m|}(\theta,\phi)\right|^2 .$$

$${}^{\pm}Y_l^{|m|} = \left[Y_l^m(\theta,\phi) \mp (-1)^m Y_l^{-m}(\theta,\phi)\right]\Theta(m)(4)$$

and $Y_l^m(\theta,\phi)$ are the spherical harmonics.

${}^{\epsilon}T_{l,m}$ *are the fitted parameters described by* (ϵ, l, m)

With $\epsilon = \pm\,;\, l = 0,1,2...;\, m = 0, \pm 1, ... \pm l$

The T are complex

We also tried the distribution (inspired on moments):

$$\boxed{I(\theta,\phi) = \sum_{L,M} \sqrt{\frac{(L-M)!}{(L+M)!}} H(LM) P_l^m(\theta) cos(m\phi) \qquad (9)}$$

where P_l^m are the Associate Legendre Polynomials.

$$\boxed{H(L,M)\ are\ the\ fitted\ parameters\ described\ by\ (L,M)}$$

With $L = 0,1,2...; M = 0, \pm 1, ... \pm L$

The H are real.

Tools of the Trade

- 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

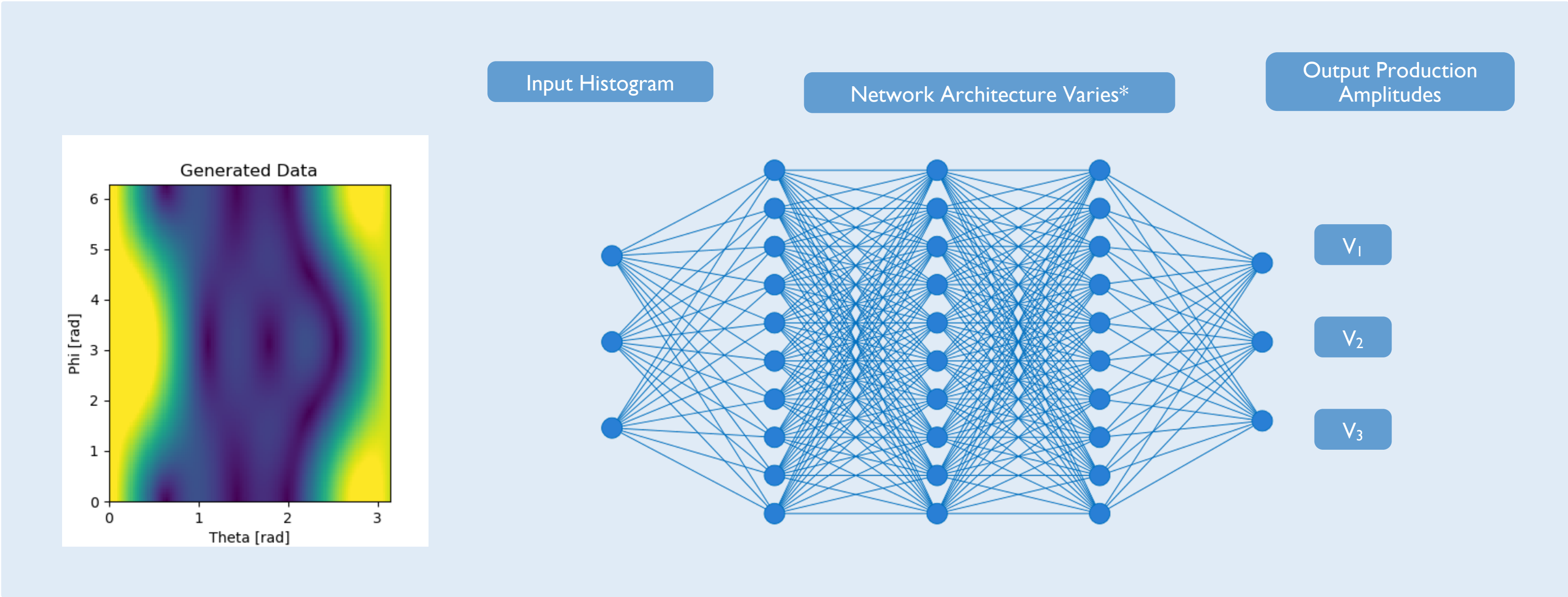
```
test = pd.read_csv("TRAIN/TRAIN.csv")
labels = pd.read_csv("TRAIN/TRAIN_labels.csv")
activation = 'relu'

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)
```

Method

- Using various Neural Network topologies to fit binned angular distributions
  - Angular distribution is described by the intensity function histogram, $I(\Phi,\theta)$
    - Creating histograms of 128x128 bins
- We started with generating angular distributions in the form of histograms stored in NumPy files (>10GB) but we changed to "on the fly" generation
- Conditions information:
  - Fix one phase angle for one of the waves.
  - Only train with a phase difference up to π otherwise there will be ambiguities
  - Regression with Neural Networks is different than traditional ML algorithms like Gradient Descent (MIGRAD-Minuit)
    - Multiple valid solutions will be given as *an average of valid solutions*, assuming training was not biased.
    - In some cases just the training will not converge

Convolutional Neural Networks (CNN)

- 128 Dense Layers – Relu activation
- 4 to 8 layers
- 20 to 50 epochs

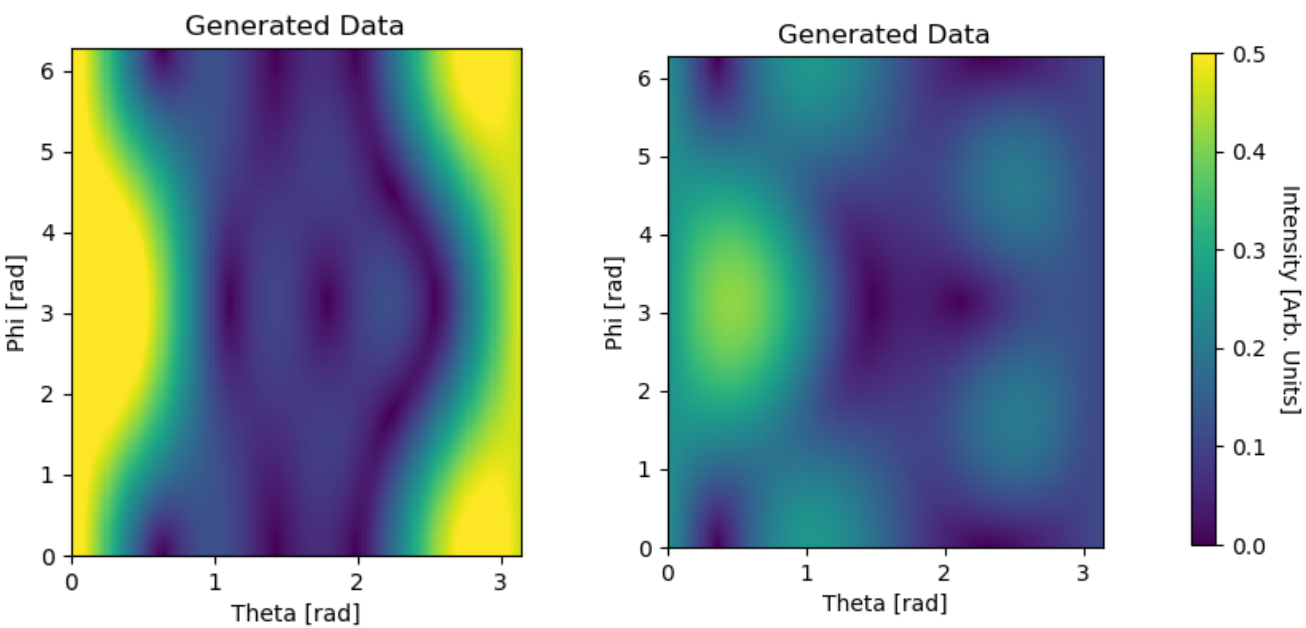


- Generate datasets using decay amplitudes with the following quantum numbers
  - L = 1,2,3 (we used up to 5)
  - m = 0,1
  - ϵ_R= -1,+1
  - 9 total waves ("18 fit parameters")

$T = Ae^{i\theta}$

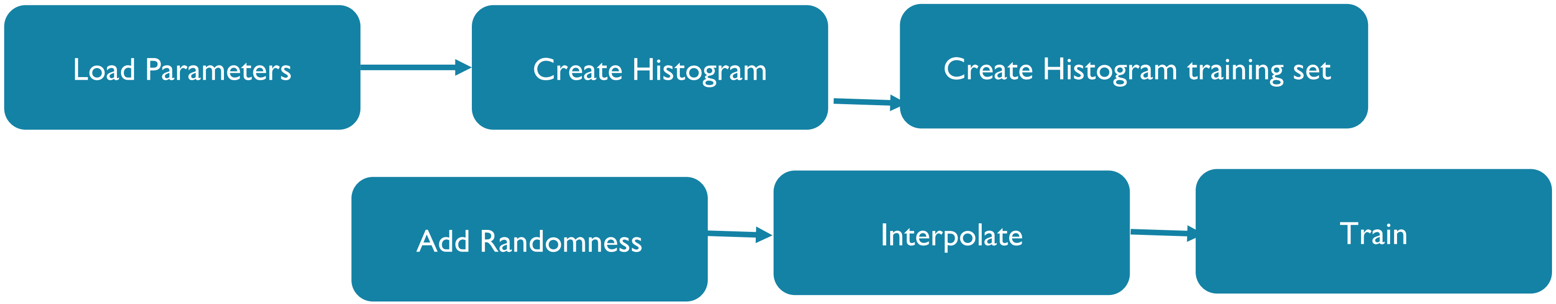
A = [0,1]

Θ =[0,2**π**]. But used [0, **π**]



“On the fly” Generator

Due to large amount of data, training set is not stored in memory



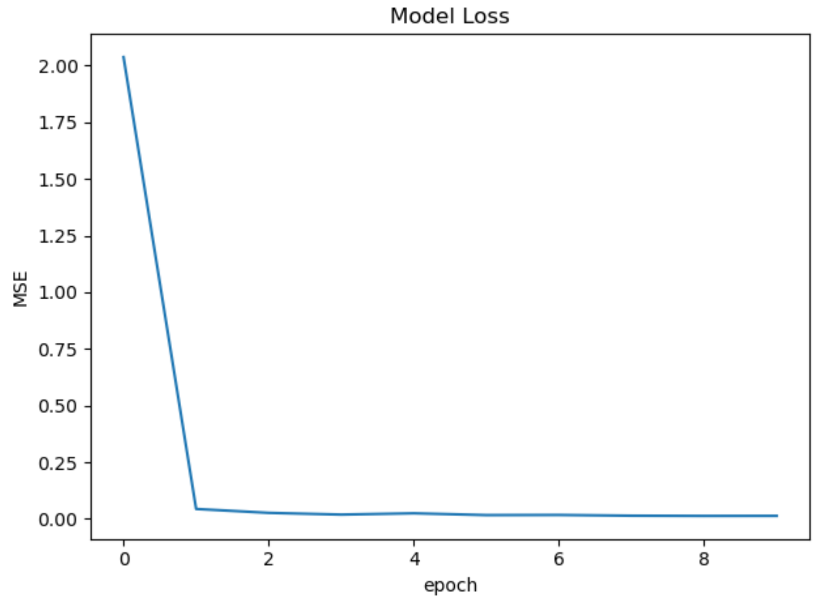

```python
def vector3_generator():
    while True:
        yield random.random(), random.random(), random.random()

if __name__ == '__main__':
    V3_gen = vector3_generator()
    for i in range(10):
        print(next(V3_gen))
```

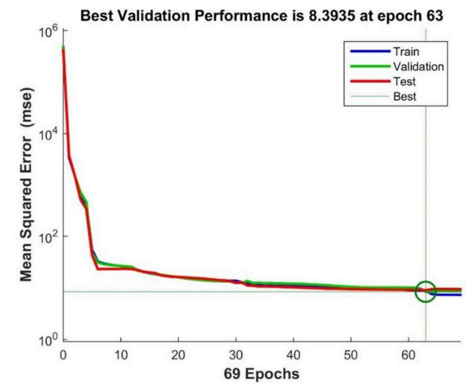
Generator Example – Returns tuple of 3 random numbers

Training Information

- How long does it take to train?

on average: Big jump initially and slow changes later – it can take a maximum of 2-3 hours (~15 epochs)





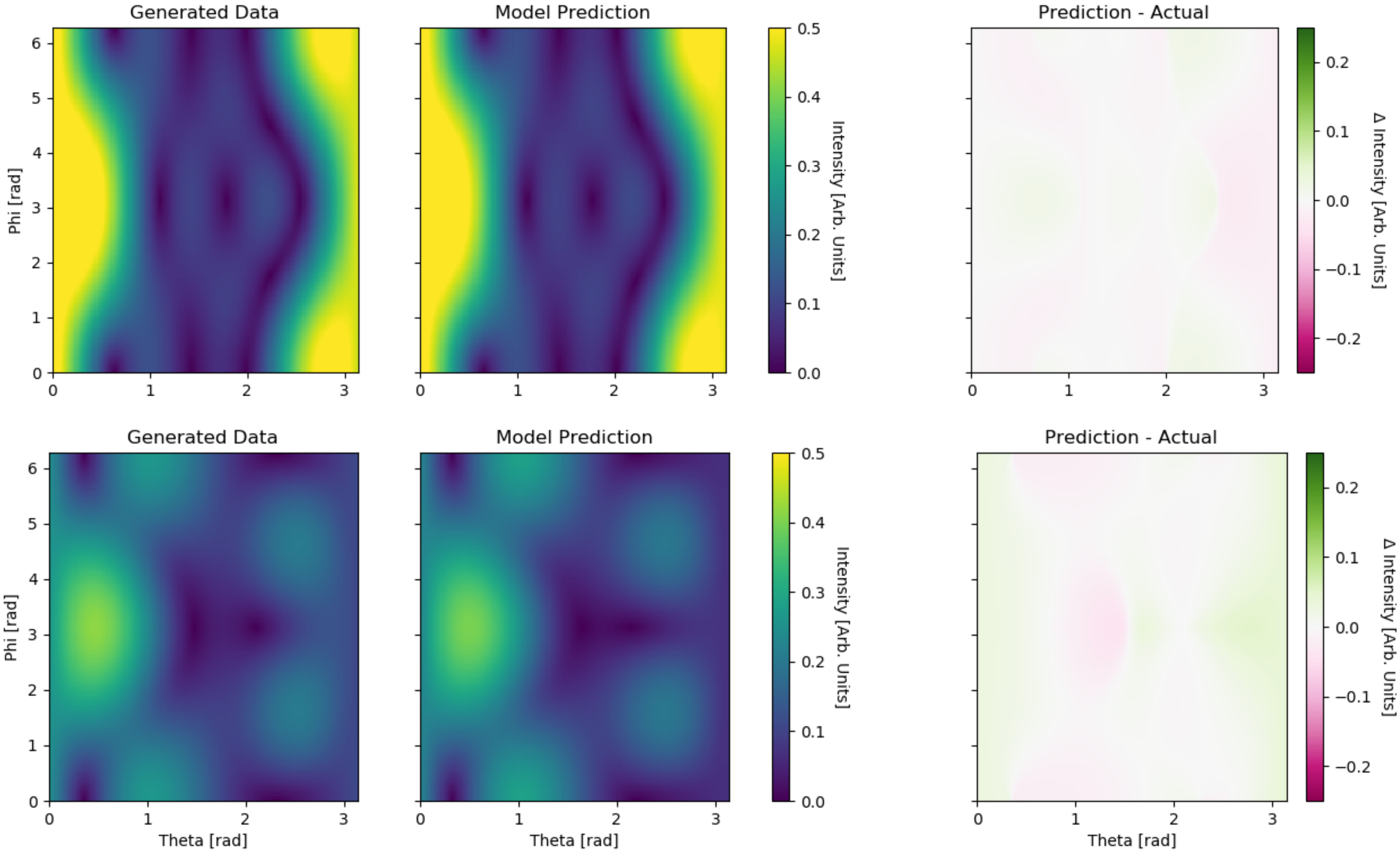
textbook

Results from fits

- We compare the generated intensity function the CNN model predictions
- Model Architecture:
  - 128x128 2D histogram as input
  - 9x128 Dense Layers – Relu activation
  - Pwa production amplitudes as output
  - Moments as output
- In order to deal with the vast amounts of data we used generators to generate data for each epoch on the fly
- PWA (T complex) about 70% (ambiguities!!) correct
- Moments (H real) about 90% correct

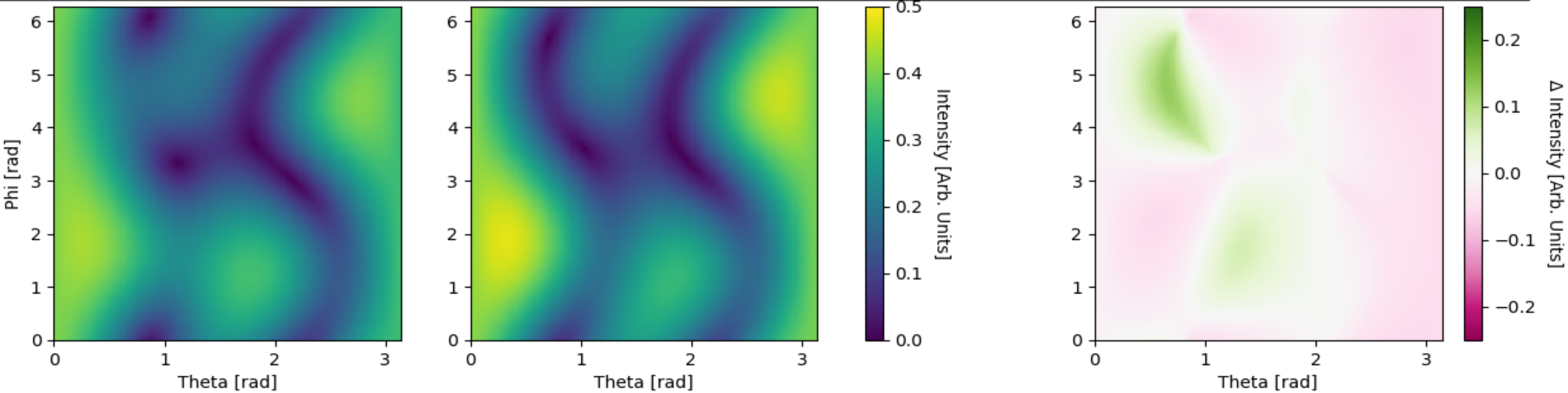
Of course these results will depend on the model (i.e. physics reactions)

PWA

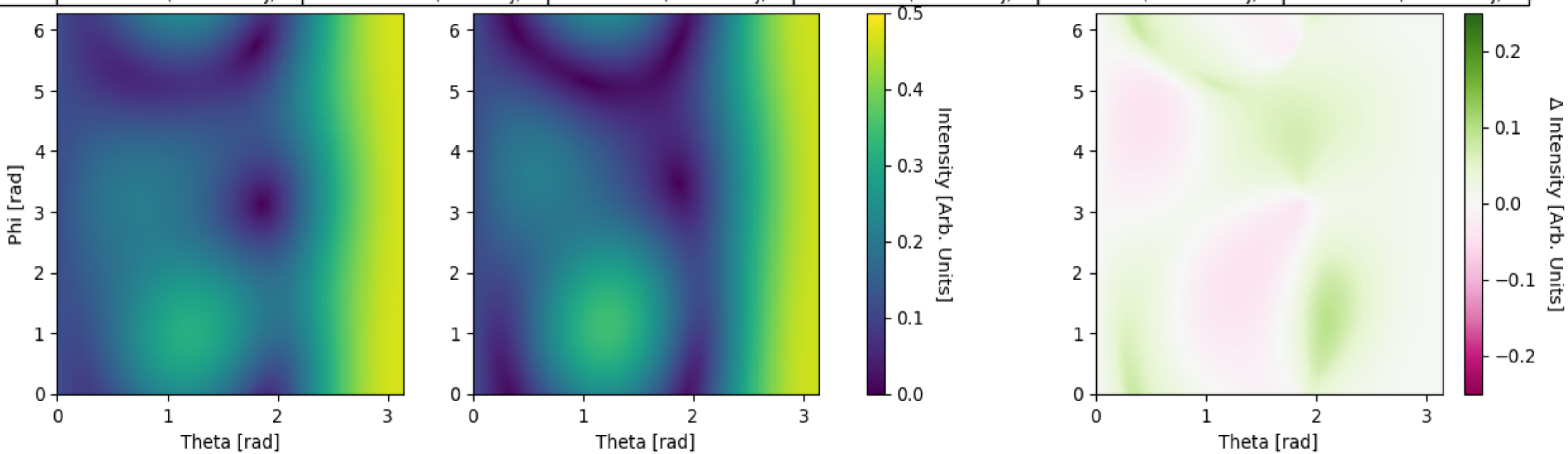

Generated Data
Model Prediction
Prediction - Actual
Phi [rad]
Theta [rad]
Intensity [Arb. Units]
Δ Intensity [Arb. Units]
0
1
2
3
4
5
6
0.0
0.1
0.2
0.3
0.4
0.5
0.2
0.1
0.0
−0.1
−0.2

PARAMETERS FIT

(ϵ, l, m)	(-1, 1, 0)	(-1, 1, 1)	(1, 1, 1)	(-1, 2, 0)	(-1, 2, 1)	(1, 2, 1)
Generated Data	(0.08+0j)	(0.2+0.07j)	(0.09+0.08j)	(0.1+0.3j)	(-0.08+0.2j)	(-0.02+0.02j)
Model Prediction	(0.1-0.02j)	(0.1+0.07j)	(0.07+0.1j)	(0.08+0.3j)	(-0.2+0.2j)	(-0.01+0.03j)

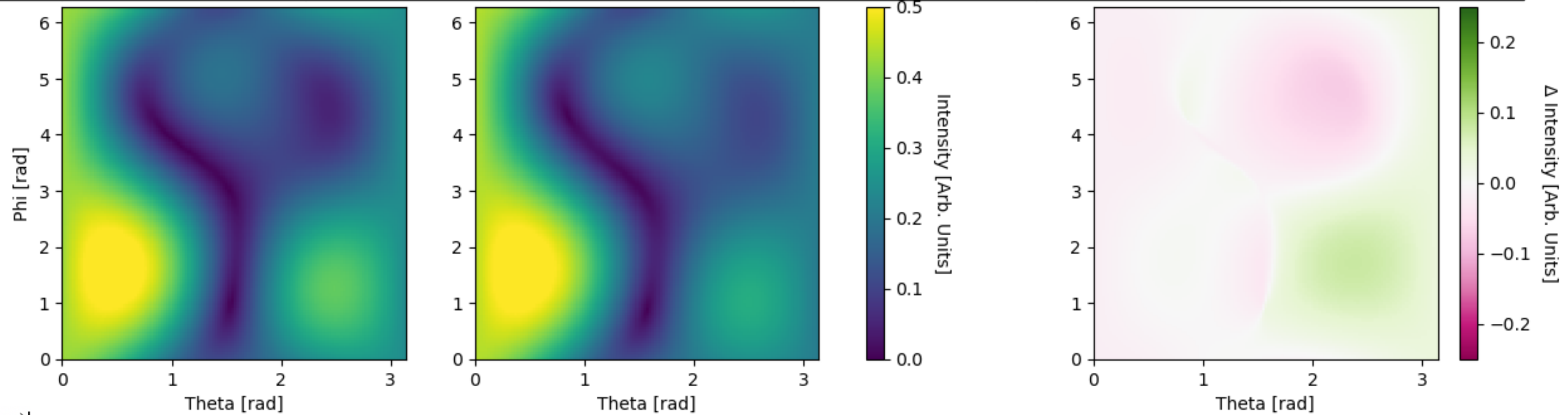


(ϵ, l, m)	(-1, 1, 0)	(-1, 1, 1)	(1, 1, 1)	(-1, 2, 0)	(-1, 2, 1)	(1, 2, 1)
Generated Data	(0.3+0j)	(0.2+0.09j)	(-0.006+0.1j)	(-0.2+0.08j)	(-0.05+0.08j)	(-0.02+0.1j)
Model Prediction	(0.2+0.007j)	(0.1+0.2j)	(-0.02+0.1j)	(-0.2+0.08j)	(-0.005+0.1j)	(-0.02+0.1j)

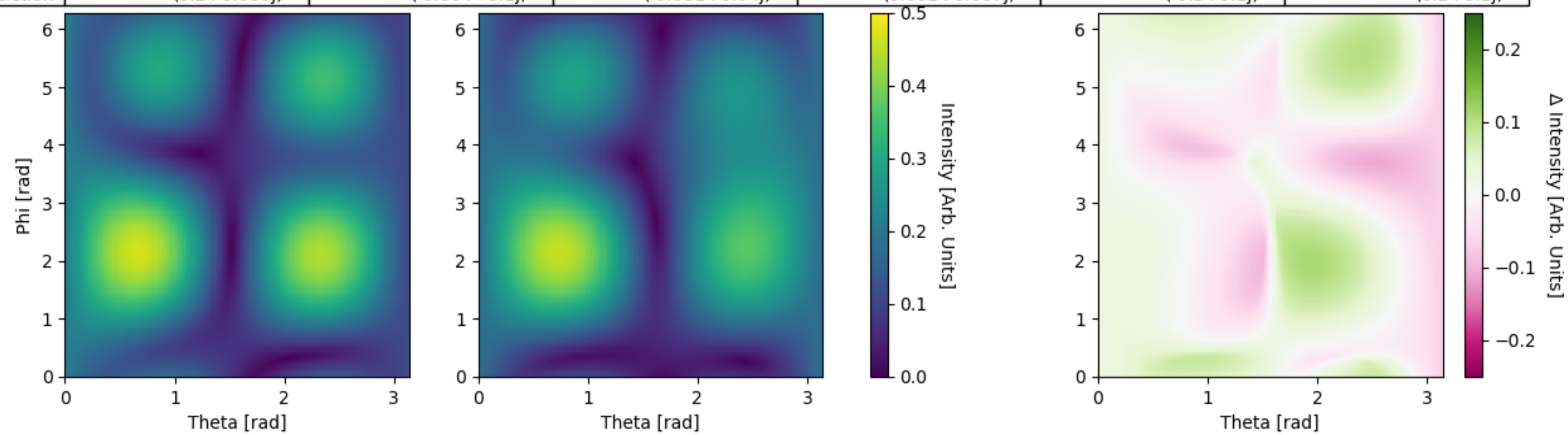


PARAMETERS FIT

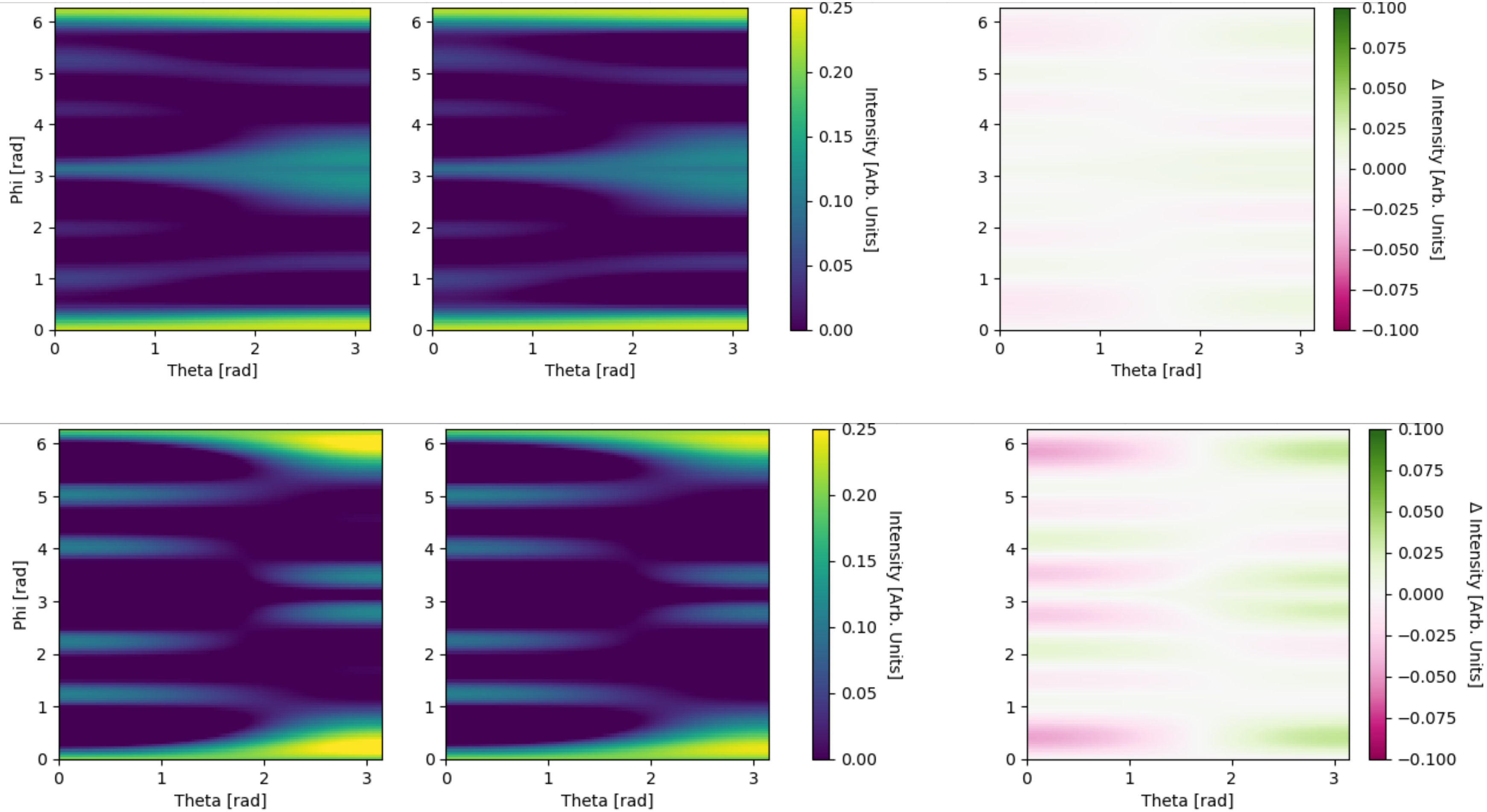
(ϵ, l, m)	(-1, 1, 0)	(-1, 1, 1)	(1, 1, 1)	(-1, 2, 0)	(-1, 2, 1)	(1, 2, 1)
Generated Data	(0.3+0j)	(-0.1+0.08j)	(0.04+0.06j)	(0.08+0.08j)	(-0.007+0.3j)	(-0.03+0.09j)
Model Prediction	(0.3+0.04j)	(-0.2+0.1j)	(0.03+0.06j)	(0.09+0.09j)	(-0.004+0.2j)	(-0.05+0.1j)



(ϵ, l, m)	(-1, 1, 0)	(-1, 1, 1)	(1, 1, 1)	(-1, 2, 0)	(-1, 2, 1)	(1, 2, 1)
Generated Data	(0.2+0j)	(0.06+0.02j)	(-0.03+0.06j)	(0.04+0.03j)	(-0.3+0.2j)	(0.2+0.2j)
Model Prediction	(0.2+0.008j)	(-0.004+0.1j)	(-0.001+0.04j)	(0.002+0.007j)	(-0.3+0.1j)	(0.2+0.1j)



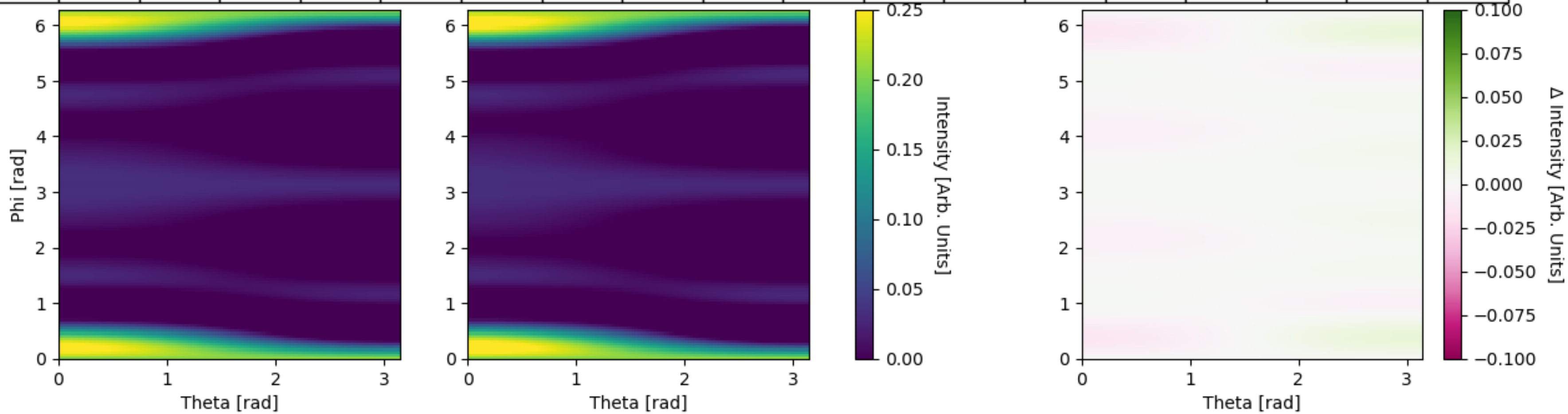
Moments


Phi [rad]
Theta [rad]
Intensity [Arb. Units]
Δ Intensity [Arb. Units]
Phi [rad]
Theta [rad]
Intensity [Arb. Units]
Δ Intensity [Arb. Units]

Moments

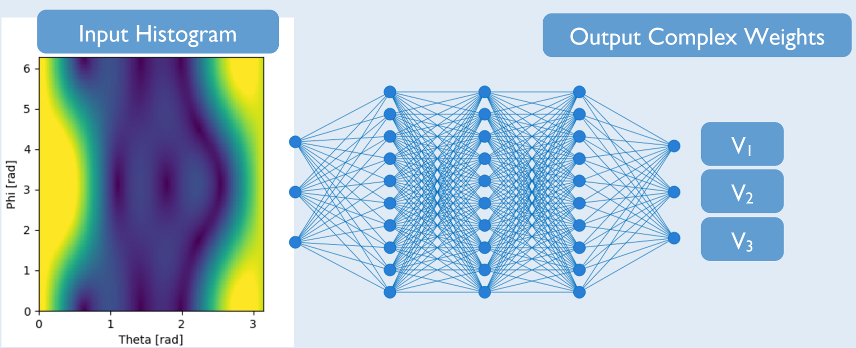
(L,M)

	(1, 0)	(1, 1)	(1, -1)	(2, 0)	(2, 1)	(2, -1)	(3, 0)	(3, 1)	(3, -1)	(4, 0)	(4, 1)	(4, -1)	(5, 0)	(5, 1)	(5, -1)	(6, 0)	(6, 1)	(6, -1)
Generated Data	4.02%	0.53%	4.06%	6.29%	8.15%	8.59%	4.81%	5.82%	9.04%	9.23%	0.67%	4.49%	5.87%	2.62%	7.14%	2.72%	7.12%	8.82%
Model Prediction	4.14%	3.57%	7.81%	6.18%	5.57%	5.67%	4.91%	3.85%	7.59%	8.77%	3.42%	7.86%	5.84%	2.96%	7.90%	2.98%	4.51%	6.47%



Complex-Valued Deep Neural Networks

- Used in MRI Reconstruction
- Using complex-valued weights and biases makes sense for regression in PWA
  - Regression in PWA involves taking real valued observables and extracting complex amplitudes from those measured values



Keras-complex arXiv:1705.09792
https://github.com/JesperDramsch/keras-complex

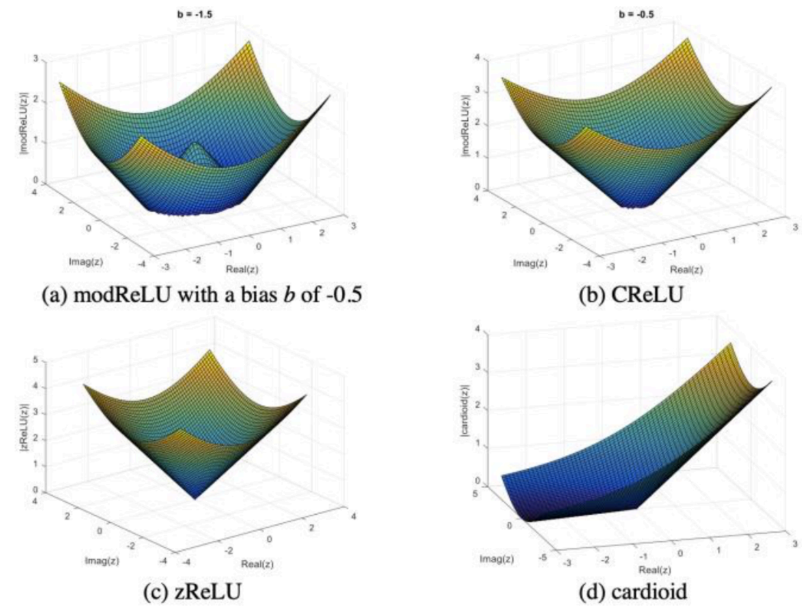


Fig. 1. Surface plots of the four tested complex-valued activation functions.

Analysis of Deep Complex-Valued Convolutional Neural Networks for MRI Reconstruction - arXiv:2004.01738

Future Work

- How to move beyond 9-12 waves for Regression
  - In real world fits, can we have a second NN
- How to better deal with Ambiguities?
- Include Uncertainties
- Include Acceptance/Resolution corrections.
- **What is of AI-PWA beyond Regression??**

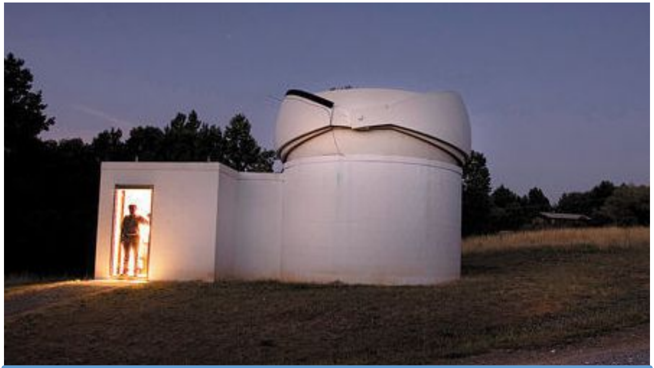
Summary

- Initial work done with PyPWA-AI **looks promising** but is also challenging.
- Fits worked much better for Moments (real-no ambiguities) than for Waves (complex-ambiguities)
- Lessons learned
    - On the fly generation
    - Ambiguities make “training” difficult (convergency)

Backup

Group Members

- Carlos Salgado (NSU/JLab)
  - Professor and Joint appointment with Hall D
  - Department of Physics
- 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


Carlos Salgado


William Phelps


Andru Quiroga