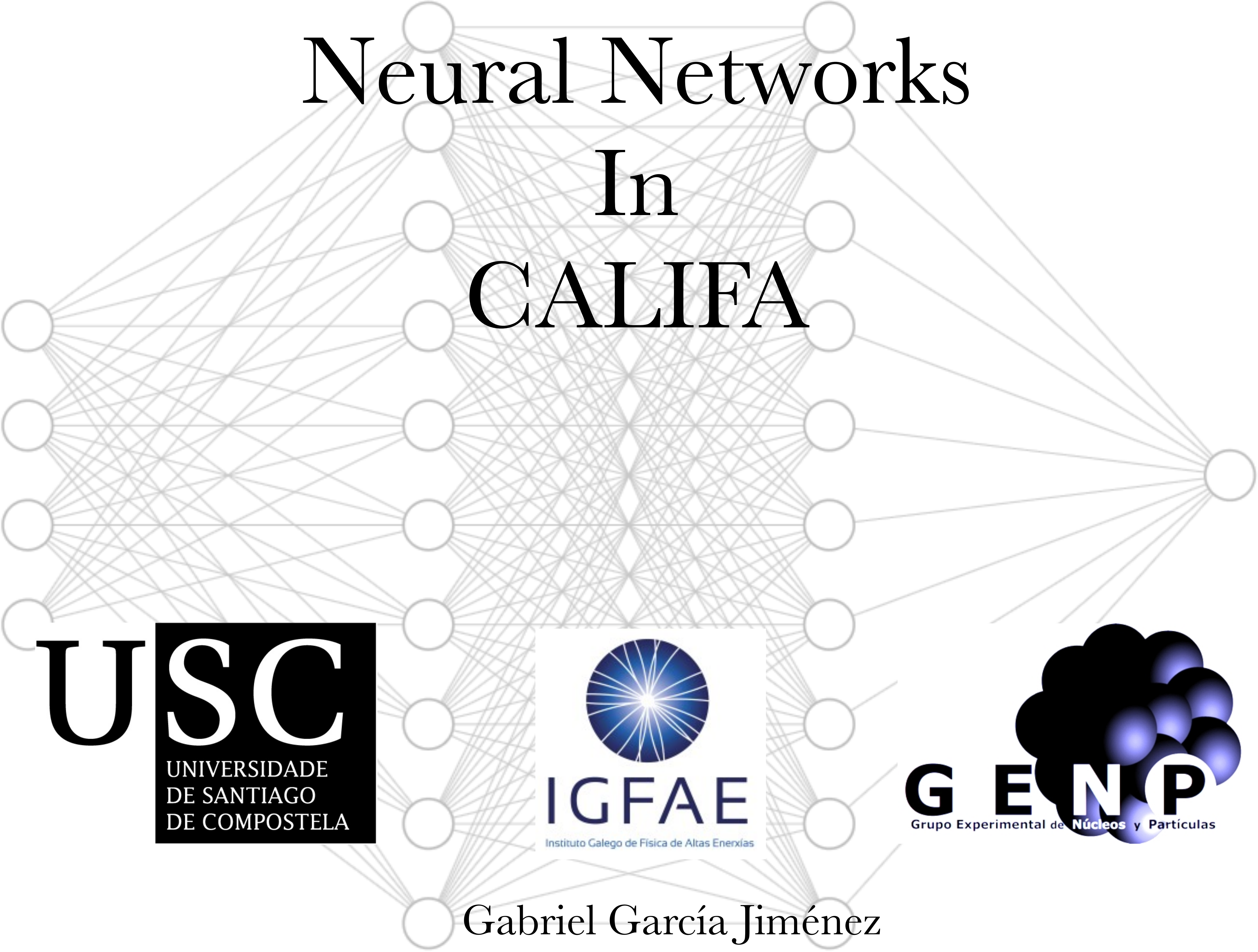
Neural Networks In CALIFA

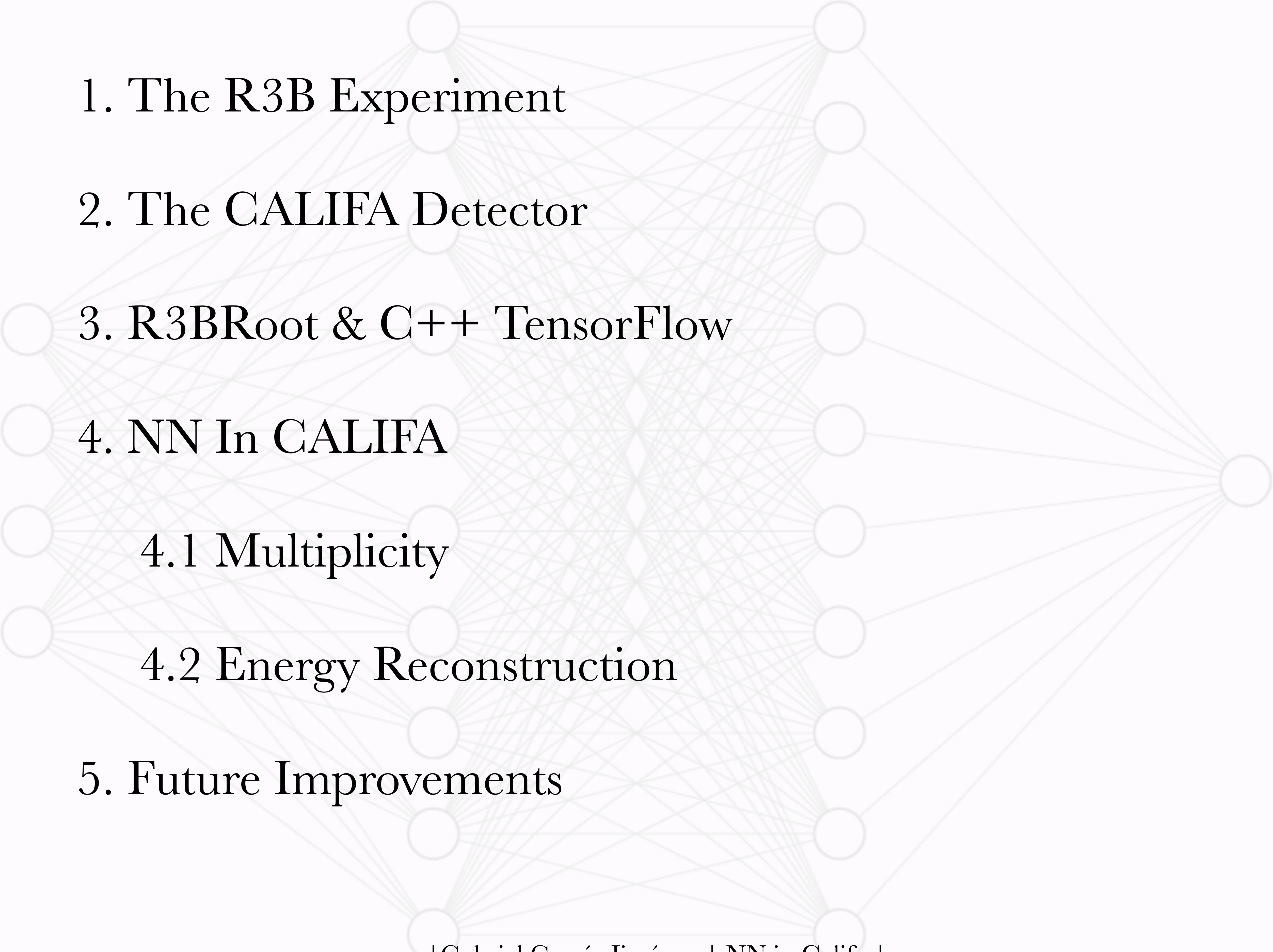


USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

IGFAE
Instituto Galego de Física de Altas Enerxías

GENP
Grupo Experimental de Núcleos y Partículas

Gabriel García Jiménez



1. The R3B Experiment

2. The CALIFA Detector

3. R3BRoot & C++ TensorFlow

4. NN In CALIFA

    4.1 Multiplicity

    4.2 Energy Reconstruction

5. Future Improvements

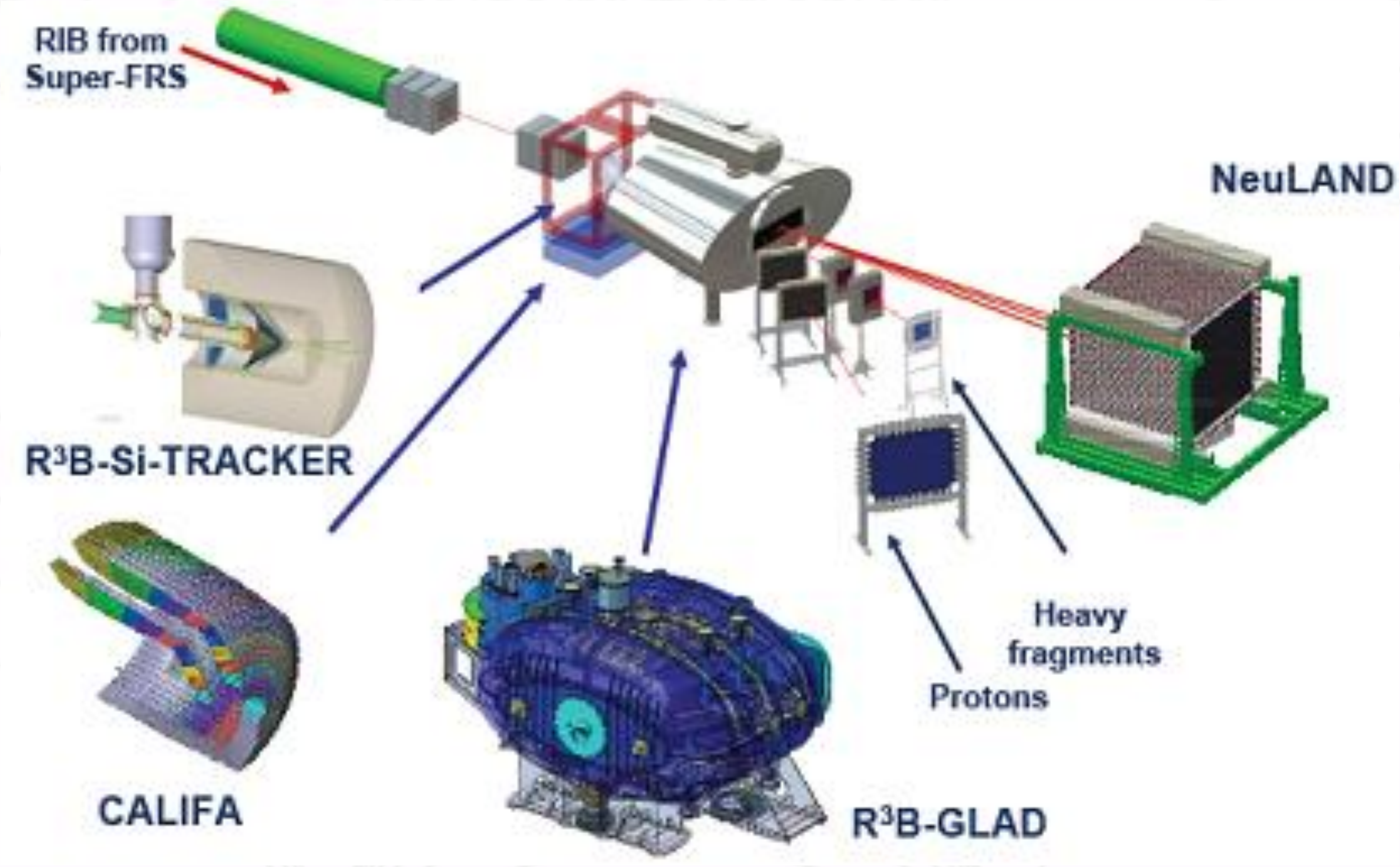
The R3B Experiment

- High Energy Beam: Up to 1000 AMeV (|p| ~ 400 GeV/c for 238U)

- Inverse Kinematics

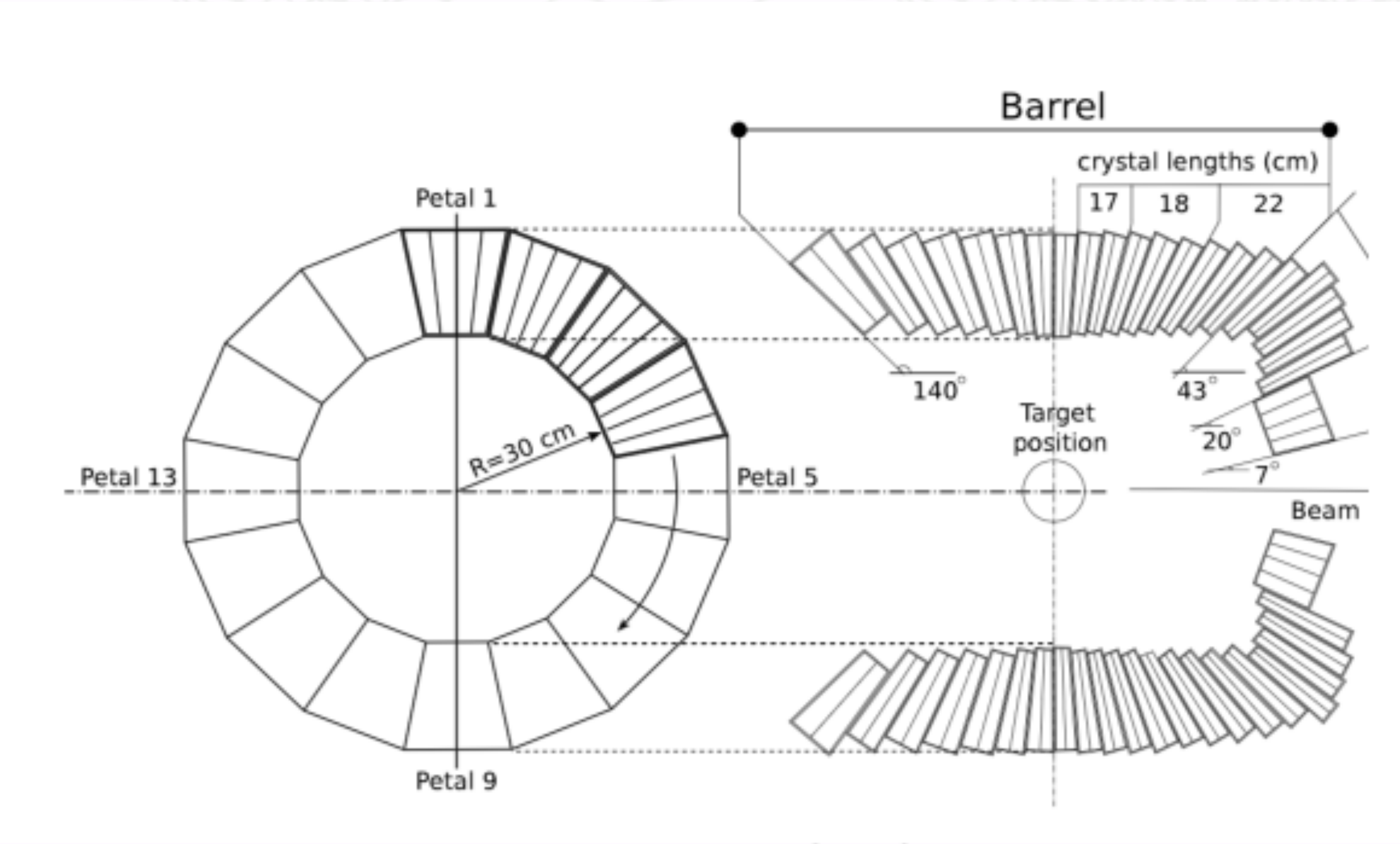
- Measurements in a Complete Kinematic Regime

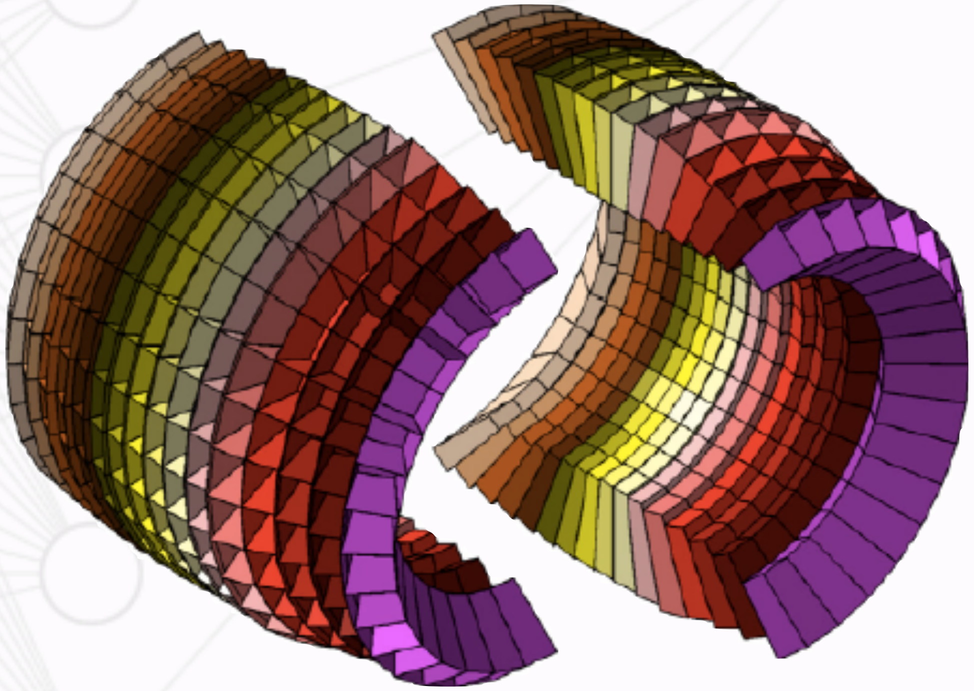
- Topics: Nuclear Structure (p2p, p2pn, fission procceses, exotic nuclei structure)



The CALIFA Detector

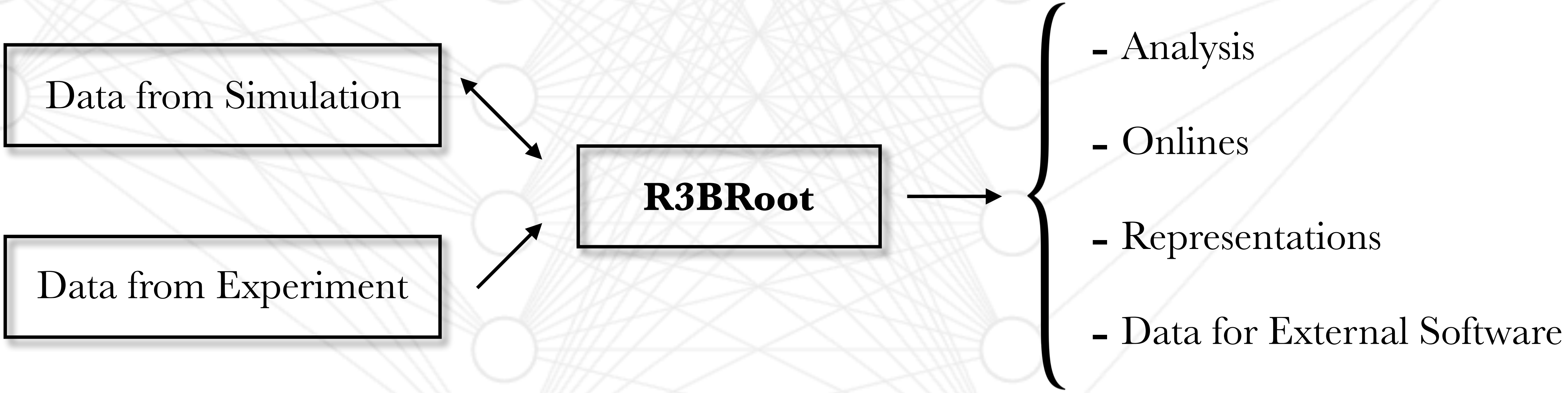
- 2432 CsI(Tl) Crystals
- Barrel : 1952 Crystals, covering $\theta = [43^{o},140^{o}]$
- Iphos: 480 Crystals, covering $\theta = [20^{o}, 43^{o}]$
- Gamma Energy Resolution: R ~ 5% - 6% (1 MeV)
- Light Particles Resolution: 1 % (protons @330 MeV)





R3BRoot Highlights

- Written in C++ (Object Oriented)
- Based on Root
- Uses Geant4 as particle transport engine
- Every operation involving any kind of data is performed in *Tasks*
- Framework manages both simulations and real data



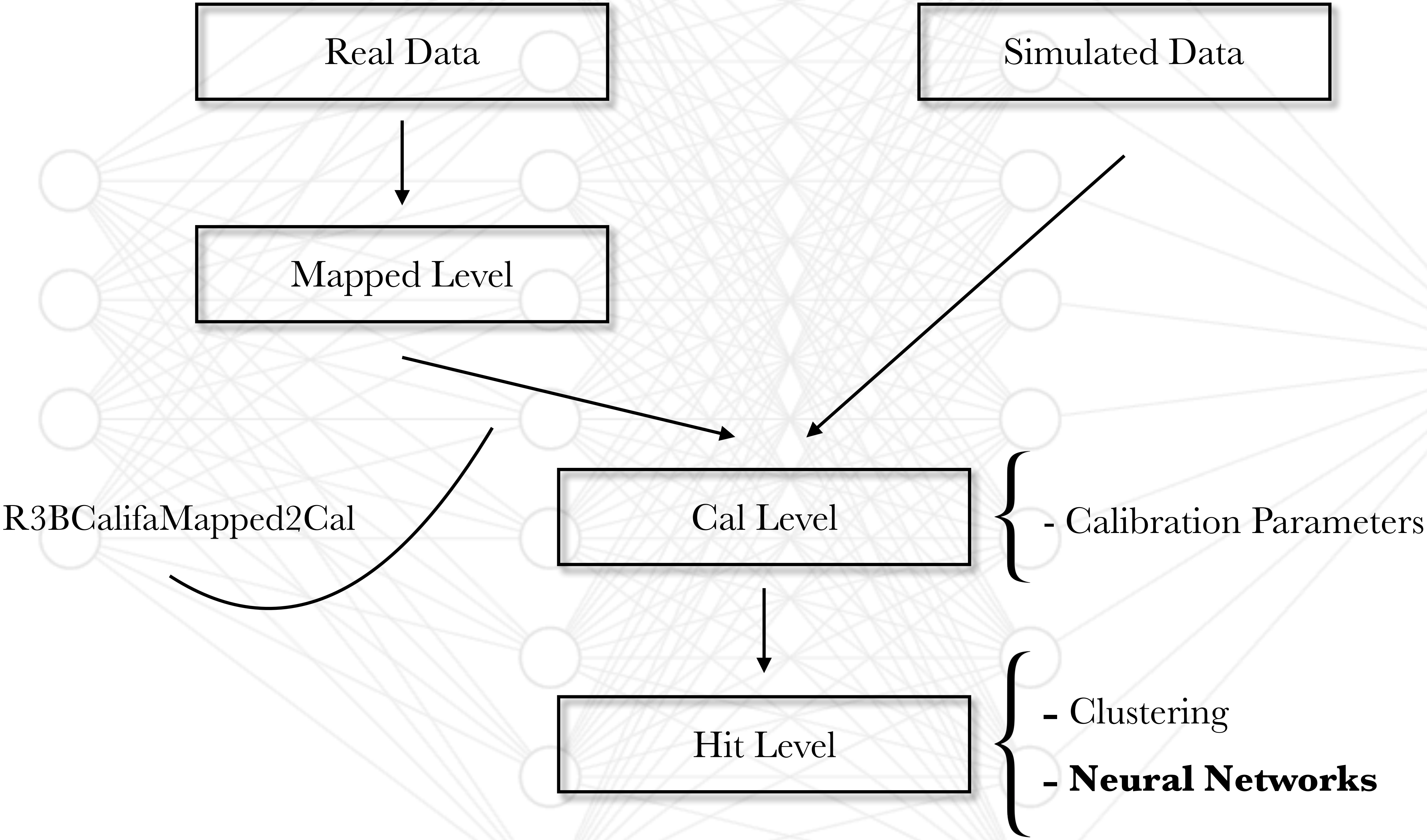
Data from Simulation

Data from Experiment

R3BRoot

- Analysis
- Onlines
- Representations
- Data for External Software

Data Levels in CALIFA



Real Data

Simulated Data

Mapped Level

R3BCalifaMapped2Cal

Cal Level

- Calibration Parameters

Hit Level

- Clustering
- **Neural Networks**

C ++ TensorFlow

As R3BRoot is written entirely in C++ , our NN framework must be developed in the same language!

- Not needed for a later analysis of data.
- Needed for a built-in tool, merged with the current set of algorithms of R3BRoot.

Comparison with common high-end frameworks in Python (*ie* Keras, Pytorch…)

- Faster. Very low level models **+** Intrinsic speed of C++
- Highly customizable models. You can do whatever you want with weights and biases at any point during training, validation or test
- More dificult to develop than its Python counterparts. Less resources !!
- Raw TF has its own types and syntax (very different from Std C++)
- Very tricky installation (for anyone interested: https://github.com/gabrigarjim/TensorFlowCMake/wiki)

NN in CALIFA: Standard Inputs

Raw Inputs

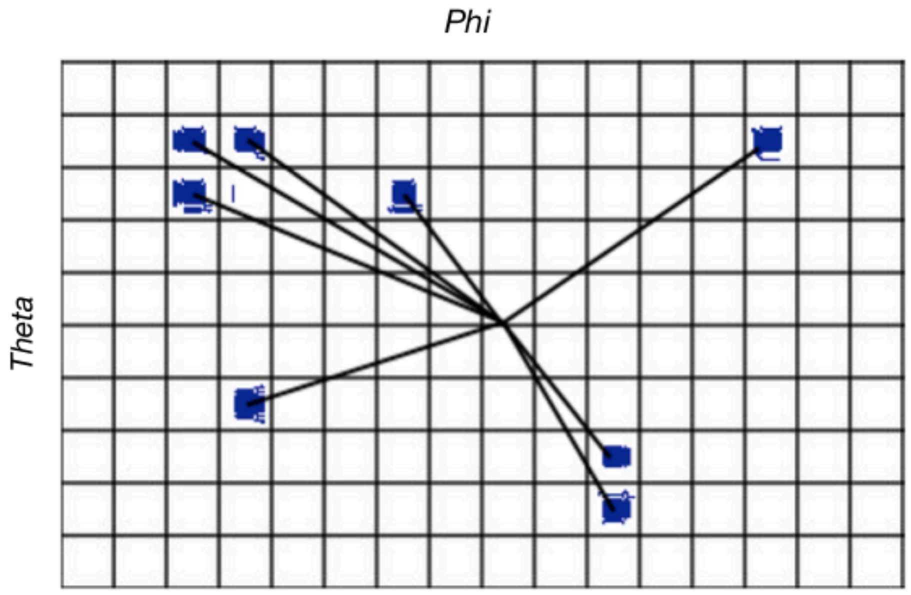
- **Matrix form:** Tensor with shape {batch,m,n}. Dificult to implement, but spatial correlations are intrisincally defined.

- **Vector form:** Tensor with shape {batch,1, 2432}. Spatial correlations are not very clear (or lost). Very easy to translate events into NN inputs.

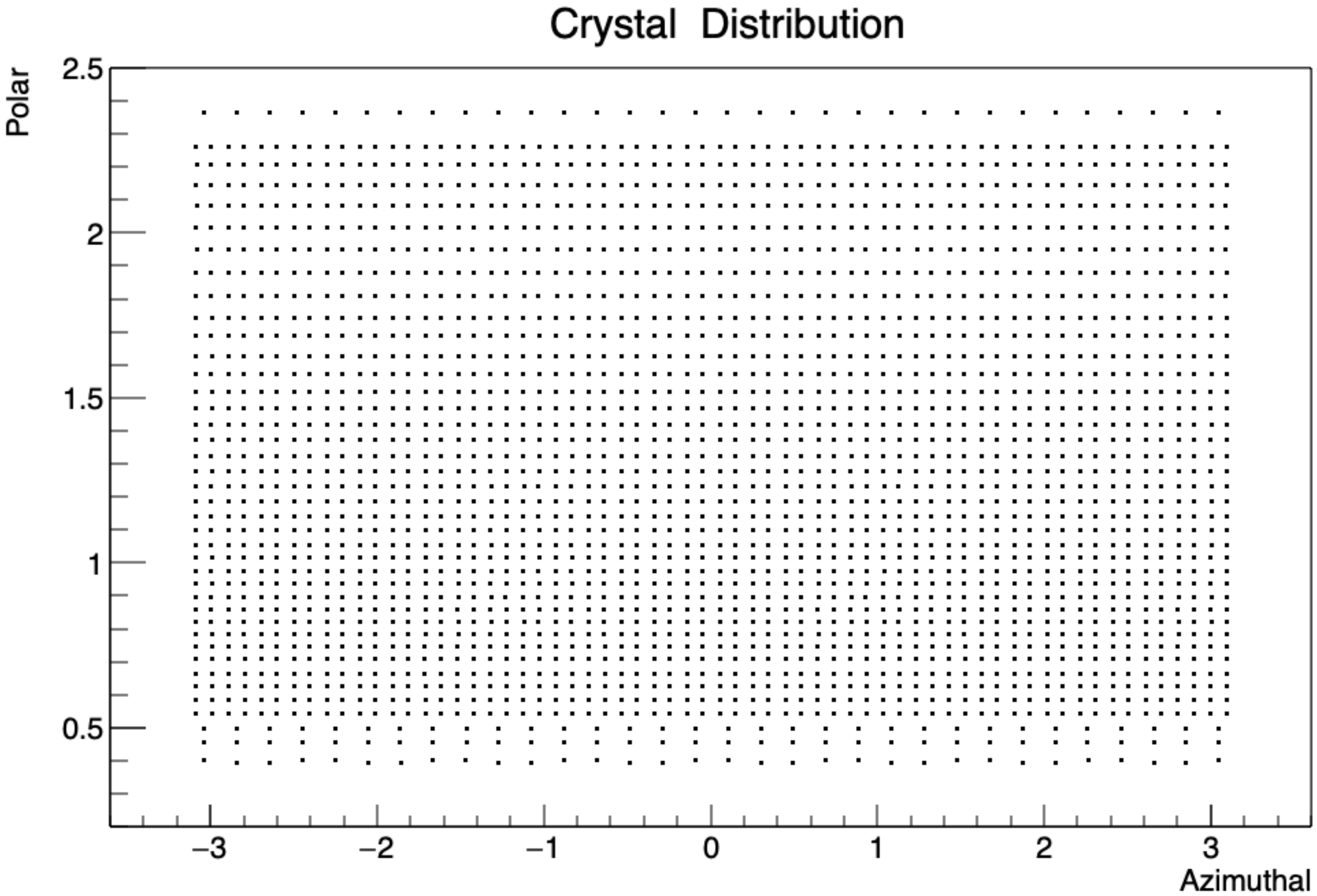
"Manufactured Inputs"

Each Event has *m* characteristics. Some examples:

- Calorimetric Sum
- Number of Hits
- Angular and energy dispersion, Angular distance
- Mixing with Clusters
- Fast and Slow components of light
- Every combination of the previous variables



NN in CALIFA: Standard Inputs



NN in CALIFA: Multiplicity

For each nuclear reaction, several particles can be generated ⟶ **Multiplicity**

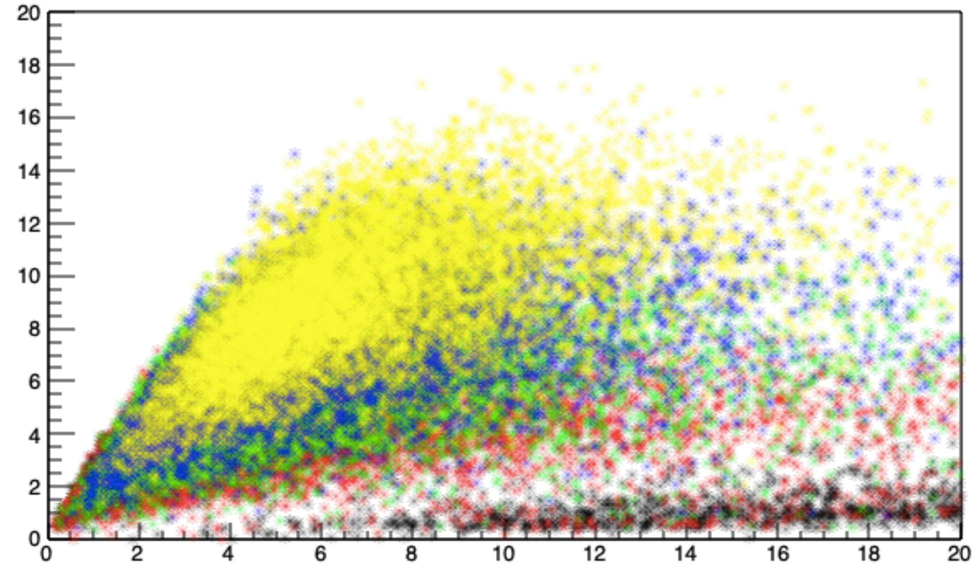
	$e_{i,j}$	$e_{i,j+1}$			
	$e_{i+1,j}$				
		$e_{i+3,j+1}$			

Some problems! One primary + Compton? Two primaries?? Four???

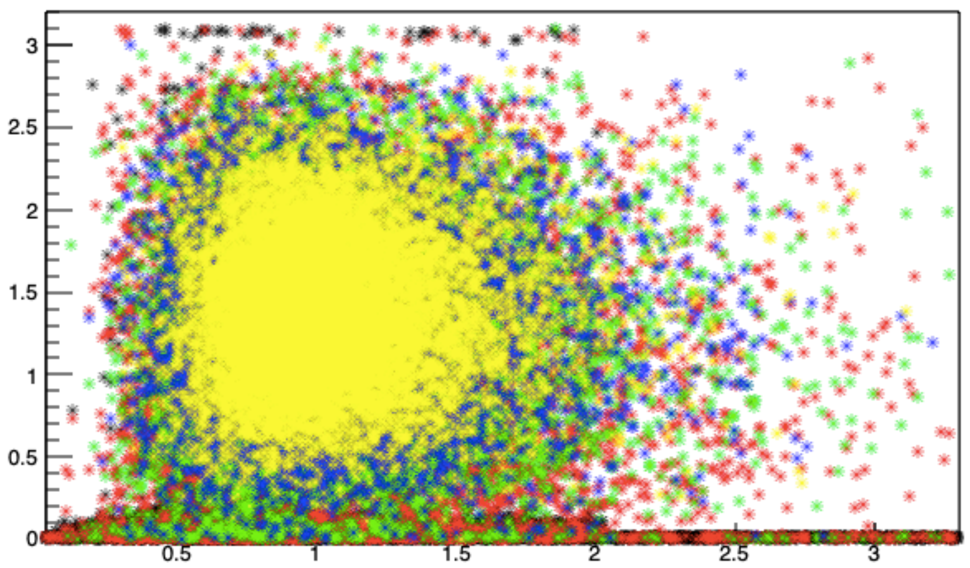
NN in CALIFA: Multiplicity

Color Code: M1 - Black, M2 - Red, M3 - Green, M4 - Blue, M5 - Yellow

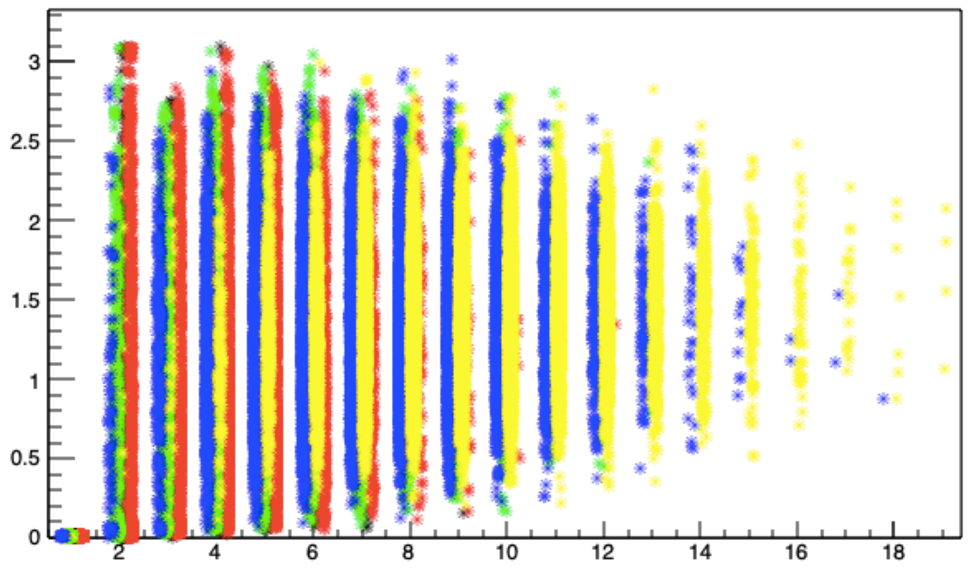
EnergySum/Distance Vs EnergySum



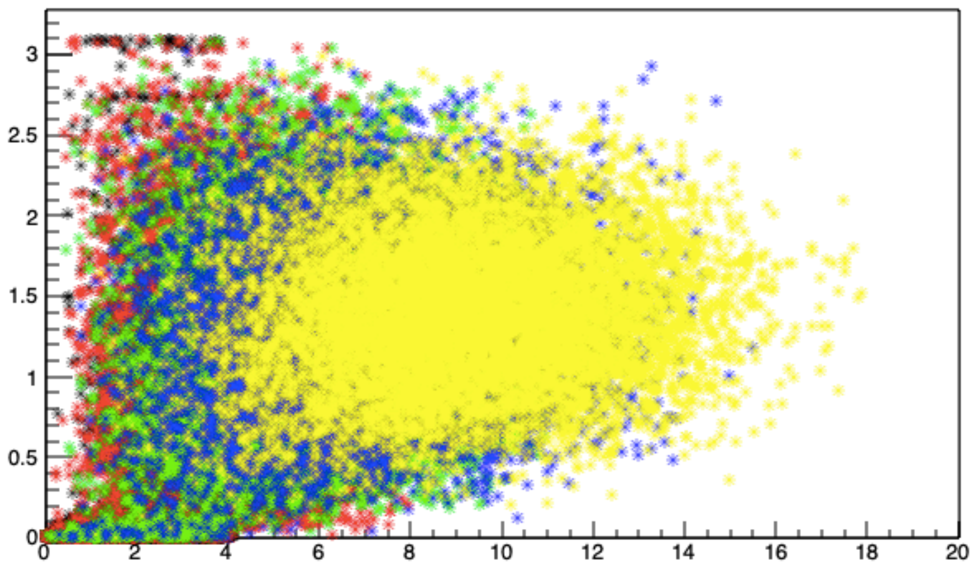
EnergySum / #Hits Vs Distance



Number of Hits Vs Distance

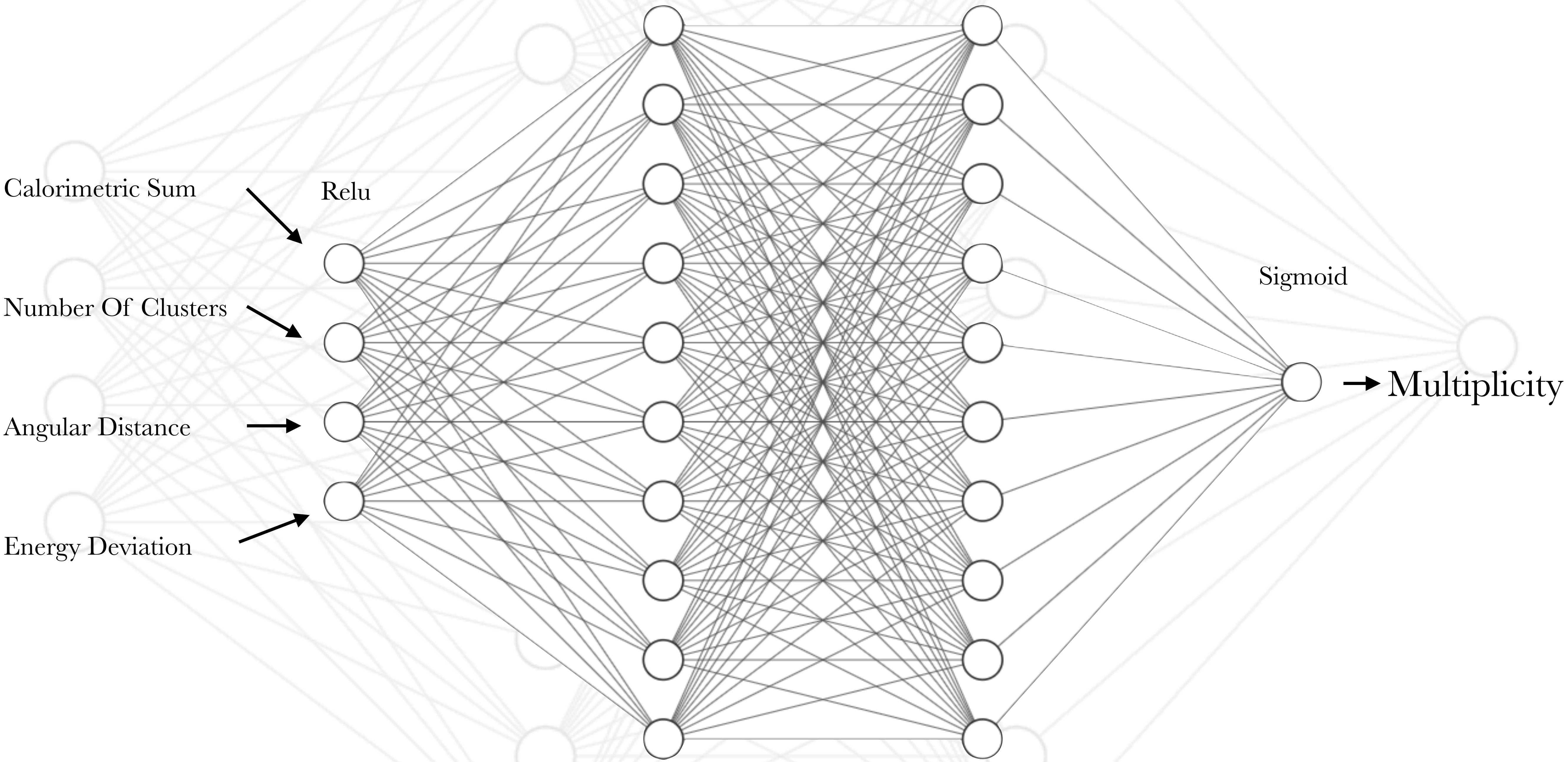


Energy Sum Vs Distance



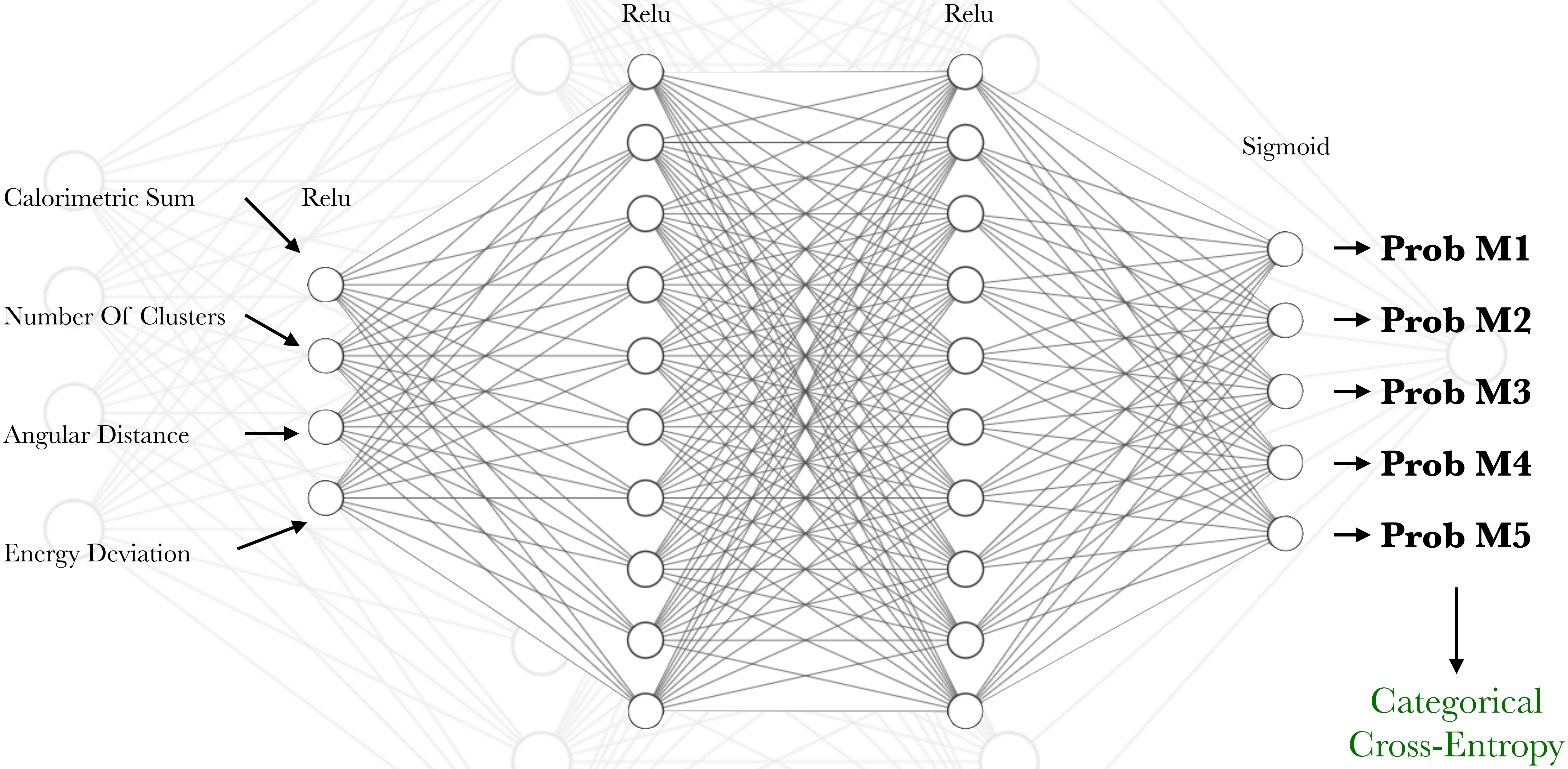
Highly Non-linear Problem!!

NN in CALIFA: Multiplicity



Model Example with 150 parameters.

NN in CALIFA: Multiplicity



Calorimetric Sum

Number Of Clusters

Angular Distance

Energy Deviation

Relu

Relu

Relu

Sigmoid

→ **Prob M1**

→ **Prob M2**

→ **Prob M3**

→ **Prob M4**

→ **Prob M5**

Categorical Cross-Entropy

Model Example with 190 parameters.

NN in CALIFA: Multiplicity

3780 Parameters, Two Hidden Layers

Neural Network

Simulated →

Reconstructed ↓

	M1	M2	M3	M4	M5
M0	11.20 %	1.96 %	0.32 %	0.07 %	0.00 %
M1	82.42 %	22.00 %	4.38 %	0.96 %	0.14 %
M2	6.10 %	61.32 %	22.46 %	6.20 %	1.16 %
M3	0.28 %	13.58 %	51.40 %	24.21 %	6.94 %
M4	0.00 %	1.06 %	19.00 %	43.24 %	24.5 %
M5	0.00 %	0.08 %	2.44 %	25.32 %	67.26 %

Classic Algorithm (Cluster)

Simulated →

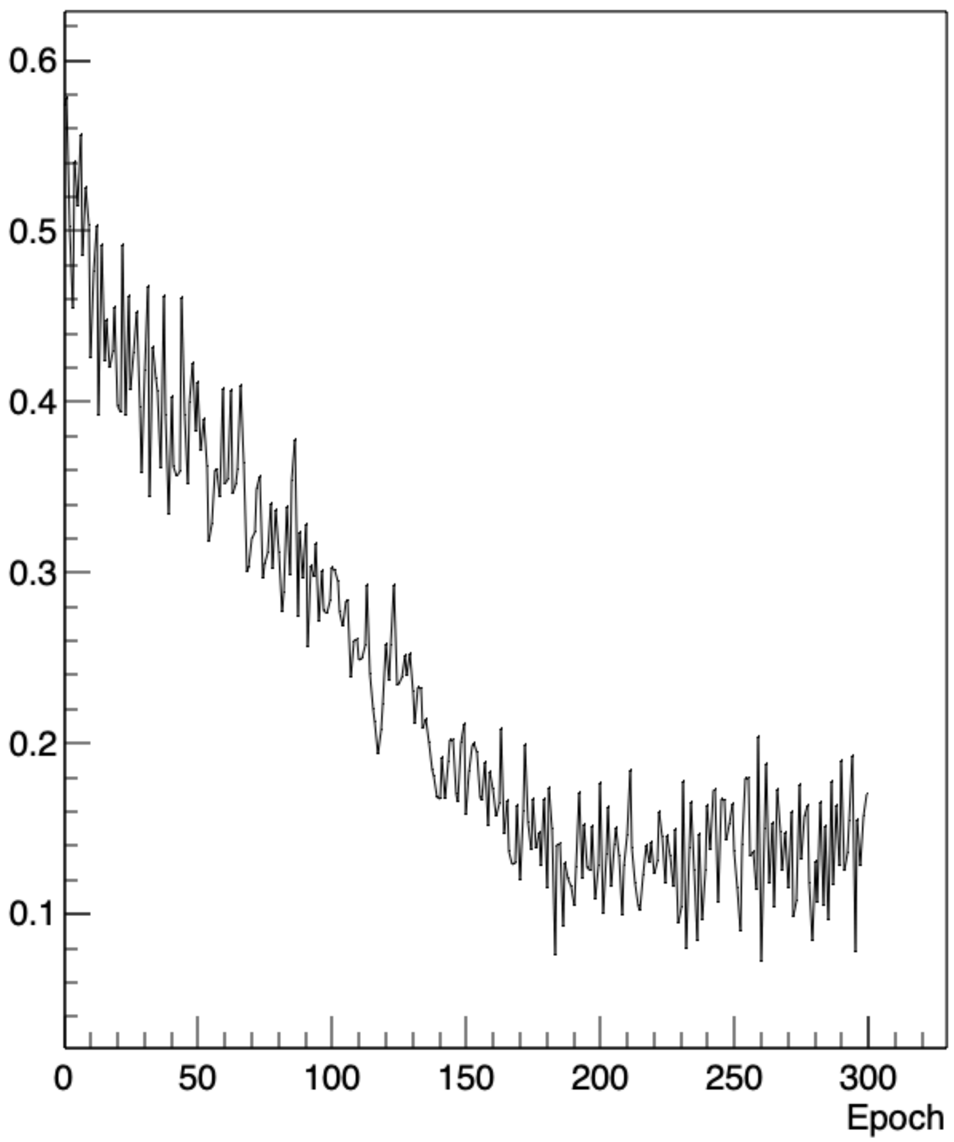
Reconstructed ↓

	M1	M2	M3	M4	M5
M0	9.88 %	1.22 %	0.28 %	2.40 %	0.25 %
M1	81.14 %	18.76 %	3.18 %	0.52 %	0.31 %
M2	7.18 %	67.54 %	23.18 %	5.90 %	0.54 %
M3	0.48 %	11.16 %	57.94 %	27.98 %	12.50 %
M4	1.22 %	1.22 %	13.36 %	49.9 %	35.81 %
M5	0.0 %	0.1 %	2.06 %	13.30 %	50.59 %

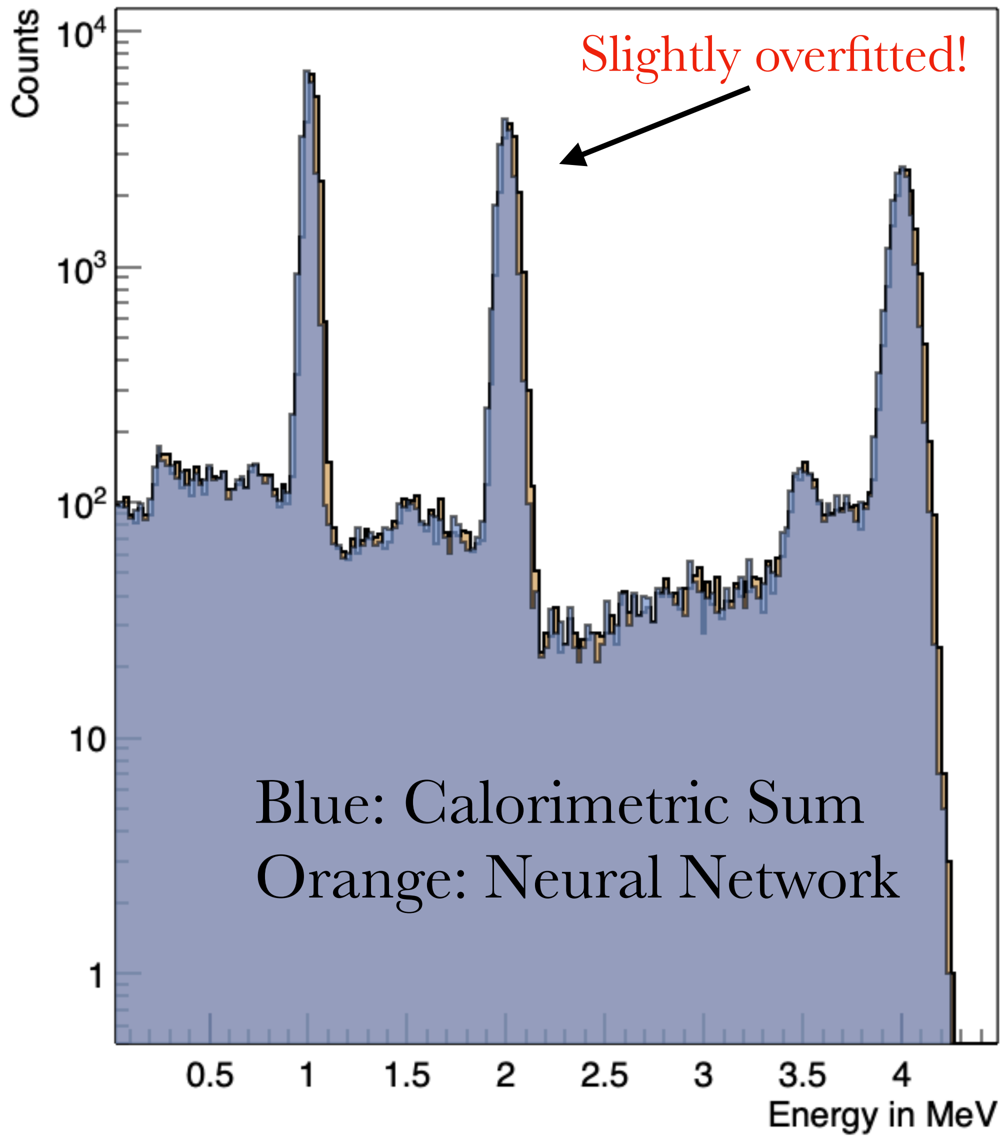
NN in CALIFA: Energy

- **Input: Raw Crystal Array**
- **12.186 Parameters, One Hidden Layer**
- **Three Discrete Energies as Testing sample: 1 MeV , 2 MeV and 4 MeV**

Loss



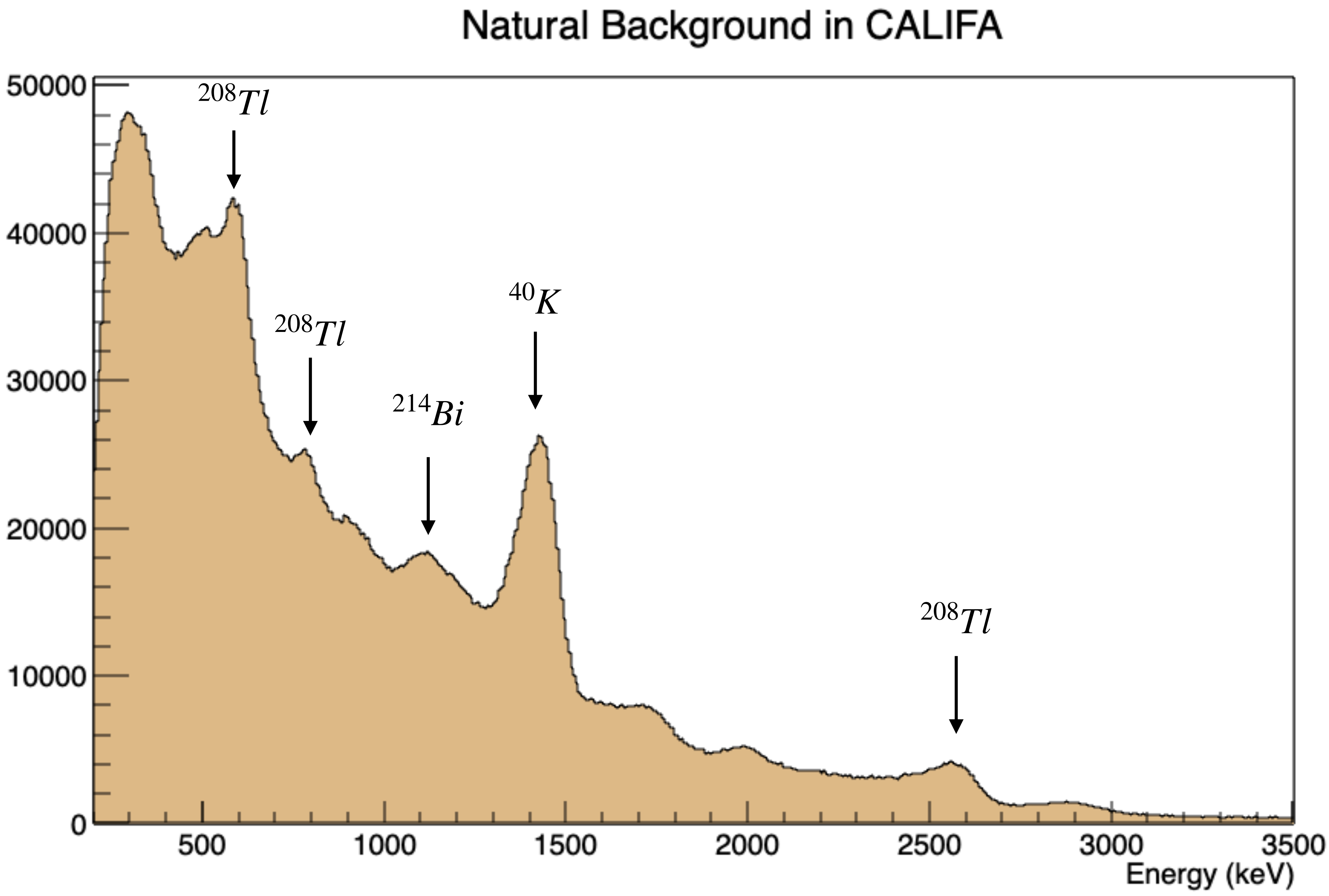
Predicted Energy



Slightly overfitted!

Blue: Calorimetric Sum
Orange: Neural Network

To be Done….


Natural Background in CALIFA
^{208}Tl
^{208}Tl
^{214}Bi
^{40}K
^{208}Tl
50000
40000
30000
20000
10000
0
500
1000
1500
2000
2500
3000
3500
Energy (keV)

To be Done….

- Real Thresholds + Resolutions + Background = Hyper Realistic Sim Events for our NN Models
- A lot of features of Python TF are not implemented yet in TF C++
- Convolutional Layers
- PID

Thanks for your Attention!!