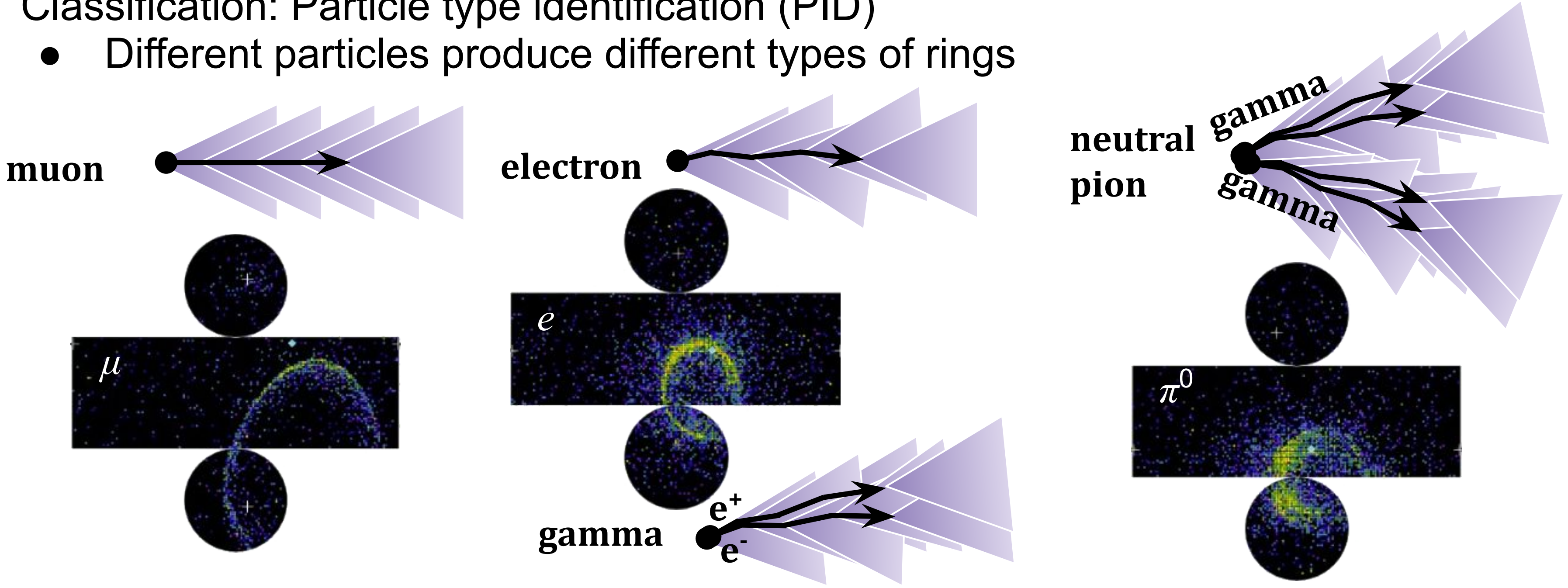
Machine Learning Reconstruction for Hyper-K

20 February 2024
N. Prouse

Reconstruction in WC detectors

Classification: Particle type identification (PID)

- Different particles produce different types of rings



Regression: reconstructing particle's properties:

- Location and time of PMT hits allows triangulating position and direction
- Amount of charge observed at PMTs gives estimate of energy

Traditional reconstruction method

fiTQun: Likelihood-based reconstruction for higher energies

- Originally developed for Super-K detector
  - Based on algorithm of MiniBooNE: https://arxiv.org/abs/0902.2222

- Uses full information of unhit PMTs + time & charge of hit PMTs:

$$\underbrace{\boxed{L}}_{\text{Likelihood to maximise}}\left(\underbrace{\boxed{\mathbf{x}}}_{\text{Candidate event hypothesis}}\right) = \prod_{j}^{unhit} \underbrace{\boxed{P_j\left(unhit|\mathbf{x}\right)}}_{\text{Probability of no hit at PMT}} \prod_{i}^{hit} \underbrace{\boxed{P_i\left(hit|\mathbf{x}\right)}}_{\text{Probability of hit at PMT}} \underbrace{\boxed{f_q\left(q_i|\mathbf{x}\right)}}_{\text{Hit charge probability density}} \underbrace{\boxed{f_t\left(t_i|\mathbf{x}\right)}}_{\text{Hit time probability density}}$$

- Probabilities calculated based on direct + scattered + reflected light

- Likelihood ratios used to distinguish particle types and single-ring / multi-ring event topology hypotheses

Machine learning reconstruction for WC

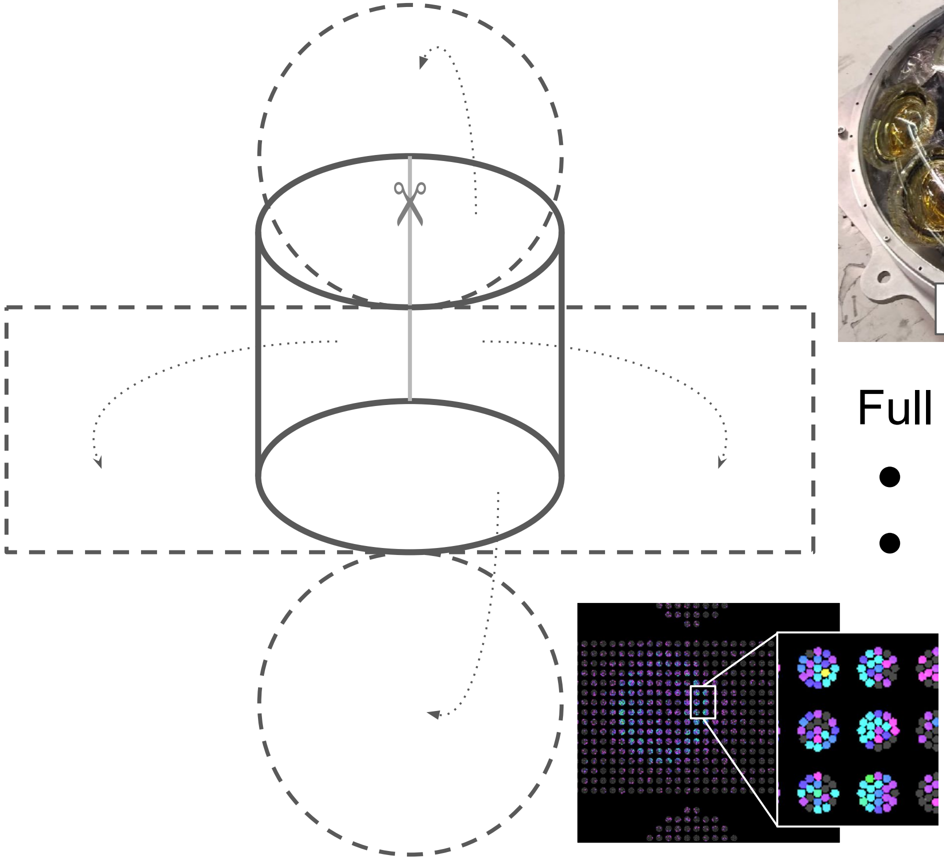
Limit of traditional maximum-likelihood reconstruction methods (fiTQun) is being reached

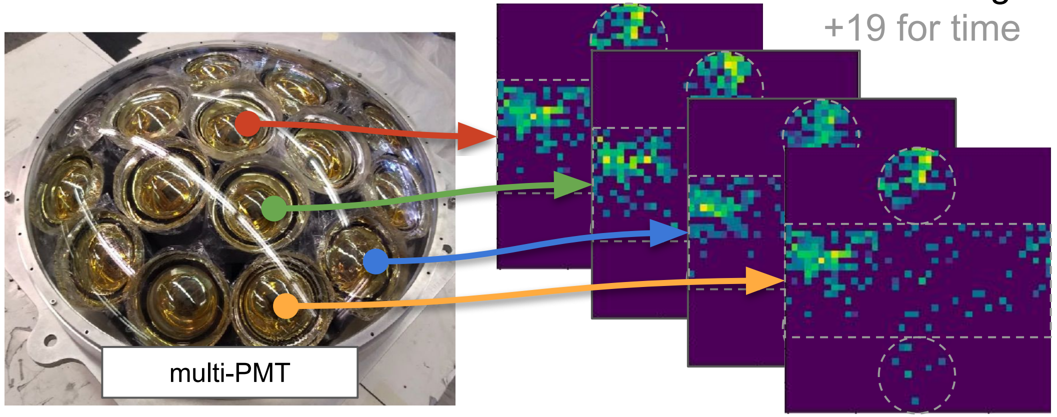
- Computation time is becoming a limiting factor
  - Larger far detector with more PMTs increases computation time
  - Smaller intermediate detector requires scaled down resolutions
  - Improving resolutions requires more complex algorithms with fewer approximations

ML and deep neural networks have potential to push reconstruction further

- Very successful in areas of computer vision and image processing
- Potential to use all information without detector model approximations
- Very fast to run once neural networks have been trained
  - fiTQun reconstruction: 1M events uses 10,000s CPU-hours
  - ML reconstruction: 1M events uses less than 1 GPU-hour
  - Opens opportunities for analyses with huge datasets not currently feasible

Image-like data for IWCD with mPMTs



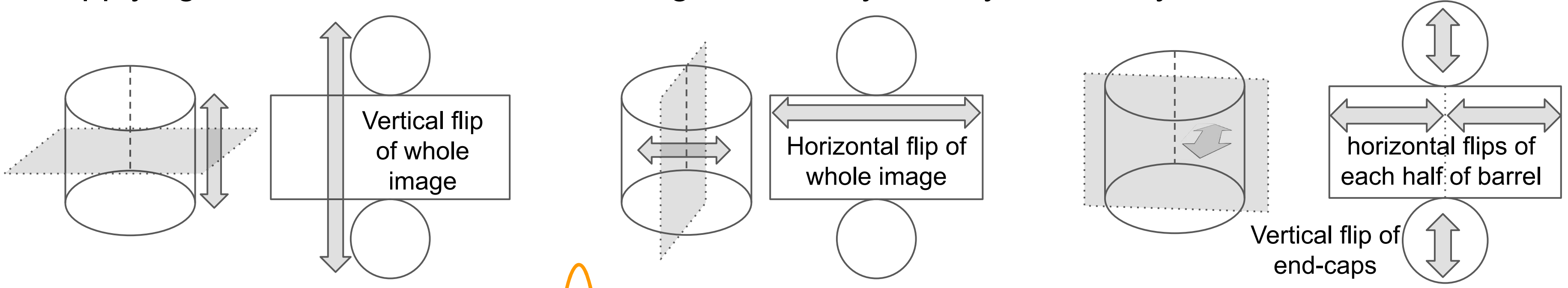


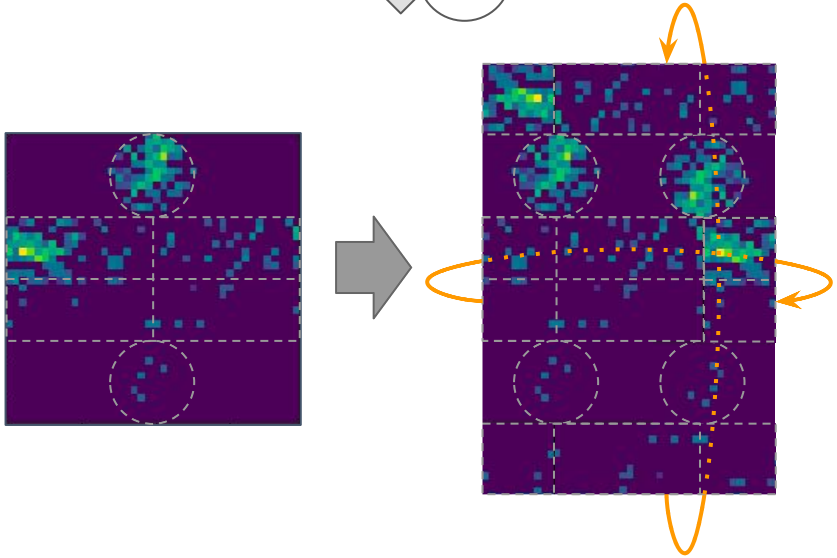
Full cylinder is unwrapped onto flat image

- One pixel per multi-PMT
- 38 channels per pixel:
  19 charge & 19 time channels of the 19 PMTs per mPMT

Data Transformations and Augmentation

Applying random transformations using detector symmetry effectively increases dataset

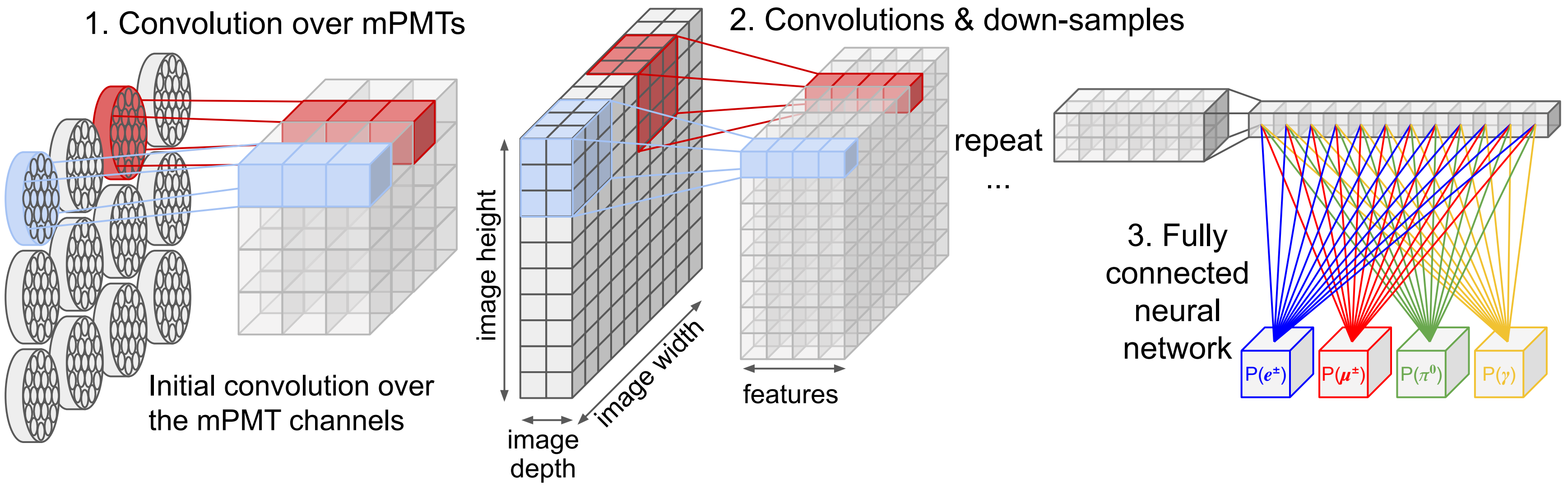

Vertical flip of whole image
Horizontal flip of whole image
horizontal flips of each half of barrel
Vertical flip of end-caps



Rearranging and duplicating the geometrical surface has several advantages

- Less dependence on “choice” of slice along barrel to unwrap cylinder
- All segments appear exactly twice with minimal blank space
- Circular boundary conditions in both directions

ResNet architecture for IWCD with mPMTs

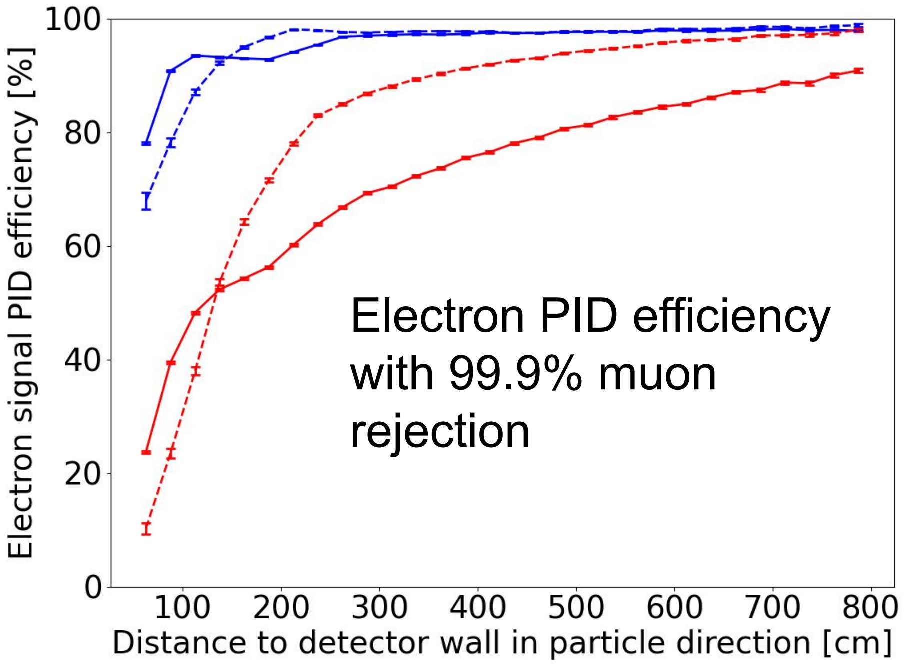

1. Convolution over mPMTs
2. Convolutions & down-samples
repeat
...
3. Fully connected neural network
Initial convolution over the mPMT channels
image height
image width
image depth
features
$P(e^{\pm})$
$P(\mu^{\pm})$
$P(\pi^0)$
$P(\gamma)$

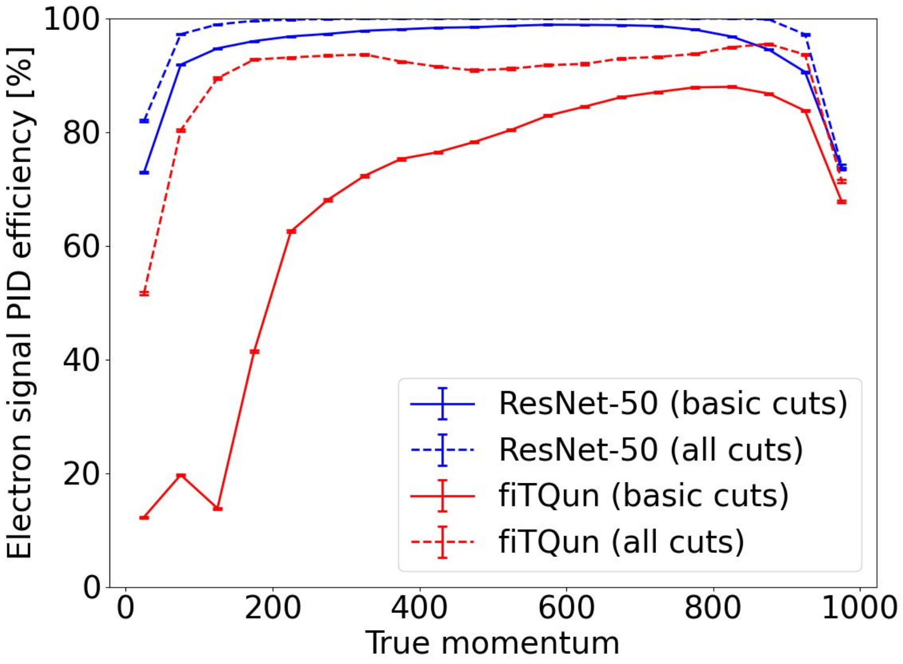
Convolutional Neural Network based on ResNet-50

- Initial ResNet 7x7 convolution with downsampling replaced by 1x1 convolution over 38 mPMT channels without downsampling
- Circular padding applied at each convolution to exploit circular boundary conditions

Also explored point-cloud based networks (PointNet, DG-CNN) and graph networks but ResNet CNN has consistently given best results so far

PID results: electron vs muon

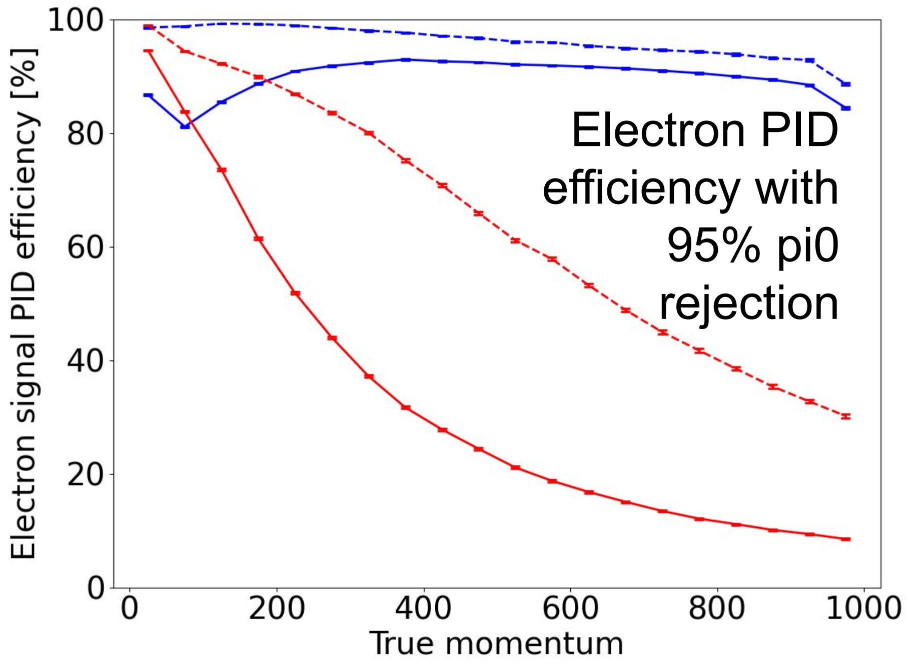


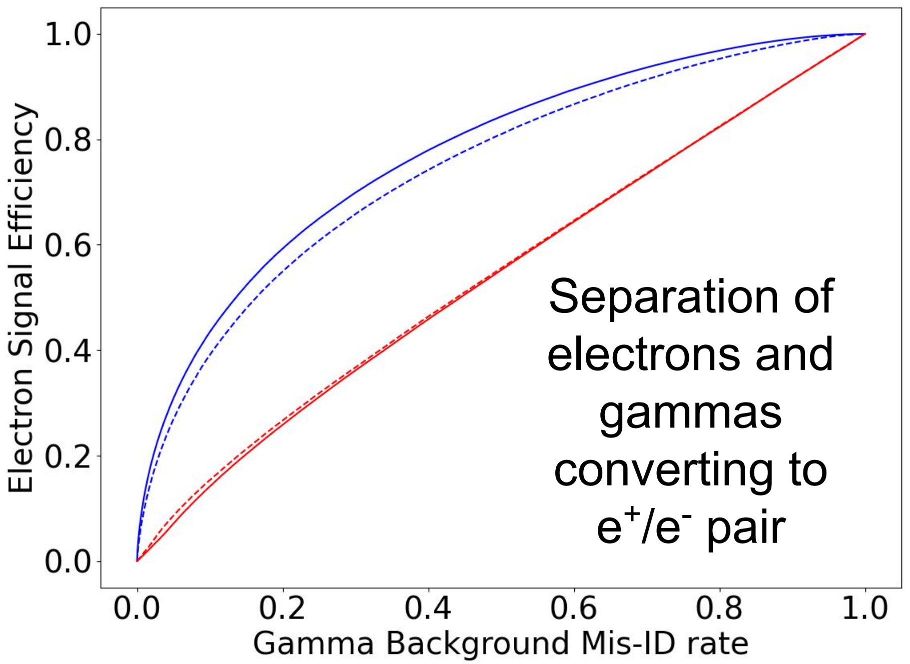


Basic cuts: >25 hits & fully contained events
All cuts: basic cuts & fiTQun fit converges & true position > 50cm from tank wall

ResNet performing better, particularly where fiTQun's likelihood model approximations fail close to detector wall

PID results: electron vs pi0 and gamma



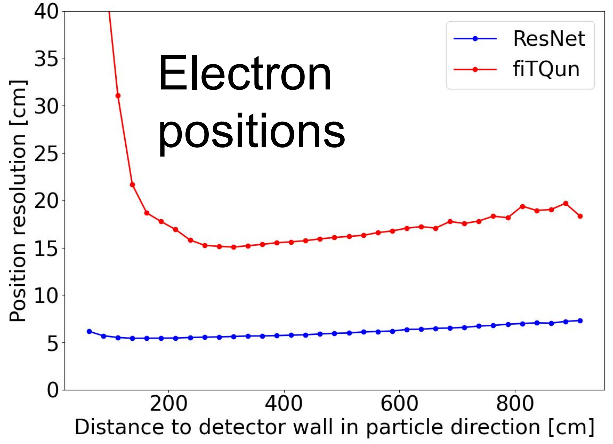


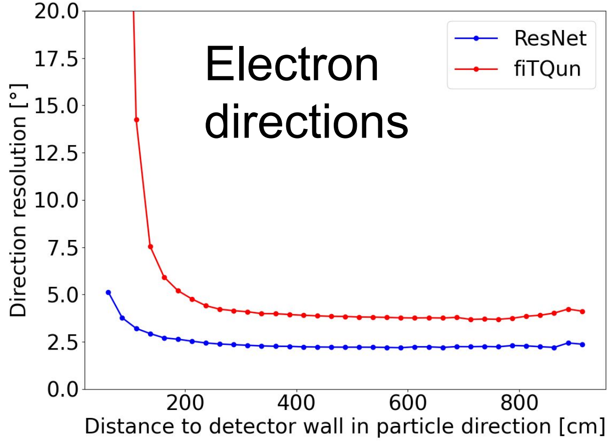
ResNet performing better, particularly at high energies where two rings from pi0 decay gammas overlap

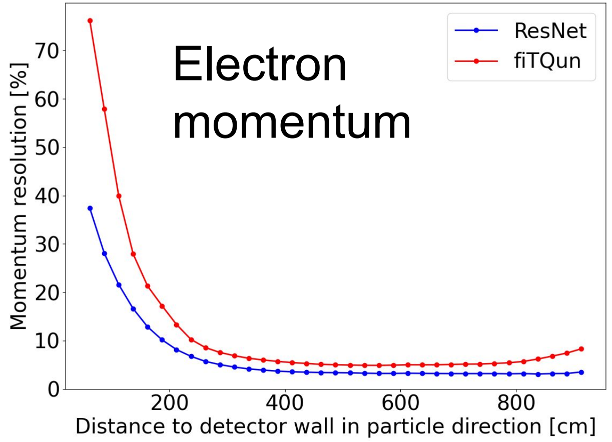
Separation of electrons and gammas never successfully used with fiTQun
Statistical separation of ~70% with ResNet

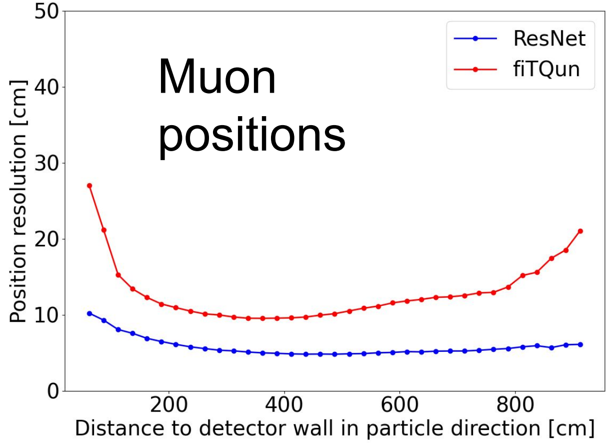
Regression results

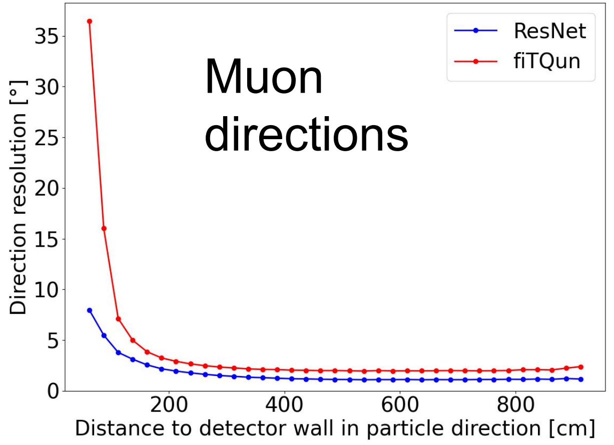
Reconstruction resolutions all reduced by 40 - 70%

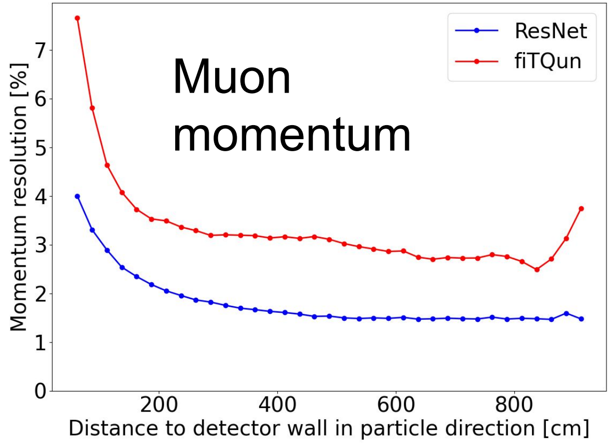






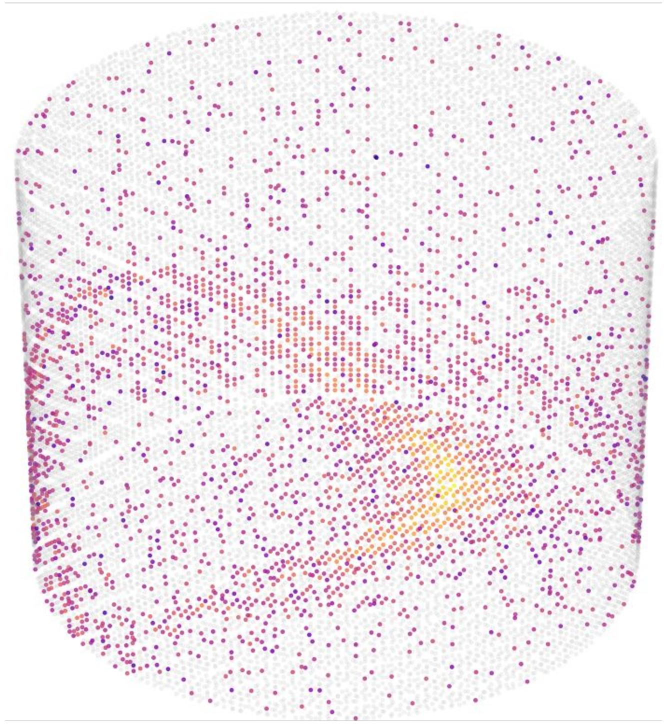






Challenges and plans

- Performance for IWCD is good - now surpassing traditional methods in all tasks we have trained
  - But relies very heavily on MC simulation
  - fiTQun also highly tuned to simulation, but designed based on physical principles
  - For ML, now planning to explore effect of imperfect simulation, to check reconstruction is robust and understand how to prevent learning artefacts of simulation
- HK far detector has more complex geometry
  - Checkerboard pattern, not simple grid of 20” PMTs
  - Mixture of 20” PMTs and multi-PMTs
  - Mapping to a 2D image should still be possible, currently working on this
    - Non-physical mapping may introduce bias in positions/directions
  - PointNet and Graph networks give more flexibility
    - Reconstruction is working but only similar or lower performance than fiTQun
- Reconstruction of more complex multi-ring events much more challenging
  - Need multiple networks or more complex network architectures



Machine learning enhanced simulation and calibration

Some new ideas for applying ML to detector simulation and calibration

Networks trained to correct for imperfect detector simulation

- Use ideas from image processing networks that modify images
- Train using real detector data to modify simulated data: remove noise, account for water quality, etc.

Real examples of ML-based image transformation

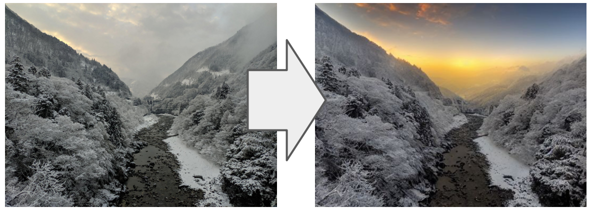
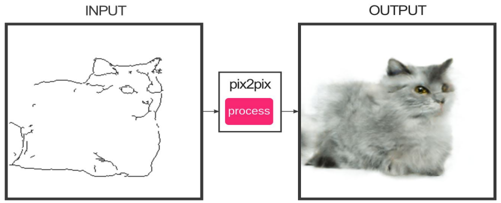


Photo from bridge near Super-K (real detector event)

“remove clouds and add sunset” (“remove dark noise”)

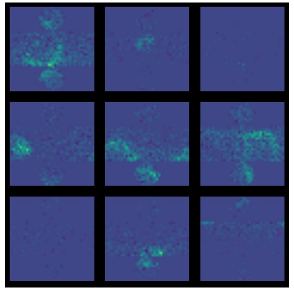


Drawing of cat (simulated event)

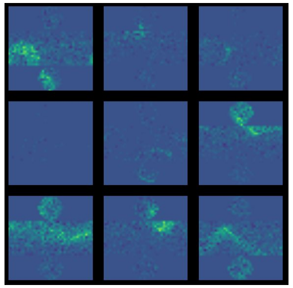
Realistic cat (looks like real data)

Differentiable detector simulation

- ML models can be trained to replicate detector simulations
- This ‘differentiable detector simulation’ can then be trained further on real data
- Use calibration data to train the simulation to account for detector effects



GAN generated events



Geant4 simulated events

First attempt at generating realistic events trained on WCSim events

Computing requirements for ML

Resource requirements and bottlenecks can be very different to traditional reconstruction

- fiTQun is *very* CPU intensive
  - Most resource intensive part of the entire HK software chain
- ML reconstruction is *very* fast to run on GPU
  - Bottleneck is usually the time taken to read data from disk (either local or network)
  - Usually run with a few CPU worker threads loading and preprocessing data to feed to GPU
  - Run in mini-batches of ~1000s events to be processed simultaneously in one iteration
  - One process runs over many minibatch iterations to minimise overhead of loading the model
  - With data on a fast local NVMe disk in efficient format, evaluating the ResNet-50 model
- Training ML models has slightly different requirements
  - Data needs to be read repeatedly in randomised order
    - Datasets often too large to fit in RAM (although newer servers with up to e.g. 1TB RAM can)
    - Move all data onto fast local disk before training
    - Use an uncompressed, unchunked data format like HDF5, stored as dense arrays of the data to be input to ML model
  - With fast enough disk, or enough RAM to store whole dataset, bottleneck becomes moving data between RAM and GPU VRAM
    - For more complex networks GPU processing itself is bottleneck
- For the ResNet-50 models, with data on a fast local NVMe disk in HDF5 format, using one NVIDIA A100 40GB and 4 CPU cores
  - Training one model (4-class PID, or a single reconstruction variable) takes ~ 48 hours
    - During development, repeat this many times e.g. with different hyperparameter configurations
  - Evaluating one model on 1M events takes less than 5 minutes