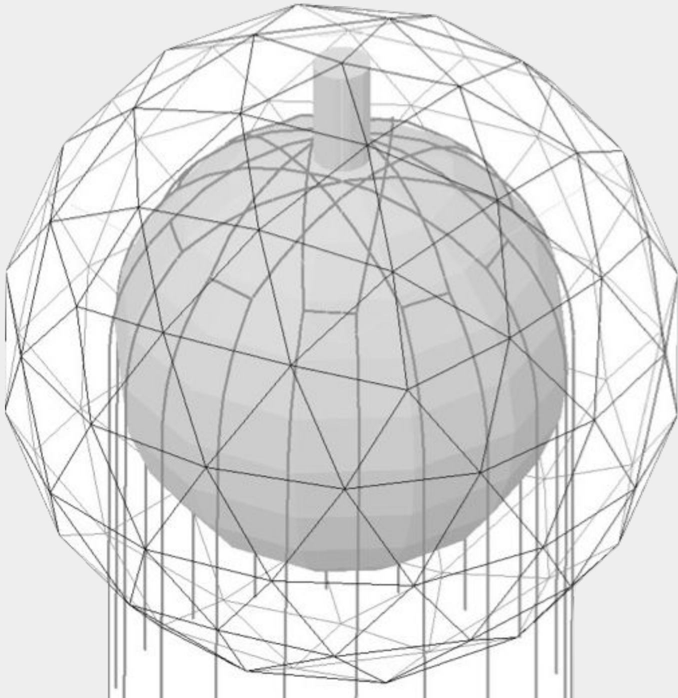
Machine Learning Project

SNO+ Neutrinos Detector

Carlos Roxo
Carolina Fernandes
Oriented by: Nuno Barros
Advanced Data Analysis Techniques

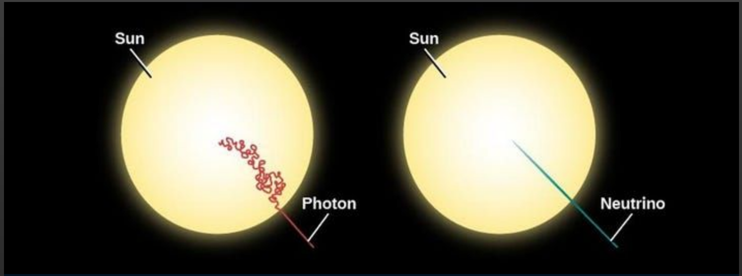


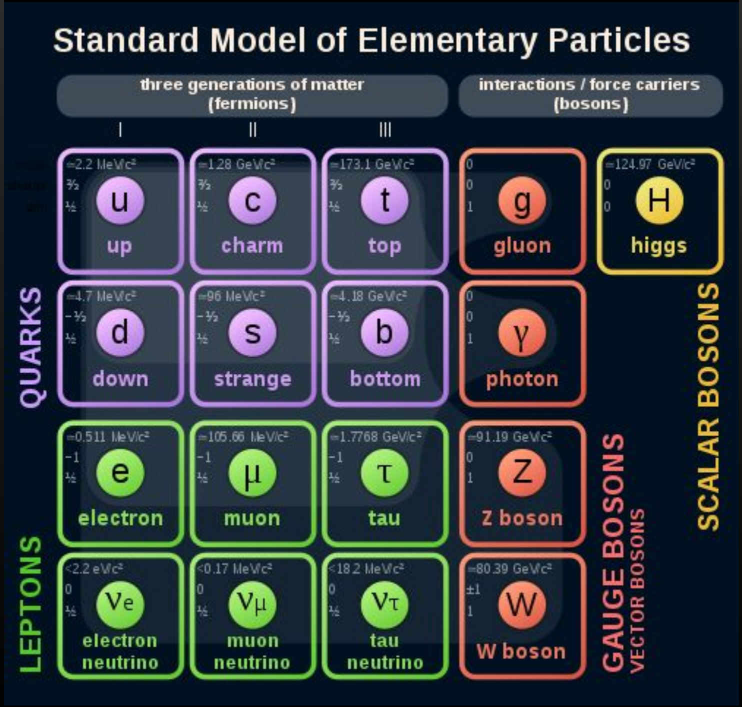
Presentation Topics:

- What are neutrinos?
- SNO+ detector components and characteristics
- Data analysis
- Data pre-processing
- Training Model
- Results

What are neutrinos?

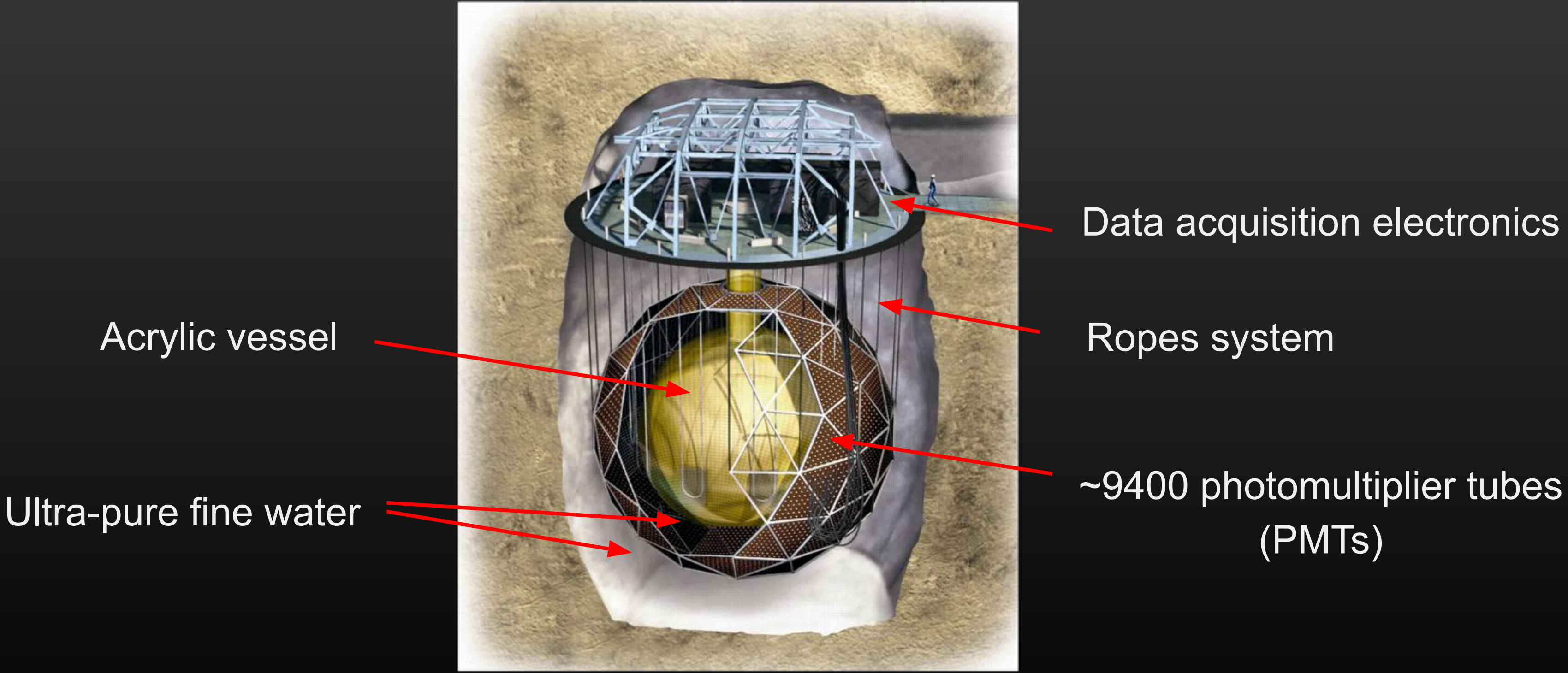
- Uncharged particles
  - They don't interact electromagnetically
- Interact very weakly
- Could be their own antiparticles
- Very light particles
- Appears in three different ways
  - electron neutrino (v_e)
  - muon neutrino (v_μ)
  - tau neutrino (v_τ)





Standard Model of Elementary Particles
three generations of matter (fermions)
interactions / force carriers (bosons)
I
II
III
≈2.2 MeV/c² ⅔ ½ u up
≈1.28 GeV/c² ⅔ ½ c charm
≈173.1 GeV/c² ⅔ ½ t top
0 0 1 g gluon
≈124.97 GeV/c² 0 0 H higgs
≈4.7 MeV/c² −⅓ ½ d down
≈96 MeV/c² −⅓ ½ s strange
≈4.18 GeV/c² −⅓ ½ b bottom
0 0 1 γ photon
≈0.511 MeV/c² −1 ½ e electron
≈105.66 MeV/c² −1 ½ μ muon
≈1.7768 GeV/c² −1 ½ τ tau
≈91.19 GeV/c² 0 1 Z Z boson
<2.2 eV/c² 0 ½ νe electron neutrino
<0.17 MeV/c² 0 ½ νμ muon neutrino
<18.2 MeV/c² 0 ½ ντ tau neutrino
≈80.39 GeV/c² ±1 1 W W boson
QUARKS
LEPTONS
GAUGE BOSONS
VECTOR BOSONS
SCALAR BOSONS

SNO+: Detector Components and Characteristics


Data acquisition electronics
Acrylic vessel
Ropes system
~9400 photomultiplier tubes (PMTs)
Ultra-pure fine water

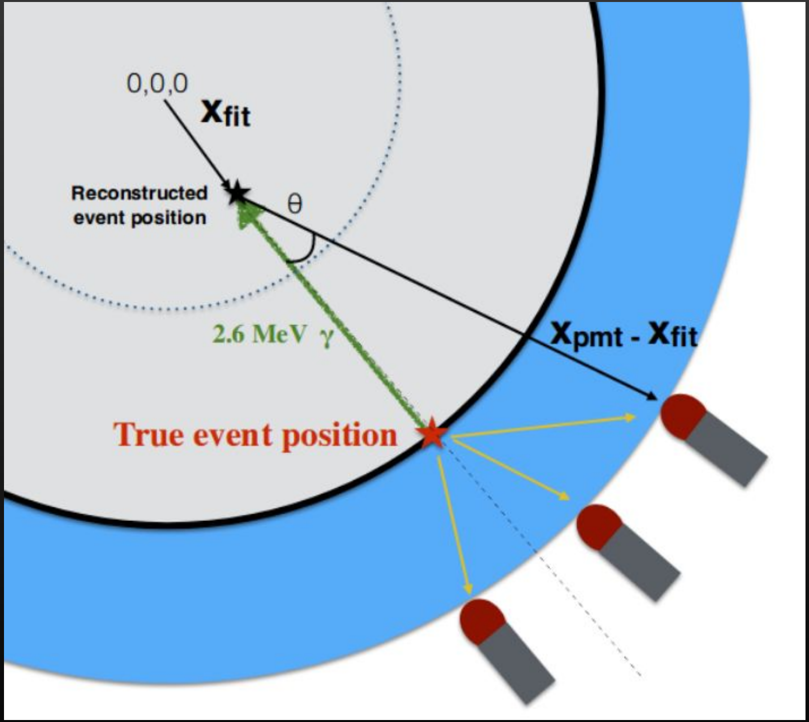
Machine learning project

Problems:

- Radioactive backgrounds from:
    - Natural radioactivity from the scintillator
    - Natural radioactivity from the detector components
    - Radioactivity from cosmic rays
- Appearance of alpha and gamma particles

Goal:

- Train a model that can distinguish between beta and gamma particles



0,0,0
$\mathbf{x}_{fit}$
Reconstructed event position
θ
2.6 MeV γ
$\mathbf{x}_{pmt} - \mathbf{x}_{fit}$
True event position

Data analysis

```
#Abrir os ficheiros de dados
data_e = uproot.open("snop_e.root")["data"]
data_g = uproot.open("snop_gamma.root")["data"]
print(data_e.show())
```

```
name                 | typename                 | interpretation
---------------------+--------------------------+-------------------
--
event                | int32_t                  | AsDtype('>i4')
hit                  | int32_t                  | AsDtype('>i4')
time                 | float                    | AsDtype('>f4')
x                    | float                    | AsDtype('>f4')
y                    | float                    | AsDtype('>f4')
z                    | float                    | AsDtype('>f4')
```

Data type: Root -> Ttree (read using uproot)

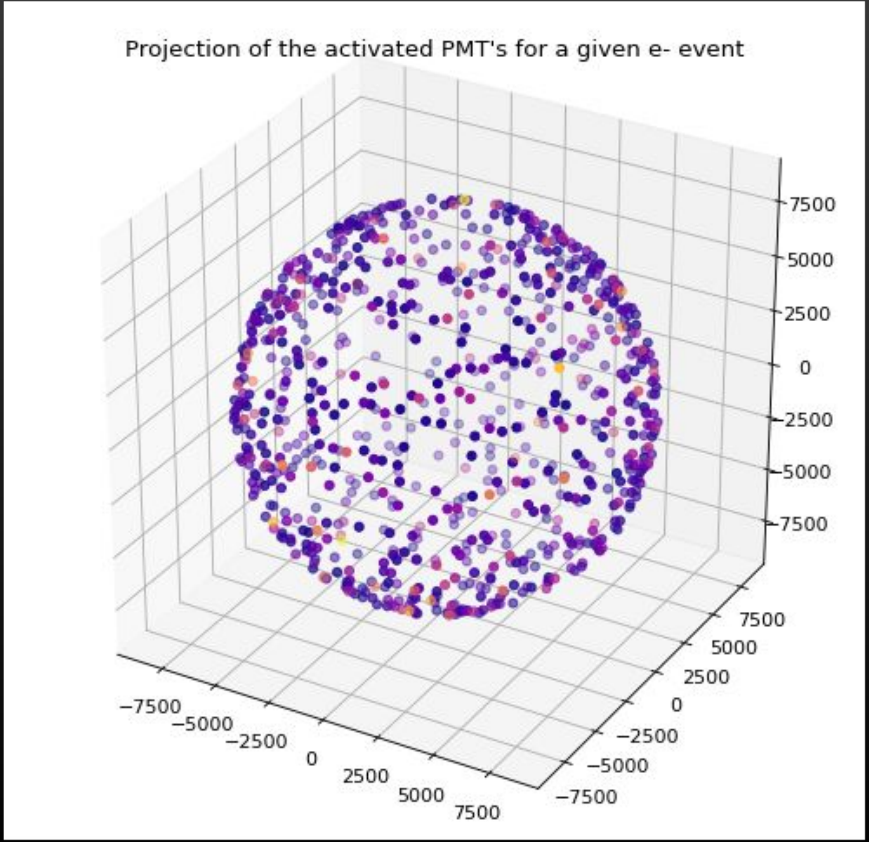
Each event was simulated on the center of the detector and with energy around 2.5 MeV

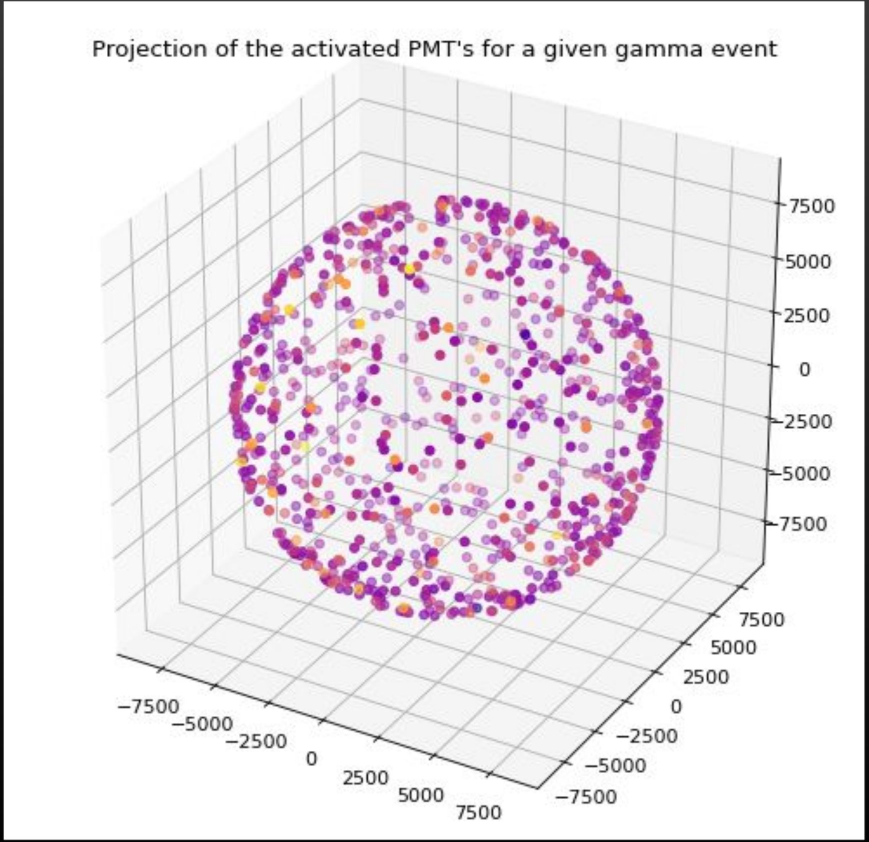
Data for 2 types of events (beta and gamma):

- Around 250k events recorded for each type
    - Around 800 hits each event
        - 6 data values each hit (the 4 later of each we´re going to use on the analysis)
            - Indexes (event, hit)
            - Position (x, y, z, time)

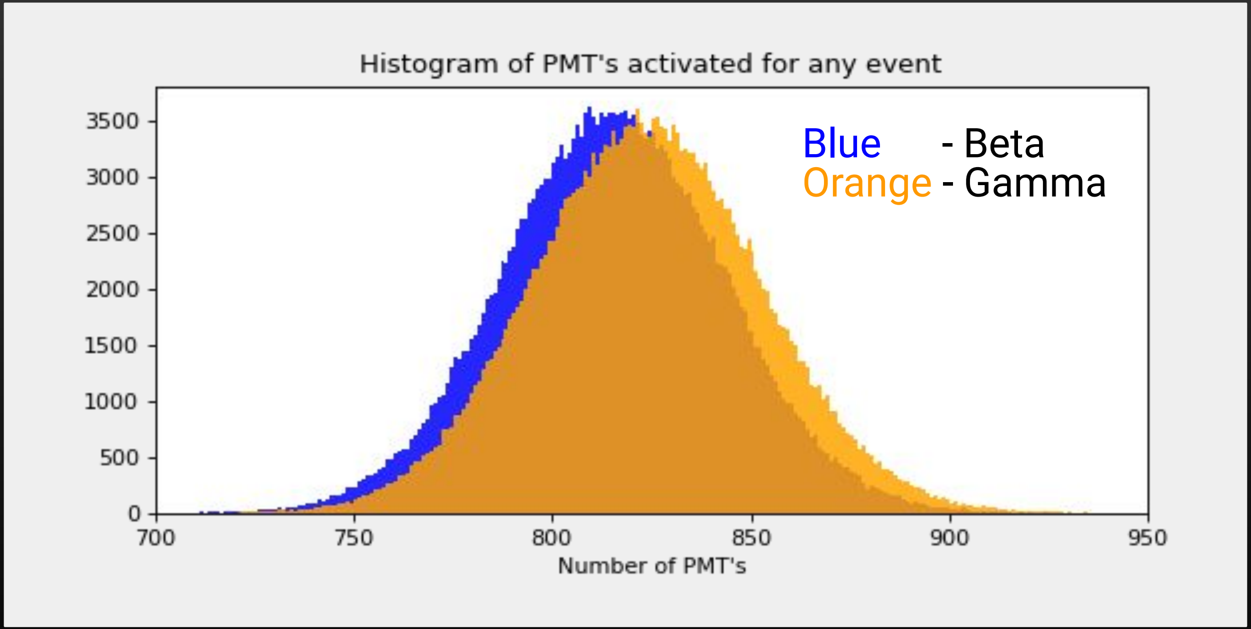
In total, around 1e+9 ints and 2e+9 floats of data (lots of data).

Illustration of enabled PMTs for a single beta and gamma event

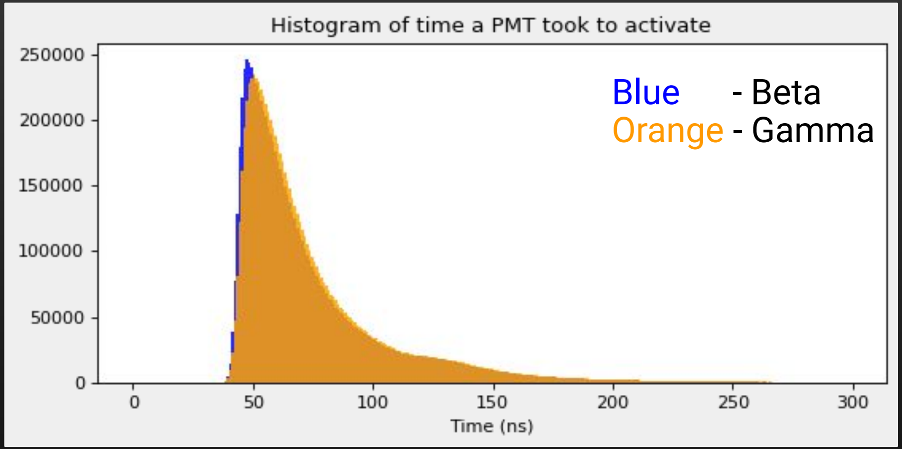


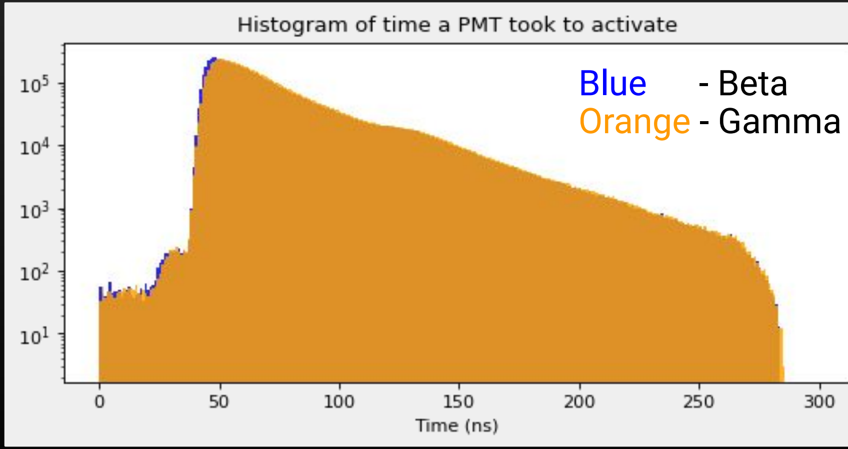


Number of hit PMT’s per event, integrated over all events



Distribution time of hit over all PMT over all events





Data pre-processing

- Changed data from Cartesian coordinates to spherical coordinates for better memory and processing power efficiency
- Data filtering (removed Nan’s)
- Quality cut in time, from around 30 to 70 ns (later hits are mostly due to re-triggers and indirect light)
- 80% of data for training and 20% for validation
- Data normalization

```
#-----------------------------------------------
r = math.hypot (x,y,z)
theta_e = np.arccos (z/r)
phi_e  = np.sign (y)  * np.arccos (x/sqrt (x**2+y**2))

#-----------------------------------------------
Tmed_e[np.where(np.isnan(Tmed_e)==True)] = 0
Pmt_e[np.where(np.isnan(Pmt_e)==True)]  = 0
Tmed_g[np.where(np.isnan(Tmed_g)==True)] = 0
Pmt_g[np.where(np.isnan(Pmt_g)==True)]  = 0

#-----------------------------------------------
X_train, X_test, y_train, y_test =
train_test_split (X,target,test_size=0.2,random_state=0)

#-----------------------------------------------
X_train = StandardScaler ().fit_transform (X_train)
```

AI Model - First Iteration

Model:

- 1 input neuron for each PMT (~9400 input neurons)
    - Hit time
- Some dense layers to let the model learn
- (Maybe some convolution layers?)
- 2 neurons as output
    - Beta
    - Gamma

Problem:

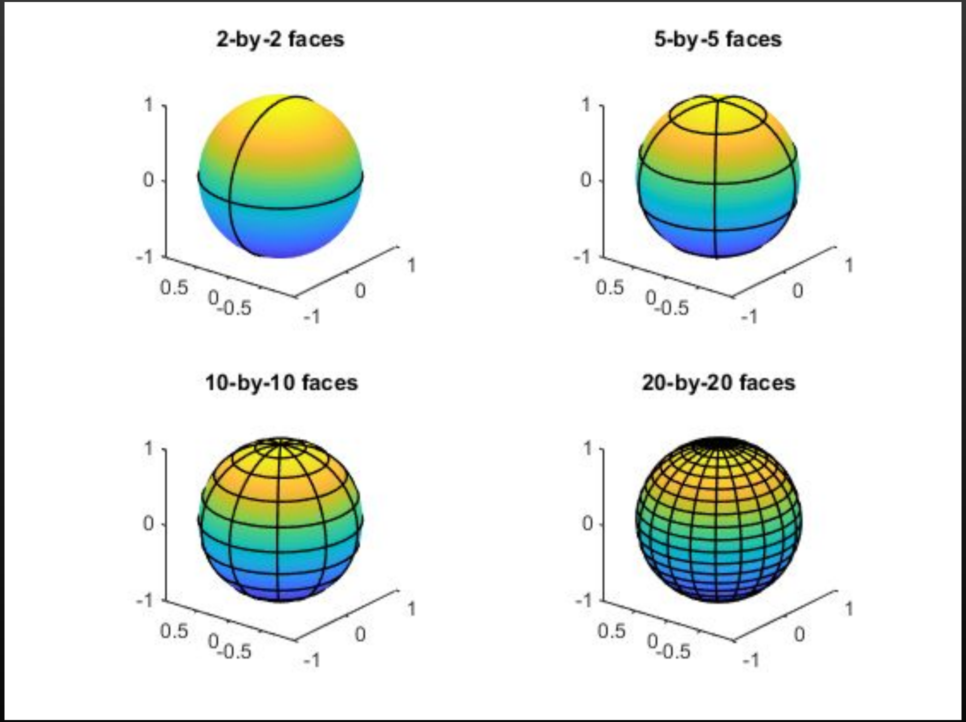
- Too intensive in processing power
- Inefficient
- Since there are around 800 hits per event, most of the input neurons aren’t used for a single event
- No need to know what PMT’s activated in specific due to the spherical symmetry of the system

```
#------------------------------------------------
model = tf.keras.models.Sequential ([
  tf.keras.layers.Flatten (),
  tf.keras.layers.Dense (128,
activation=tf.nn.relu ),
  tf.keras.layers.Dense (128,
activation=tf.nn.relu ),
  tf.keras.layers.Dense (2,
activation=tf.nn.softmax )
])
opt = tf.keras.optimizers.Adam (learning_rate= 0.001)
model.compile (optimizer=opt ,
              loss= 'mean_squared_error' ,
              metrics= ['accuracy' ])
```

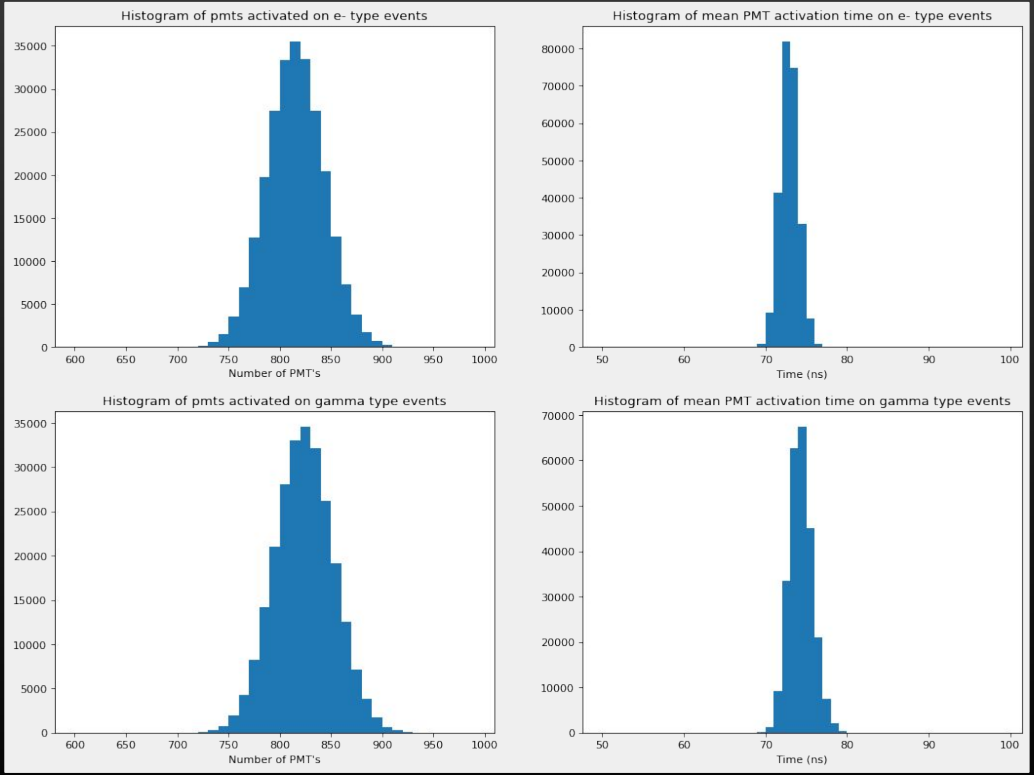
AI Model - Second Iteration

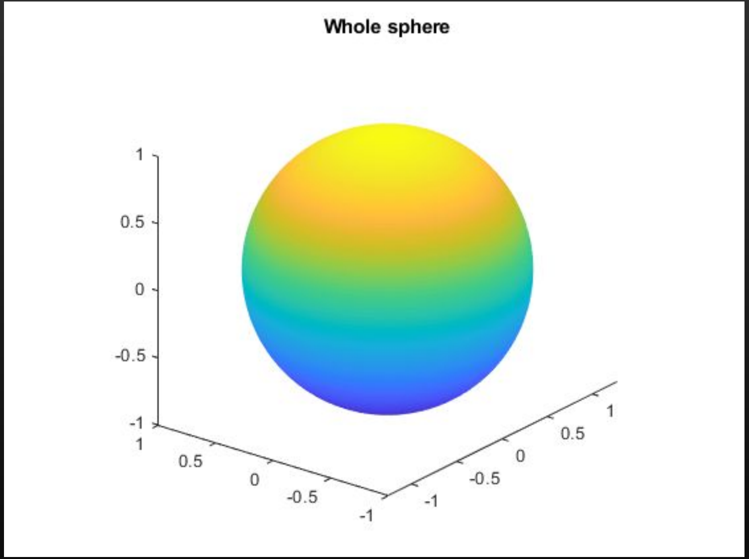
Model:

- Data grouped in regions
- 2 input neuron for each division of the sphere
    - Mean hit time of that region
    - number of PMT’s activated on that region
- Dense layers
- 2 neurons as output
    - Beta
    - Gamma

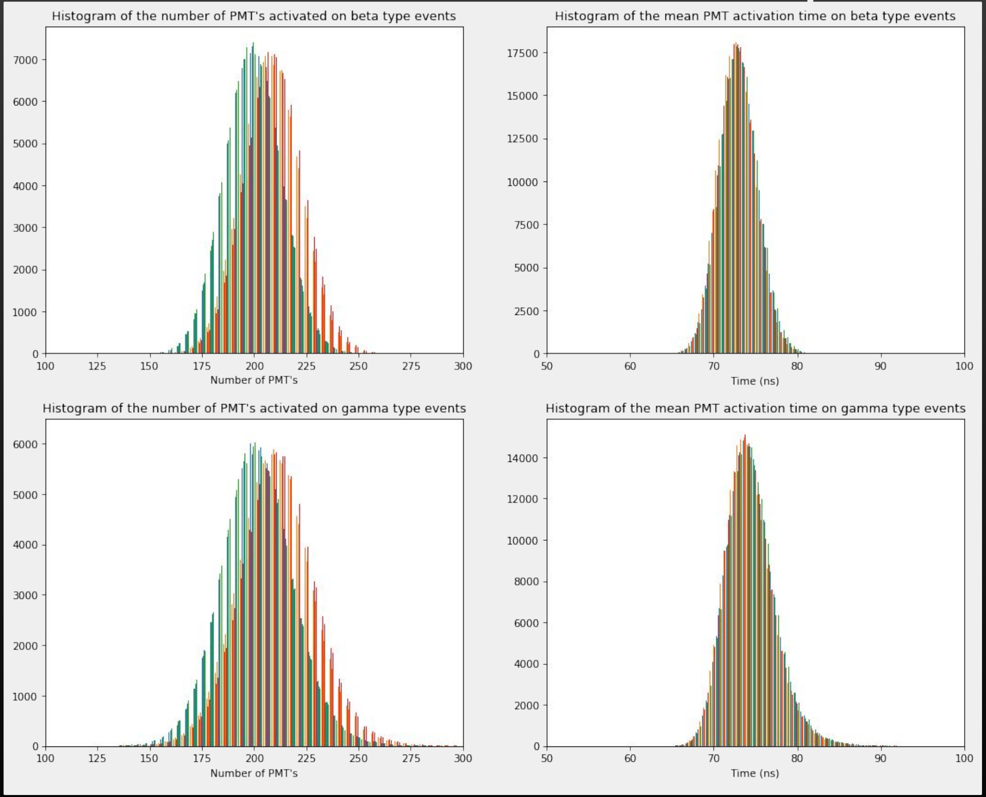


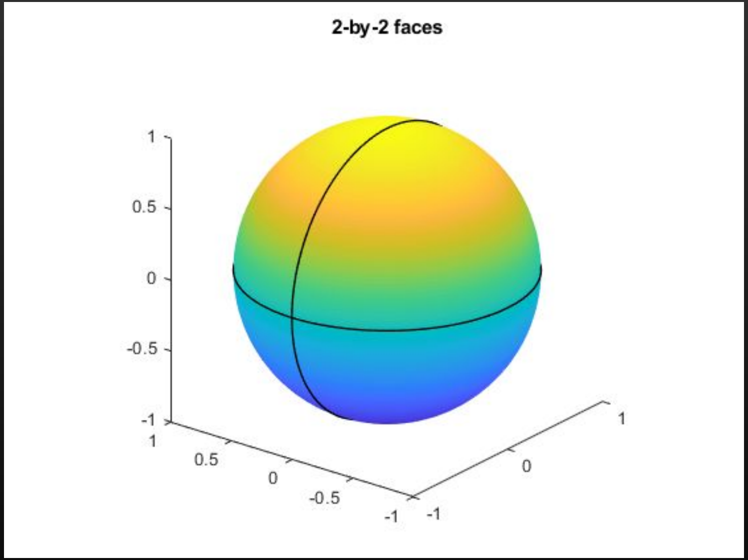
Number of hit PMT’s and mean time with half spheres



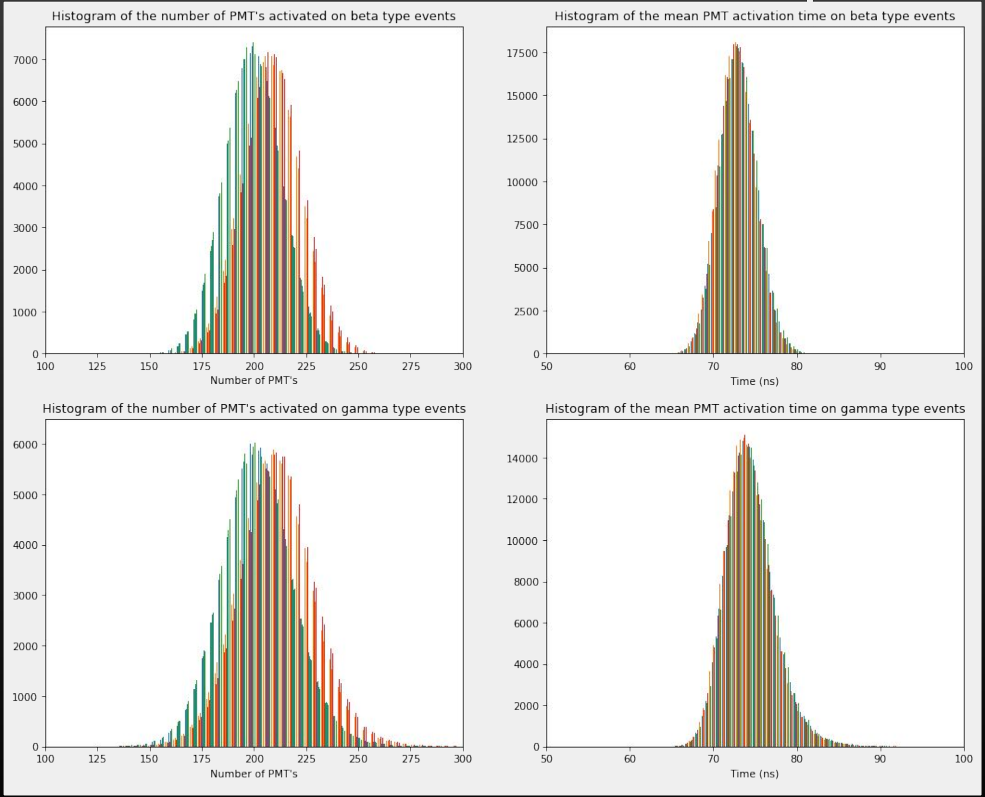


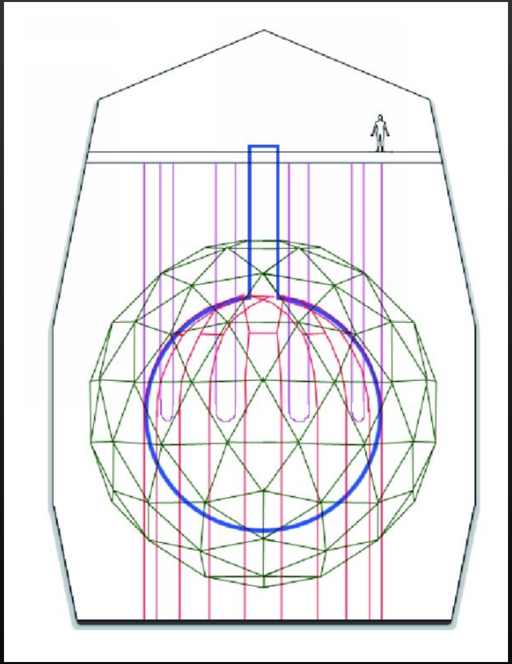
Number of PMT’s activated and mean time with quarter spheres





Number of PMT's activated and mean time with quarter spheres





Results

```
model.fit(xs, ys, epochs=10)
model.evaluate(np.array(X_test, dtype=float),  np.array(y_test,
dtype=float))


Epoch 1/10
12501/12501 [==============================] - 33s 3ms/step - loss:
0.2500 - accuracy: 0.5009
Epoch 2/10
12501/12501 [==============================] - 32s 3ms/step - loss:
0.2500 - accuracy: 0.4989
Epoch 3/10
12501/12501 [==============================] - 34s 3ms/step - loss:
0.2500 - accuracy: 0.4997
Epoch 4/10
12501/12501 [==============================] - 32s 3ms/step - loss:
0.2500 - accuracy: 0.4995
Epoch 5/10
12501/12501 [==============================] - 32s 3ms/step - loss:
0.2500 - accuracy: 0.4997
Epoch 6/10
12501/12501 [==============================] - 32s 3ms/step - loss:
0.2500 - accuracy: 0.5005
Epoch 7/10
12501/12501 [==============================] - 32s 3ms/step - loss:
0.2500 - accuracy: 0.4999
Epoch 8/10
12501/12501 [==============================] - 36s 3ms/step - loss:
0.2500 - accuracy: 0.5007
Epoch 9/10
12501/12501 [==============================] - 33s 3ms/step - loss:
0.2500 - accuracy: 0.5000
Epoch 10/10
12501/12501 [==============================] - 33s 3ms/step - loss:
0.2500 - accuracy: 0.5000
3126/3126 [==============================] - 6s 2ms/step - loss:
0.5000 - accuracy: 0.5000
[0.49999749660491943, 0.5000150203704834]
```

```
for i in range(10):
  a = model.predict ( [X_train[i]])
  b = y_train[i]
  print(np.round(a,3))
  print(b)


1/1 [==============================] - 0s 116ms/step
[0.289 0.711]
1
1/1 [==============================] - 0s 34ms/step
[0.654 0.346]
1
1/1 [==============================] - 0s 32ms/step
[0.704 0.296]
0
1/1 [==============================] - 0s 36ms/step
[0.051 0.949]
1
1/1 [==============================] - 0s 36ms/step
[0.553 0.447]
0
1/1 [==============================] - 0s 34ms/step
[0.677 0.323]
0
1/1 [==============================] - 0s 50ms/step
[0.478 0.522]
1
1/1 [==============================] - 0s 38ms/step
[0. 1.]
1
1/1 [==============================] - 0s 45ms/step
[0.694 0.306]
1
1/1 [==============================] - 0s 38ms/step
[0.613 0.387]
0
```

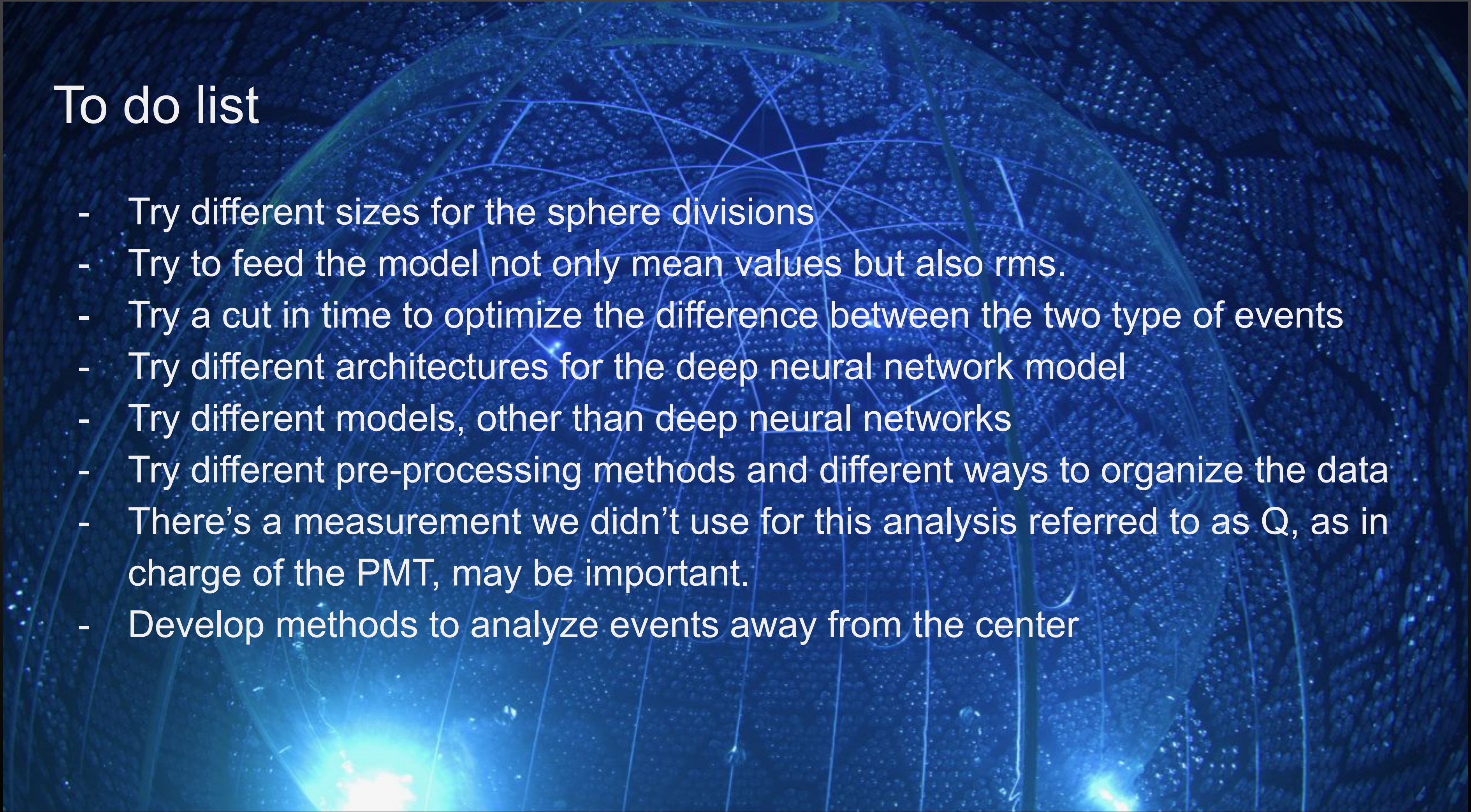
Discussion of results

Conclusions:

- Seems the model wasn't able to discriminate the events, it wasn't more accurate than flipping a coin. Reasons for this may be:

    - 4 pixels per event seems way too low
      The time distributions are too similar for the means of such a large group to hold any significant information

    - It's important to consider the geometry of the detector.
      The difference in the number of PMT's that are hit doesn't does not depend on the physics, i.e, is random (scintillation light is emitted isotropically)



To do list

- Try different sizes for the sphere divisions
- Try to feed the model not only mean values but also rms.
- Try a cut in time to optimize the difference between the two type of events
- Try different architectures for the deep neural network model
- Try different models, other than deep neural networks
- Try different pre-processing methods and different ways to organize the data
- There’s a measurement we didn’t use for this analysis referred to as Q, as in charge of the PMT, may be important.
- Develop methods to analyze events away from the center