Machine learning applications for JUNO

Maxim Gromov[1,2]
on behalf of the JUNO Collaboration

[1]*Skobeltsyn Institute of Nuclear Physics Lomonosov Moscow State University*
[2]*Joint Institute for Nuclear Research*

Reconstruction and Machine Learning in Neutrino Experiments

17.09.2019

JUNO Detector

Calibration Room

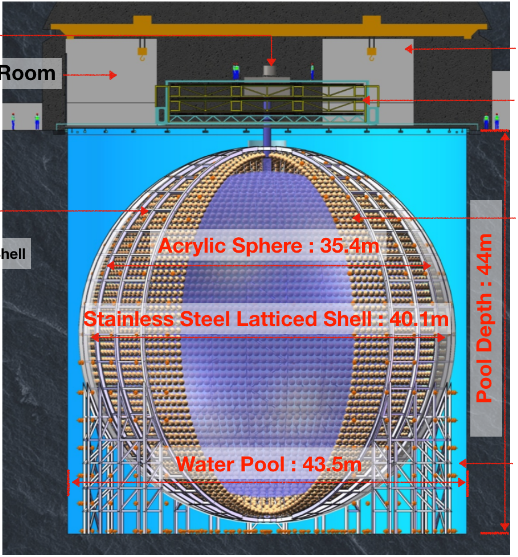
Pure Water Filling Room

Central Detector

Acrylic Sphere
Stainless Steel Latticed Shell

Liquid Scintillator

20 kton



LS Filling Room

Top Tracker

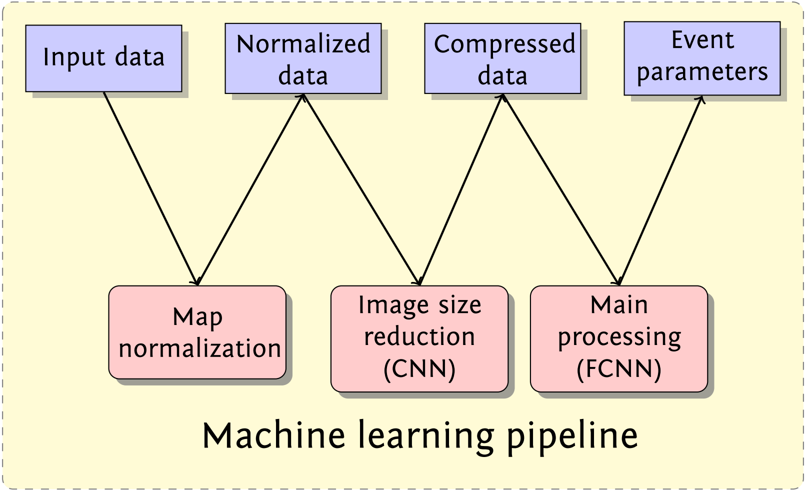
PMT

18,000 20-inch PMTs
25,000 3-inch PMTs
78% Coverage

Water Cherenkov

35 kton pure water
2,000 20-inch veto PMTs

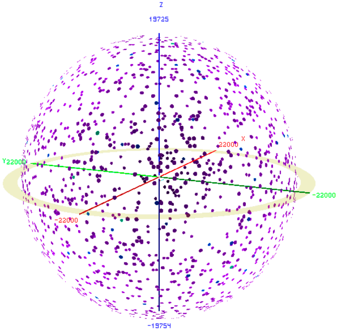
Common aspects of machine learning for JUNO



FCNN – **F**ully **C**onnected **N**eural **N**etwork
CNN – **C**onvolutional **N**eural **N**etwork

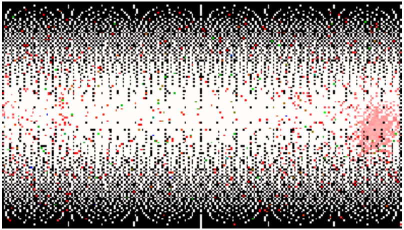
Aspect №1: Input data preparation

Step 1: original info for each PMT:
charge, 1st hit time, mean hit time,
PMT positions, σ (hit time)
Step 2: event on the $\phi - \theta$ map of the PMT positions

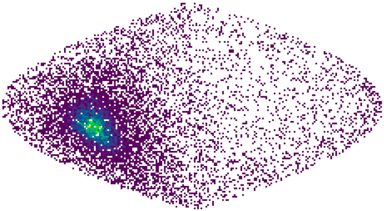


Step 3: avoid event splitting by rotating the image
Step 4: restoration of the uniform density of PMTs:
squeezing pixels in ϕ-direction

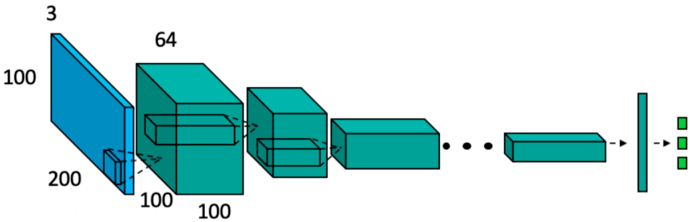
$\phi - \theta$ map of the PMT positions

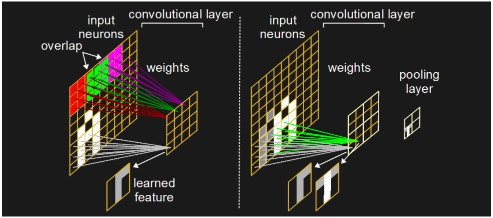


Sinusoidal projection: squeeze ϕ-bins



Aspect №2: Reduction of data dimension





Convolutional Neural Network (CNN)

Applying a set of filters (kernels) to extract feature maps

Non-linear down-sampling (pooling) to reduce the map size

Alternate use of convolutional and pooling layers

Position reconstruction: some first attempts

Convolutional neural network

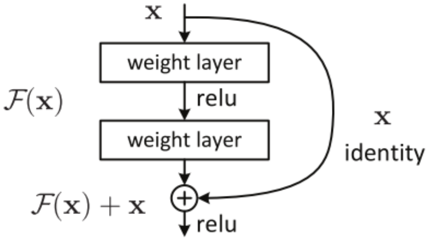
- Project charge and time data of all PMT's using the Mollweide projection
- 4 convolutional, 4 pooling and 2 FC layers
- Error in absolute distance is $\approx$ 11 cm (5 MeV, 140k events)
- Training time is huge, but prediction is fast

Feedforward neural network

- Inputs are mean First Hit Time, total number of PE and 3 cartesian coordinates retrieved from charge center method
- Error in absolute distance is $\approx$ 15 cm (5 MeV, 140k events)
- Training is very fast

Position reconstruction: CNN architecture

How to avoid network degradation at the last (deep) layers?



- Let's define the residual function $\mathrm{F(x) = H(x) - x}$
- reframed $\mathrm{H(x) = F(x) + x}$, where F(x) and x represents the stacked non-linear layers and the identity function (input=output)
- Easy to optimize $\mathrm{F(x)}$
- Hard to optimize $\mathrm{H(x)}$ (direct mapping from x to y)

Position reconstruction: CNN architecture

- Usage of residual blocks instead of “plain” connections[1]
- 50 convolutional layers, 25 million parameters
- Batch Normalization layer after each convolutional one
- ReLU activation after each Batch Normalization layer
- He Normal (He-et-al) weight initialization[2]
- L2 Regularization
- Adam Optimizer (without learning rate decay), or Stochastic Gradient Descent with Nesterov Momentum[3]and exponential learning rate decay (decay rate=0.9, decay steps=420000)

[1] He K. et al. Deep residual learning for image recognition, 2016

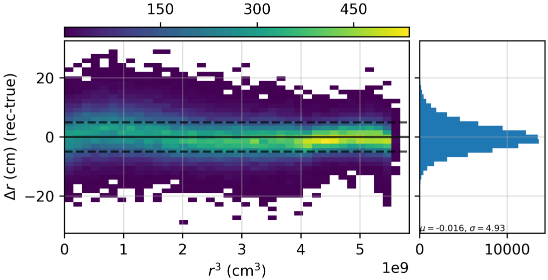
[2] He K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015

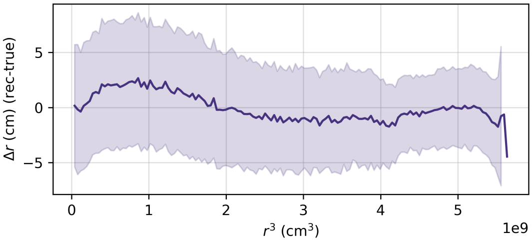
[3] Sutskever I. et al. On the importance of initialization and momentum in deep learning, 2013

Position reconstruction: input data

- 1 million e+ events
- Continious in [0, 10] MeV
- Events uniformly distributed in the detector
- Training: 900k events
- Testing: 100k events
- Validation: 5k events for each discrete energy in {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} MeV
- Sinusoidal projection, 256x128, 2 channeled images (charge and time)

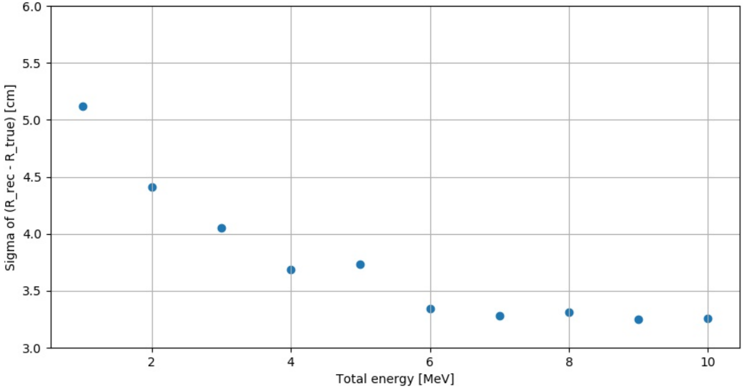
Position reconstruction: some results





Position reconstruction: some results

σ of $(\mathrm{R_{rec}} - \mathrm{R_{true}})$ @ 1MeV is 5.1 cm



Energy reconstruction: input data

Input (features):

$N_{p.e.}$ total number of photo-electrons

r_{cc}, Z_{cc} charge center $\frac{1}{N_{PMT}} \sum_i^{N_{PMT}} \vec{x}_i n_i^{p.e.}$

t_{fh} mean first hit time
(counted from the time of the first PMT hit)

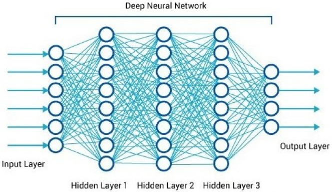
Dataset: positrons uniformly distributed in CD + dark noise

1. $E_{kin} : 0 - 10$ MeV,
900k/100k – training/validation
2. $E_{kin} : 0, 1, ..., 10$ MeV,
11x10k – testing

Energy reconstruction: two ML techniques

Boosted Decision Trees (BDT)

Deep Neural Networks (DNN)

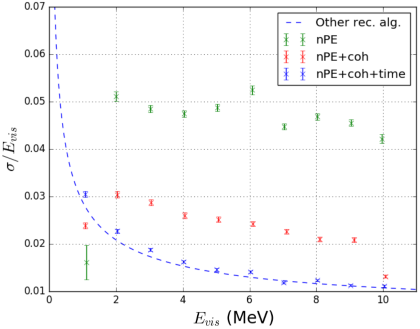


TensorFlow

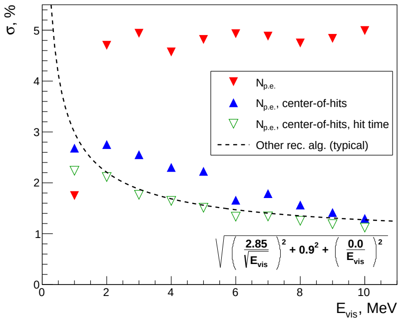
Which one is better?

Energy reconstruction: resolution

Boosted Decision Trees



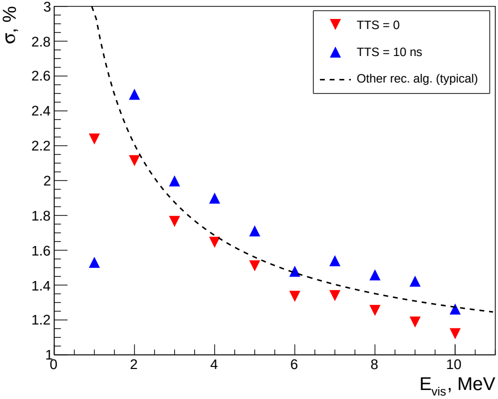
Deep Neural Network



- As good as traditional (JUNO default) methods
- Similar performance, however DNN is little better

Energy reconstruction: resolution with TTS

Effect of Time Transit Spread (TTS):



(studied only with DNN)

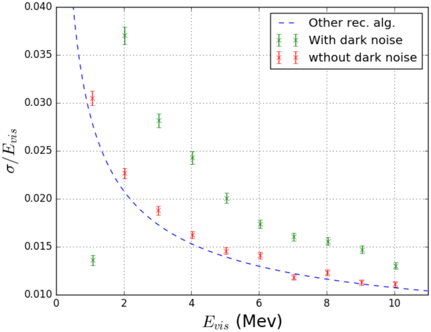
TTS in JUNO PMTs:

Hamamatsu:	3 ns
NNVT:	18 ns

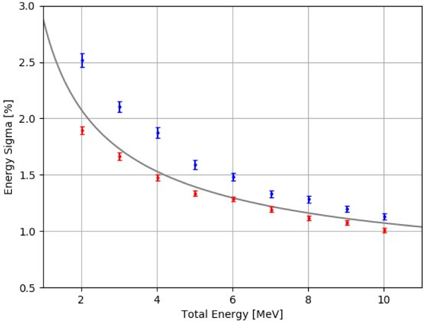
Energy reconstruction: resolution with DN

Effect of **D**ark **N**oise (DN):

BDT



DNN



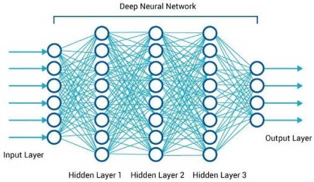
- DNN suffer less

Energy reconstruction: BDT vs. DNN*

Boosted Decision Trees (BDT)

- Less accurate
- Faster **:
  Training: several minutes
  Reco: 2 seconds/100k events
- Suffers more from DN
- Requires less data for training

Deep Neural Networks (DNN)



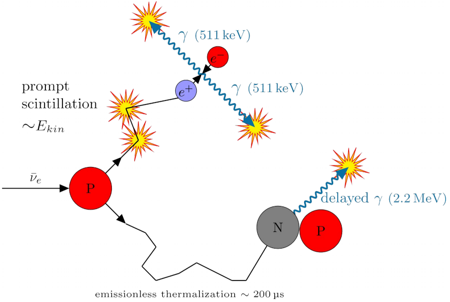
- More accurate
- Slower **:
  Training: half an hour
  Reco: 15 seconds/100k events
- Suffers less from DN
-

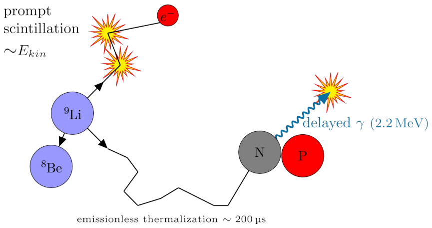
* – based on this study

** – tested on regular laptop

Particle identification: motivation

Different particles have different pulse shapes,
this is especially pronounced in the slow component of scintillation
Main detection channel: Inverse **Beta Decay** (IBD)
Possible e^-/γ, α/β discrimination
e^+/e^- discrimination: a key point to reduce backgrounds (^{9}Li/^{8}He)





Small difference in shape and timing ($+3$ ns for ortho-positronium)
due to the lack of an annihilation signal for backgrounds

e^+/e^- discrimination: methods, results and problems

e^- & e^+ events: $E_{vis} = 1 - 10$ MeV, 100k+100k events, uniform distribution
Sampling: training : validation : test is 80% : 10% : 10%
Time profile: first 400 ns, 1 ns per 1 bin

Type 1: FCNN
Network structure:

- one simple hidden layer
- 20 nodes in the hidden layer
- activation function: ReLU
- optimizer: Adam

Input data: time profile for each e^+

- PMT trigger time
- only one time profile

Type 2: CNN
Network structure:

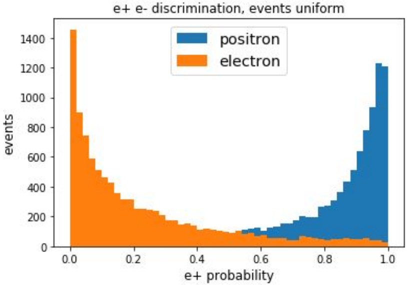
- 4 layers of 3D convolutions
- 5 separable 2D conv. layers
- activation function: ReLU (softmax for the output layer)

Input data: 8 time profiles for each e^+

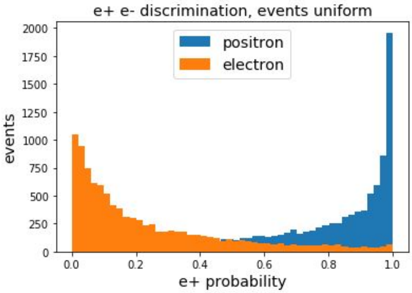
- arrival time & total signal
- simple $\phi - \theta$ map: 8 pixels, 2x4 map
- time profile for each pixel

e^+/e^- discrimination: methods, results and problems

Type 1: FCNN
Accuracy 87.5%



Type 2: CNN Accuracy 83.0%

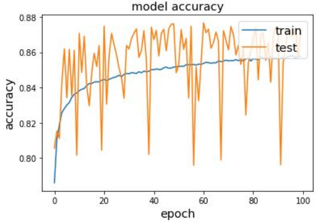


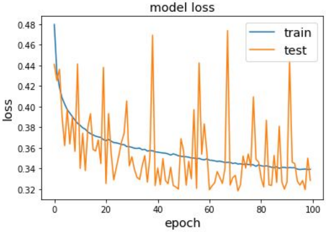
e^+/e^- discrimination: methods, results and problems

Let's take a closer look

Type 1: FCNN

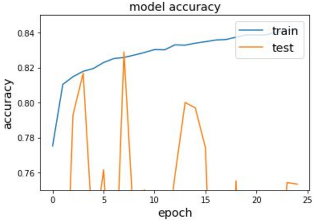
Accuraccy 87.5%

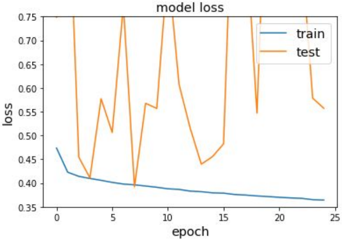




Type 2: CNN

Accuraccy 83.0%





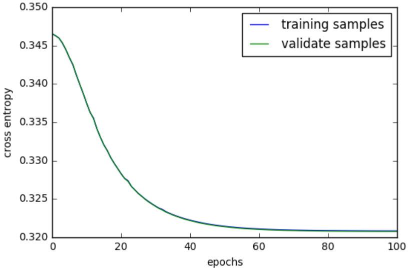
e^+/e^- discrimination: methods, results and problems

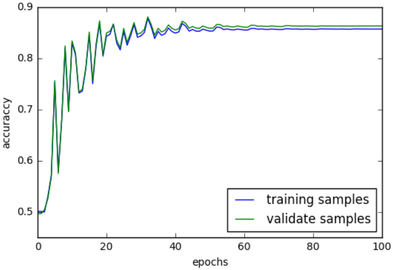
Another attempt to perform PSD

Type 3: FCNN with 2 hidden layers with 1200 and 20 nodes

Events are placed in the center

The rest options are the same

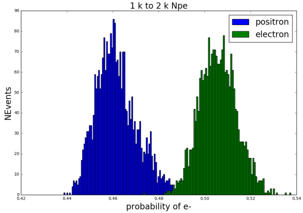


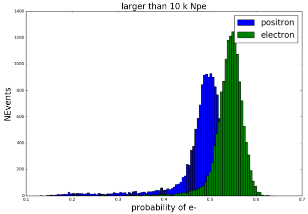


Loss function: cross entropy $\epsilon = \frac{1}{N} \sum_i^N (y_{\text{real}} \log y_{\text{predict}})$

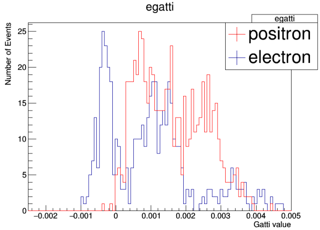
e^+/e^- discrimination: comparison Gatti with FCNN

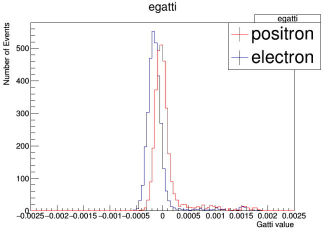
FCNN





Gatti's method for e^-/e^+





Deterioration of the e^+/e^- discrimination with increasing energy

Muon reconstruction: motivation

Cosmic muon is one of the major sources of neutrino backgrounds.
To efficiently veto these backgrounds,
we need to reconstruct the trajectory of muon.

Traditional approach: Fastest Light Model (FLM)

Problems:

- Reflection and refraction of optical photons, latency of the light scintillation and time resolution of the PMTs may affect the precision of this method. Complex optical model.
- Additional corrections to the First Hit Time (FHT) bias are necessary for these methods.

Machine learning approach: Reconstruction with CNN

- No need to consider optical model
- Without any corrections

Muon reconstruction: input and output data

As a supervised deep learning problem,
we need plenty of labelled training samples

2 maps as inputs: PE number (Q) and First Hit Time (T)

320k simulated muon events

- Uniform randomly choosing injecting point and direction on the surface of the detector
- Labeled with injecting point and direction from simulation
- Using the mean energy of the muons across the detector (200 GeV, fixed)
- Gaussian uncertainty of time measurement, TTS= 3ns
- Electronic simulation is not included
- Divide into 300k training set, 20k testing set

6 outputs: injecting point (x_0,y_0,z_0), injecting direction (p_{x0},p_{y0},p_{z0})

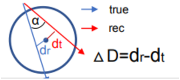
Muon reconstruction: network structure

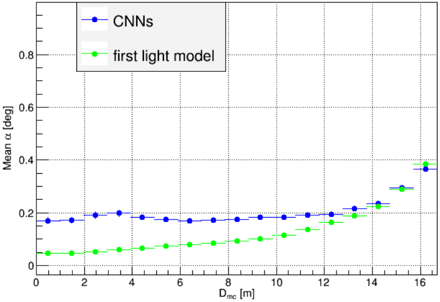
Final network structure is a result of search for an optimal network architecture

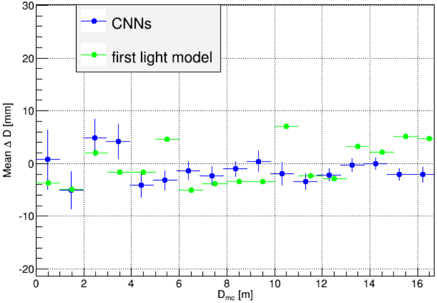
- 2 convolutional layers
- Convolutional filters per layer: 16
- Convolutional filter size: 5x5
- Pooling filter size: 5x5
- Activation function: ReLU
- Optimization algorithm: gradient descent optimizer, $batch\_size = 128$
- FCNN part: 3 hidden layers, 1024, 512, 256 nodes
- Number of the network parameters: 23M
- Loss Function: L1

Muon reconstruction: comparison FLM with CNN

The performance of the CNN method is comparable to the performance of the FLM muon reconstruction method







- The deflection angle α is less than 0.4 degree
- The distance error $\Delta\mathrm{D}$ is smaller than 1 cm

Muon reconstruction: time costs

Method	Hardware	Time per event [ms]	Comments
1. FLM	one CPU	~ 5000	
2. CNN (batch size=1)	one CPU (E5-2650 v4)	~ 950	~ 5x speed up vs 1
3. CNN (batch size=1)	one GPU (Tesla V100)	~ 9	~ 500x speed up vs 1

Batch size means the number of events reconstructed at the same time

Reconstruction speed is greatly improved with the CNN approach

Lowering the multi-messenger trigger

Ultimate technical goal:
set the multi-messenger (MM) trigger as low as possible

Benefits:

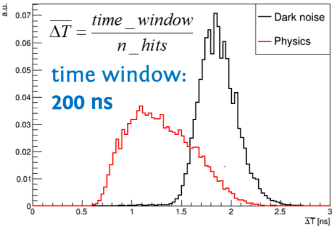
- Significantly increase number of detected events in case of supernova
- Several times more $\nu - \mathrm{p}$ Elastic Scattering (ES) events
- Observation and research of the neutronization burst
- SN can be found 50 ms earlier (small bonus)

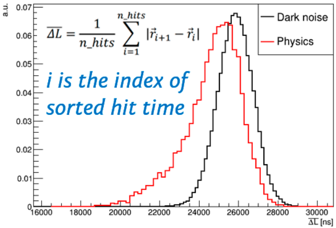
Situation: sinking into a sea of dark noise and ^{14}C radioactivity

- 20-inch PMT dark noise rate ~ 50 kHz per PMT
- ^{14}C radioactivity (optimistic case): ~ 100 kHz for 20 kton LS

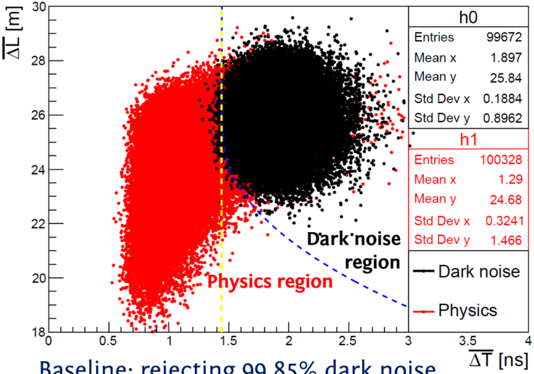
JUNO MM trigger: dark noise rejection in 2D parameter space

Idea: Physics-like hits will cluster in hit time & location of hit





A 2-D discriminator is considered

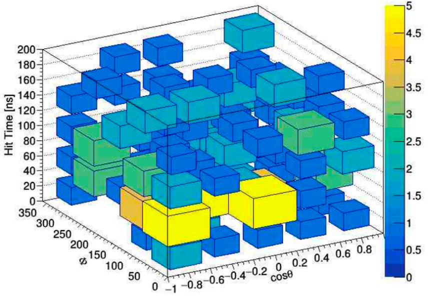


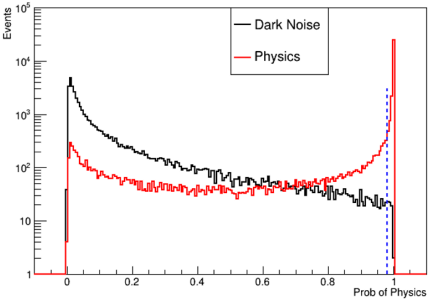
Baseline: rejecting 99.85% dark noise, while retaining 70% physics

JUNO MM trigger: dark noise rejection with machine learning

Deep learning techniques

- Time window: 200 ns
- Input parameters:
  - Location of hit PMT (Φ, $\cos\theta$)
  - Hit time for each hit
  - Nhits

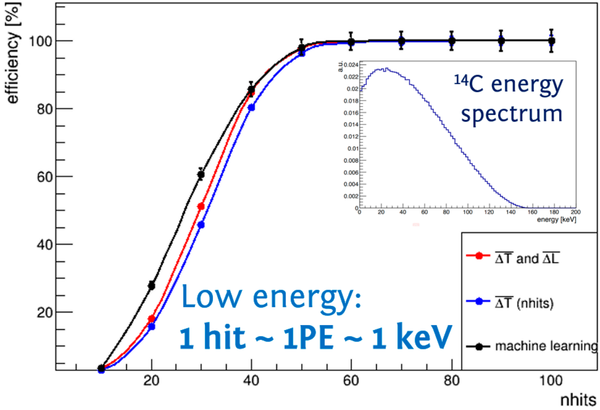




Baseline: rejecting 99.85% dark noise, While retaining 71% physics

JUNO MM trigger: comparison of different approaches

Filtering efficiency when rejecting 99.85% dark noise



Physics: ^{14}C

Retain 100% efficiency at ~50 keV

Still retain ~20% efficiency at ~20 keV !

Everything needs to be calibrated with real data!

General problems and recommendations

Problems:

- Input data sufficiency:
  $\mathrm{V_{CD}} \sim 23200\,\mathrm{m}^3$,
  if only 1 event per single cubic volume (10x10x10 cm^3)
  $\Longrightarrow 23.2 \cdot 10^6$ events required
- Input data source: simulation, calibration, real data after special offline analyses?
- Energy and position dependencies
- Electronic simulation influence
- Dark noise impact
- Difficulties on boundaries

Recommendations:

- Neural network structure optimization
- Input data should be divided into 3 sets:
  training, validation, test

Conclusions

- Machine learning works!
- Can be 500x speed up in comparison with traditional approaches
- Easy to achieve rough results, hard – accurate ones
- Reduction of data dimension is a challenge
- Required solid and sufficient dataset for training
- Loss functions without spikes for all data subsets
- Consideration other methods:
  Boosted Decision Trees (BDT) &
  General Regression Neural Network (GRNN)

Thank you for your attention!