Determination of H2 CMS Barrel Test Beam Calorimeter Response Correction To Pion Beams with Deep Neural Networks

Daniel Li[1], Sergei Gleyzer[2], Emanuele Usai[1], Meenakshi Narain[1], Ulrich Heintz[1], Sitong An[3,4], Jason Terry[1], Andrew Dabydeen[1]

[1]Brown University (US), [2]University of Florida (US), [3]Carnegie Mellon University (US), [4]CERN (CH)

Outline

1. Test Beam Setup and Dataset
2. Beam Distribution Corrections – Analytic Method
3. Beam Distribution Corrections – Neural Network Method
   i. Machine Learning Motivation
   ii. Model Features
   iii. Parameterization
   iv. Results
4. Summary and Conclusions

Background

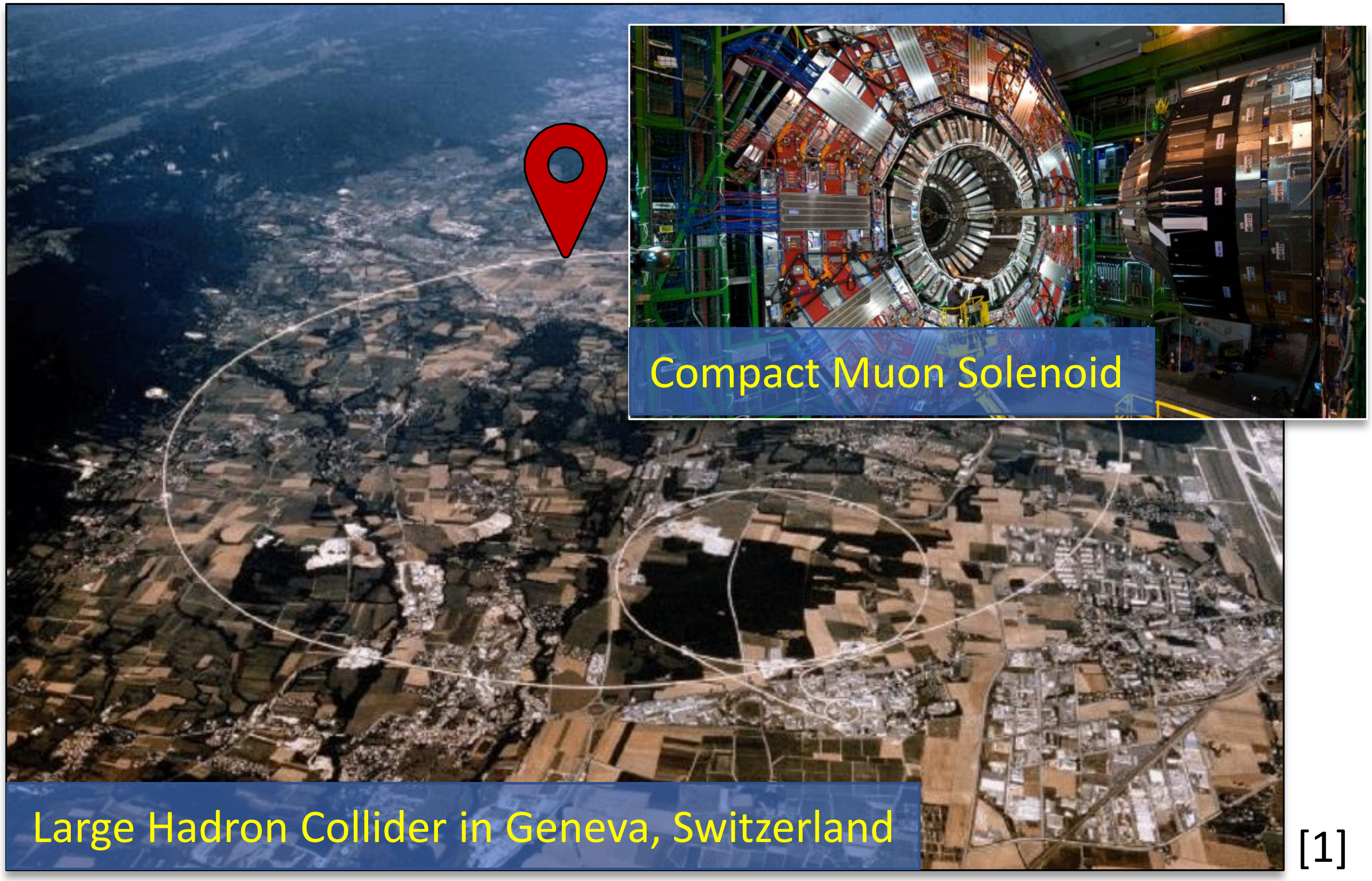
Compact Muon Solenoid (CMS) is one of the general-purpose physics detectors at the Large Hadron Collider (LHC)

- ❑ $13\ \mathrm{TeV}$ proton-proton collisions

CMS detector consists of four layers:

1. Inner Tracker
2. Electromagnetic Calorimeter (ECAL)
3. Hadronic Calorimeter (HCAL)
4. Outer Tracker

Focusing on the *ECAL* and *HCAL*



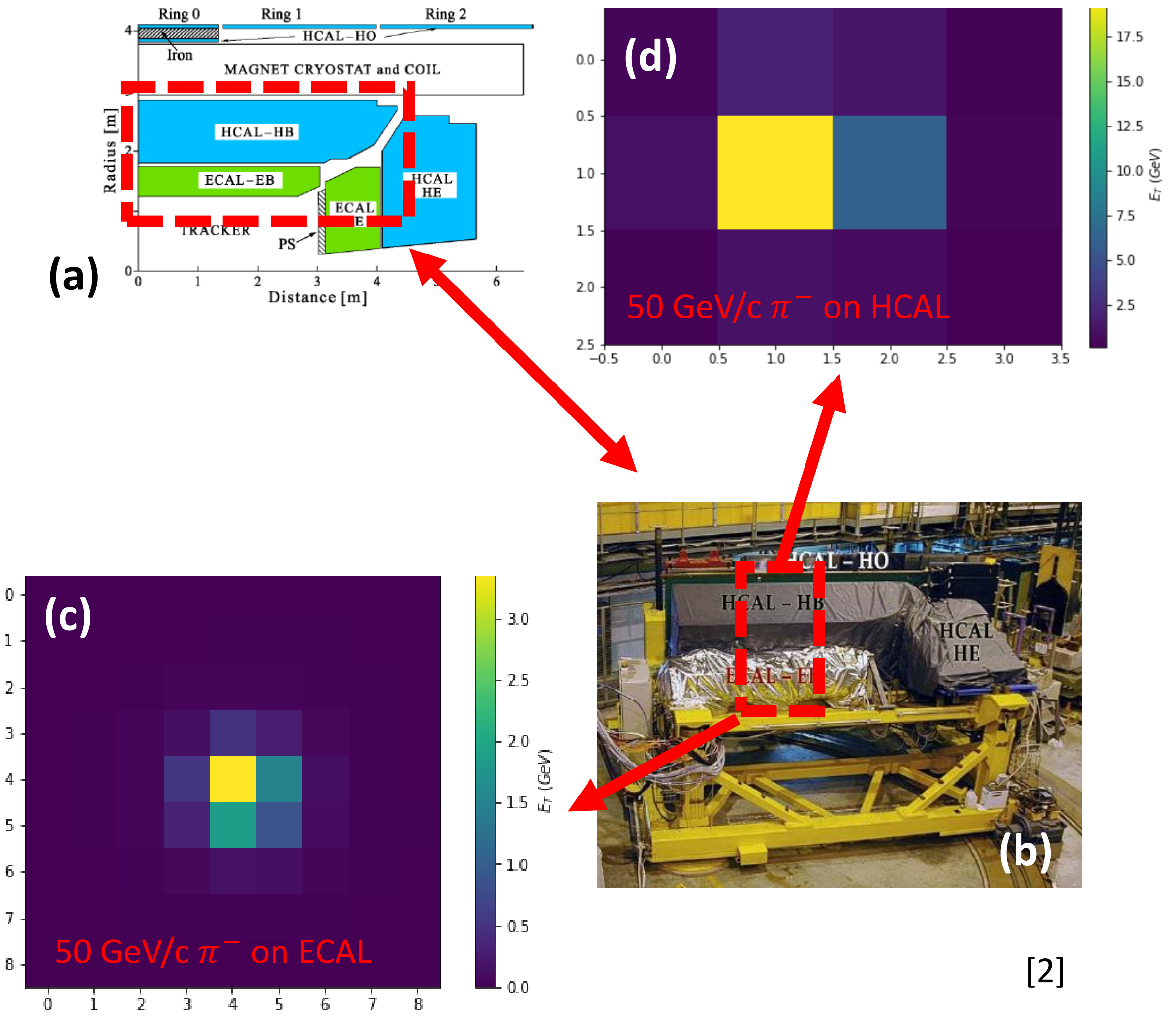
[1]

Test Beam Setup and Dataset

H2 CMS ECAL and HCAL as detectors
- ECAL: 9x9 crystals (Figure c)
- HCAL: 3x4 towers (Figure d)

π^- beam incident upon ECAL and HCAL ranging from 2 to 300 GeV/c (nominal momenta)
- 2, 3, 4, 5, 6, 7, 8, 9, 20, 30, 50, 100, 150, 200 and 300 GeV/c

Data is reconstructed energy images
- ECAL Energy + HCAL Energy ≠ Nominal Energy



[2]

Analytic Corrections

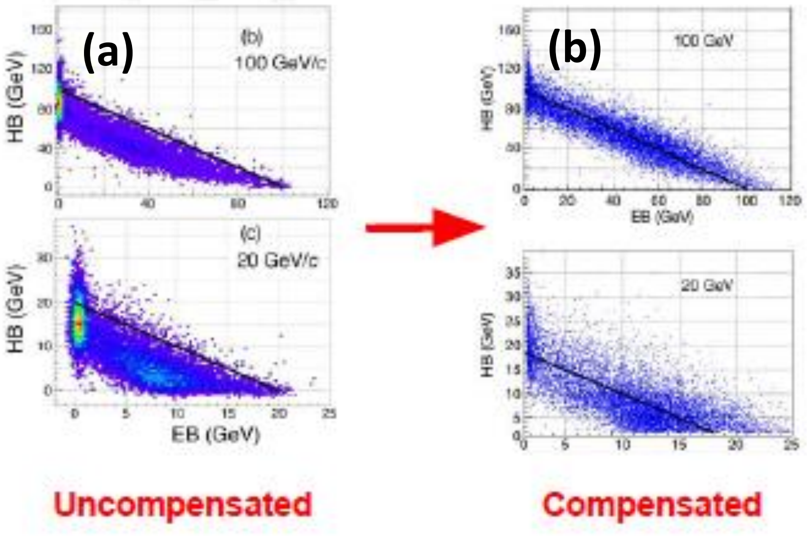
(Abdullin et al, 2009): event-by-event raw energy corrections can be obtained by compensating the raw energy response

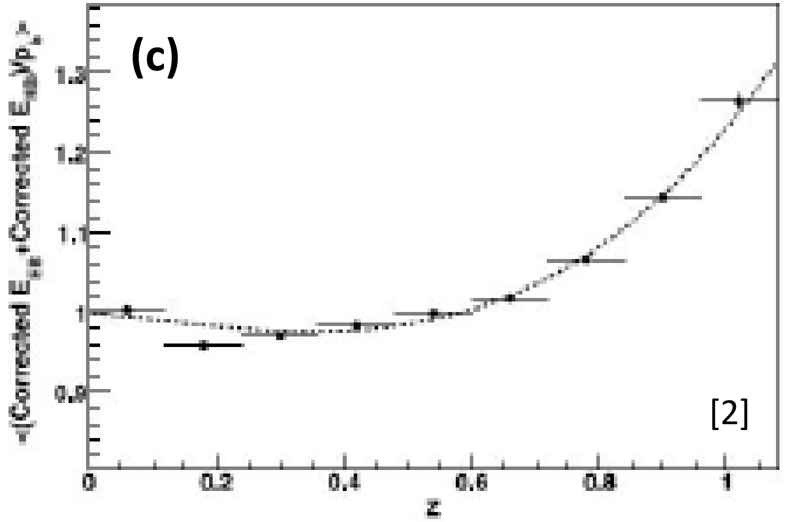
- *Raw energy = (7x7 ECAL energy sum) + (3x3 HCAL energy sum)*

Parameterization of the corrected raw energy $(E^*_{ECAL} + E^*_{HCAL})$ uses a third-order nonlinear function of $Z = E_{ECAL}/(E_{ECAL} + E_{HCAL})$ (Figure C)

Energy resolution is computed from the mean and RMS of a Gaussian fit about the parameterized beam distributions

- Only data between 5 to 300 GeV/c is fit





Machine Learning Motivation

What are the necessary ingredients for training a neural network?

- ✓ **Large dataset:** thousands of events for each nominal beam energy (~6000)
- ✓ **"Truth" values:** nominal beam energy (2 GeV, 3 GeV,…300 GeV)

Use neural network to learn differences in images with similar domains

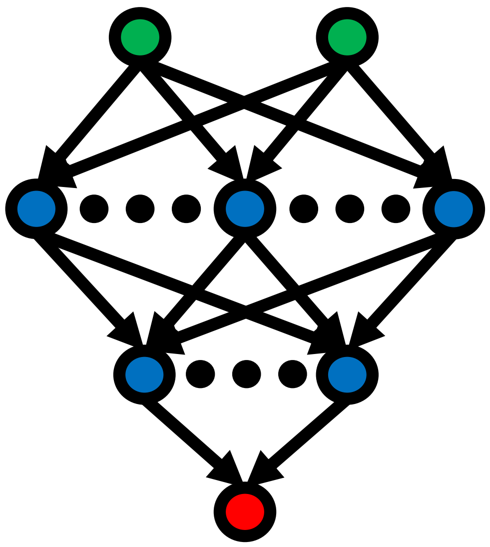
- ❑ Bypass dependence on prior-knowledge
- ❑ Direct dependence on energy response
- ❑ Scales naturally with arbitrary data complexity

We train convolutional and dense neural networks that apply event-by-event corrections to the raw energy

- ❑ Results compared to Abdullin et al [2] for reasonability

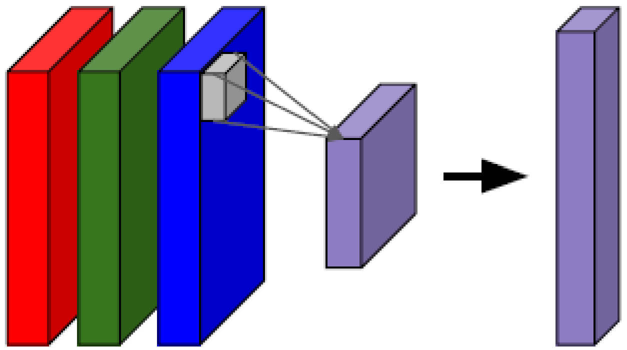
Dense Neural Network

- Three types of layers:
  1. Input (number of pixels)
  2. Hidden (custom)
  3. Output (corrected energy)



Convolutional Neural Network

- Four types of layers:
  1. Input (EB and HB image)
  2. Convolution/Pooling (custom)
  3. Hidden (custom)
  4. Output (corrected energy)



Neural Network Model Features

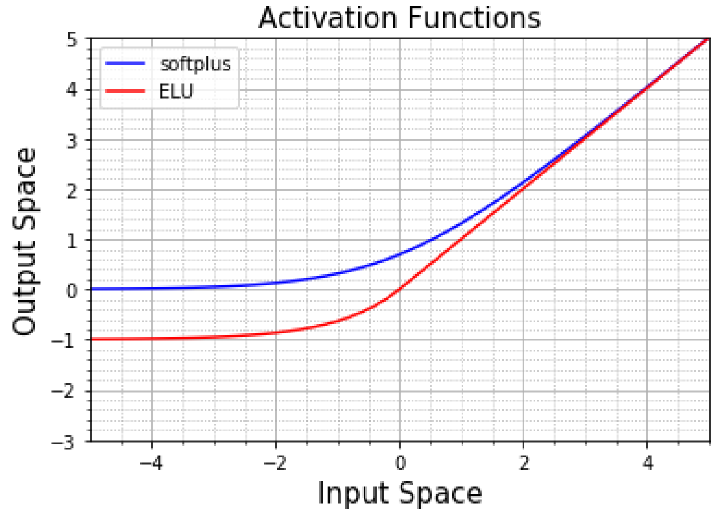
A model architecture is characterized by its hyper parameters:

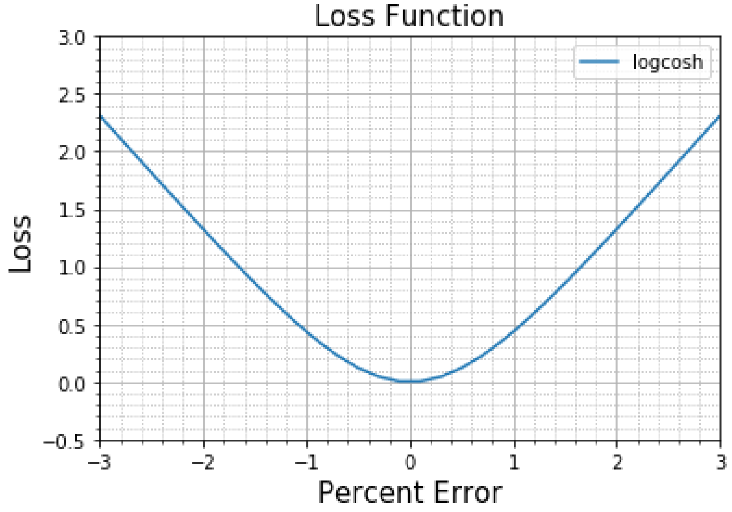
Hyper Parameter	CNN	DNN
Batch Size		
Dropout		
Dense Layer		
Initial Nodes		
Convolutional Layer		
Kernel Size		
Filter Size		
Activation Function		
Optimizer		
Learning Rate		
Loss Function		
Patience		

**The high-lighted hyper parameters were optimized using Bayesian optimization

Neural Networks for Energy Correction

Hyper Parameter	CNN	DNN
Optimizer	Adam	Adam
Loss Function	% logcosh	% logcosh
Dense Activation Function	Softplus	Softplus
Convolutional Activation Function	ELU	--





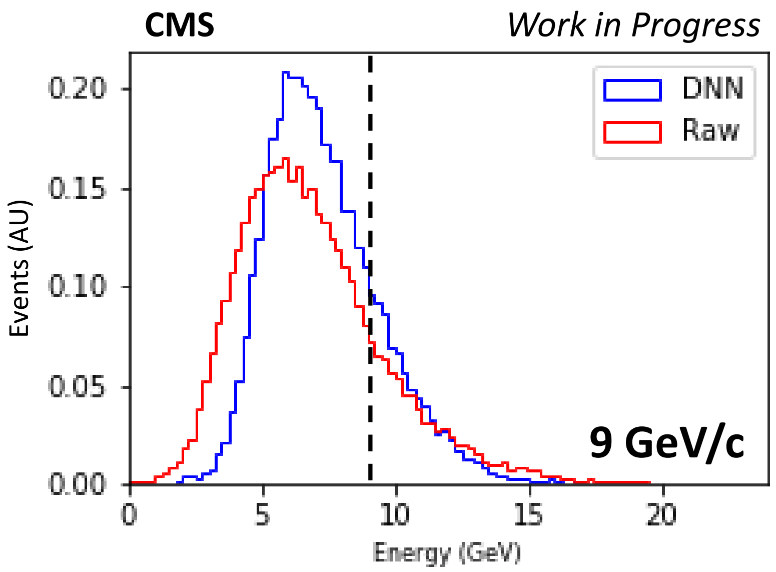
- ❑ Adam is a gradient-based stochastic optimizer designed for training deep neural networks
- ❑ Softplus maps negative values into a positive output space and has non-vanishing gradient
  - ❑ Data contains negative inputs after subtracting off the pedestal
- ❑ ELU decreases negative inputs and has a non-vanishing gradient
- ❑ Patience is the number of epochs before early stopping

Energy Response Parameterization

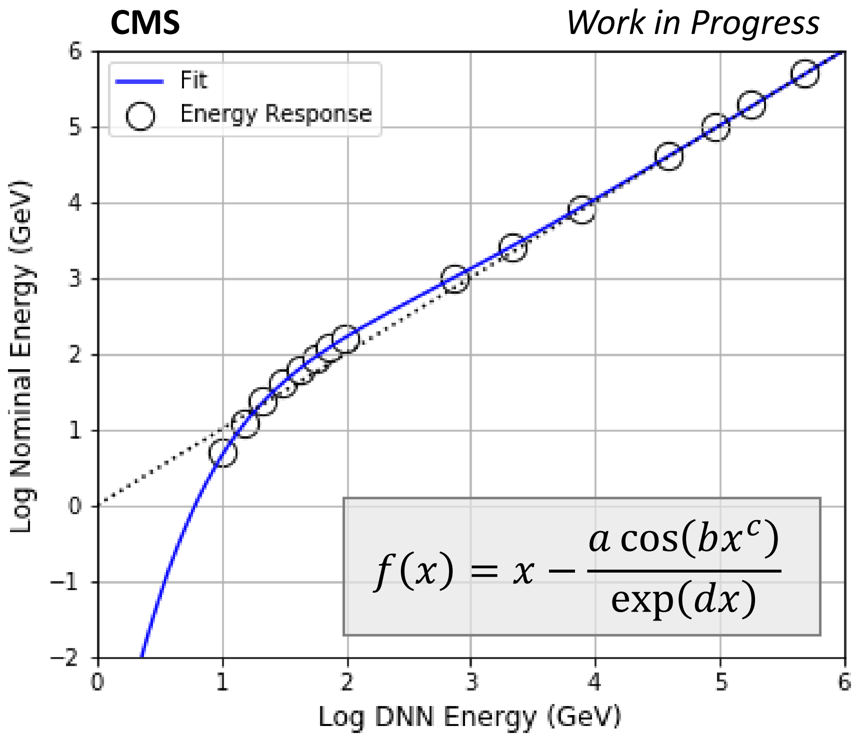
Trained network has a bias in the low energy ($\leq 20\ GeV$) which is parameterized

- ❑ Parameterization applied event-by-event to the predicted data

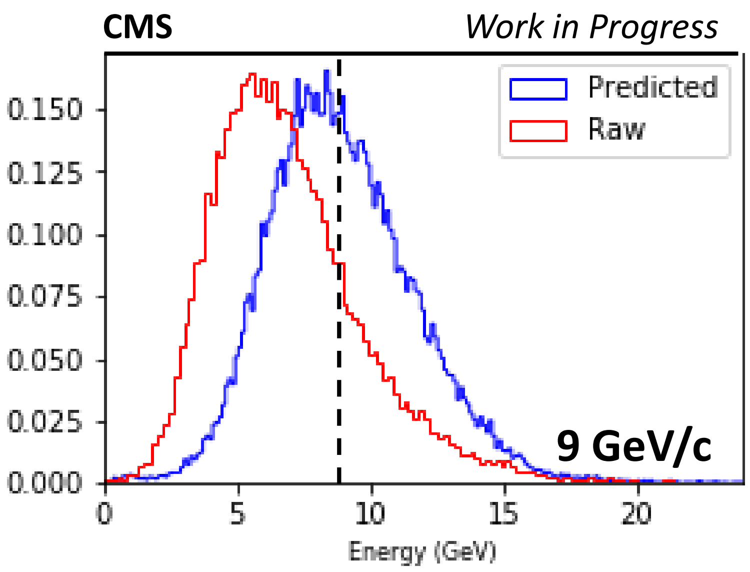
1. Neural Network Output



2. Parameterize Energy Response

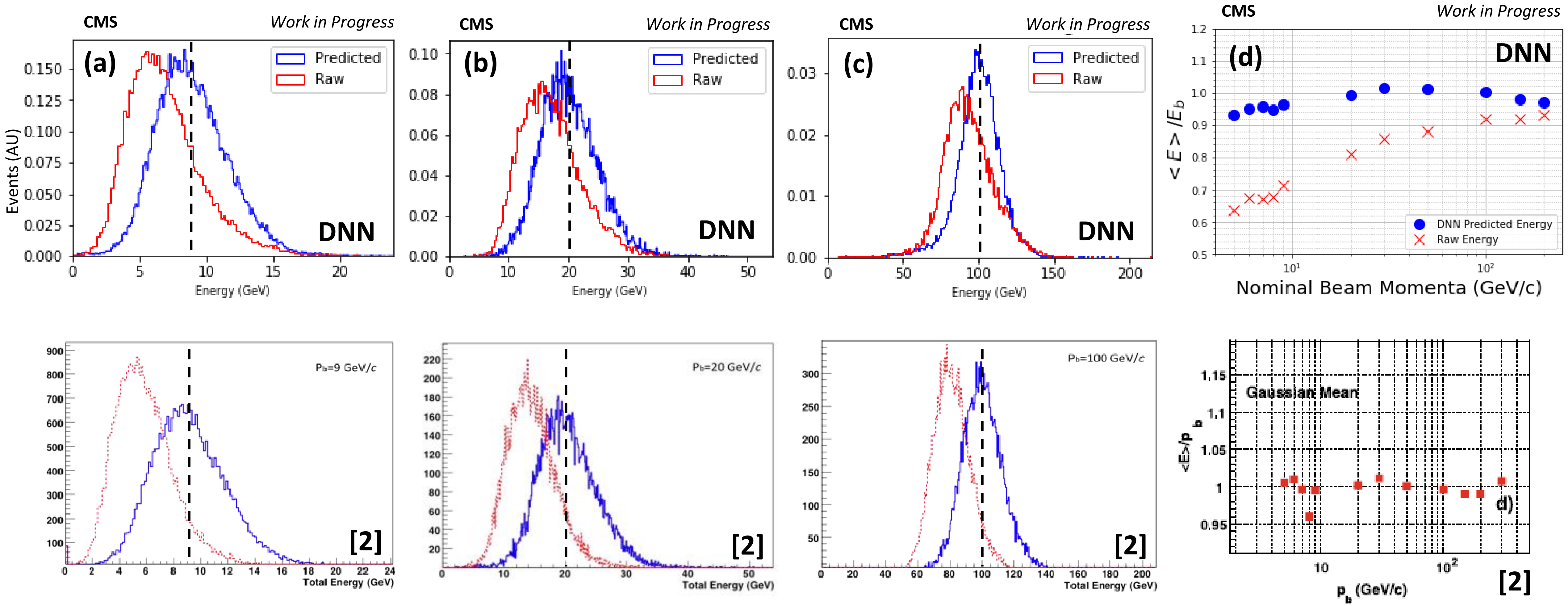


3. Event-by-event Correction



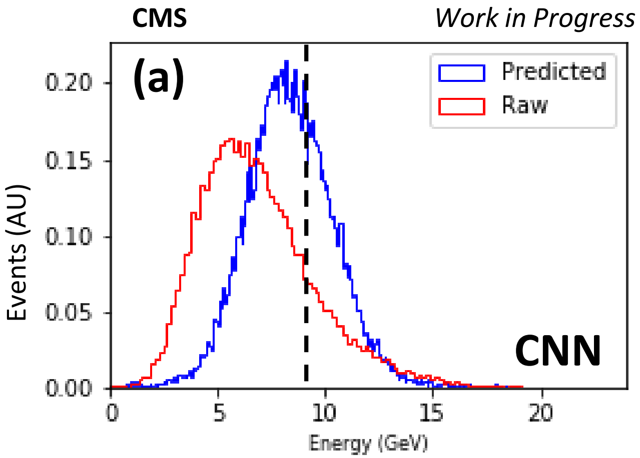
Energy Distribution Corrections: DNN

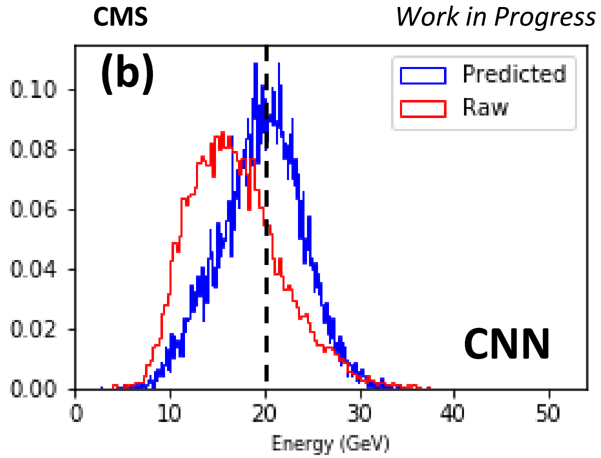
9 GeV/c | **20 GeV/c** | **100 GeV/c**

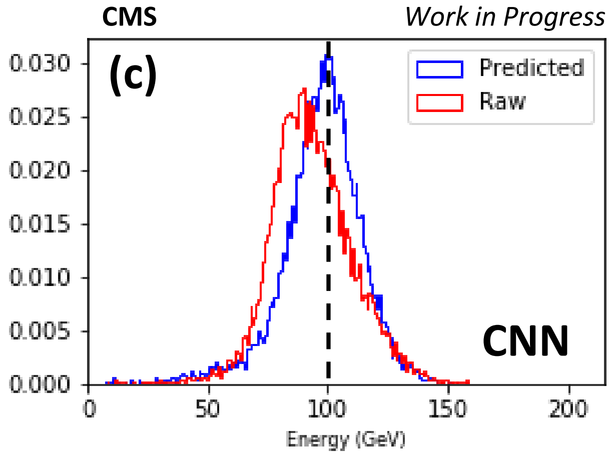


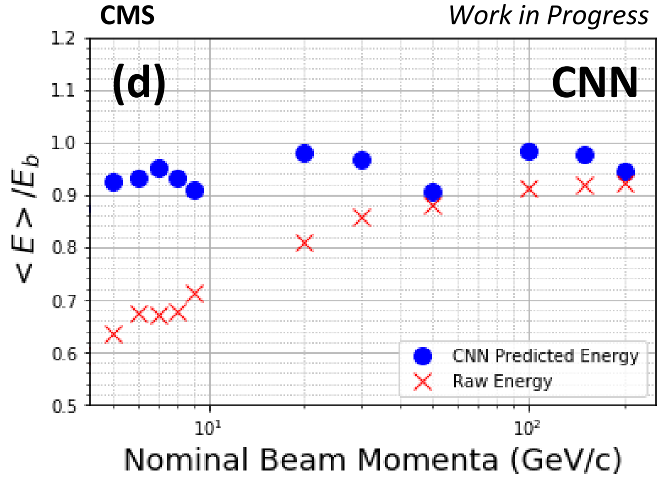
Energy Distribution Corrections: CNN

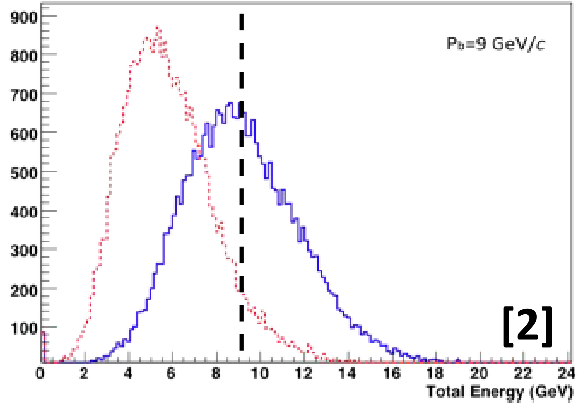
9 GeV/c | **20 GeV/c** | **100 GeV/c**

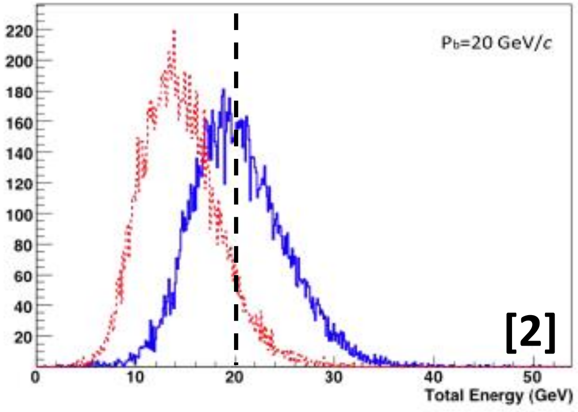


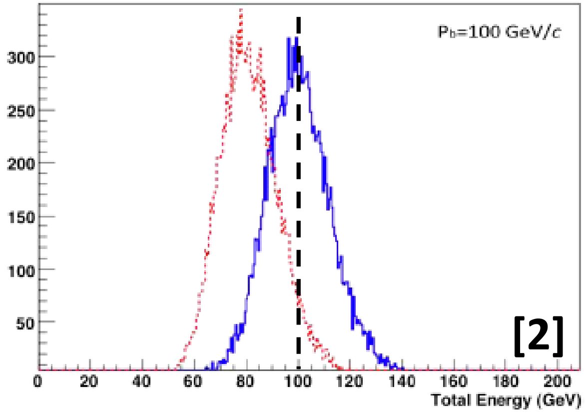


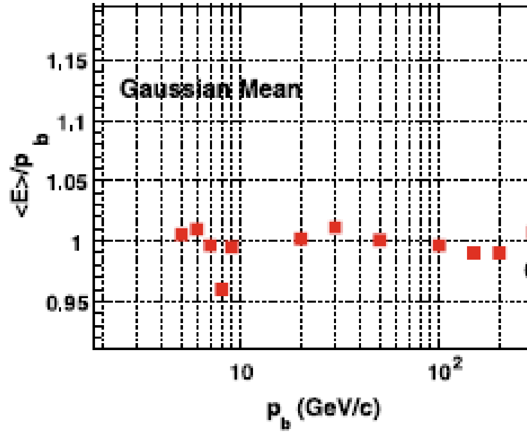






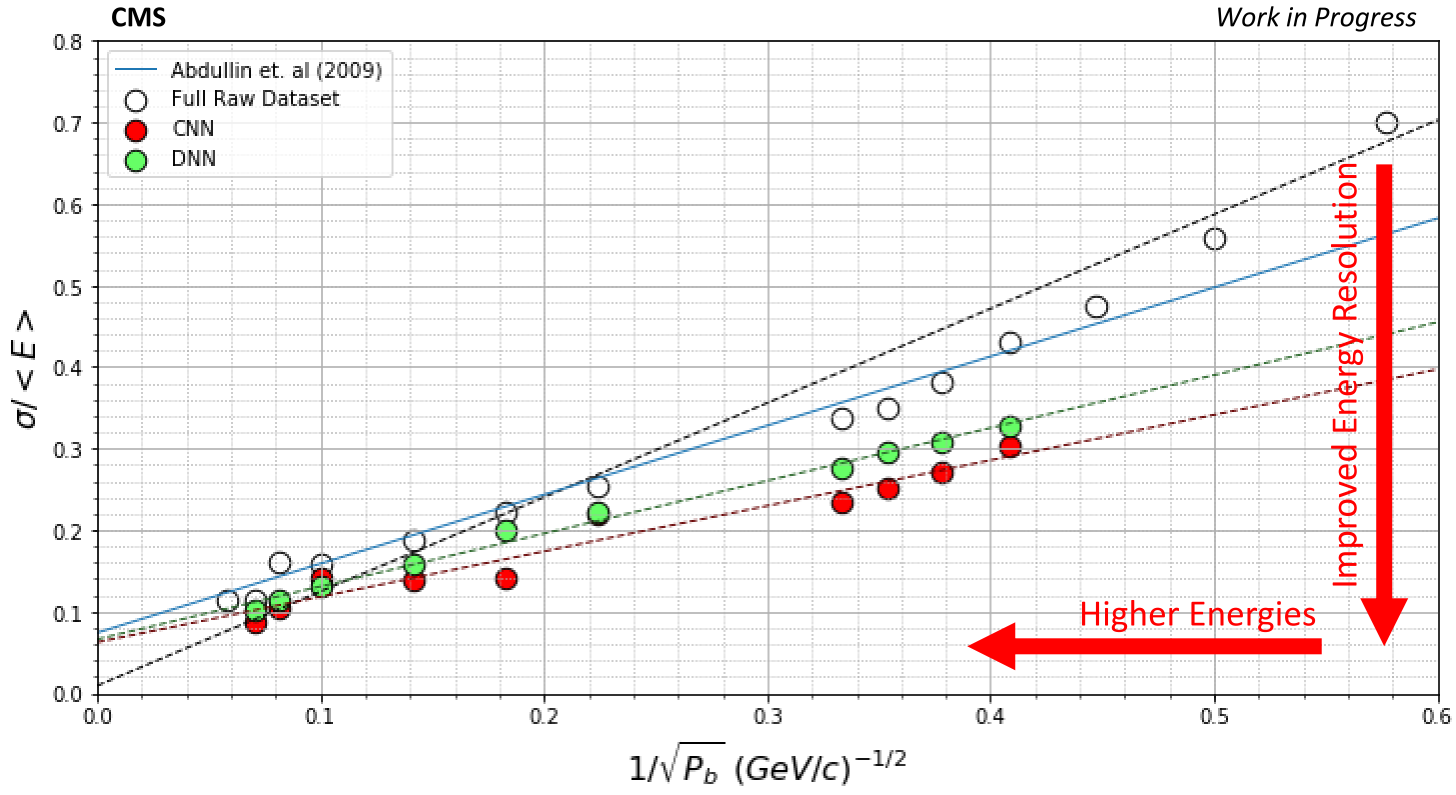


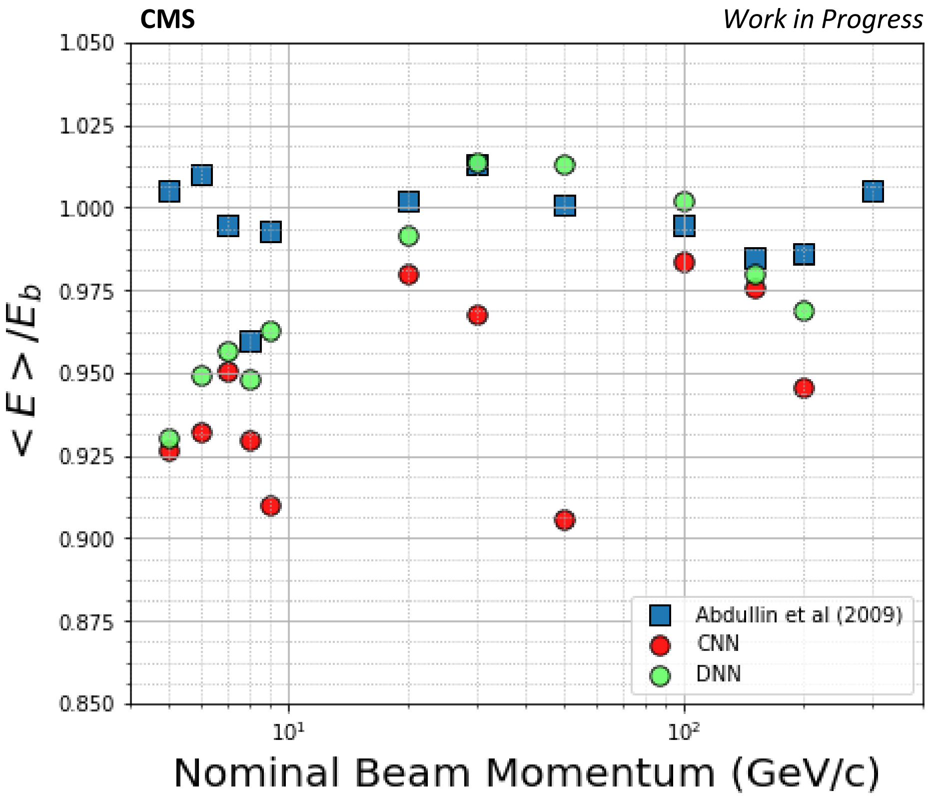




[2]

Energy Resolution and Response Results





Method	Regression	Mean Ratio
CNN	$0.558/\sqrt{P_b} \oplus 0.063$	0.946 ± 0.027
DNN	$0.648/\sqrt{P_b} \oplus 0.066$	0.974 ± 0.027
Abdullin et al. [1]	$0.847/\sqrt{P_b} \oplus 0.074$	0.996 ± 0.014

Interpolation Performance (1)

For application, model must perform well on datasets not exposed to during training

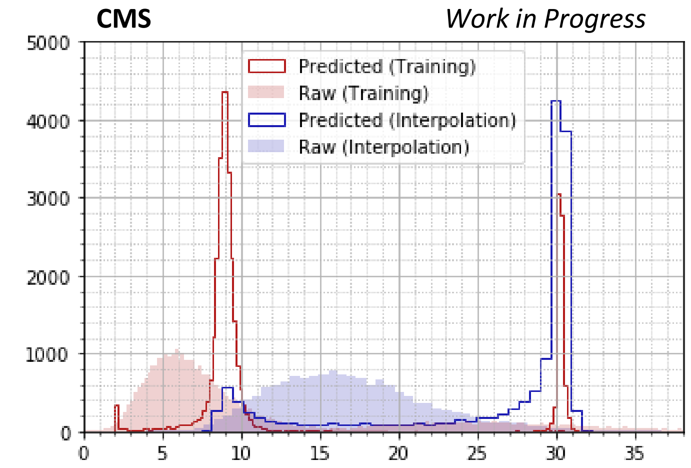
- ❑ Sparse variety → classification

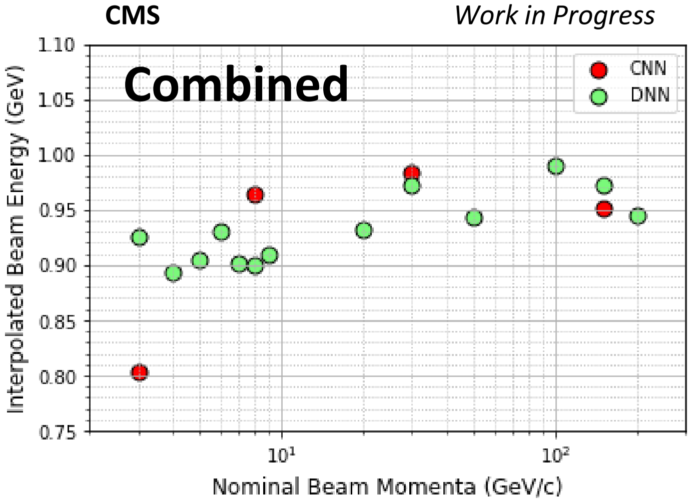
All-but-one training and interpolate on the excluded dataset to check for regression

- ❑ Beam distributions for 150 GeV DNN and CNN retain the energy distribution form

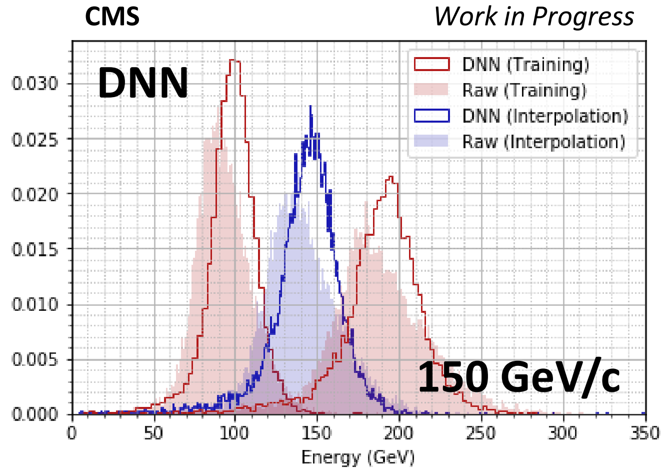
The lower left plot shows the energy response of the interpolated values

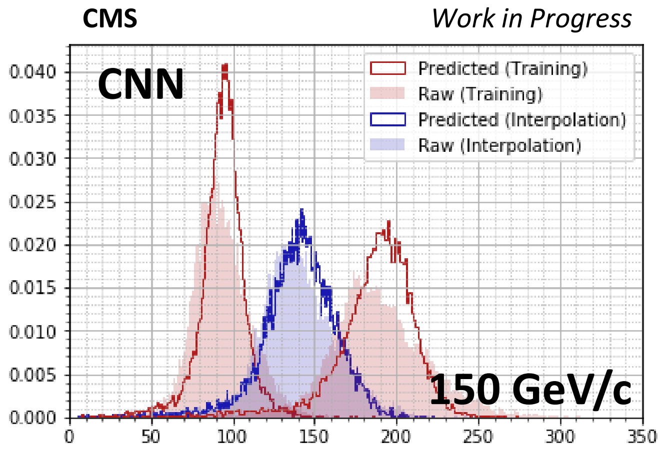
Poor Interpolation





Better Interpolation

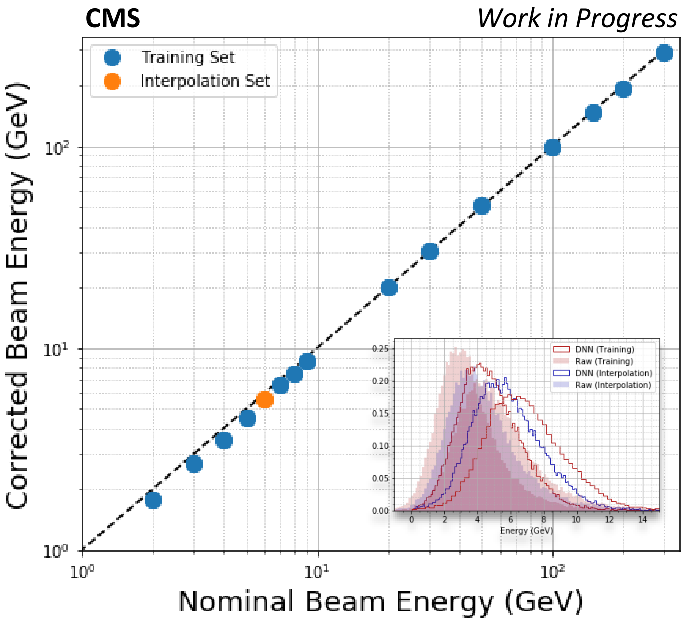


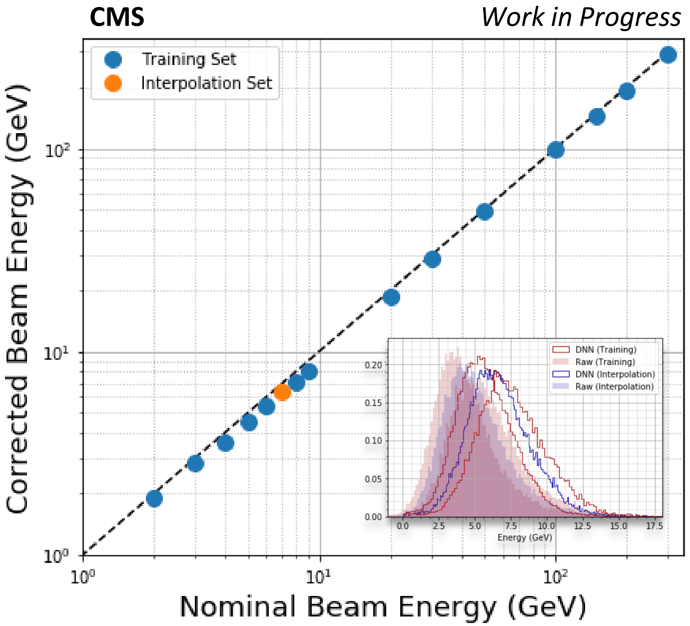


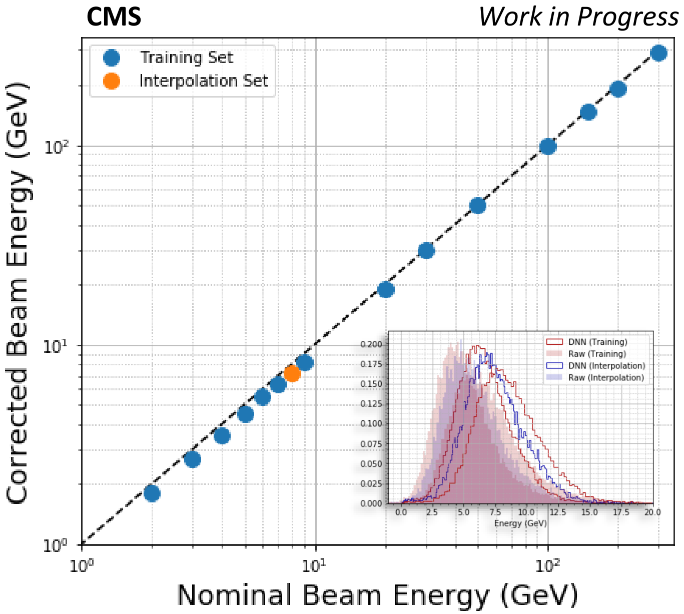
Interpolation Performance (2): DNN

6 GeV/c | **7 GeV/c** | **8 GeV/c**

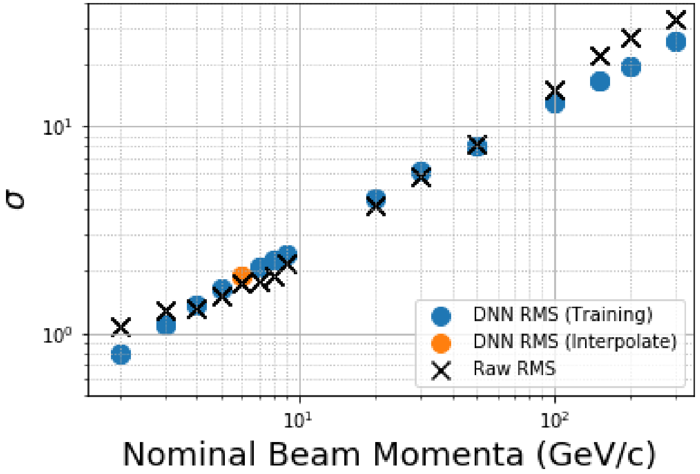
Mean

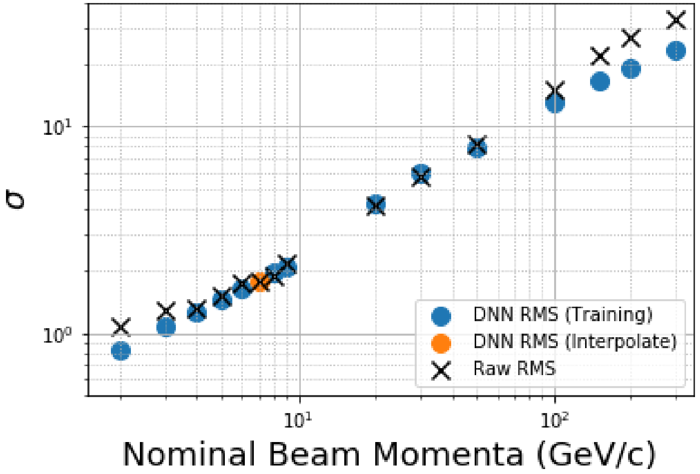


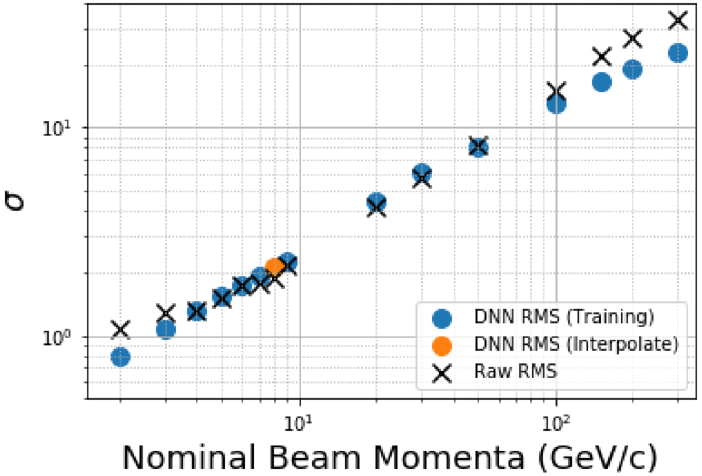




RMS







Summary and Conclusion

Different forms of neural networks (CNN and DNN) can be trained to predict the true calorimeter energy response

- By parameterizing the network predictions, corrections can be further improved
- Training is independent of contextual knowledge and energy dependent

The neural network models require a training dataset that has sufficient overlap between neighboring energy's beam distributions for interpolation

This is a work in progress which can be further improved by using simulation data at select energies

Thank You for Listening!

Questions?

References

[1] cds.cern.ch

[2] S Abdullin et al., Eur. Phys. J. C **60**, 359 (2009)

[3] The CMS Collaboration, "Observation of Higgs boson decay to bottom quarks," PRL **121** (2018)

Analytic Correction Process

1. Parameterize $\langle \pi/e_{HB} \rangle$ as a function of E_{HB} using either Wigmans' parameterization ($> 8\ GeV/c$) or a logarithmic function ($\leq 8\ GeV/c$),

$$\langle \pi/e_{HB} \rangle = \frac{1 + (e/h - 1) \times 0.1 \log(E_{HB})}{e/h}, \qquad (E_{HB} > 8\ \ GeV/c)$$

$$\langle \pi/e_{HB} \rangle = 0.179 \pm 0.005 \log(E_{HB}) + 0.413 \pm 0.005, \qquad (E_{HB} \leq 8\ \ GeV/c)$$

2. Parameterize $\langle \pi/e_{EB} \rangle$ as a function of E_{EB} using,

$$\langle \pi/e_{EB} \rangle = \frac{\langle E_{EB} \rangle}{E_b - E_{HB}^*}$$

$$E_{HB}^* = E_{HB}/(\pi/e_{HB}), \qquad E_{EB}^* = E_{EB}/(\pi/e_{EB})$$

3. Determine the corrected responses, E_{HB}^* and E_{EB}^*, which can then be parameterized to provide the corrected compensated response as a function of $Z = E_{EB}/(E_{EB} + E_{HB})$,

$$\left\langle \frac{E_{EB}^* + E_{HB}^*}{E_b} \right\rangle = (0.412 \pm 0.045)Z^3 - (0.096 \pm 0.058)Z^2 - (0.084 \pm 0.018)Z + 1.00$$

Neural Network Model Features

A model architecture is characterized by its hyper parameters:

Hyper Parameter	Description	CNN	DNN
Batch Size	A fraction of the training set optimized together in a single epoch		
Dropout	Fraction of nodes in a dense layer suppressed in a training epoch		
Dense Layer	Receives fully-connected inputs and produces a fixed number of outputs		
Initial Nodes	Number of outputs in the first hidden dense layer		
Convolutional Layer	Receives an image as an input and produces a smaller image		
Kernel Size	The weighted mask's dimensions that convolves an image		
Filter Size	The number of convolved images to be considered together for an output		
Activation Function	Maps the phase space of an input to a desired output phase space		
Optimizer	Function that determines how to optimize weights		
Learning Rate	The rate at which the optimizer adjusts weight values		
Loss Function	The metric used for optimization		
Patience	Number of epochs allowed before early stopping		

**The high-lighted hyper parameters were optimized using Bayesian optimization

Bayesian Hyper Parameter Optimization

Using Scikit-Optimize python library

Define a hyper parameter space to survey and the number of random starts and total starts for the optimizer to survey

- The optimizer assumes that the hyper parameter space is Gaussian distributed about the optimal set of hyper parameter

The metric used for the model optimization is,

$$\frac{Mean\ Absolute\ Error}{Epochs\ until\ Early\ Stopping}$$

- Weighting the loss by how many epochs until early stopping ensures robustness during training and punishes sudden loss drops that eventually diverge (associated with larger learning rates)