

# Muon reconstruction of JUNO experiment with convolutional neural networks

Lu Wang, Kun Zhang, Miao He, Tao Lin, Weidong Li

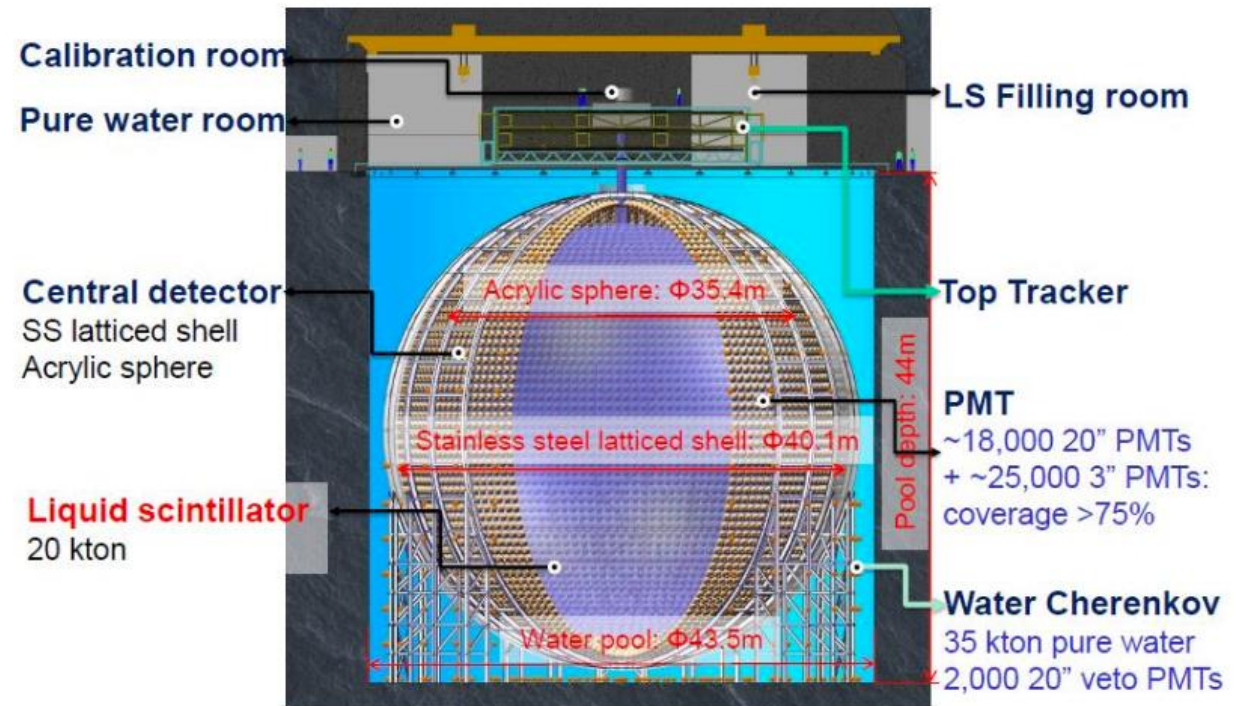
Institute of High Energy Physics, CAS, Beijing

# Outline

- Introduction
- Implementation
- Results
- Summary and future works

# The JUNO Experiment

- The Jiangmen Underground Neutrino Observatory (JUNO) is a multiple purpose neutrino experiment to determine neutrino mass hierarchy and precisely measure neutrino oscillation parameters.
- Central Detector
  - spherical container, radius=17.7m
  - 20k ton liquid scintillator
  - ~18k 20 inch PMTs, catching the lights emitted by the liquid scintillator
- Plans to take data in 2020.

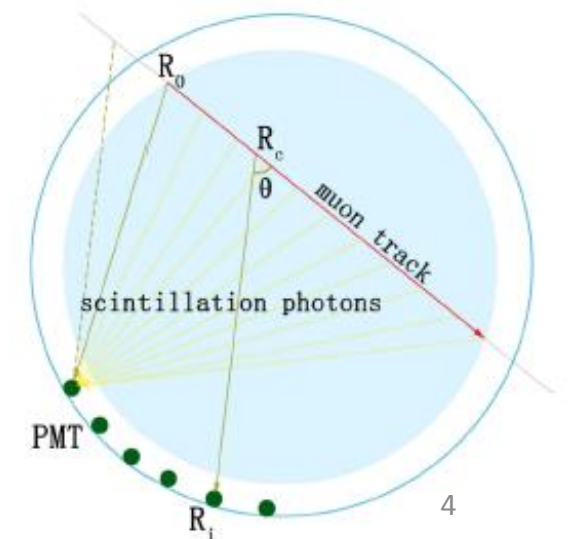


# Cosmic muon reconstruction : why and how?

- Cosmic muon is one of the major sources of neutrino backgrounds. To efficiently veto these backgrounds, we need to reconstruct the trajectory of muon.
  - Event rate of reactor antineutrinos is  $\sim 60/\text{day}$  , event rate of cosmic muon is  $\sim 3 \text{ HZ}$
  - It is better to veto certain volumes of the detector along the muon trajectory than to veto all the PMT signals in the time window of muon event.
- Traditional reconstruction is based on the “Fastest Light Model”.
  - The model predicts the First Hit Time ( FHT) of each PMT with track parameters and PMT positions.
  - By minimizing the prediction error, we can achieve the track parameters.

$$\chi^2 = \sum_i \left( \frac{T_i^{mes} - T_i^{pre}}{\sigma_i} \right)^2$$

- Reflection and refraction of optical photons, latency of the light scintillation and time resolution of the PMTs may affect the precision of this method.
- Additional corrections to the FHT bias are sometimes necessary for these methods.



# Applying deep learning in muon reconstruction

- Deep learning has been widely used in HEP computing.
  - Higgs search, Jet tagging, event classification etc.,
- Inspired by these works, we tried to use convolutional neural networks (CNN), a sub class of deep learning algorithm, to solve the muon reconstruction problem.
- This method treats the detector as a camera and the PMTs as pixels.
- Regression instead of classification,
  - $f(\text{Image of detector}) \rightarrow$  muon trajectory defined by injecting point and injecting direction
  - Similar to the problem of object detection in computing vision.

# Outline

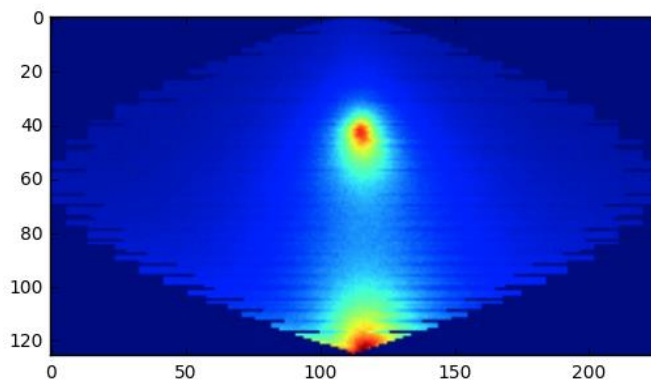
- Introduction
- **Implementation**
- Results
- Summary and future works

# Dataset

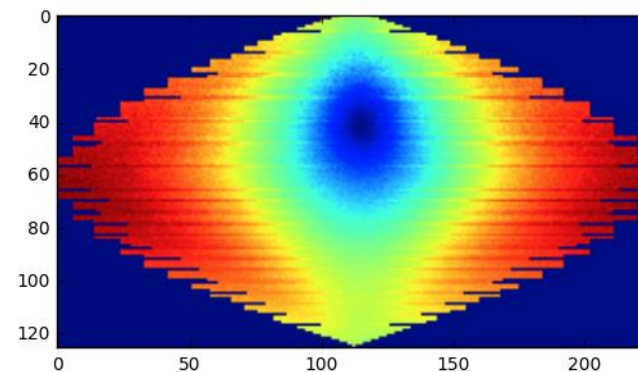
- 120 K events generated by Juno offline software as a labeled dataset.
- Simulation conditions:
  - Uniform randomly choosing injecting point and direction on the surface of the detector
  - Gaussian uncertainty of time measurement, TTS= 3ns
  - Electronic simulation is not included
  - Fixing energy of 200 GeV, the mean energy of the muons across the detector
- Divide into 100K training set, 20k evaluation set.

# Image Construction

- Arrange the ~18 k PMTs into a image like the world map,
  - two channels: PE number (Q) and First Hit Time (T)
  - PMTs of same latitude in a same line of the image, centered by longitude 0°
  - other pixels are padded with zero
- For each image, we normalized the image channel by channel:



Channel Q of a sample Image



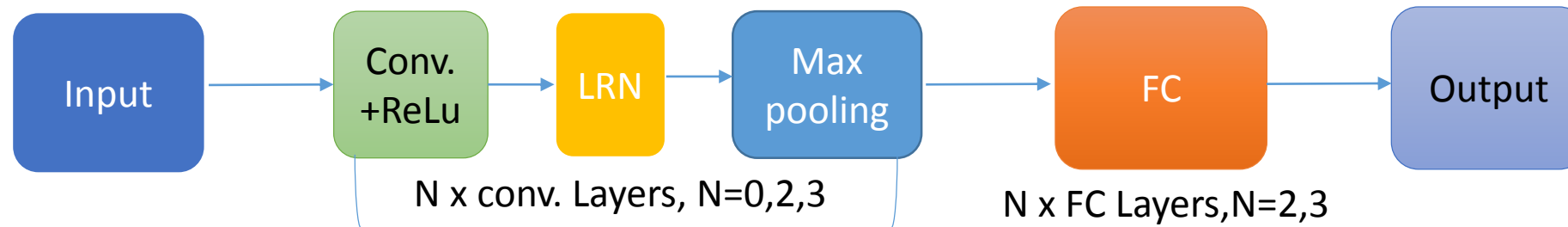
Channel T of a sample Image

$$X = \frac{X - \mu}{\sigma}$$

After normalization,  $\mu=0$ ,  $\sigma=1$



# Network Architectures



- Output: injecting point  $(x_0, y_0, z_0)$ , injecting point  $(p_x_0, p_y_0, p_z_0)$
- Convolution Stride:  $[1, 1, 1, 1]$ , Max Pooling Stride:  $[1, 3, 3, 1]$
- LRN Layer parameters: bias=1.0, alpha=0.001 / 9.0, beta=0.75
- Convolution Filters per layer: 16
- Loss Function: L1
- Weight decay of Conv. Layers: 0, Weight decay of FC layers:  $\mu=4e-4$ ,  $\sigma=0.04$

# Network Architectures

Name	Conv. Filter Size	Pooling Filter Size	Convolutional Layers(#)	Fully Connection Layer Configuration	Network Parameters(#)
2conv-5-3-1024-512-256	5x5	3x3	2	[Previous Layer]x1024x512x256x6	180M
2conv-5-5-1024-512-256	5x5	5x5	2	[Previous Layer] x1024x512x256x6	23M
2conv-5-3-384-192	5x5	3x3	2	[Previous Layer] x384x192x6	70M
2conv-5-5-384-192	5x5	5x5	2	[Previous Layer] x384x192x6	9M
3conv-5-3-384-192	5x5	3x3	3	[Previous Layer] x384x192x6	124M
3conv-5-3-384-192	5x5	5x5	3	[Previous Layer] x384x192x6	5M
DNN-1024-512-256		NA		[ PMTsx2 ] x1024x512x256x6	36M

# Training Platform

- Server Configuration
  - HP ProLiant DL380 Gen9, Intel(R) Xeon(R) CPU E5-2650 v4, 64GB RAM
- GPU
  - Nvidia Tesla K80

- Software

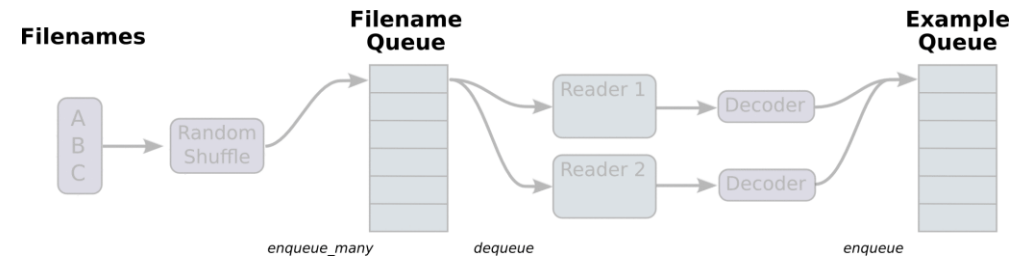
- TensorFlow v0.12, Python 2.7

- Data format

- ROOT ---Python,PyROOT,TensorFlow---> TFRecord format ---tf.tfrerecordreader---> Training Process

- I/O

- `tf.train.shuffle_batch` to read and shuffle training data with multi-thread

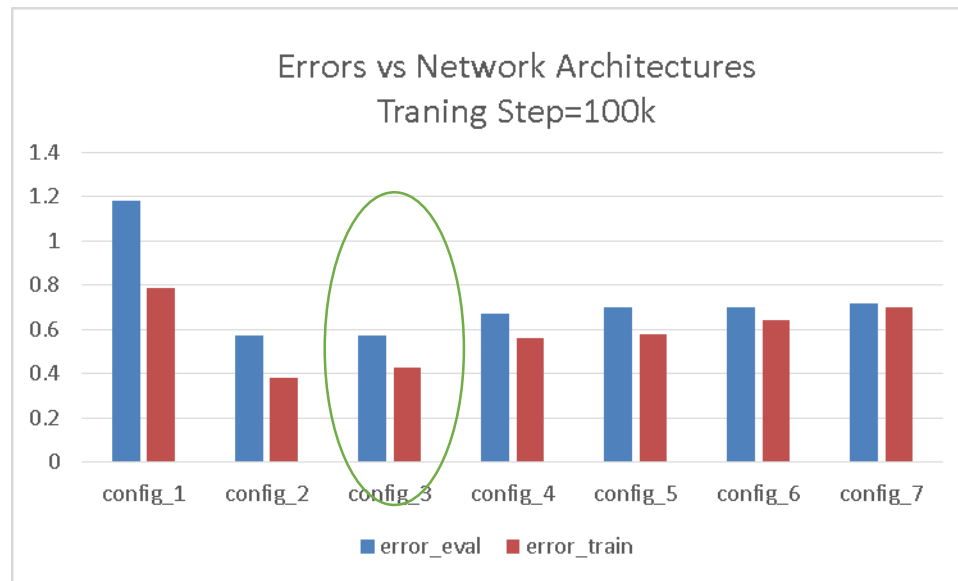


# Training Parameters

- Optimization Algorithm
  - Gradient Descent Optimizer, `batch_size=128`
- Initial Learning Rate
  - Set initial learning rate to a number in linear range [0.01,0.1,10]
  - Select the initial learning rate according to the evaluation result after a quick training (one epoch of training samples)
  - For most of our network architectures, we find 0.1 as the best choice.
- Learning Rate Decay
  - Every 32 epochs, decay rate=0.3
- Weight Regulator during Training
  - `variable_averages = tf.train.ExponentialMovingAverage(0.9999, global_step)`

# Evaluation Error vs Network Architectures

- Choose the best network configuration by comparing the prediction errors and overfittings after 100k step train.



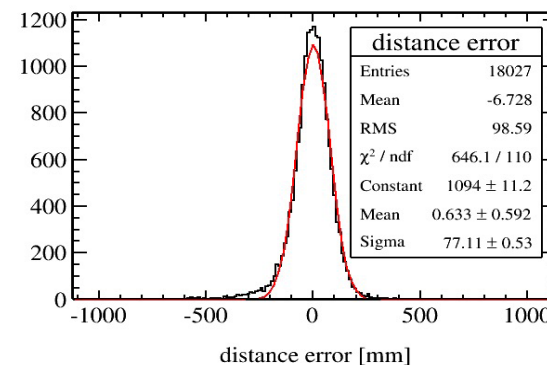
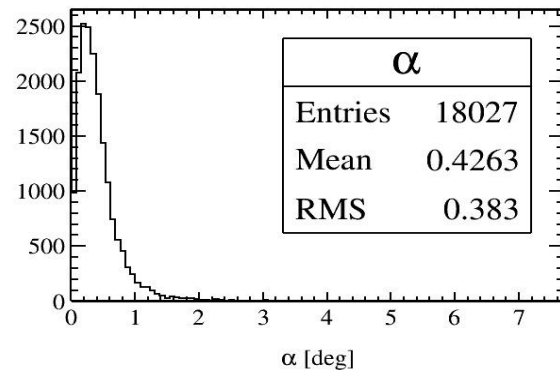
Name	Configuration	Parameters(#)
config_1	dnn(2PMTs-1024-512-256-6)+L1	36M
config_2	cnn(5*5*16conv-3*3max_pool)*2-1024-512-256-6-l1	180M
config_3	cnn(5*5*16conv-5*5max_pool)*2-1024-512-256-6-l1	23M
config_4	cnn(5*5*16conv-3*3max_pool)*2-384-192-6-l1	70M
config_5	cnn(5*5*16conv-5*5max_pool)*2-384-192-6-l1	9M
config_6	cnn(5*5*16conv-3*3max_pool)*3-384-192-6-l1	124M
config_7	cnn(5*5*16conv-5*5max_pool)*3-384-192-6-l1	5M

# Outline

- Introduction
- Implementation
- **Results**
- Summary and future works

# Performance with the best architecture

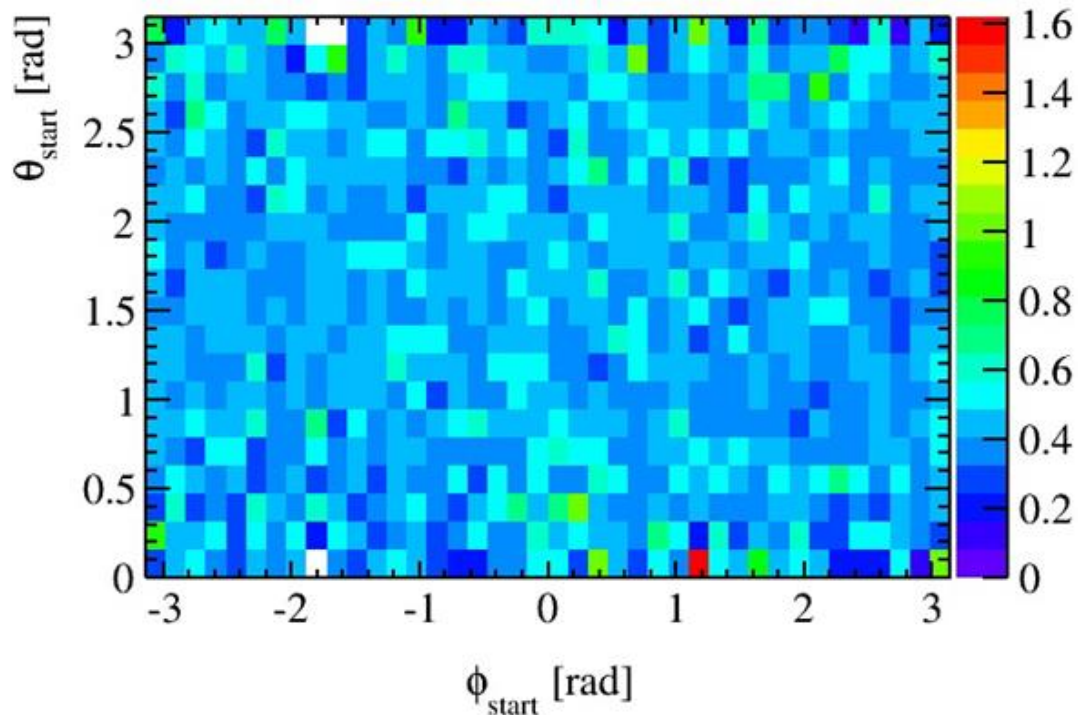
- Angle Error ( $\alpha$ )
  - The angle between direction<sub>labeled</sub> and direction<sub>reconstructed</sub> ,  $< 0.5$  degree
- Distance Error
  - Distance means the distance from the the trajectory to the center of sphere.
  - Distance Error= distance<sub>labeled</sub> – distance<sub>reconstructed</sub> ,  $\sim$ several centimeters



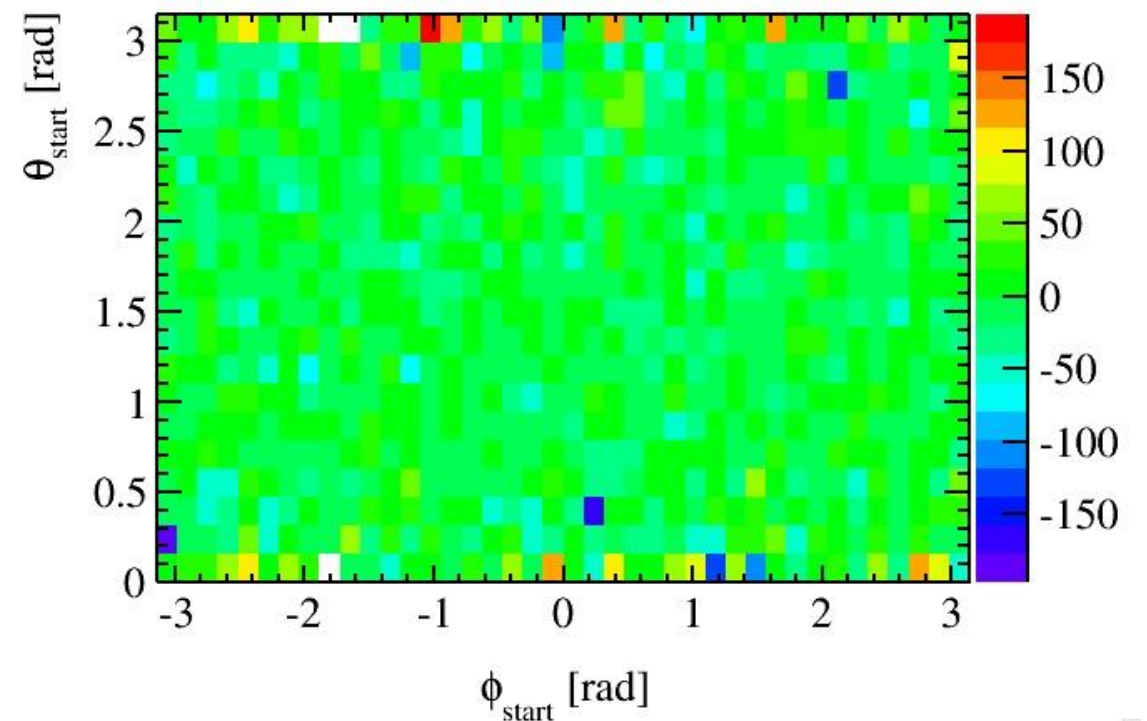
- Comparable to the performance of traditional reconstruction method, validates the potential of CNN based reconstruction.

# Errors versus the injecting point

Angle Error versus injecting point



Distance Error versus injecting point

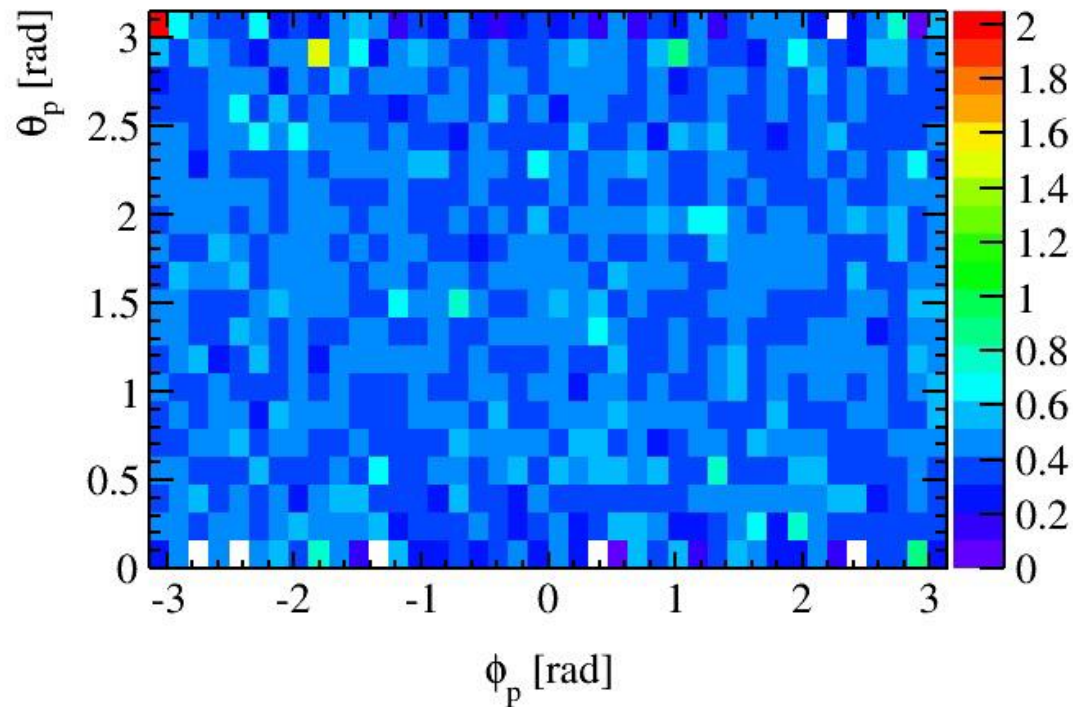


Uniform distribution , so that the image arrangement has not affected the fairness of prediction results.

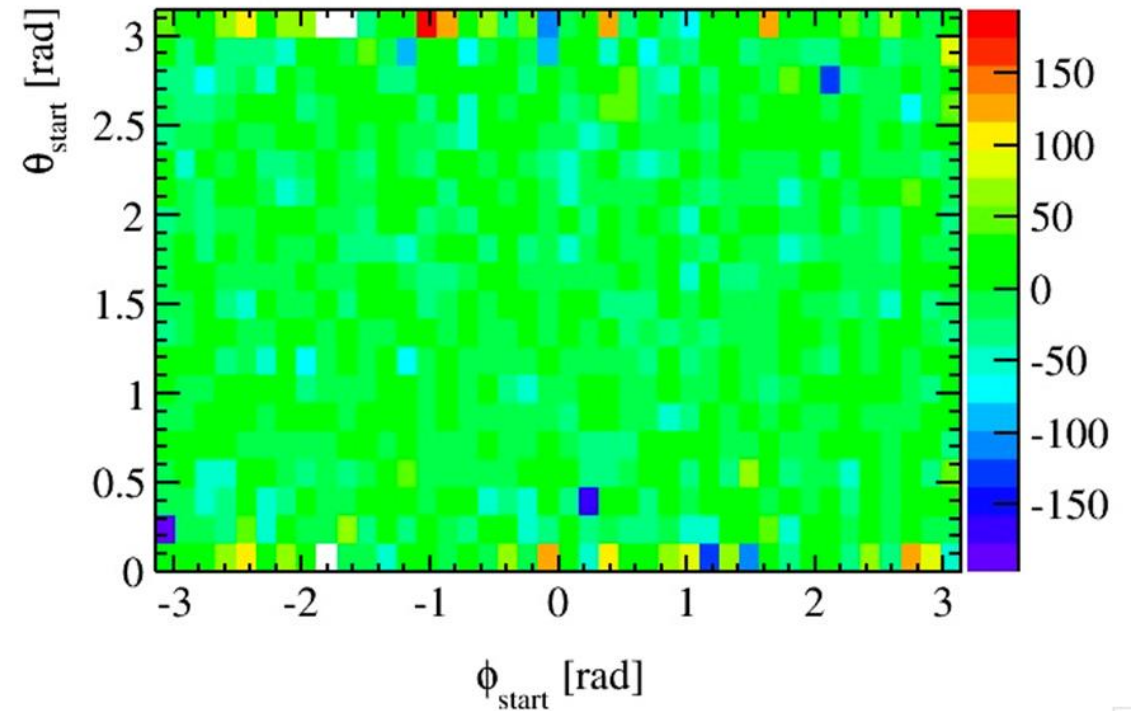


# Errors versus the injecting direction

Angle Error versus injecting direction

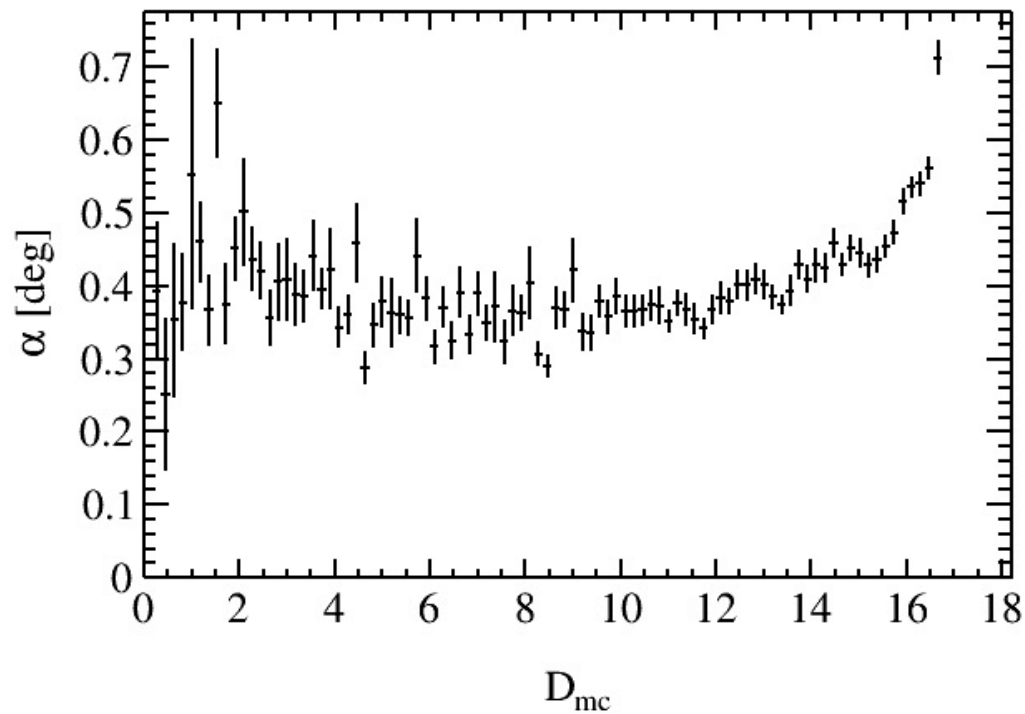


Distance Error versus injecting direction

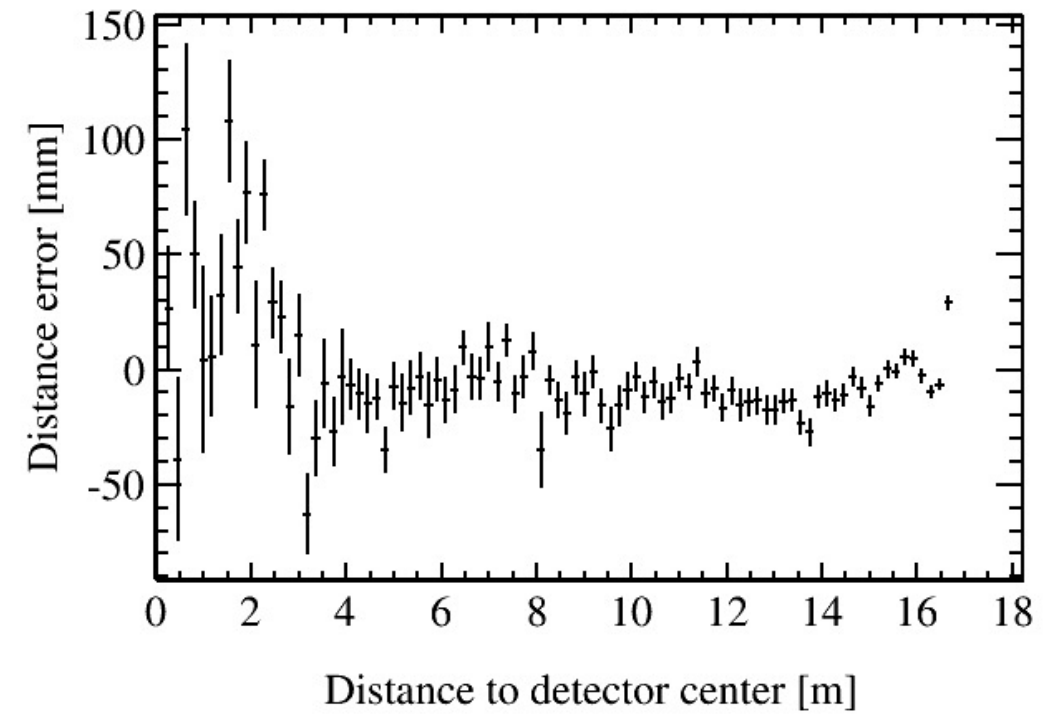


# Errors versus the distance to detector center

Angle Error versus distance to center of sphere



Distance Error versus distance to the center of sphere



- The larger errors near the detector center may be caused by deficiency of data samples in this area.
- It is difficult to predict injecting angle for tracks near the surface of the detector.

# Outline

- Introduction
- Implementation
- Results
- **Summary and future works**

# Summary

- We tried to solve the muon reconstruction problem of JUNO experiment by convolutional neural network.
- Preliminary results trained with limited simulation events and on top of a small CNN shows the potential and advantage of this method.
  - Angle Error < 0.5 degree, Distance Error ~several centimeters, comparable to traditional method
  - Additional corrections of FHT bias in traditional reconstruction can be avoided.

# Future Works

- The network complexity is limited by the size of training dataset. In the future, we will try to:
  - Add a rotation step in data preprocessing to fully leverage the spatial symmetry of the detector.
  - Extend the training to a multi-GPU environment, to speed up the training process and enable larger networks.
- CNN-based Classification of bundle events and single events.
- CNN-based Reconstruction of bundle muons.

Thank you!