# ML for hKLM

Rowan Kelleher, Anselm Vossen

Duke University

October 28, 2025

# hKLM at Second Detector

- **hKLM**: Subsystem for 2nd Detector
    - Neutral hadron energy measurement
    - Neutron vs $K_L$ identification
    - Muon vs Pion identification
- **Machine Learning applications for reconstruction and performance measurements**

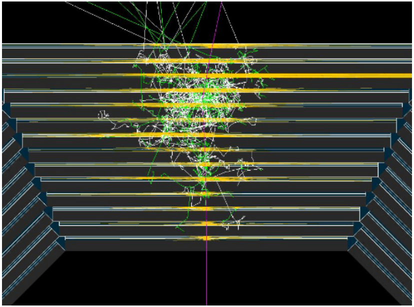
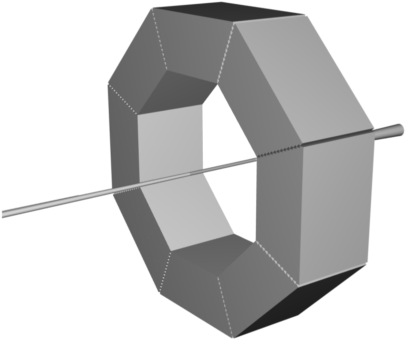


Figure: Charged hadron simulated event in hKLM detector

# hKLM Design

- **Barrel**: eight staves form around beam-pipe
- Alternating **iron-scintillator layers** within each stave
- SiPMs read out on each end of bar



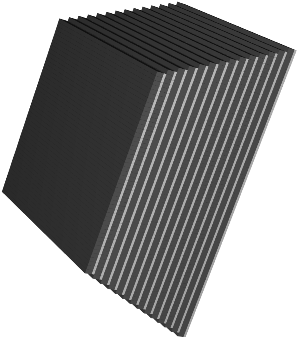
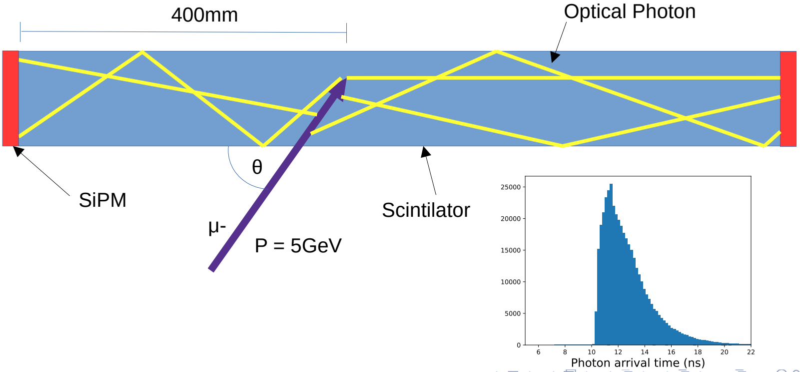


Figure: hKLM Sectors around beam pipe (left) and alternating iron-scintillator structure for one sector (right)

# Optical Photon Parameterization

- ~**10k optical photons** generated per charged track per layer
- Speed up: parameterize photon yield and timing using charged track information
  - angle, momentum, and position of hit on scintillator bar
- Timing follows unknown PDF: **learn with Normalizing Flow**



# Photon Timing

- **Normalizing flow** (NF): learns **transformation from normal distribution to unknown PDF**

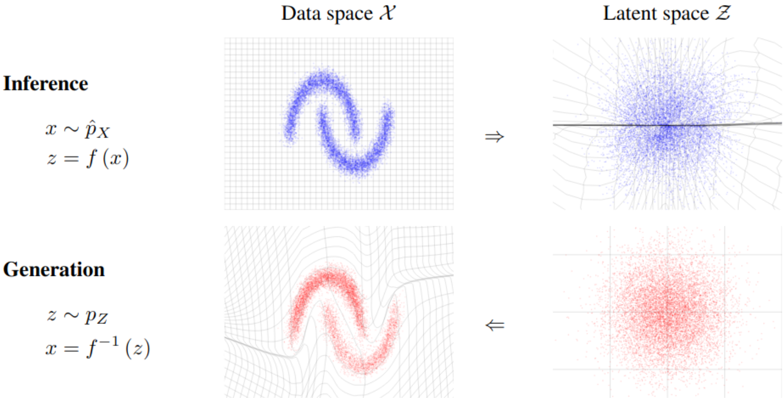



Figure: Normalizing Flow Transformation (image from Dinh et. al.)

# Photon Parameterization Results

- Good agreement between **NF and simulation**
  - First photon time distributions match
- Current resolution: $\sigma = 95ps$
- 20x faster than simulation

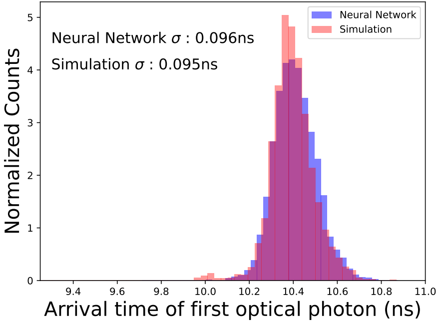
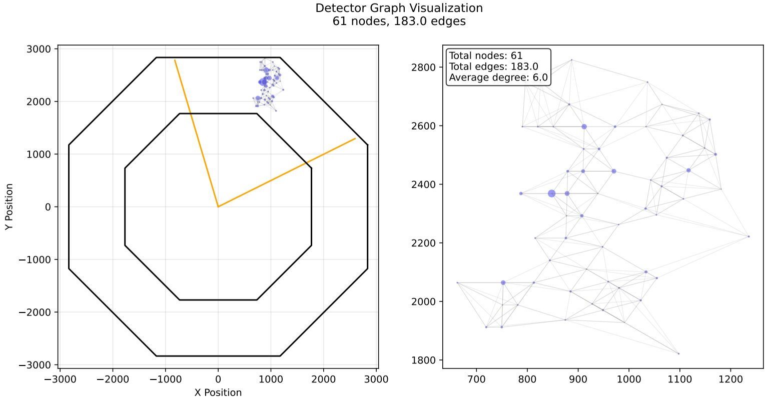



Figure: Histogram of first photon arrival time at SiPM for each event using simulation (red) and ML prediction (blue).

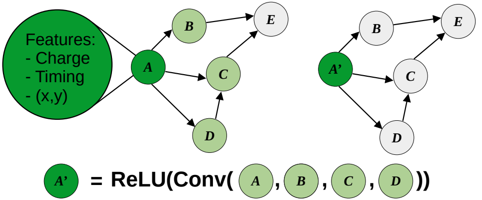
# Graph Neural Networks

- Event information:
  - **Event level**: # of hits, total charge, max charge
  - **SiPM level**: SiPM charge, SiPM timing
- Goal: predict particle energy or particle ID



# Graph Neural Networks (GNNs)

- **Input:** graph
- **Output:** class or value
- Useful for irregular input structures:
  - Detector responses with **varying numbers of hits**
- Message passing in GNNs:
  - Each node **considers its neighborhood** in the current layer when computing next layer representation



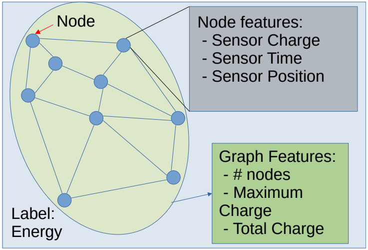




Figure: GNN convolution from one layer to the next (left) and detector response graph representation (right).

# GNN Results

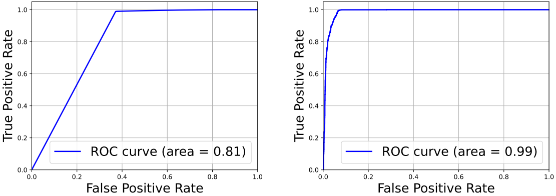



Figure: ROC curves for 5GeV MuID performance using conventional method (left) and GNN (right)

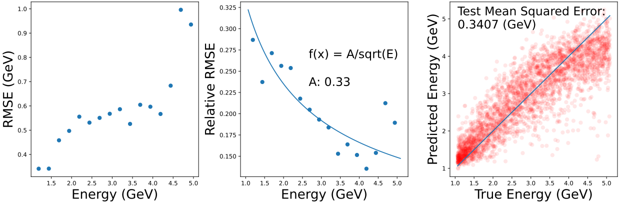



Figure: Neutron error vs Energy (left) relative error vs energy (middle) and predicted vs true energy (right)

# Optimization

- Utilize Bayesian optimization implemented by AID(2)E framework
- Provide set of geometry parameters
  - # layers, Iron, scintillator thicknesses
- Optimize for multiple objectives
  - **Maximize particle ID, momentum resolution**

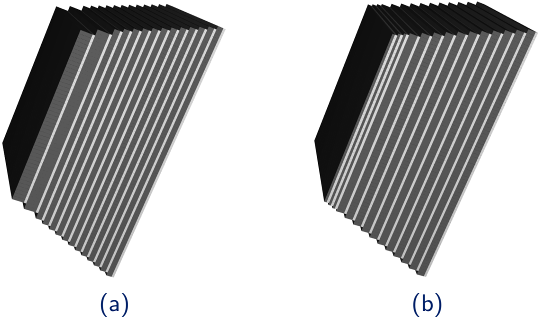
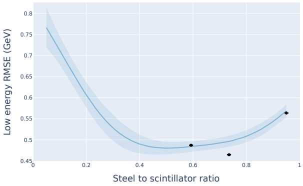
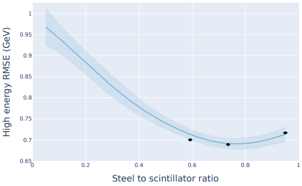



Figure: Geometry variations with large (a) and small (b) pre-shower steel layers

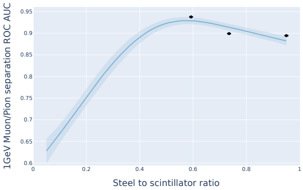
# Optimization Results

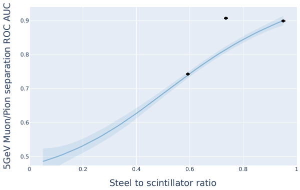
- MuID performance:



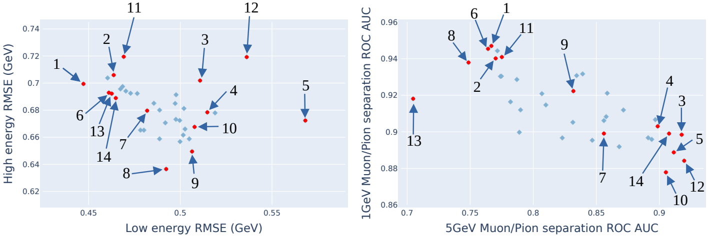

- Neutron energy performance:



# Optimization Results

- ML aided workflow:
  - Generate geometry with **AID(2)E**
  - Simulate detector response with DD4HEP
  - Sample optical photon arrival times with **NF**
  - Predict outcome with **GNN**
  - Report outcome to optimization framework, repeat
- Example: **tradeoff between high and low energy objectives**




**(Steel Ratio, Number of Layers):** 1. (0.63, 18) 2. (0.59, 16) 3. (0.95, 18) 4. (0.92, 18) 5. (0.95, 17) 6. (0.65, 17) 7. (0.82, 17) 8. (0.59, 17) 9. (0.77, 16) 10. (0.89, 18) 11. (0.56, 18) 12. (0.95, 15) 13. (0.50, 18) 14. (0.74, 14)

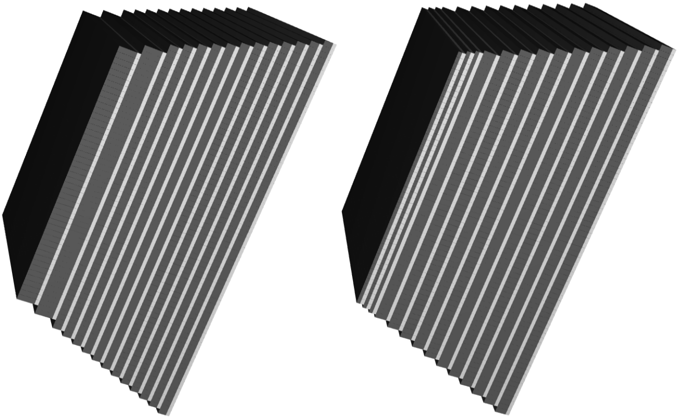
# References


- L. Dinh et al. "DENSITY ESTIMATION USING REAL NVP" (2017).

# Backup Slides

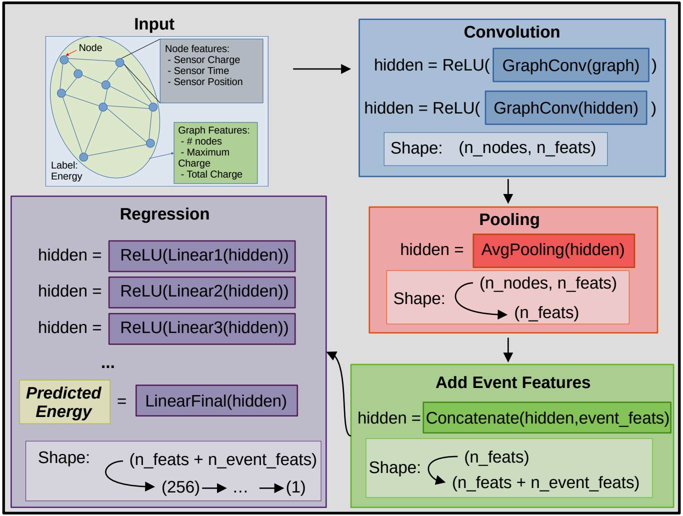
# Different Geometries

- Going beyond a uniform steel to scintillator ratio:
  - Pre-shower layers
  - Thicker (or thinner) steel in first N layers.



(a) Thick Pre-shower (b) Thin Pre-shower

# GNN Diagram



Input
Node
Node features:
- Sensor Charge
- Sensor Time
- Sensor Position
Graph Features:
- # nodes
- Maximum Charge
- Total Charge
Label: Energy
Convolution
hidden = ReLU( GraphConv(graph) )
hidden = ReLU( GraphConv(hidden) )
Shape: (n_nodes, n_feats)
Pooling
hidden = AvgPooling(hidden)
Shape: (n_nodes, n_feats)
(n_feats)
Add Event Features
hidden = Concatenate(hidden,event_feats)
Shape: (n_feats)
(n_feats + n_event_feats)
Regression
hidden = ReLU(Linear1(hidden))
hidden = ReLU(Linear2(hidden))
hidden = ReLU(Linear3(hidden))
...
Predicted Energy = LinearFinal(hidden)
Shape: (n_feats + n_event_feats)
(256) … (1)

# GNN Convolution

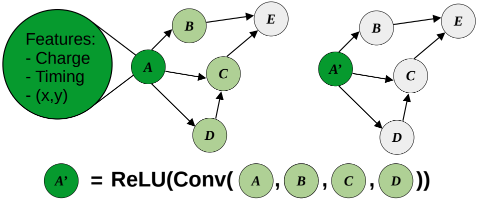
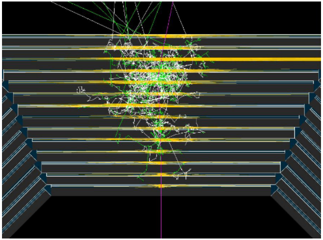


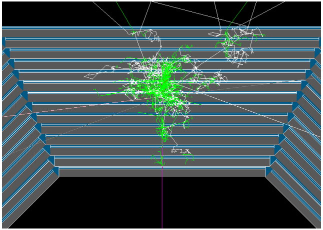

Figure: Depiction of GNN convolution from one layer to the next

# Simulation

- DDSIM program runs **Geant4 detector simulation** with Geometry built with DD4HEP, produces **ROOT file**
- Output includes:
  - Truth information about simulated particles
  - Optical **photon arrival time** at SiPM
  - **Energy deposited** by charged tracks in scintillator



(a) With Optical Photons

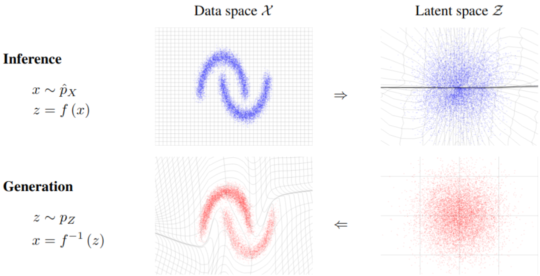


(b) No Optical Photons

Figure: DD4HEP Simulation Visualization

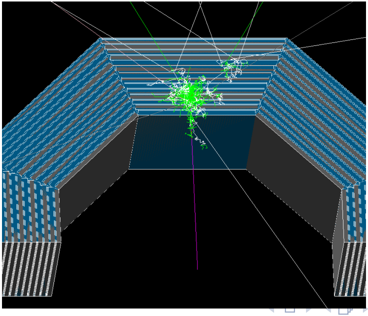
# Normalizing Flows

- Normalizing flows (NFs) are generative models that **learn the probability density function of a complex distribution.**
- NFs transform samples from a complex probability distribution to a simple distribution via a sequence of **invertible, differentiable functions.**
- Train network by **minimizing negative log-likelihood**



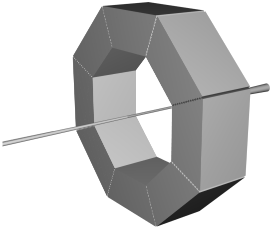
# Introduction to EIC KLM

- Proposed sub-detector for the EIC second detector for **muon and neutral hadron PID, timing and energy resolution**
- **Iron/scintillator sampling calorimeter** integrated into magnetic flux return, similar to Belle II K-Long muon detector (KLM)
- Simulation based optimization of detector geometry

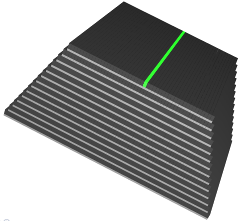


# Detector Geometry

- Staves arranged in polygon barrel around inner detectors
- Super-layers within staves contain **one iron and one scintillator layers**
- Layers have **vertical segmentation** allowing for SiPM placement on ends
- Dimensions: up to optimization!
- Implementation: **DD4HEP** toolkit integrated with **Geant4** simulation program



(a) Full Barrel



(b) Individual Sector

# Future Applications

- MC Affinity for **EIC projections** utilizing existing EIC Monte-Carlo Simulation software
- Di-hadron affinity: non-trivial to apply affinity model to di-hadrons
  - Treat di-hadron as $P_h$ and calculate each $R_i$ for the single dihadron
  - Require each hadron to fall within TMD region, i.e. $R_i < 0.3$ for each hadron
  - Multiply affinities of each hadron
- Input **learned partonic distributions** into method 1
  - Model probabilities with sum of gaussians
  - Utilize Normalizing-Flow neural networks to learn the probabilty density function of each parameter
  - Replace normal distributions with learned distributions

# Photon Yield

- Photon yield: calculate analytically
- Total photons produced: $N_\gamma = 10 * EDep_{KeV}$
- Fraction of photons reaching SiPM: $R = \frac{A}{x+B} + C$ where x is the hit distance from SiPM, A,B,C fit from data

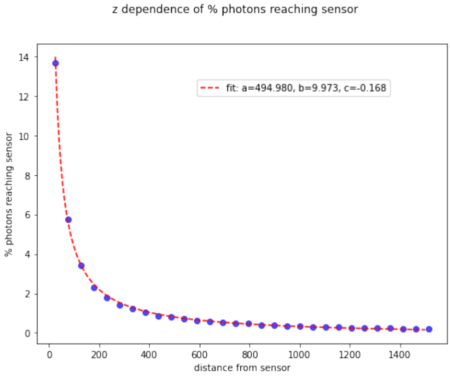



Figure: Position Dependence of Photon Yield

# DDSIM

- MCParticles bank provides direct access to MCParticles in Geant4 simulation
- HcalBarrelHits bank provides information about:
  - Optical Photons hitting sensor connected to SiPM readout
  - Charged particles producing optical photons in scintillator layers

# Change of Variables, architecture

### 3.1 Change of variable formula

Given an observed data variable $x \in X$, a simple prior probability distribution $p_Z$ on a latent variable $z \in Z$, and a bijection $f : X \to Z$ (with $g = f^{-1}$), the change of variable formula defines a model distribution on $X$ by

$$p_X(x) = p_Z\big(f(x)\big)\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right| \tag{2}$$

$$\log\left(p_X(x)\right) = \log\Big(p_Z\big(f(x)\big)\Big) + \log\left(\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|\right), \tag{3}$$

where $\frac{\partial f(x)}{\partial x^T}$ is the Jacobian of $f$ at $x$.

Figure: realNVP change of variables (Dinh et. al.)

simple bijections as an *affine coupling layer*. Given a $D$ dimensional input $x$ and $d < D$, the output $y$ of an affine coupling layer follows the equations

$$y_{1:d} = x_{1:d} \tag{4}$$

$$y_{d+1:D} = x_{d+1:D} \odot \exp\big(s(x_{1:d})\big) + t(x_{1:d}), \tag{5}$$

where $s$ and $t$ stand for scale and translation, and are functions from $R^d \mapsto R^{D-d}$, and $\odot$ is the Hadamard product or element-wise product (see Figure 2(a)).

Figure: Architecture (Dinh et. al.)

# Clustering in SIDIS Events

- Started looking at response to SIDIS events
- Limit input particles to Kaons and Muons
- Treat each particle produced within solenoid radius as a primary

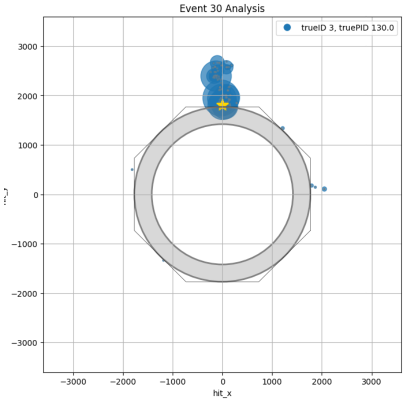



Figure: Hadronic Event: K_L interacted with solenoid/

# Decay Event

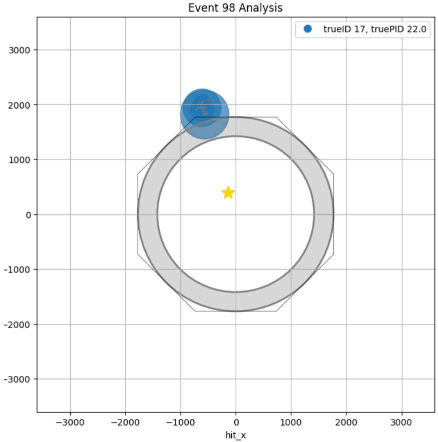
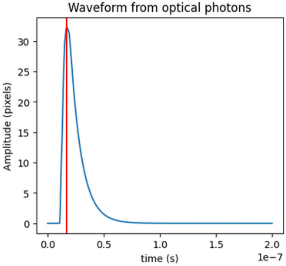



Figure: Decay Event: Kaon (gold star) decayed, and a photon produced hits in the detector

# SiPM Modeling

- **Optical Photon arrival times not given by detector**
- Actual detector output: electrical signal from SiPM
- Observables:
  - Integrated charge
  - Time of pulse peak

$$V(t) = V_0(1 - e^{-t/\tau_r}) \cdot e^{-t/\tau_f}$$



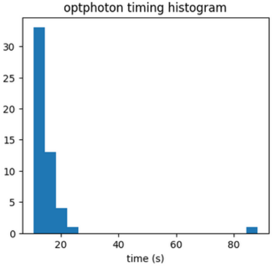




Figure: Optical photon actual arrival times (right) and modeled waveform (left)