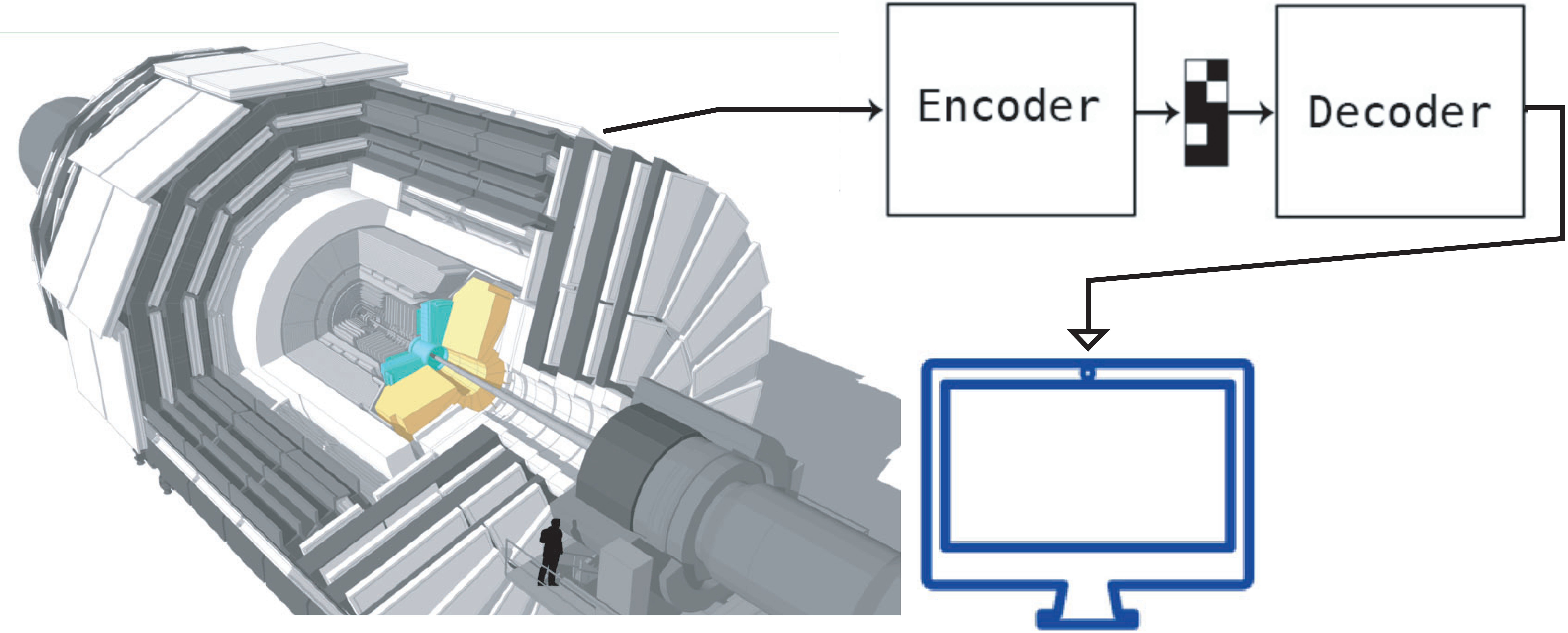
A Reconfigurable Neural Network ASIC for Front-end Data Compression at HL-LHC - Update

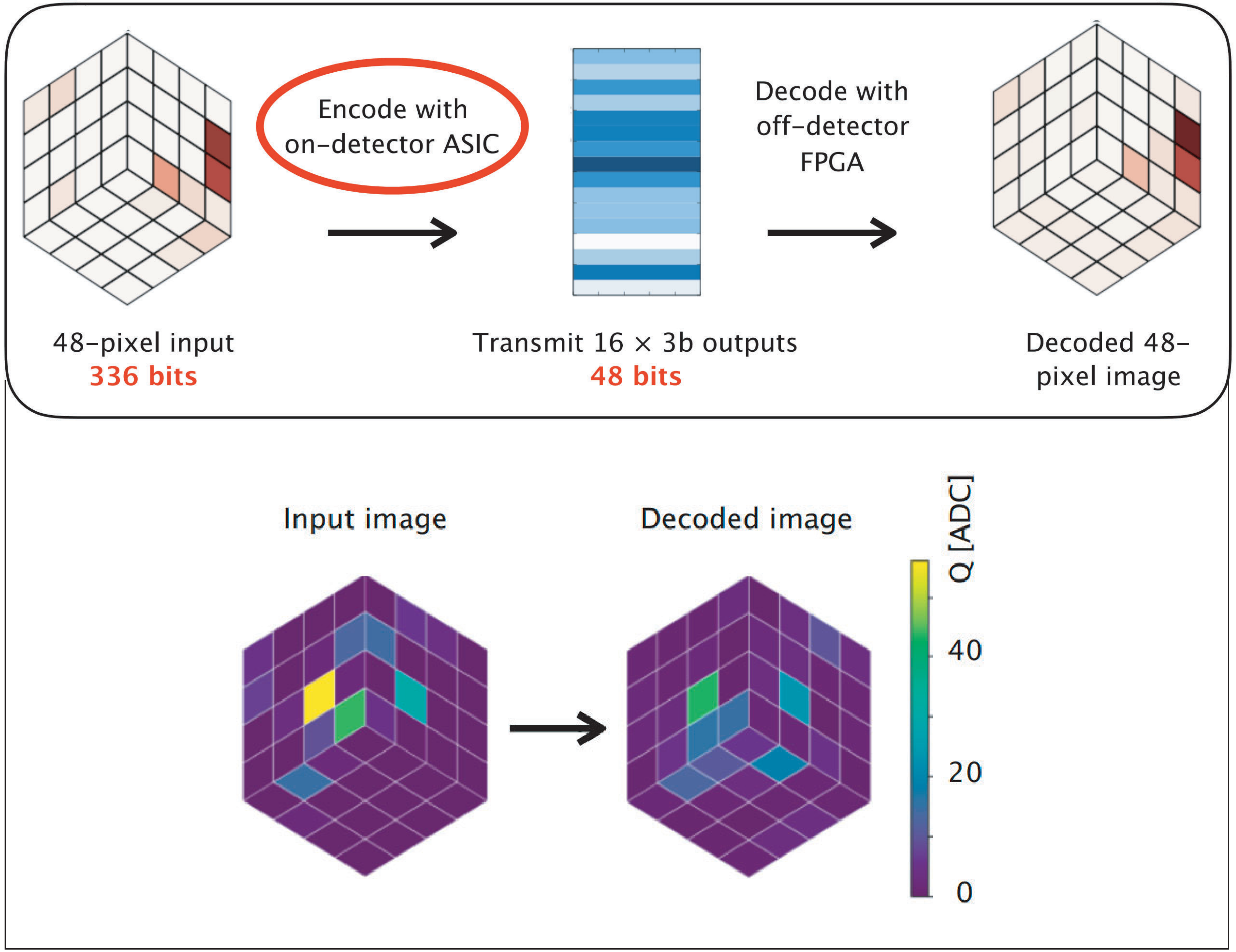


Recap

- As equipment/experiments become more sensitive, more data is required to be transmitted, making conventional transmission methods less effective

-Machine Learning could improve data compression for transporting offsite for further analysis

-Compression performance quantified by *Energy Mover's Distance* (energy x distance)



Methodology and Training Models

1. Architectures of encoder models obtained from running program; parameters varied for training models, such as:
    a. input data set
    b. pooling/stride (kept constant in this presentation)
    c. loss function (kept constant in this presentation)

2. Models obtained are transferred to different repository; with models implemented, jobs are ran in crab to obtain root files

3. root files are converted to HDF files via running condor jobs

4. HDF files added to Jupyter Notebook to obtain corrections and evaluate performance

Models Presented

1. Threshold0/All TC: saving all data recorded (used as baseline)

2. Threshold 1.35 mipT : saving all trigger cells which are above certain energy threshold

3. BC + STC: mixture of 'best-choice' (taking max values) and 'super trigger cell' (taking average of groups of cells)

4. AE Tele Stride ttbar: model trained on ttbar samples

5. AE Tele Stride Ele: model trained on electron samples

6. AE Tele Stride Ele CALQ: model trained on electron samples WHERE training target for events is set to zero when energy sum threshold is not met

*Trained encoder models

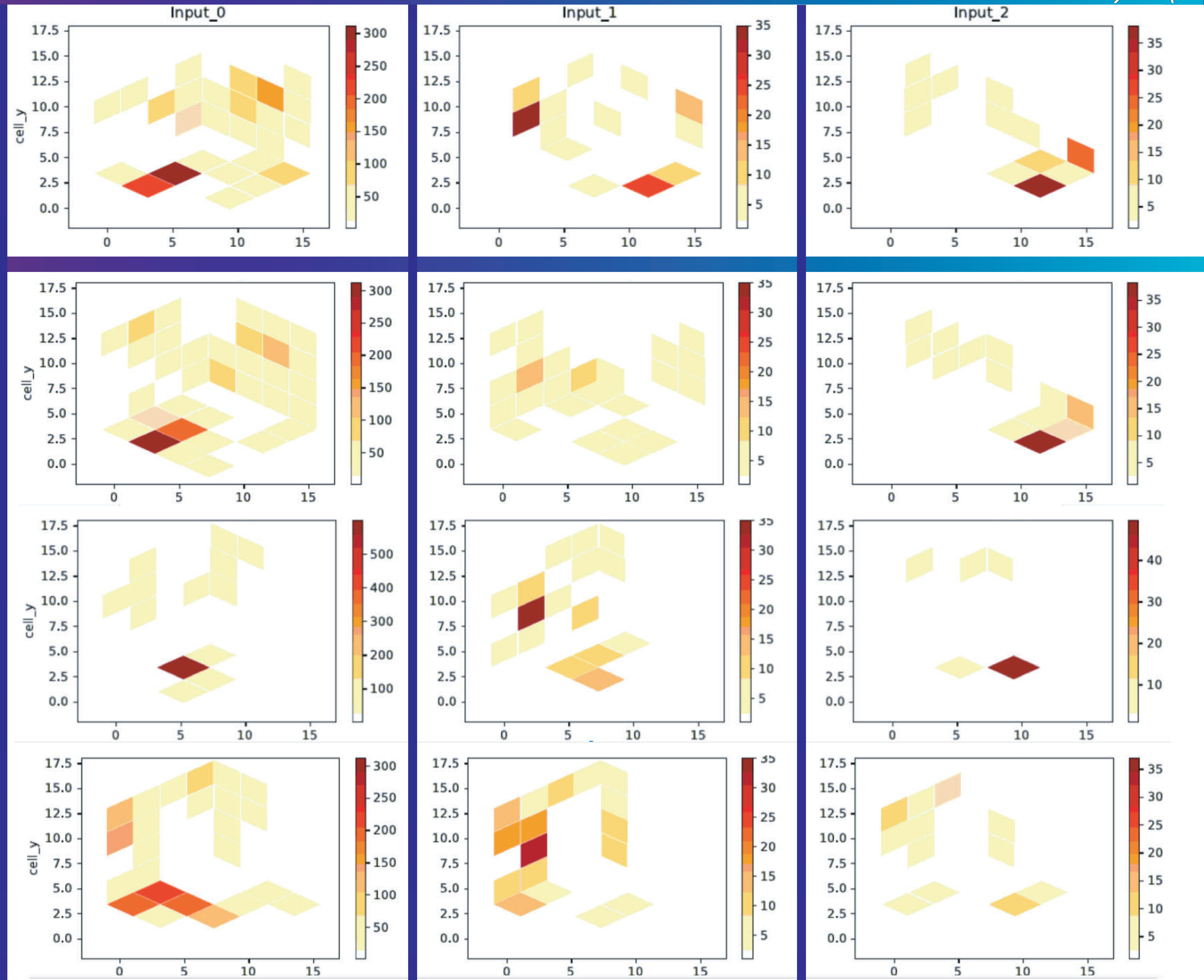
Trigger Cell Example Example (ttbar events)

Input Events (ttbar)

Decoded Image from ttbar-trained Model

Decoded Image from Electron-trained Model

Decoded Image from Electron-trained Model (cut applied)



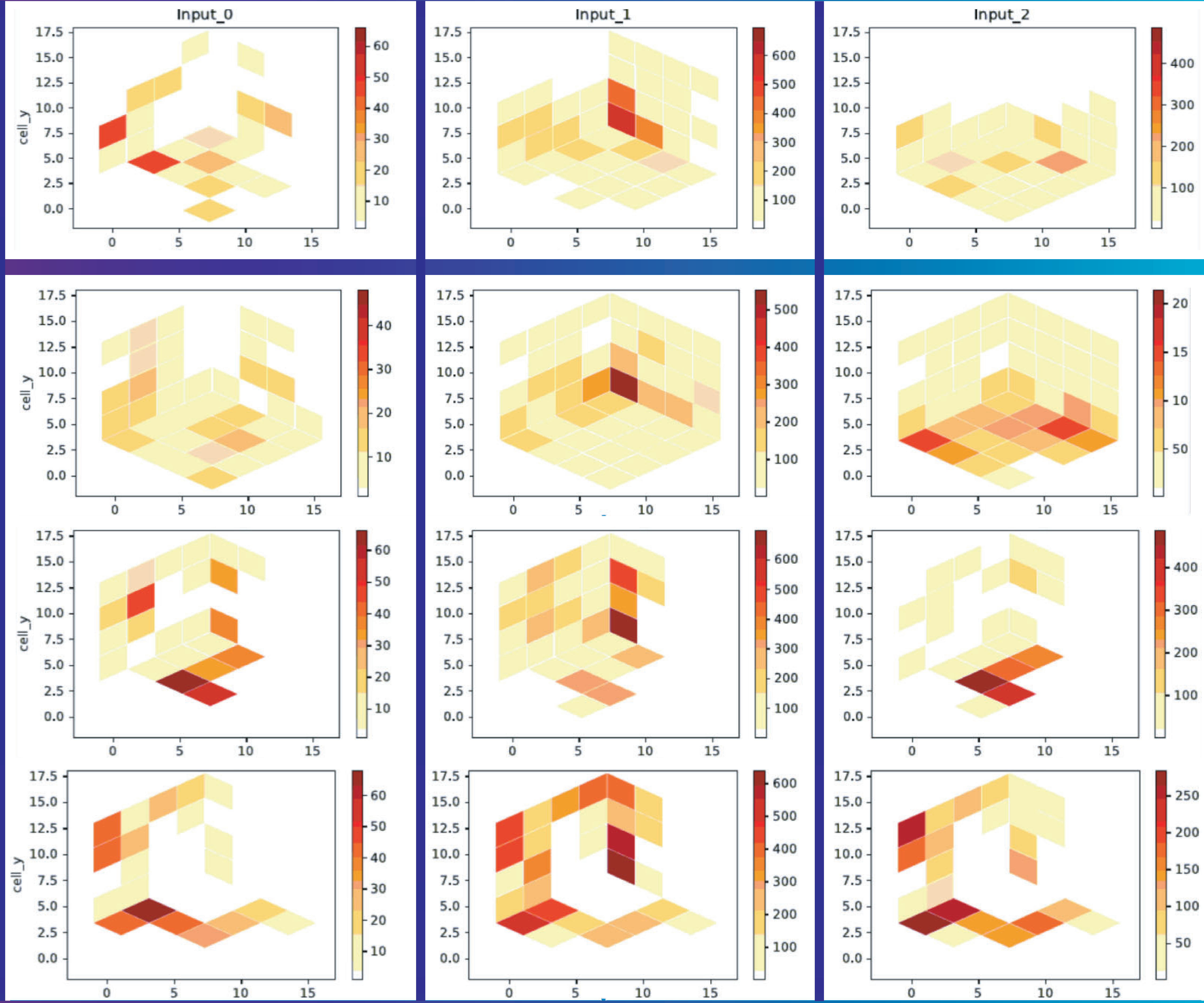
Trigger Cell Example Example (electron event)

Input Events (electron)

Decoded Image from ttbar-trained Model

Decoded Image from Electron-trained Model

Decoded Image from Electron-trained Model (cut applied)



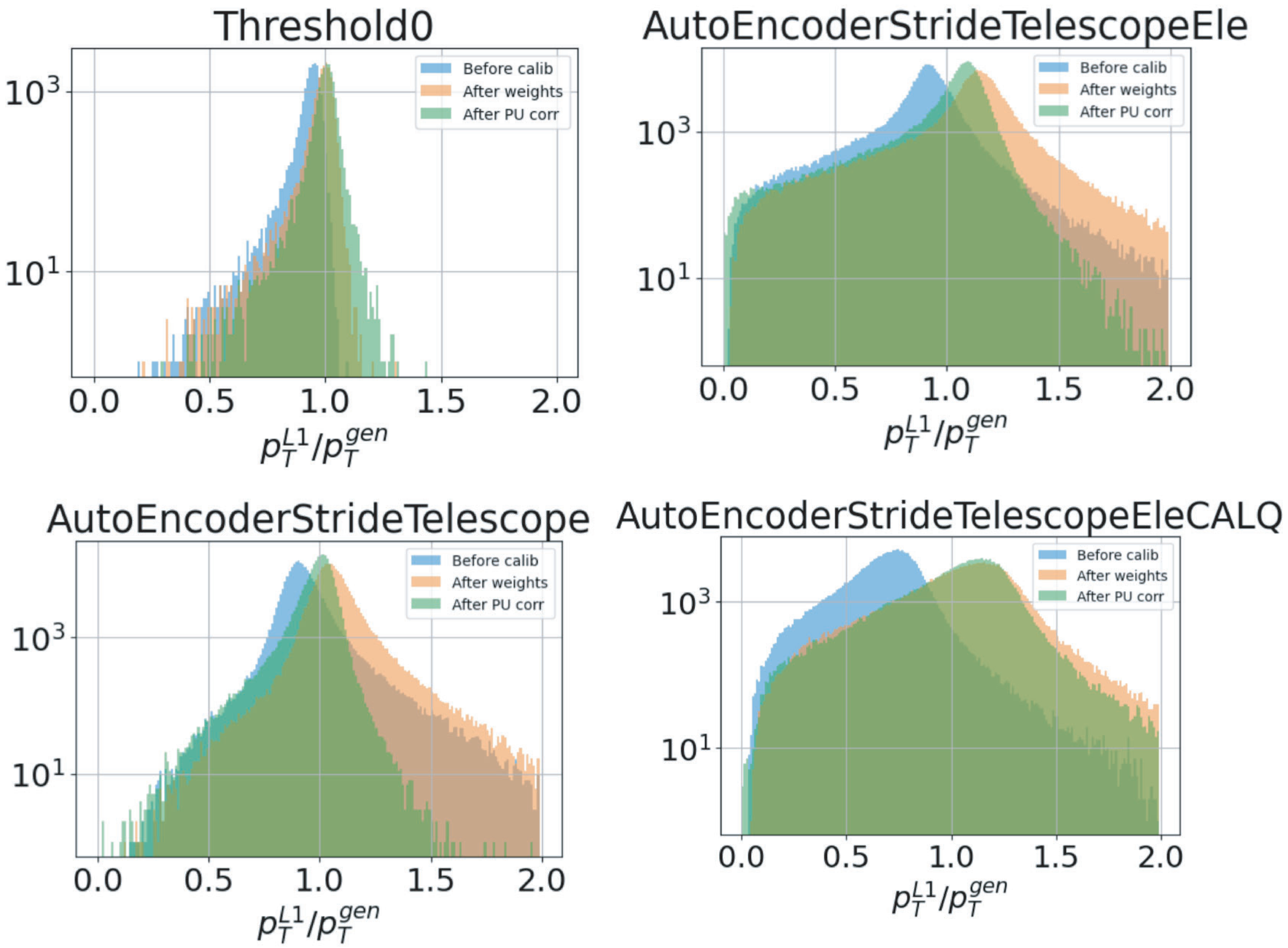
Corrections and pT Distributions

-Obtained models are ran on two samples:

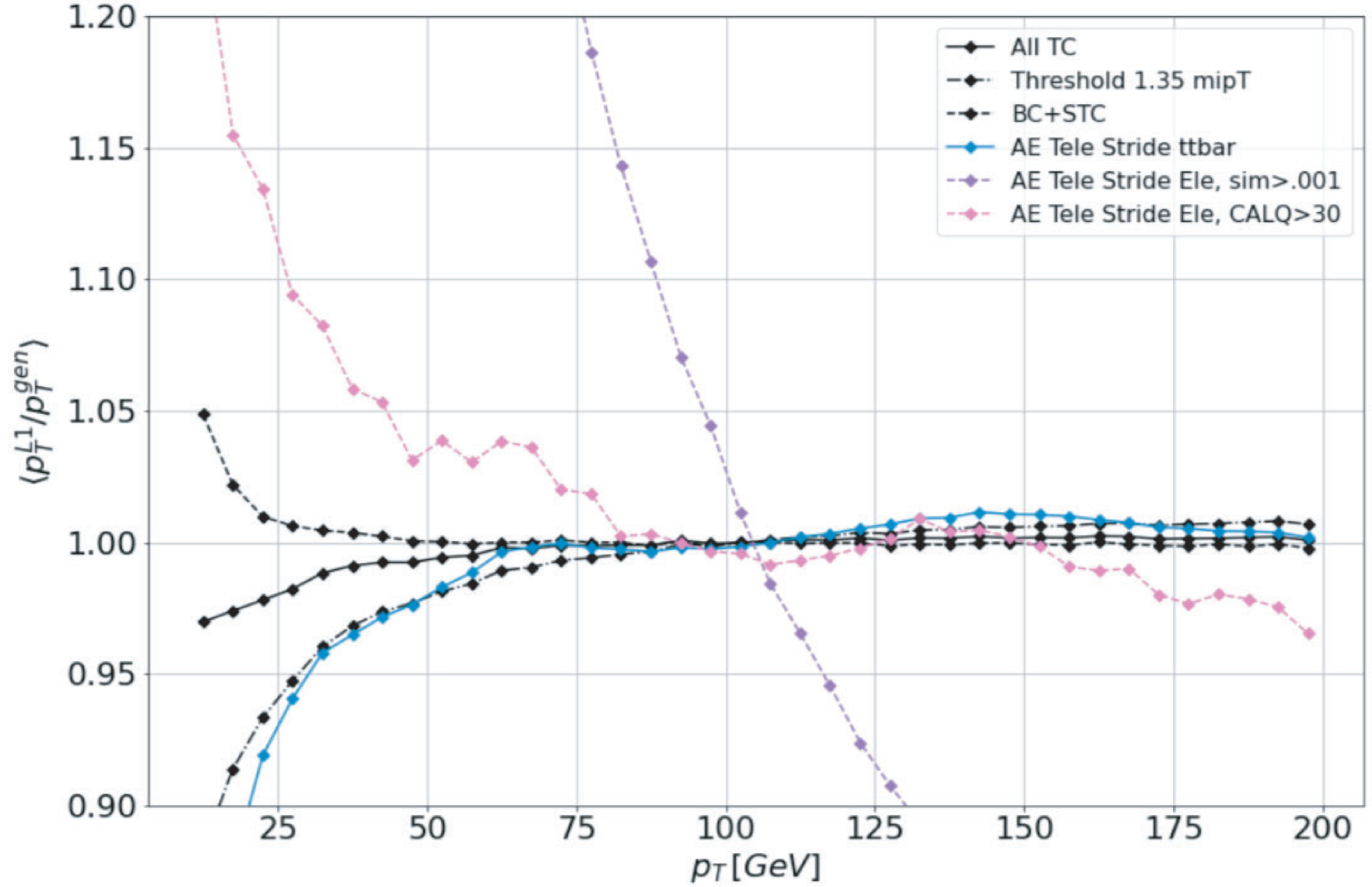
1. 0 Pileup Photons Events
2. 200 Pileup Electron Events

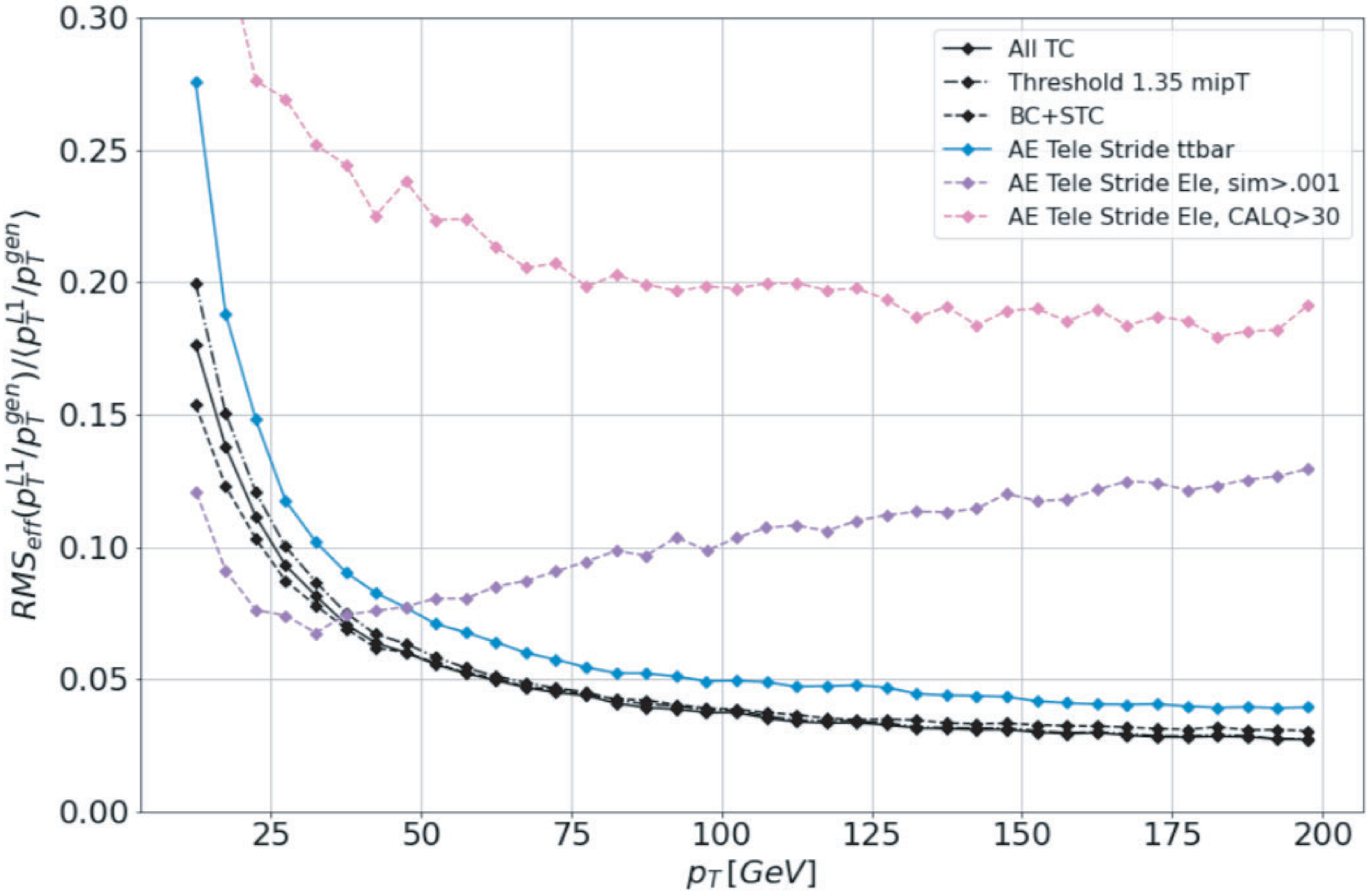
- Corrections are applied using truth pT and η

- Much wider distributions for training models (particularly those trained on electrons)



pT Distributions after Corrections applied

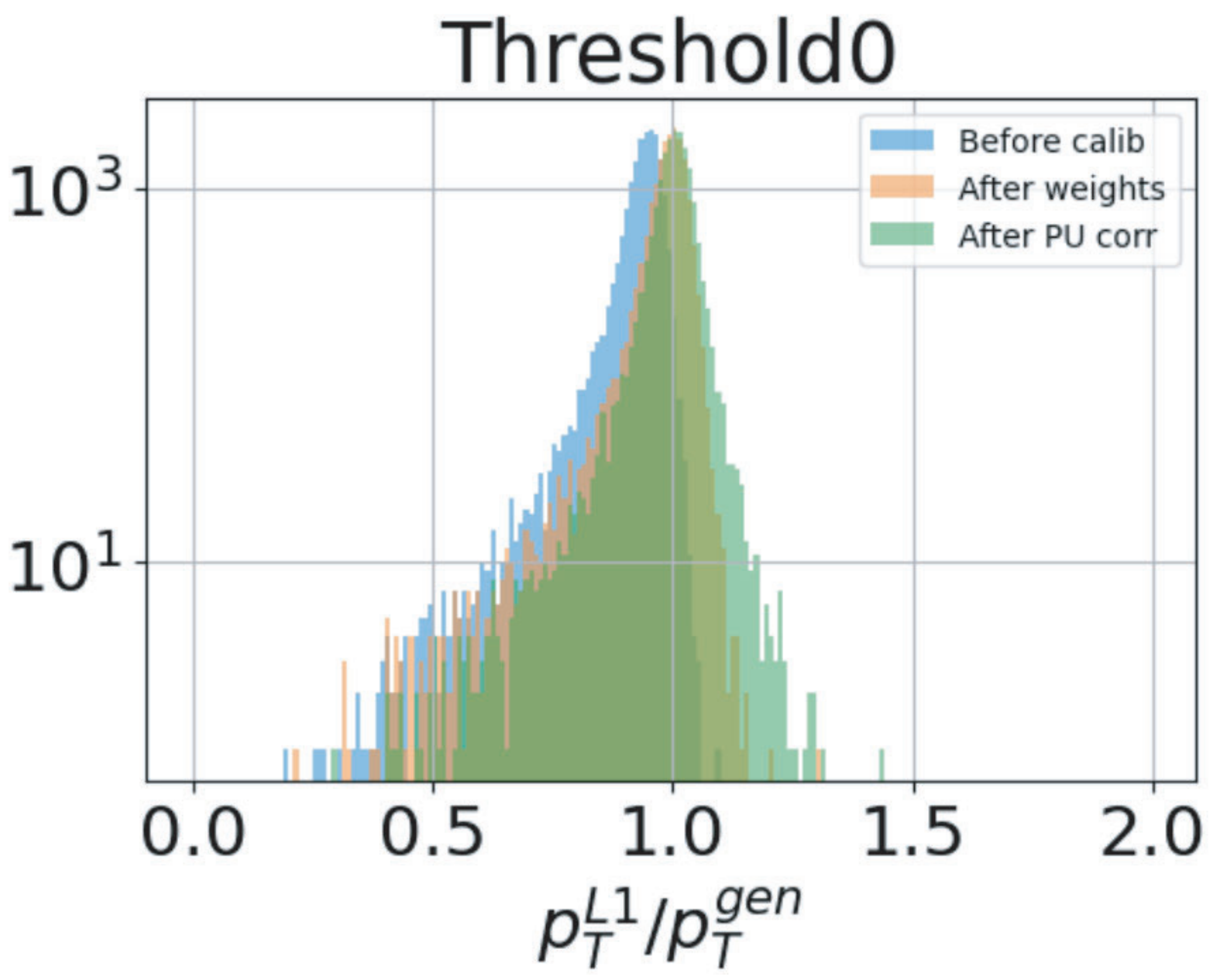


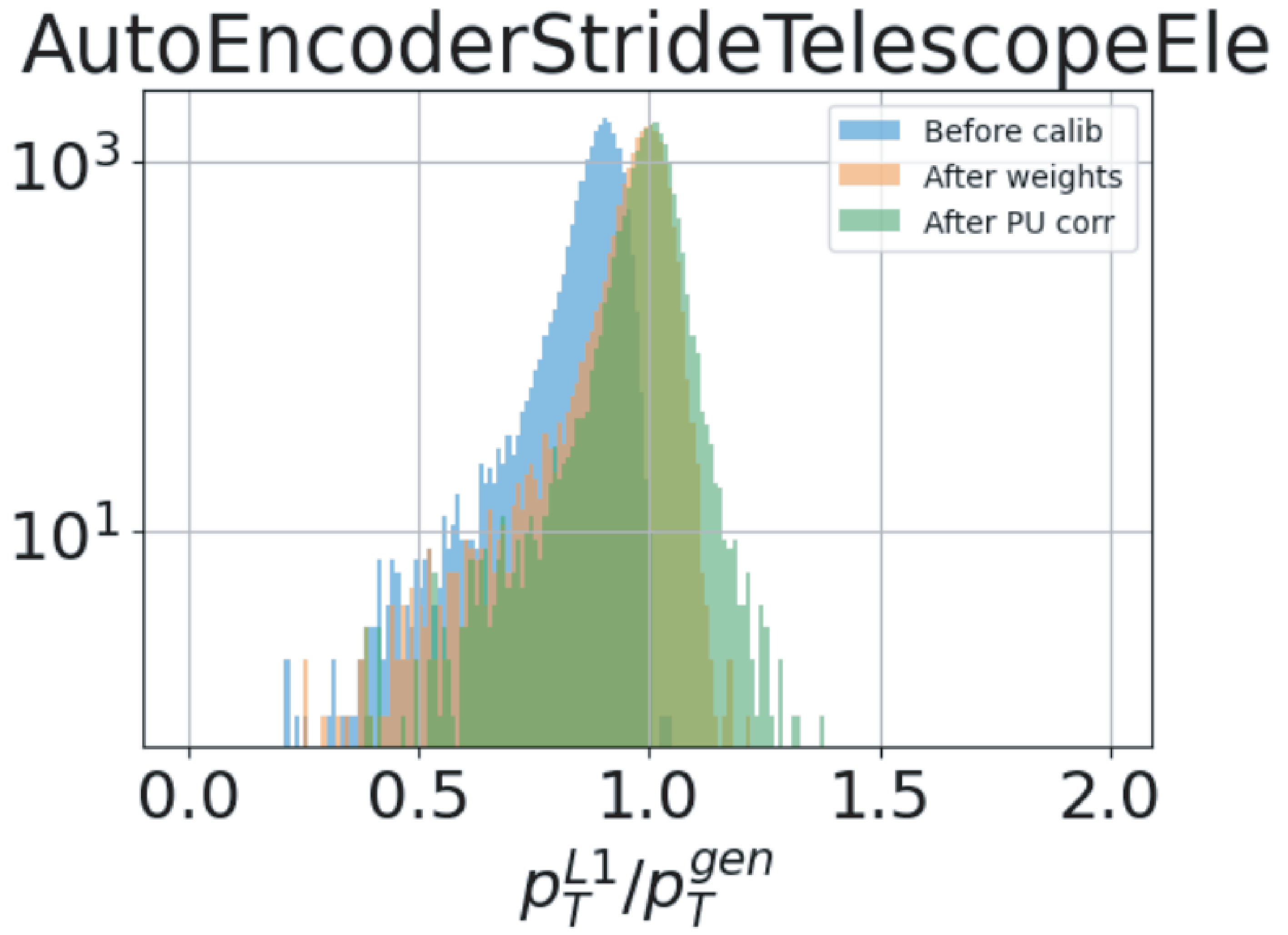


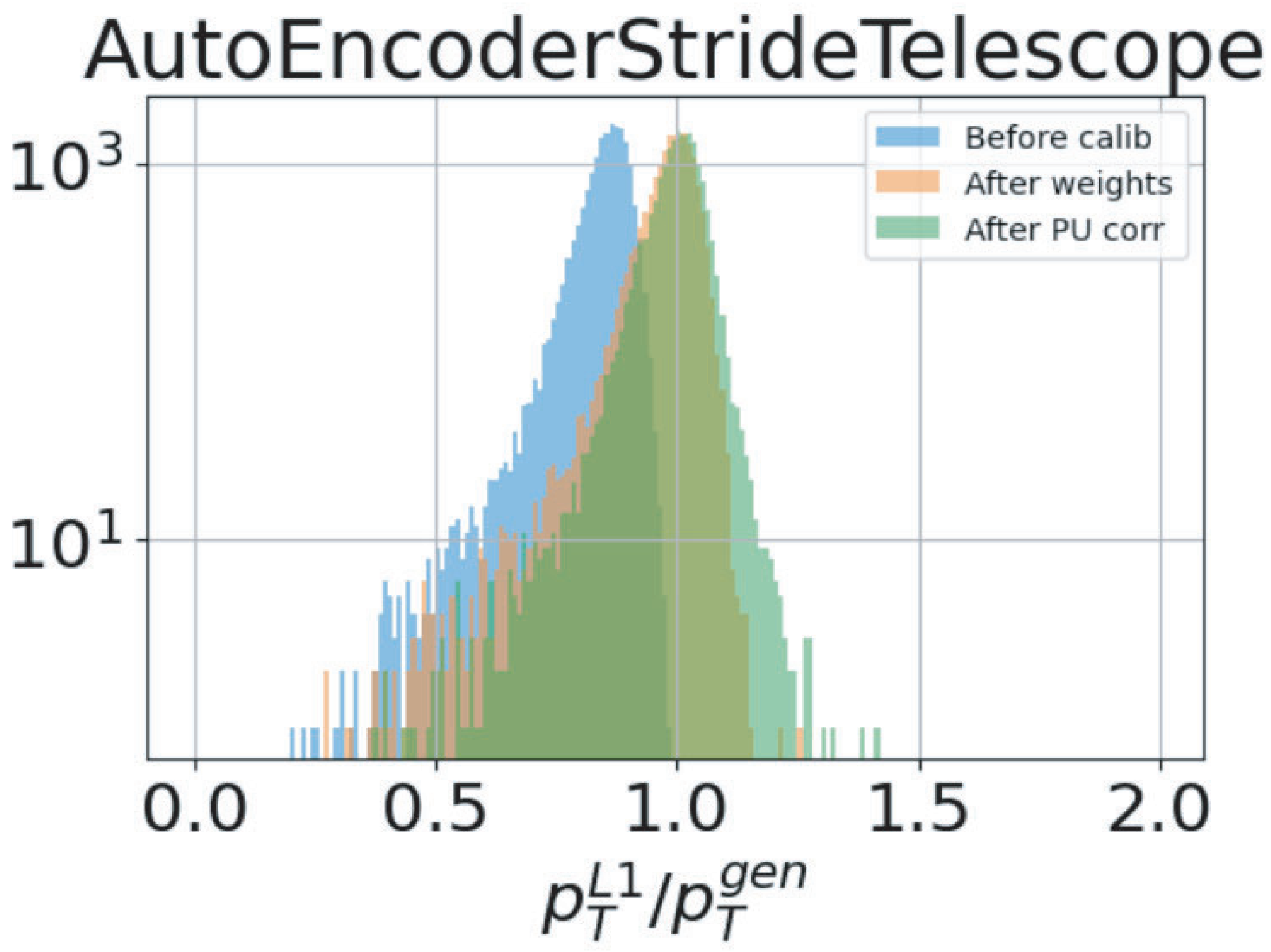
-To better understand issues with models, running models on 0 Pileup Electron events and 0 Pileup Photon Events instead

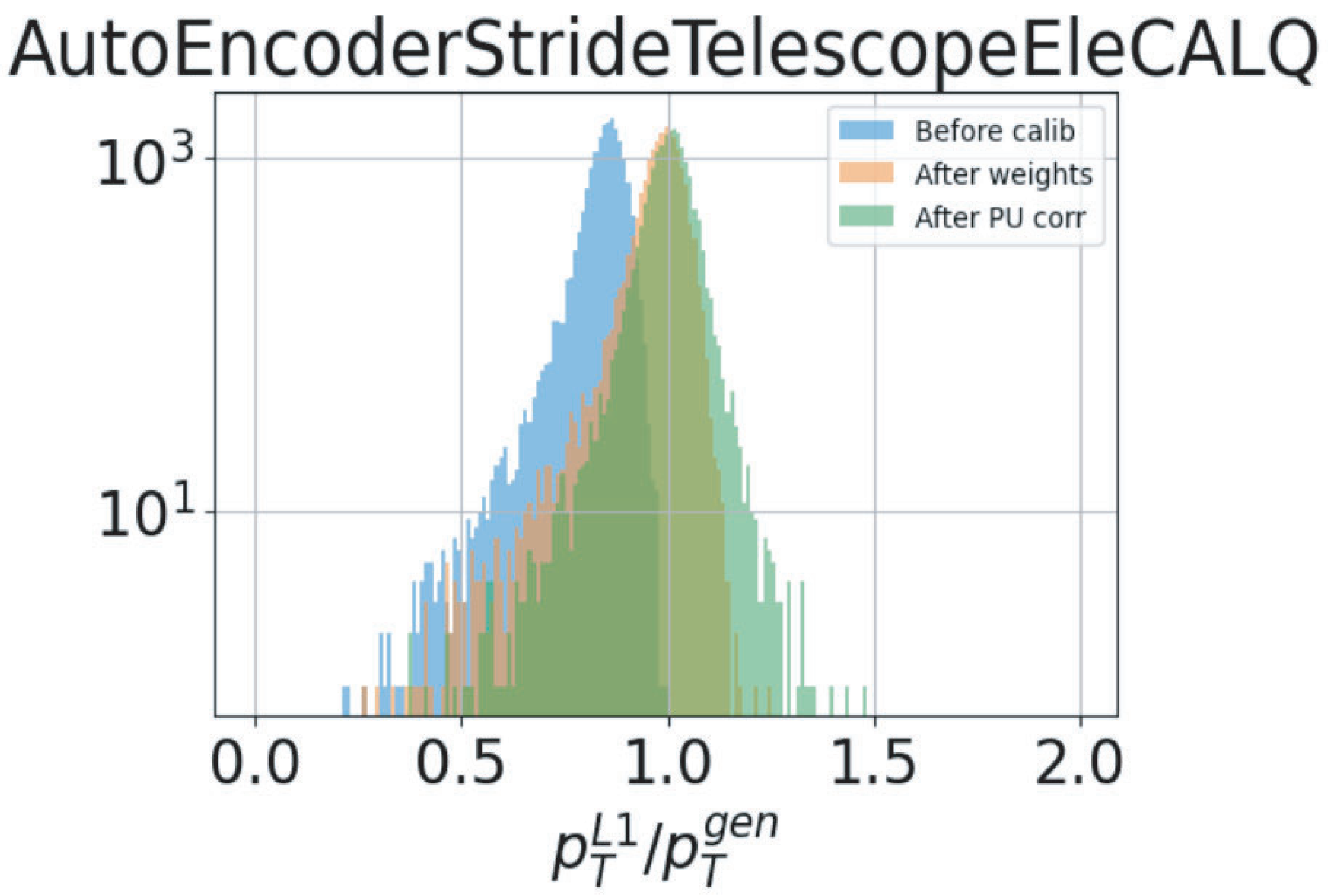
- Yields cleaner signal for models to encode and hence thinner distributions

- will investigate energy/position resolutions and corrections

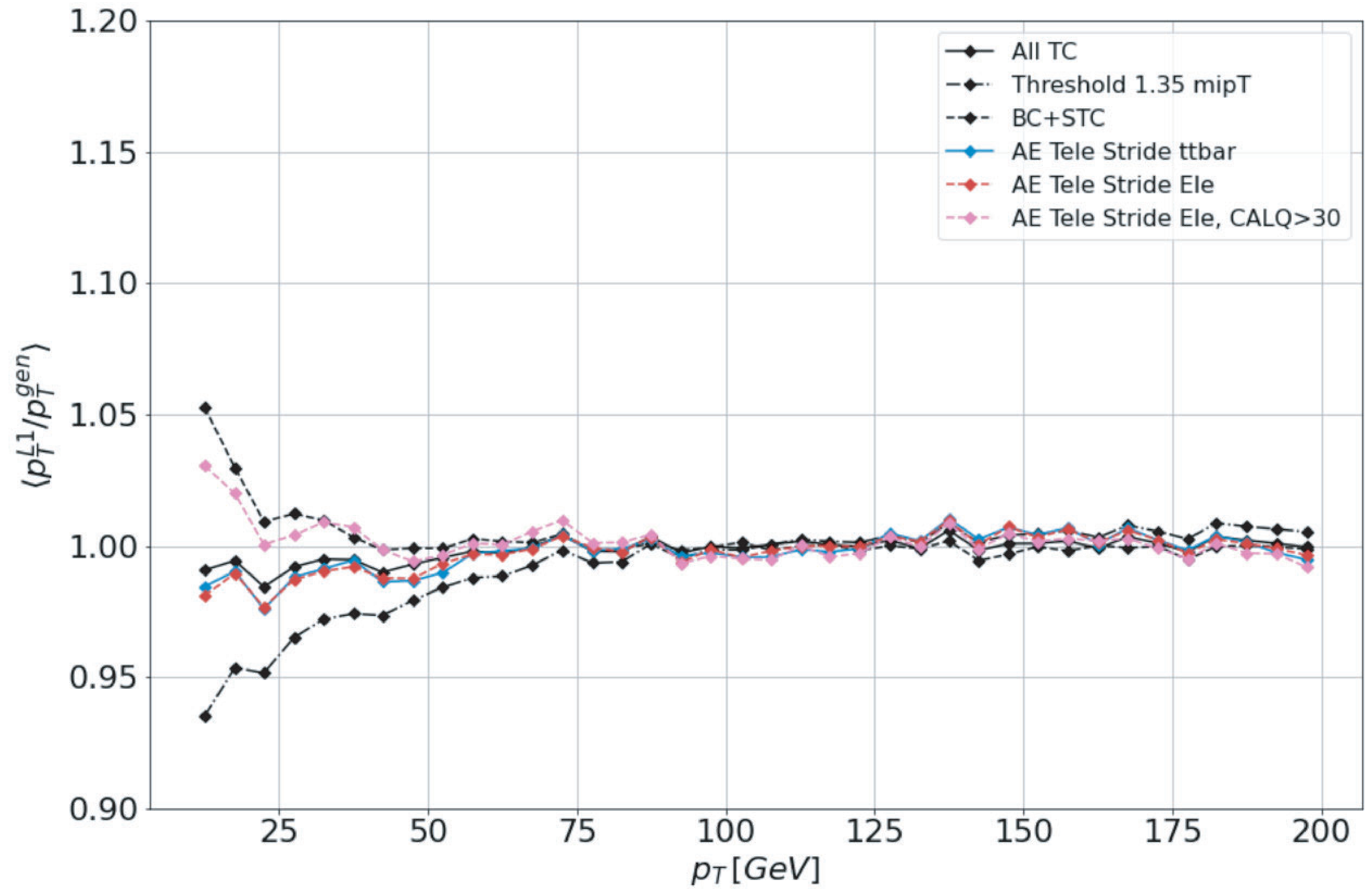


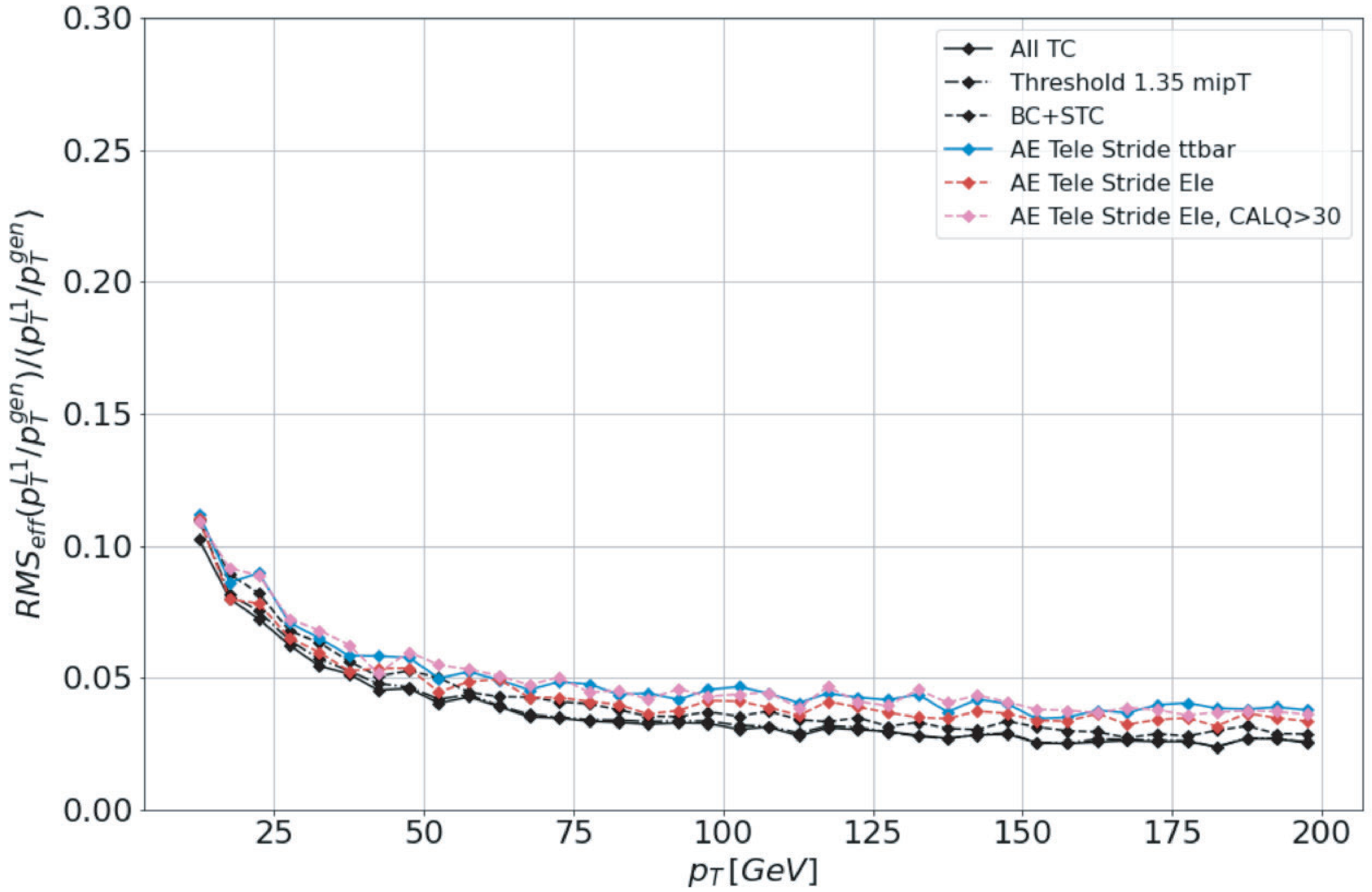






pT Distributions after Corrections Applied (0 Pileup data)





Future Plans

- Investigate resolutions of 0 pileup data

- Fix issues with 200 pileup electron data (training, corrections, etc.)

- Understand why electron-trained models perform more poorly

- Determine which models are most effective