Tracking with ML

29th January 2025

Jeremy Couthures

ATLAS EXPERIMENT

LAPP Laboratoire d'Annecy de Physique des Particules

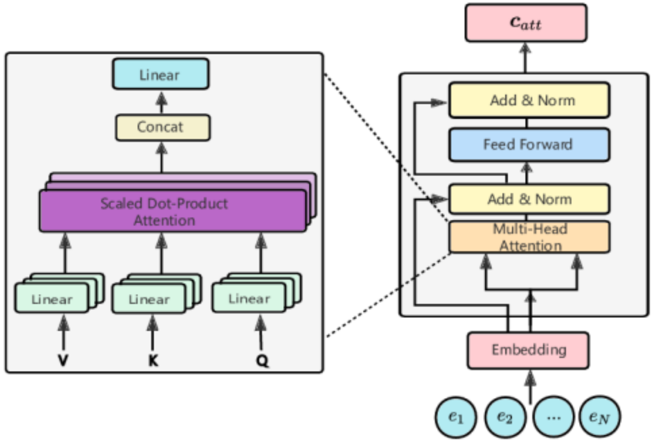
ATRAPP

UNIVERSITÉ SAVOIE MONT BLANC

TrackFormer

- **Transformer for track parameter regression**
- **Tested on several dataset: ToyTracks, Acts, TrackML**

- **Regression in pt and pz**
- **Shown promising results**

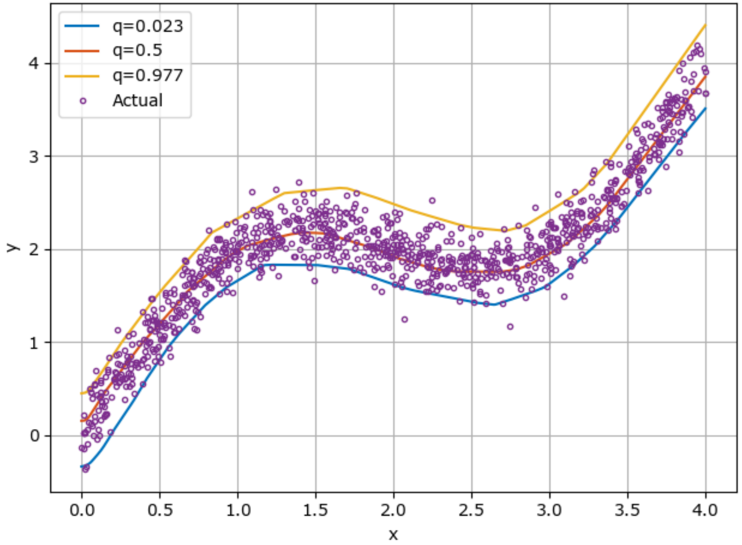


Sequences were padded to a fixed length

TrackFormer loss functions

Mean squared error:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



Quantile loss:

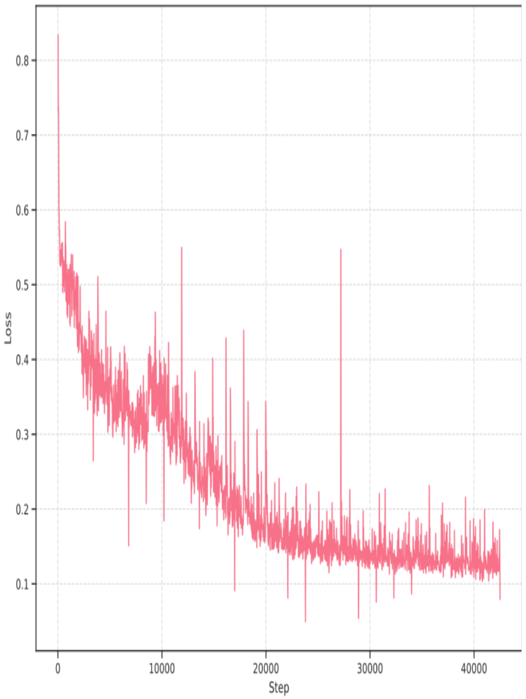
$$\mathrm{QL} = \frac{1}{n}\sum_{i=1}^{n}\left(\max(q(y_i - \hat{y}_i), (q-1)(y_i - \hat{y}_i))\right)$$

TrackFormer training

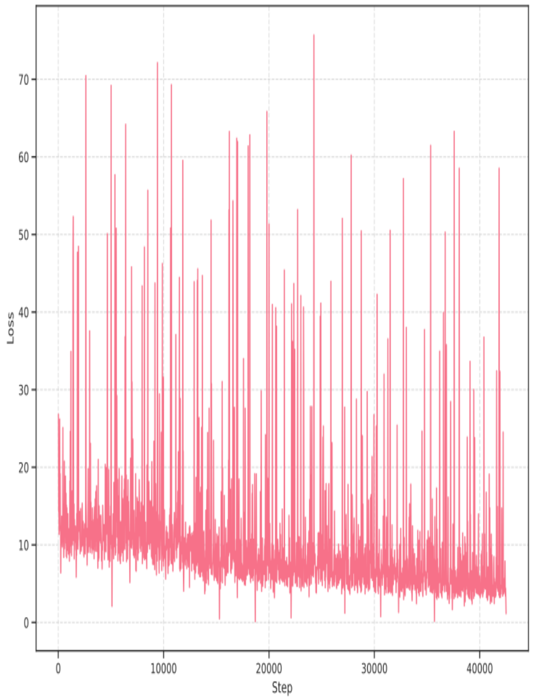
- **TrackML dataset**

By far, this aspect presented the most challenges. When performing 'on-the-fly' preprocessing and data loading, a significant hurdle was the long training time of 23 hours for 20 epochs. This was problematic because transformers are well-known for their speed and performance advantages. However, it became apparent that the transformer was not utilising its full capabilities. Firstly, even with a large model having 22 million parameters, the GPU was not being fully utilised, which was a clear indication that something was amiss. Instead, the next best option was to use the main memory and optimise data loading with PyTorch functions such as pinned memory and persistent workers. This resulted in a dramatic speedup, with training time reduced to 30 minutes for 50 epochs—a 48x speedup. This also ensured more efficient use of GPU resources, with GPU utilisation remaining at a constant 97% throughout training. In contrast, the earlier loading method caused GPU utilization to fluctuate between 0% and 100%, mostly staying at 0% due to training waiting for data retrieval.

However, this approach has its limitations. Loading an entire dataset into main memory is not always feasible due to the resource-intensive nature of this process, requiring over 50 GB of CPU RAM for large datasets. This presents an opportunity to develop a custom data-loading pipeline that strikes a balance between on-the-fly prepreprocessing and loading data into GPU RAM. Additionally, it was found that batch size played a crucial role in stabilising training, with an optimal batch size determined to be around 3,000.

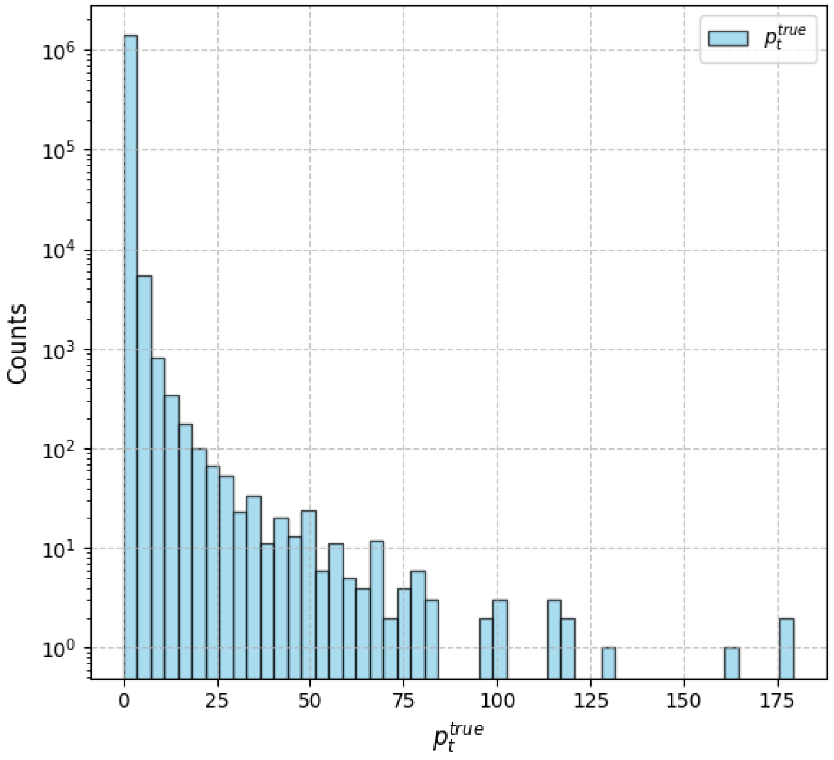


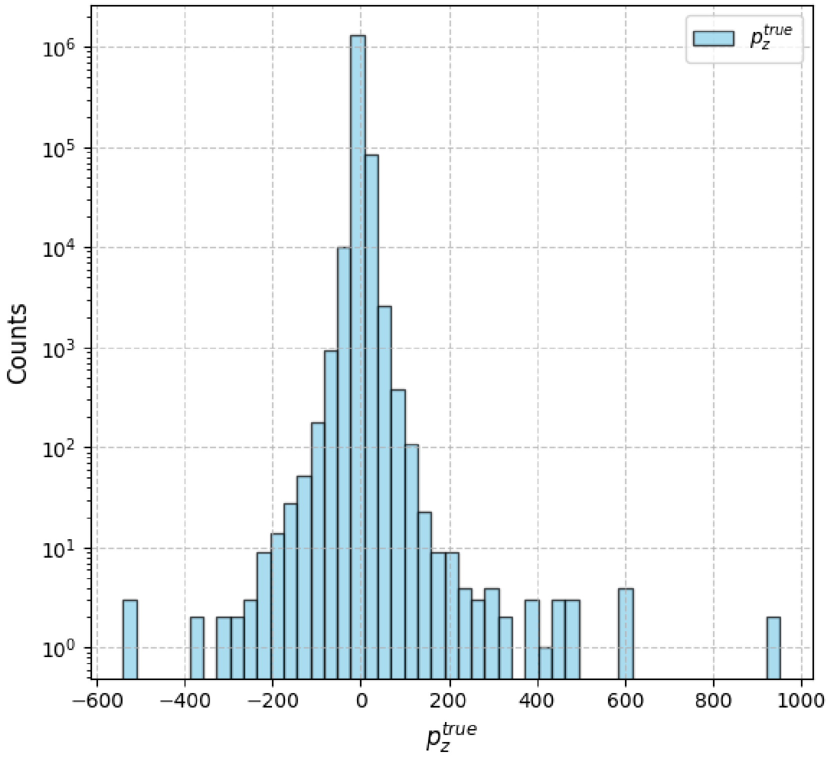
(a) Transformer train quantile Loss



(b) Transfomer MSE train Loss

Target variables





TrackFormer report results

Metric	Transformer MSE	Transformer Qloss
p_t MAE	0.2212 ± 0.0003	0.0718 ± 0.0004
p_z MAE	0.7048 ± 0.0018	0.4648 ± 0.0021

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

$$|p_t^{true} - p_t^{pred}| / p_t^{true}$$

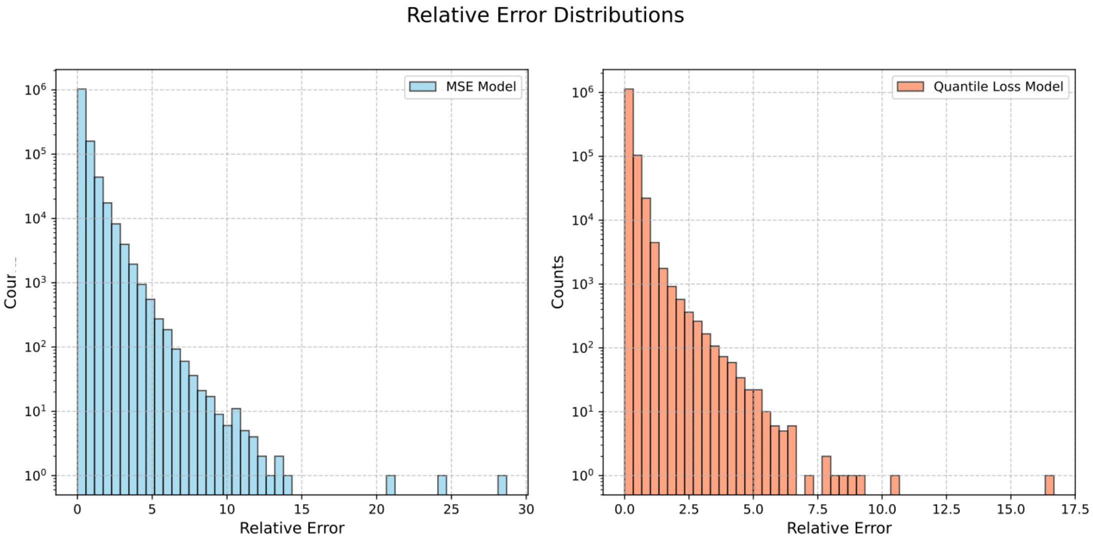


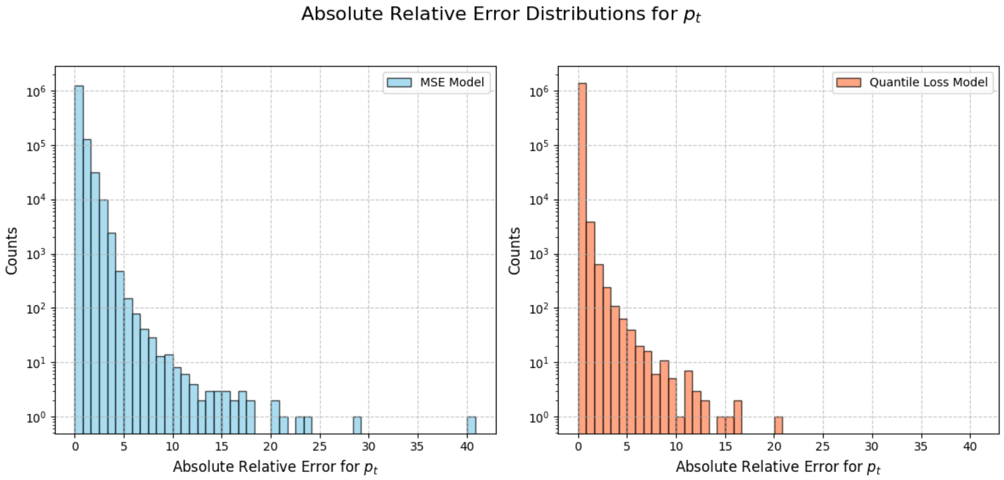
Figure 5.4: Distribution of the absolute error

TrackFormer my results

Dataset used: TrackML first file (1770 events)

- training: ~11 222 272 particles
- Validation: ~1 334 272 particles
- Testing: 1 404 273 particles

$$|p_t^{true} - p_t^{pred}| / p_t^{true}$$



Mine

100 epochs

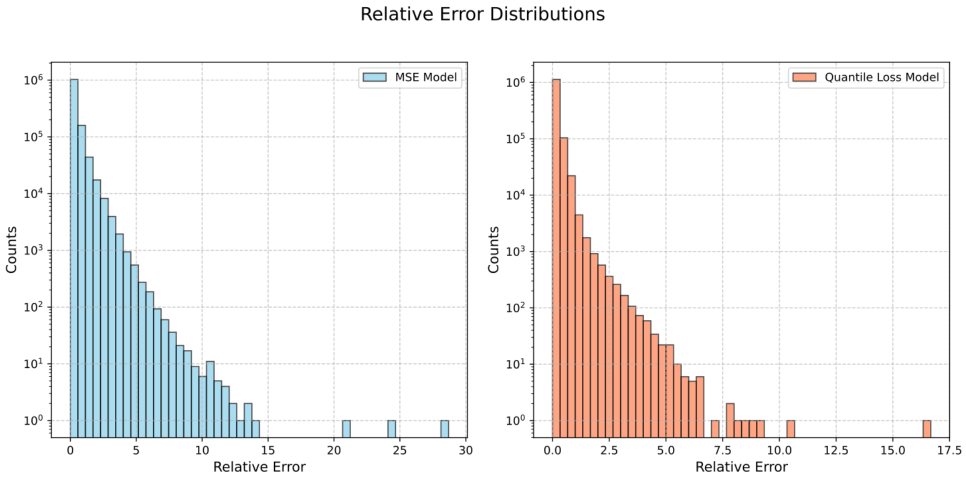
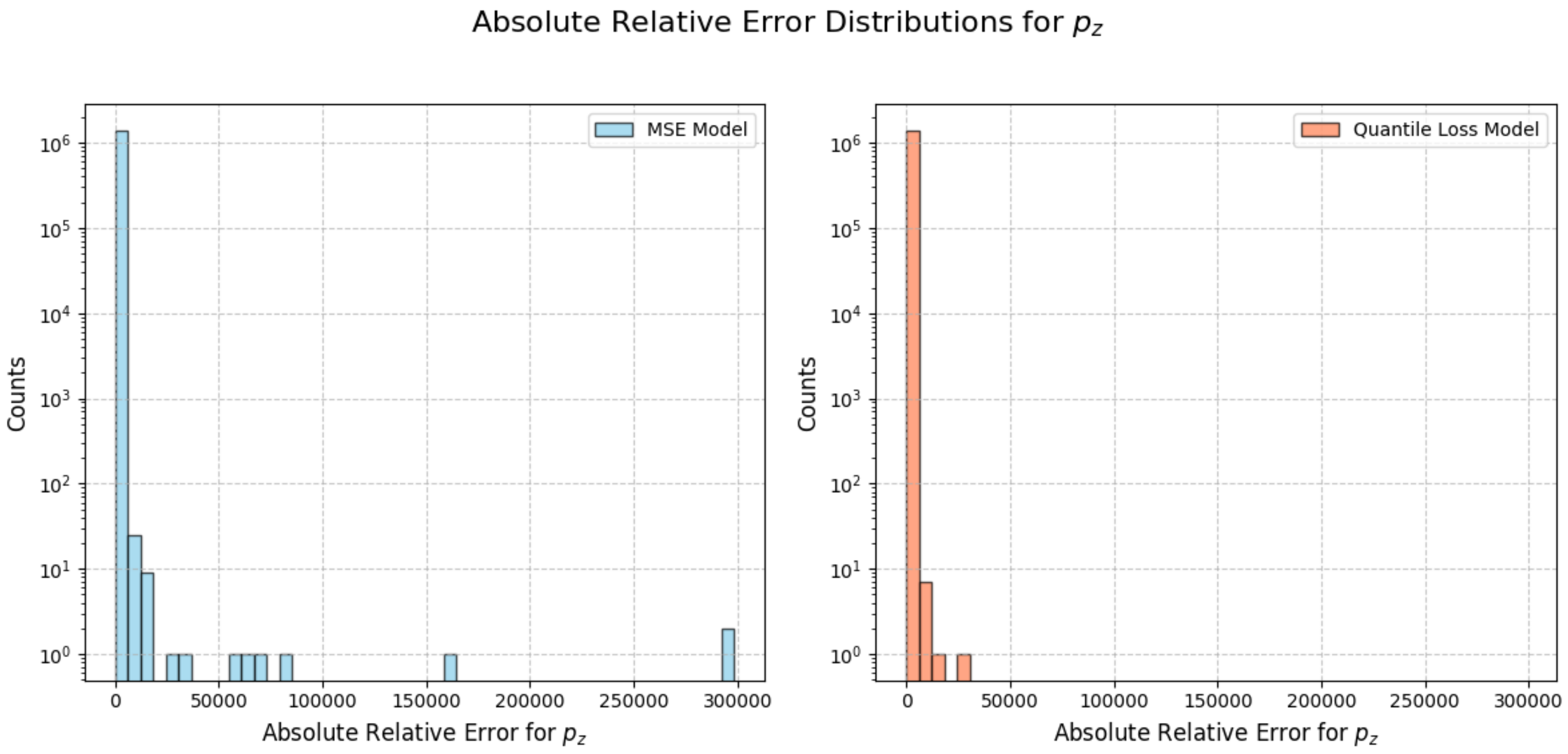


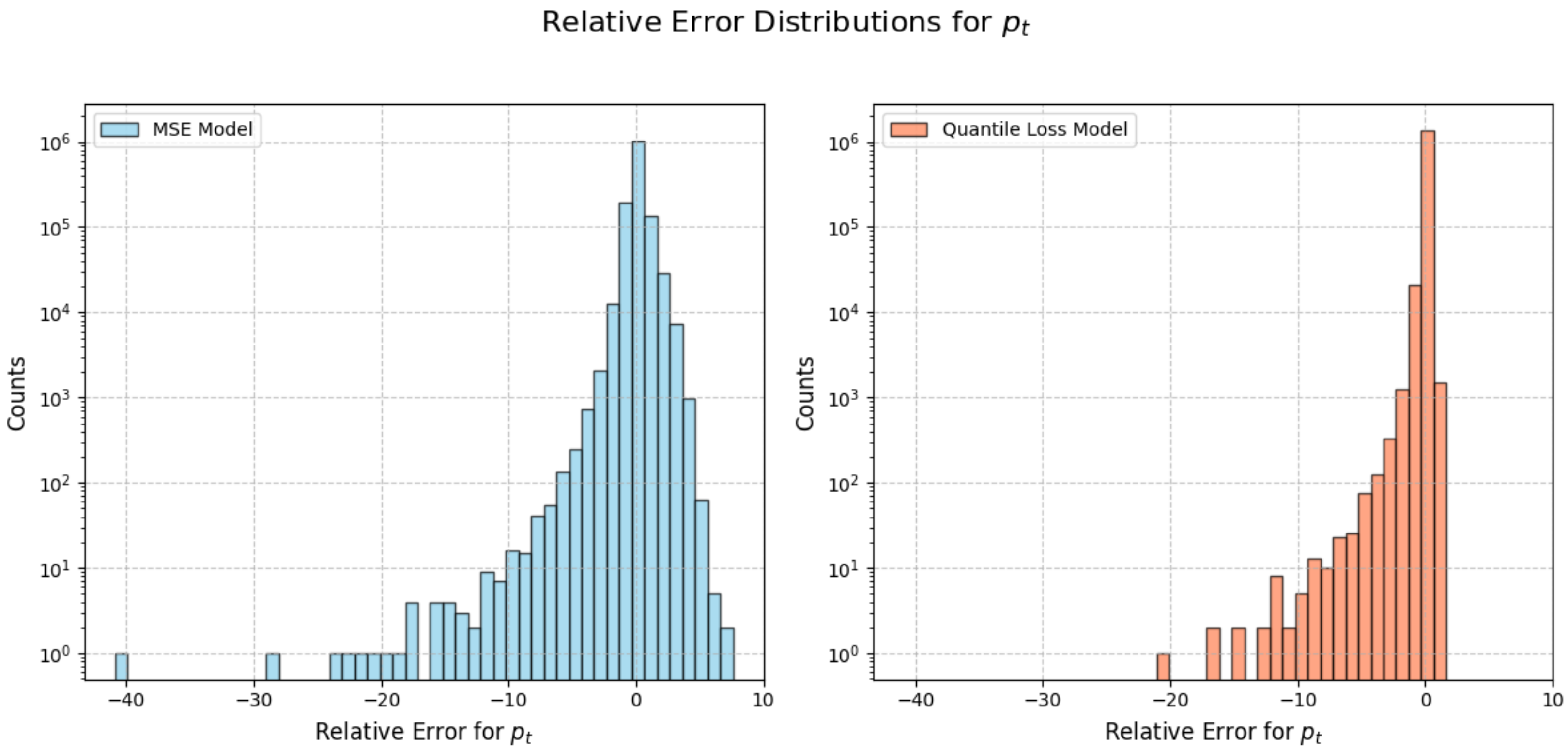
Figure 5.4: Distribution of the absolute error

Report

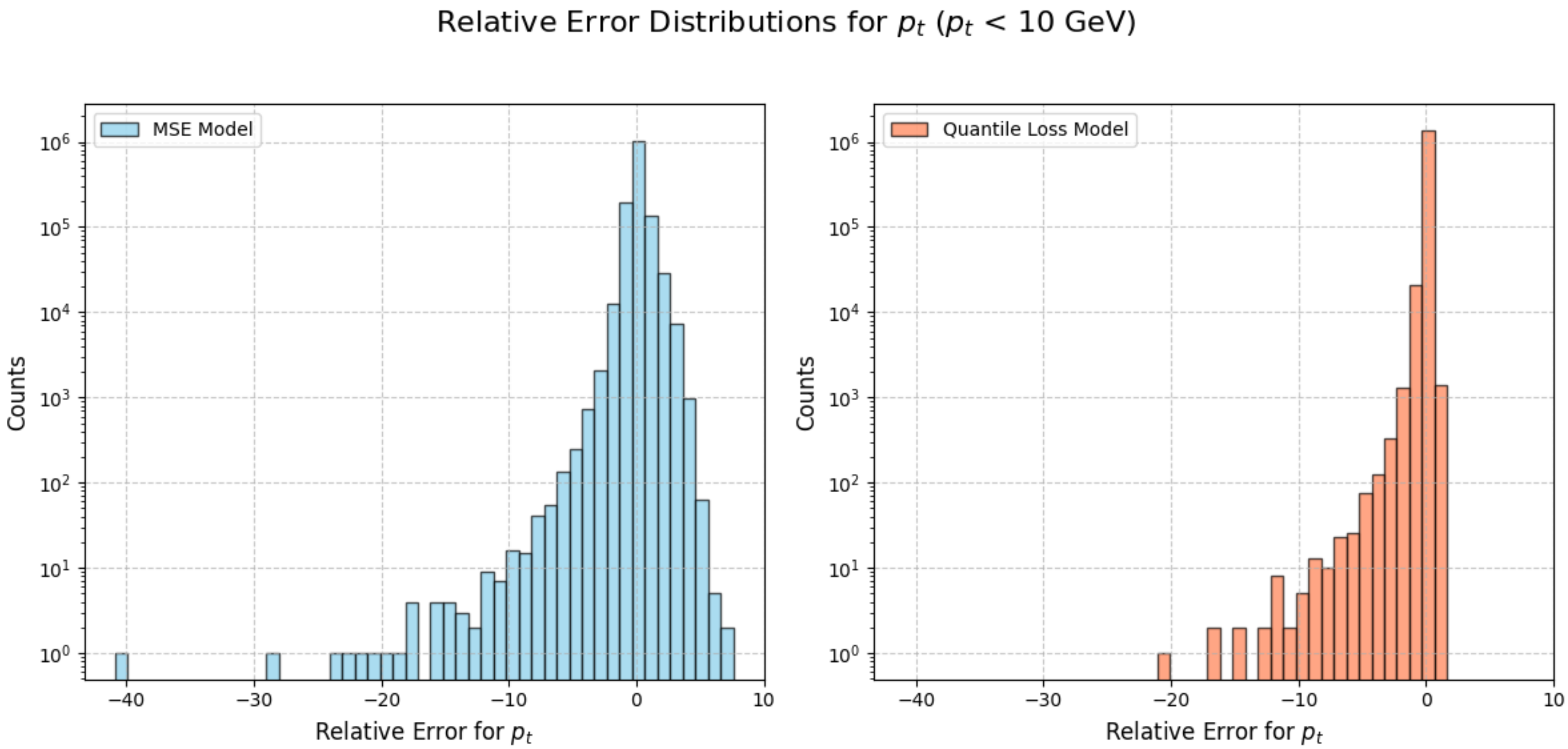
More results



More results

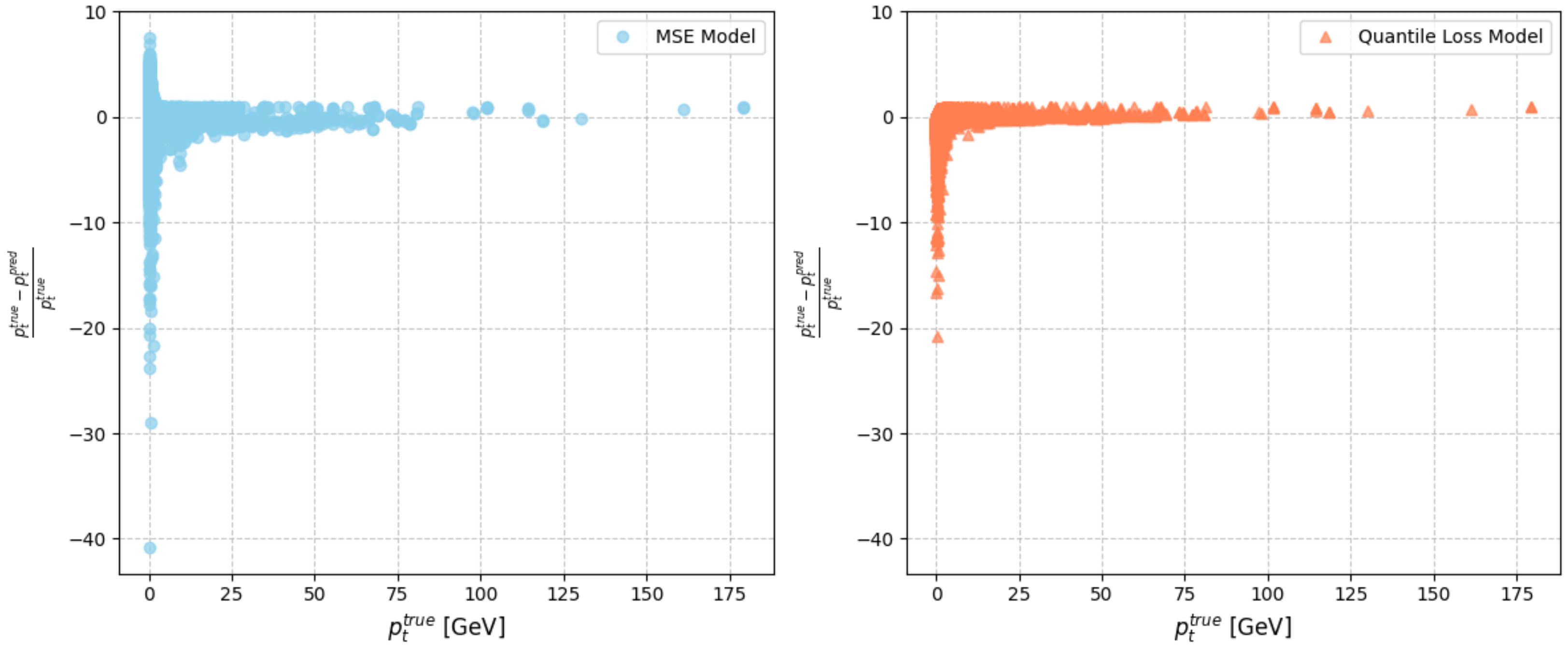


More results

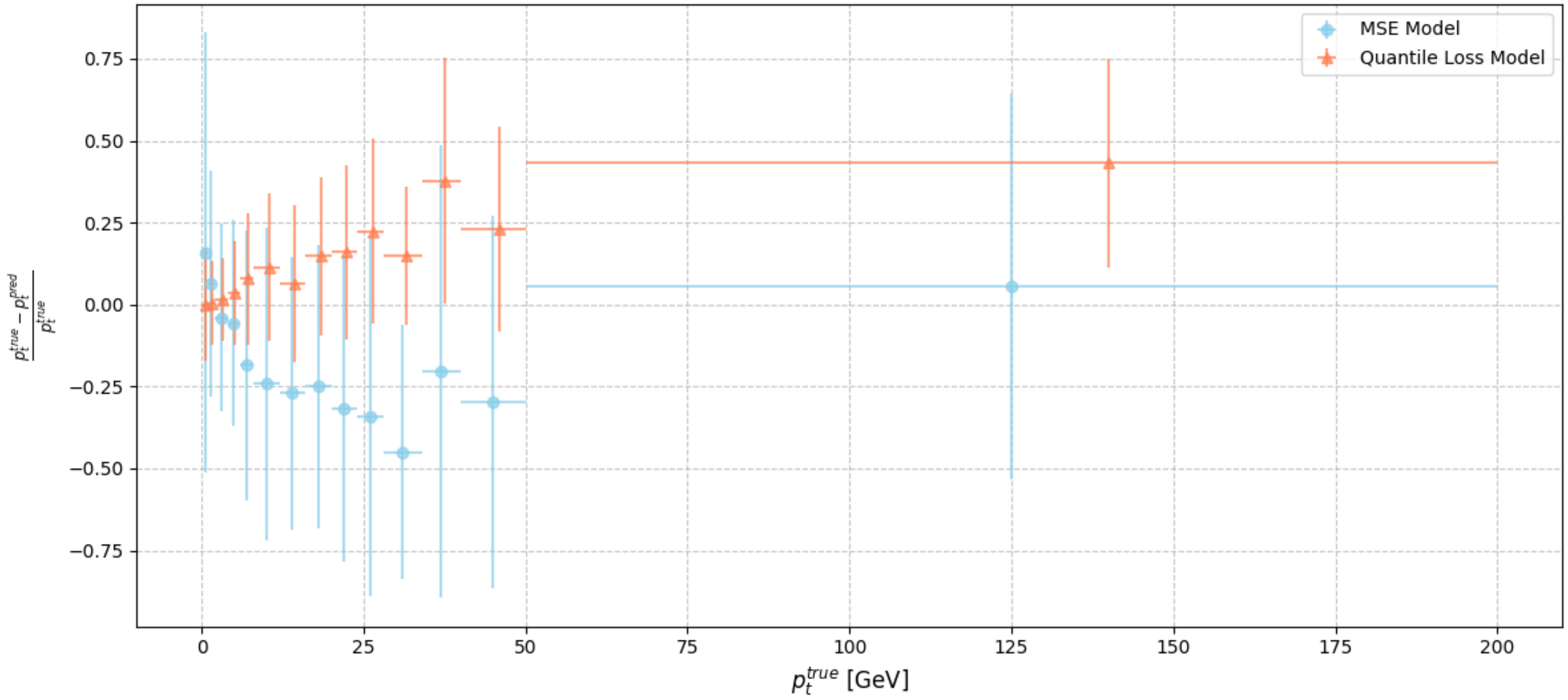


More results

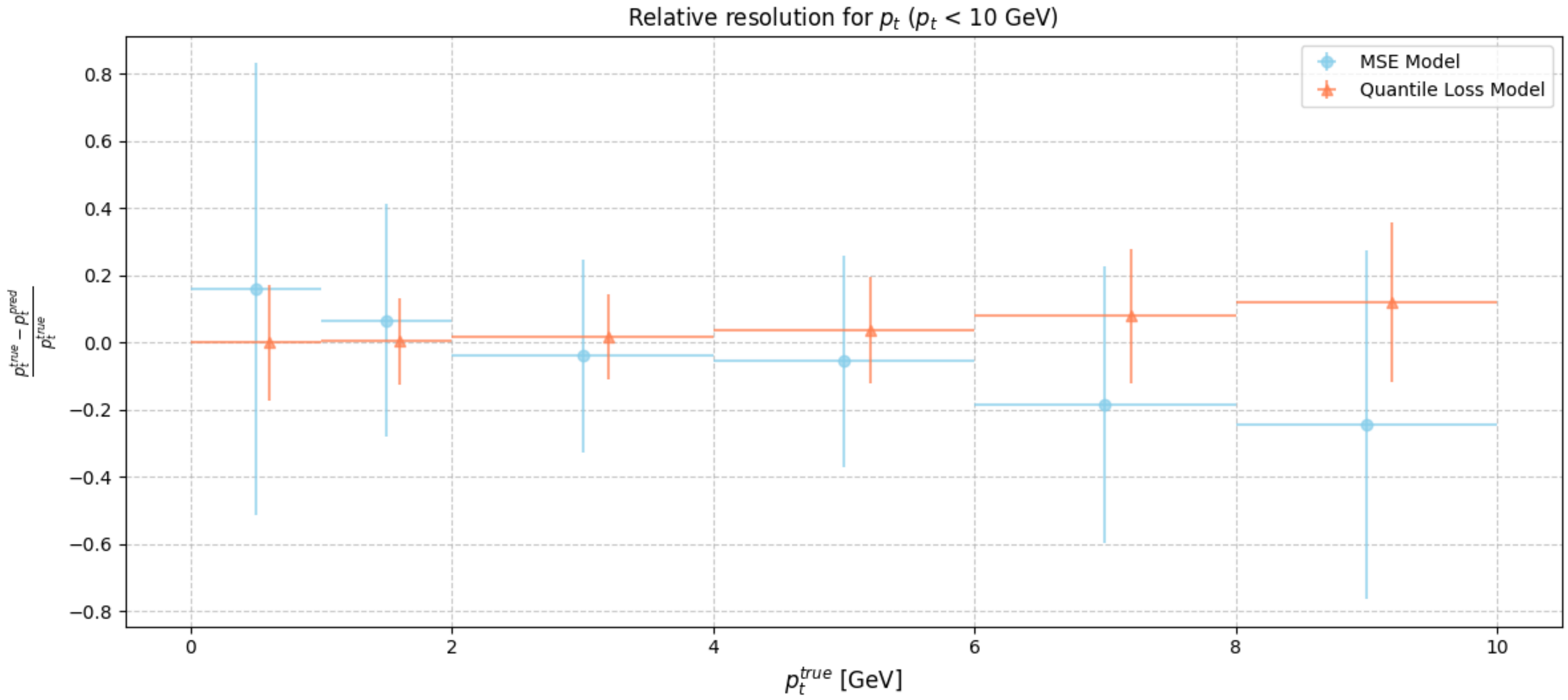
Realtive error resolution for p_t



More results

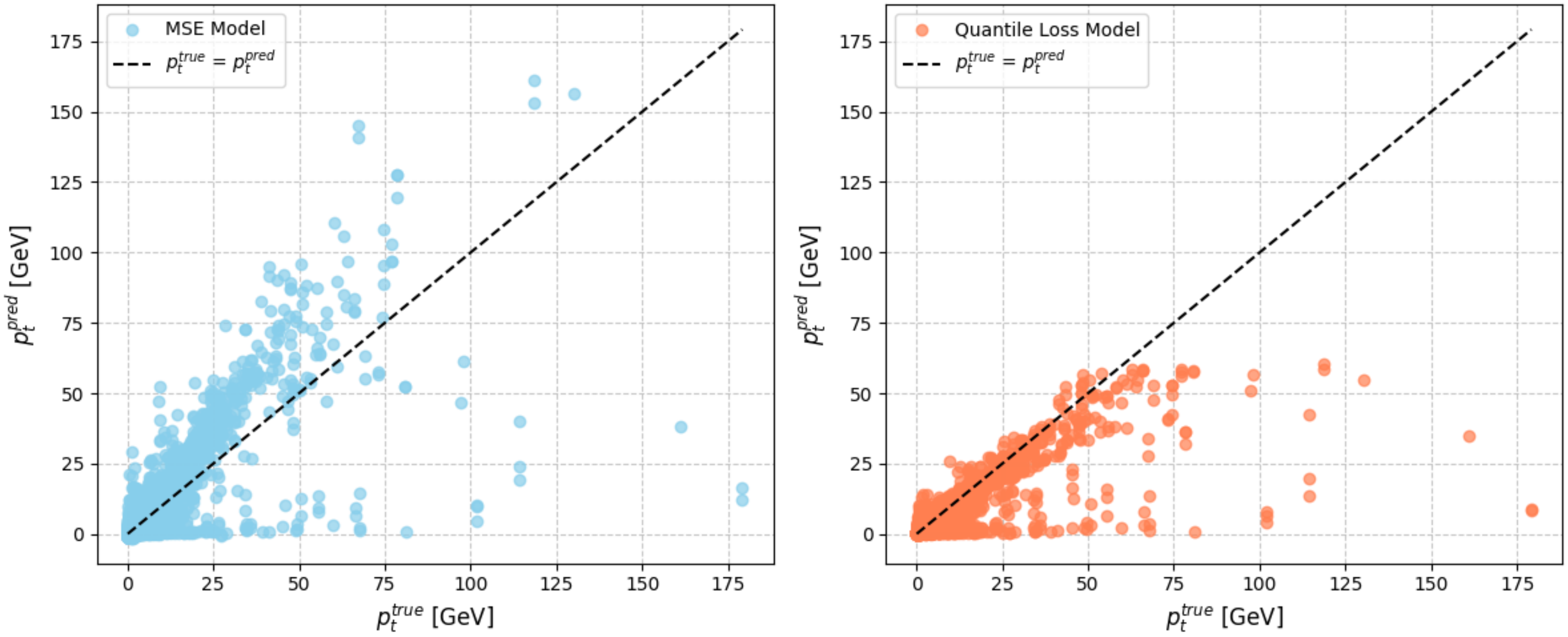


More results

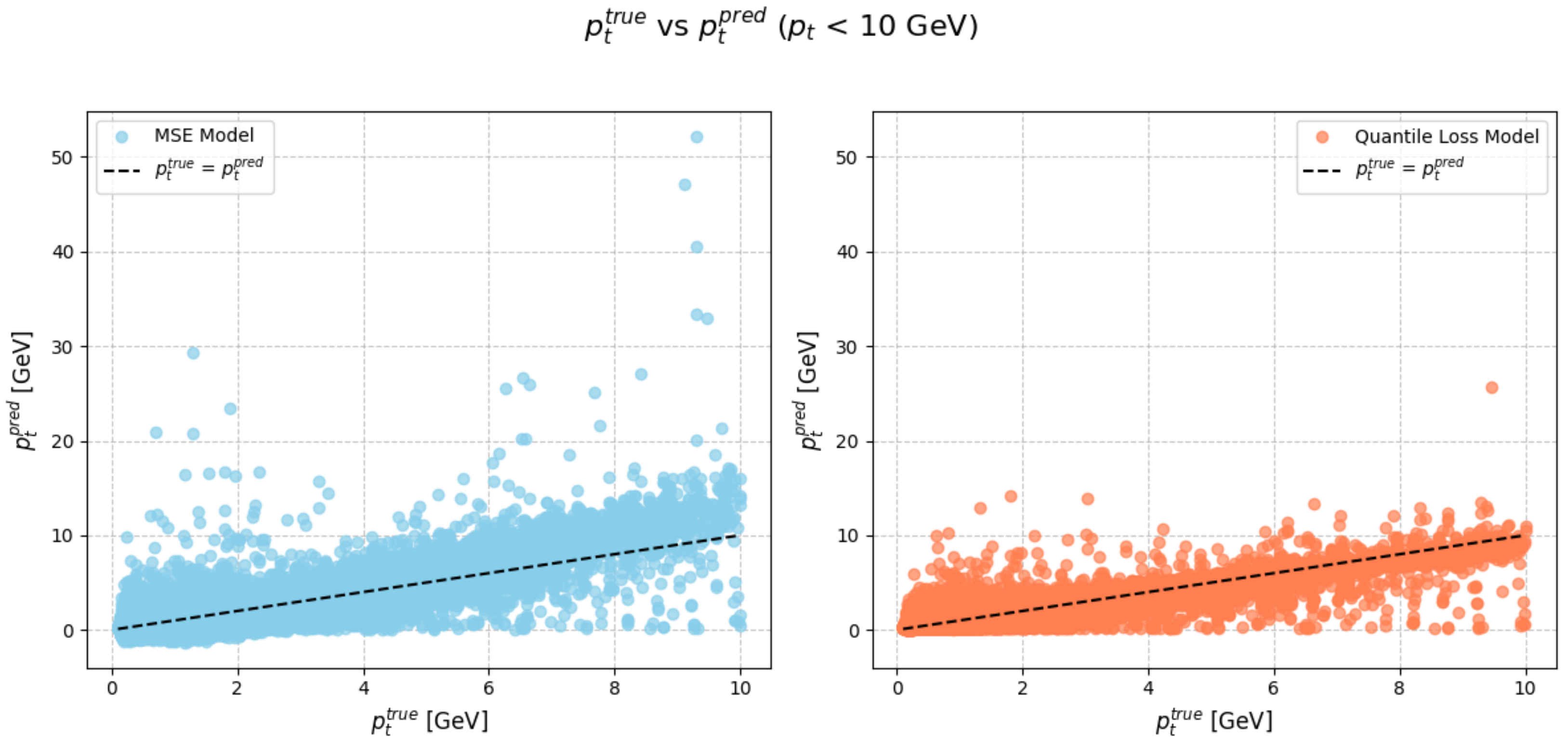


More results

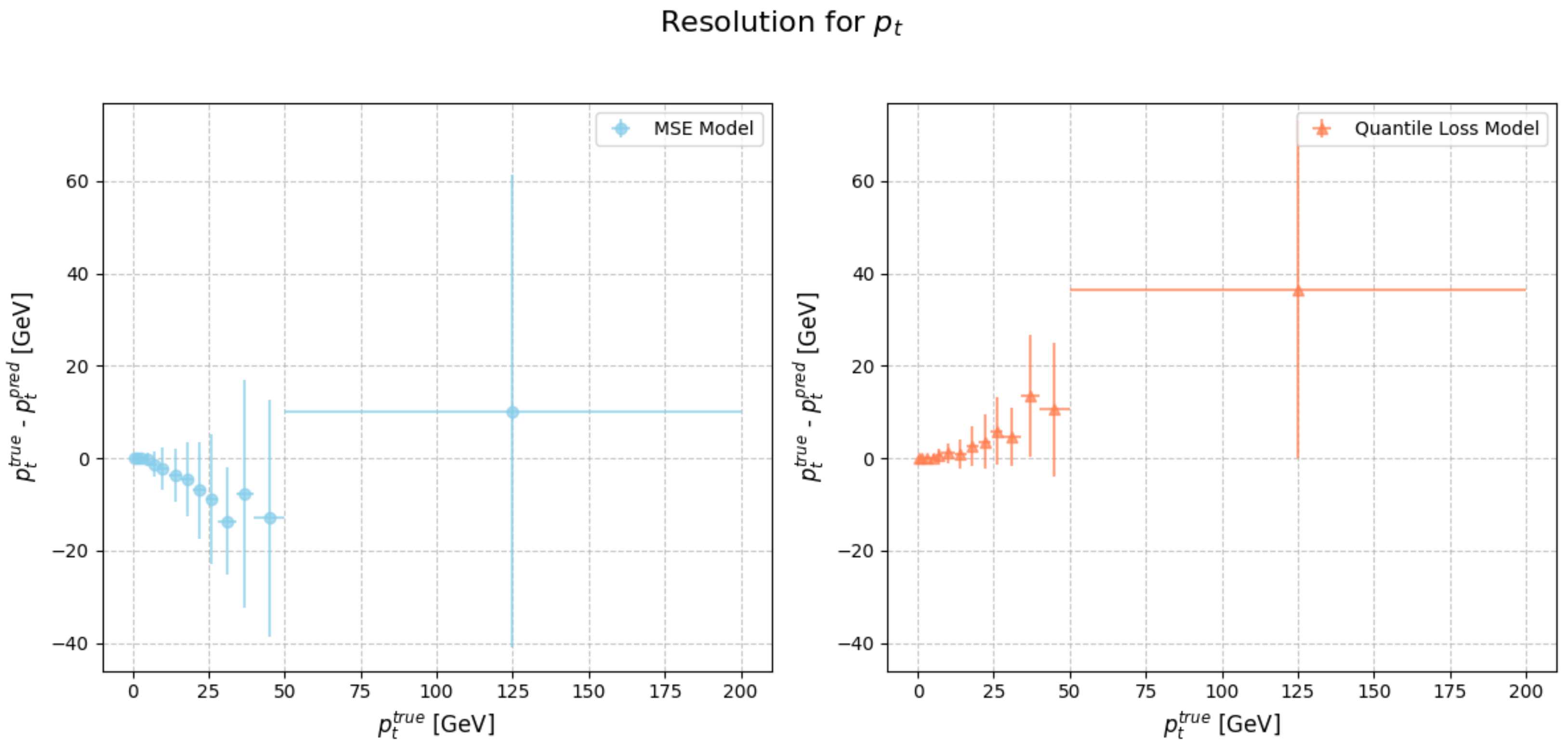
p_t^{true} vs p_t^{pred}



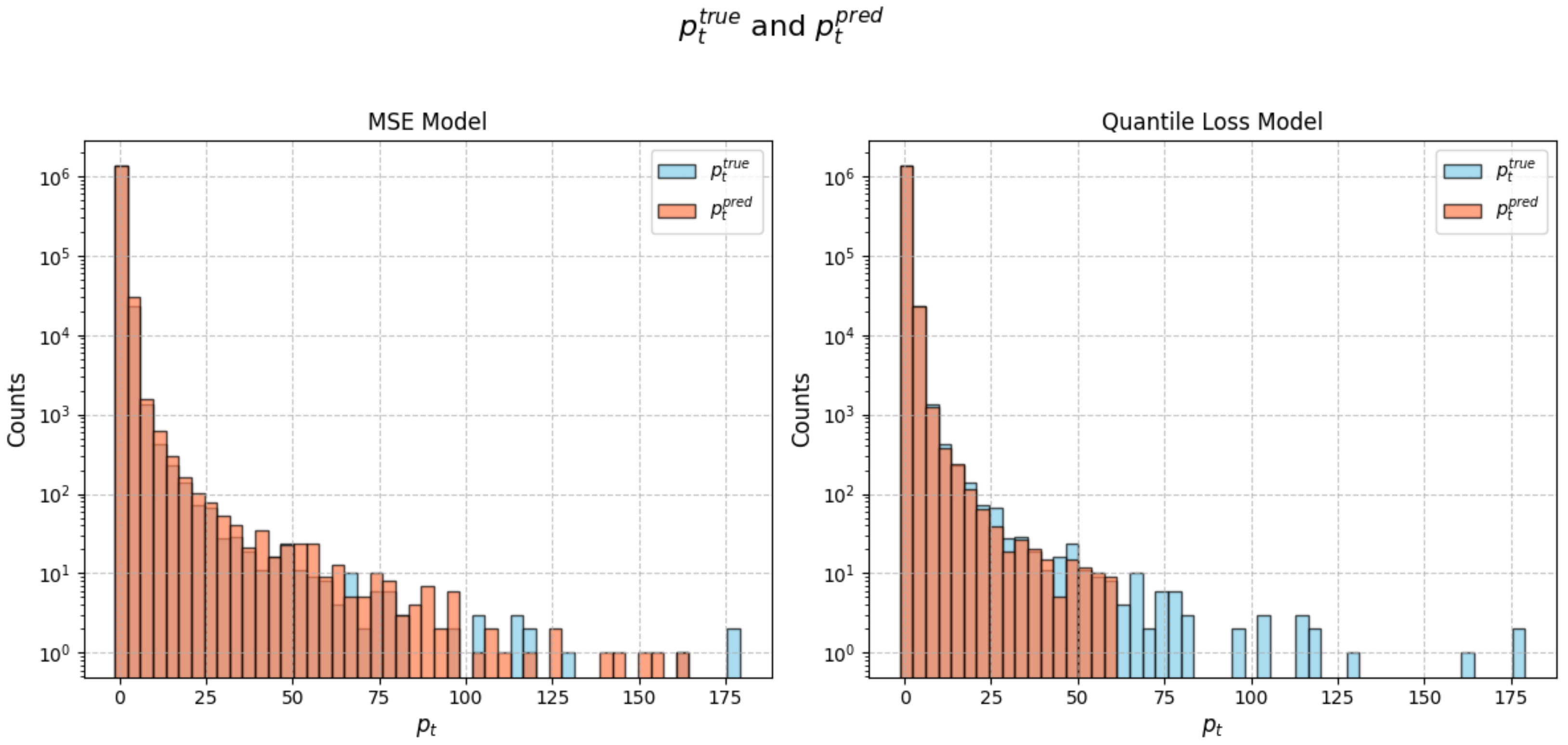
More results



More results

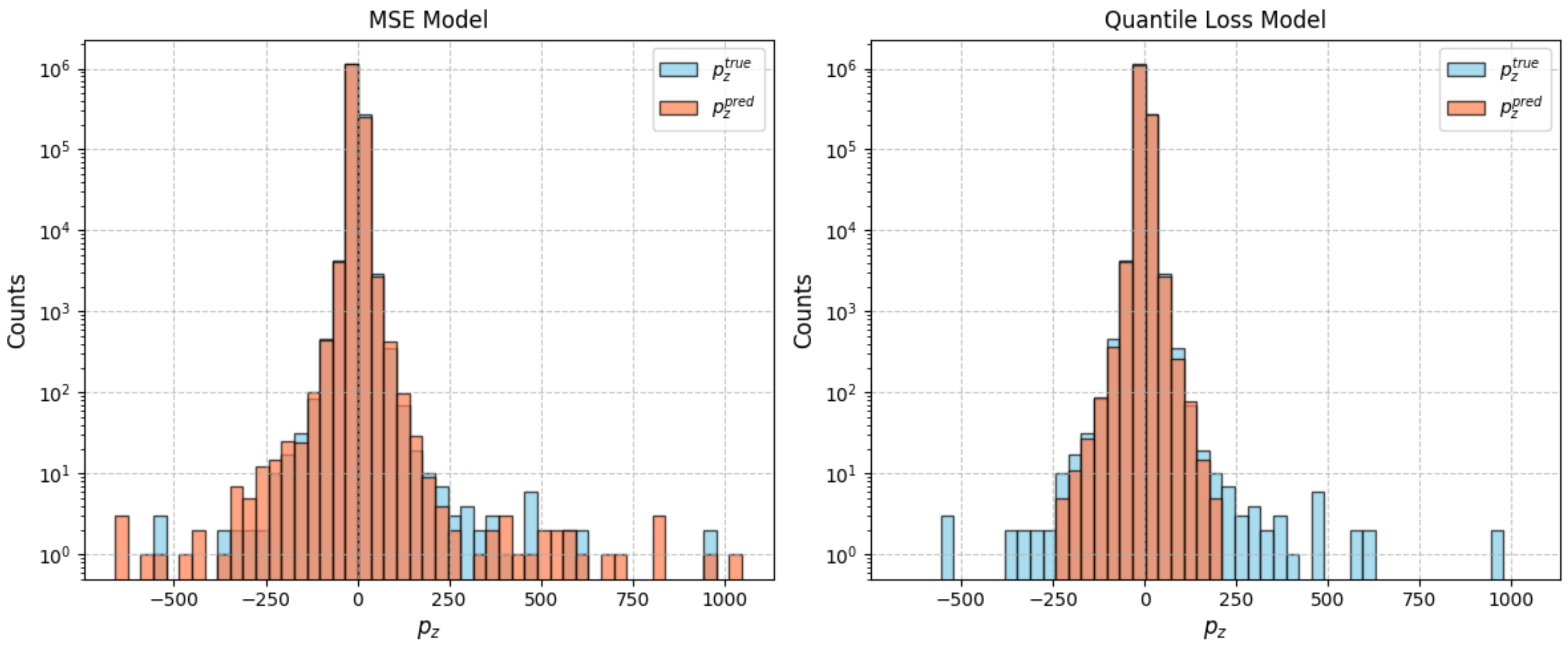


More results



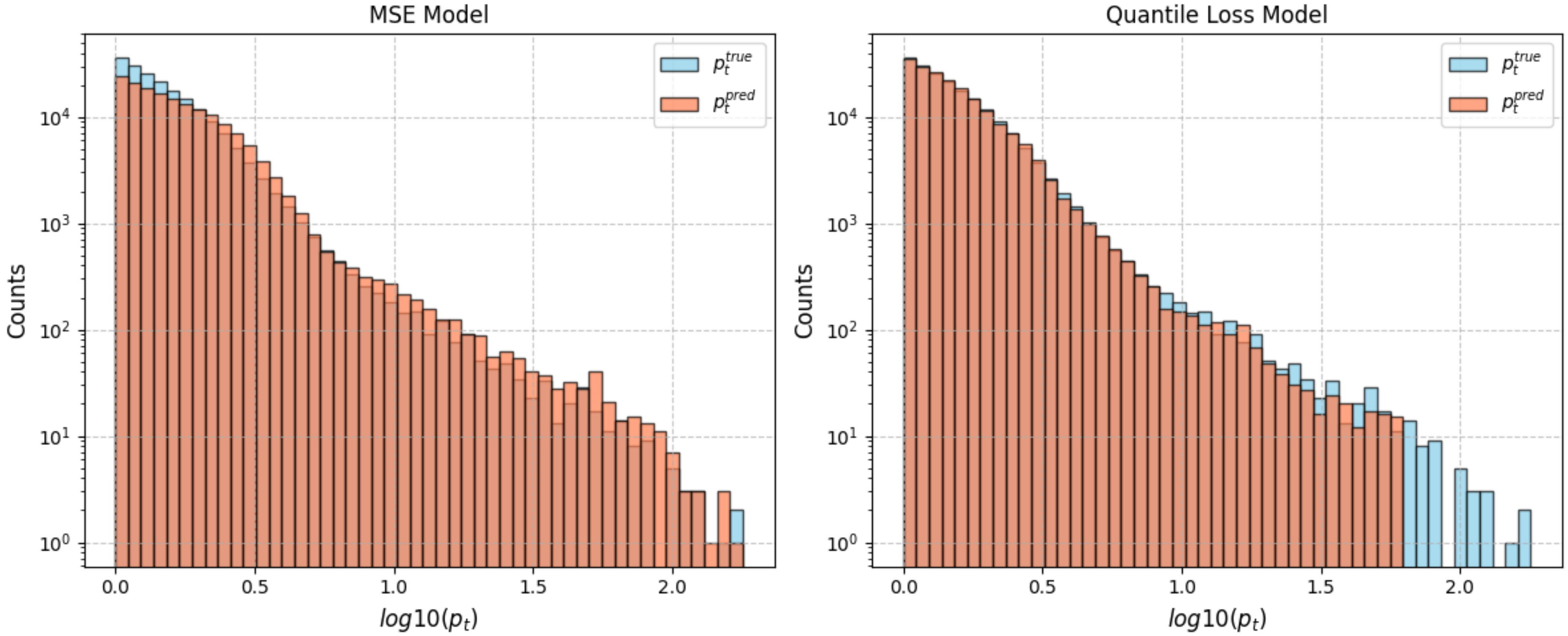
More results

p_z^{true} and p_z^{pred}



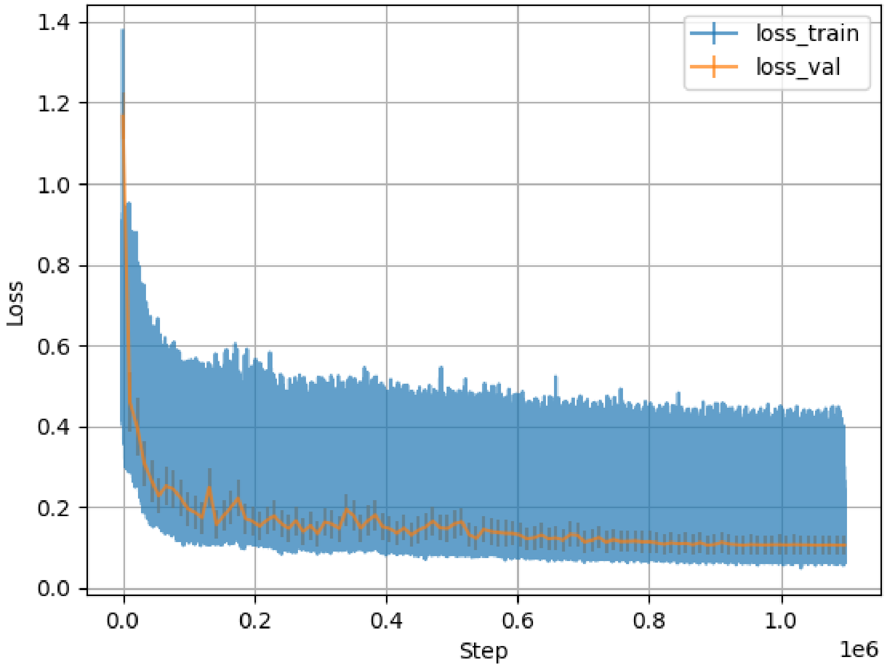
More results

$log10(p_t^{true})$ and $log10(p_t^{pred})$

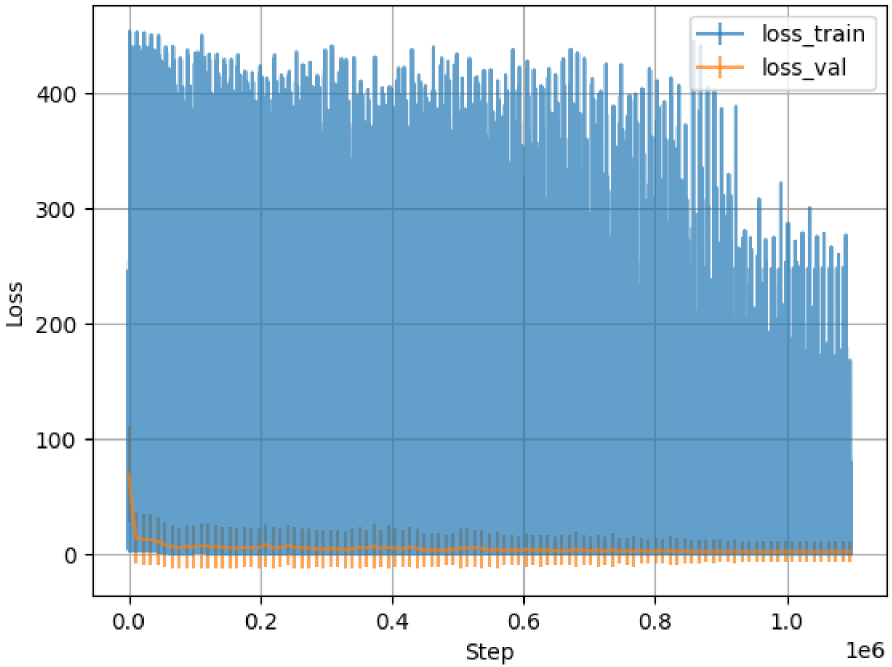


More results

Quantile Loss

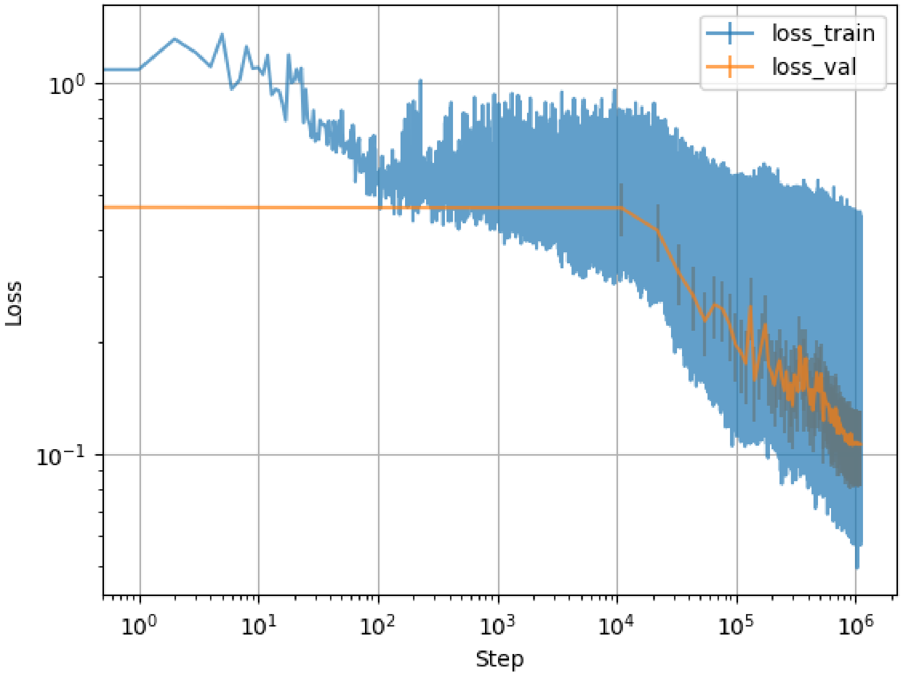


MSE Loss

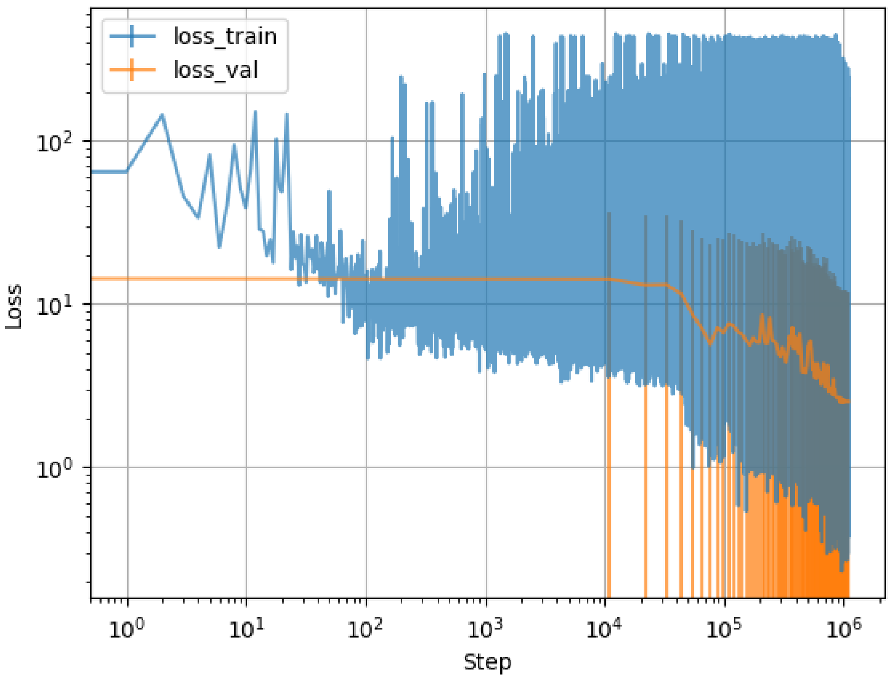


More results

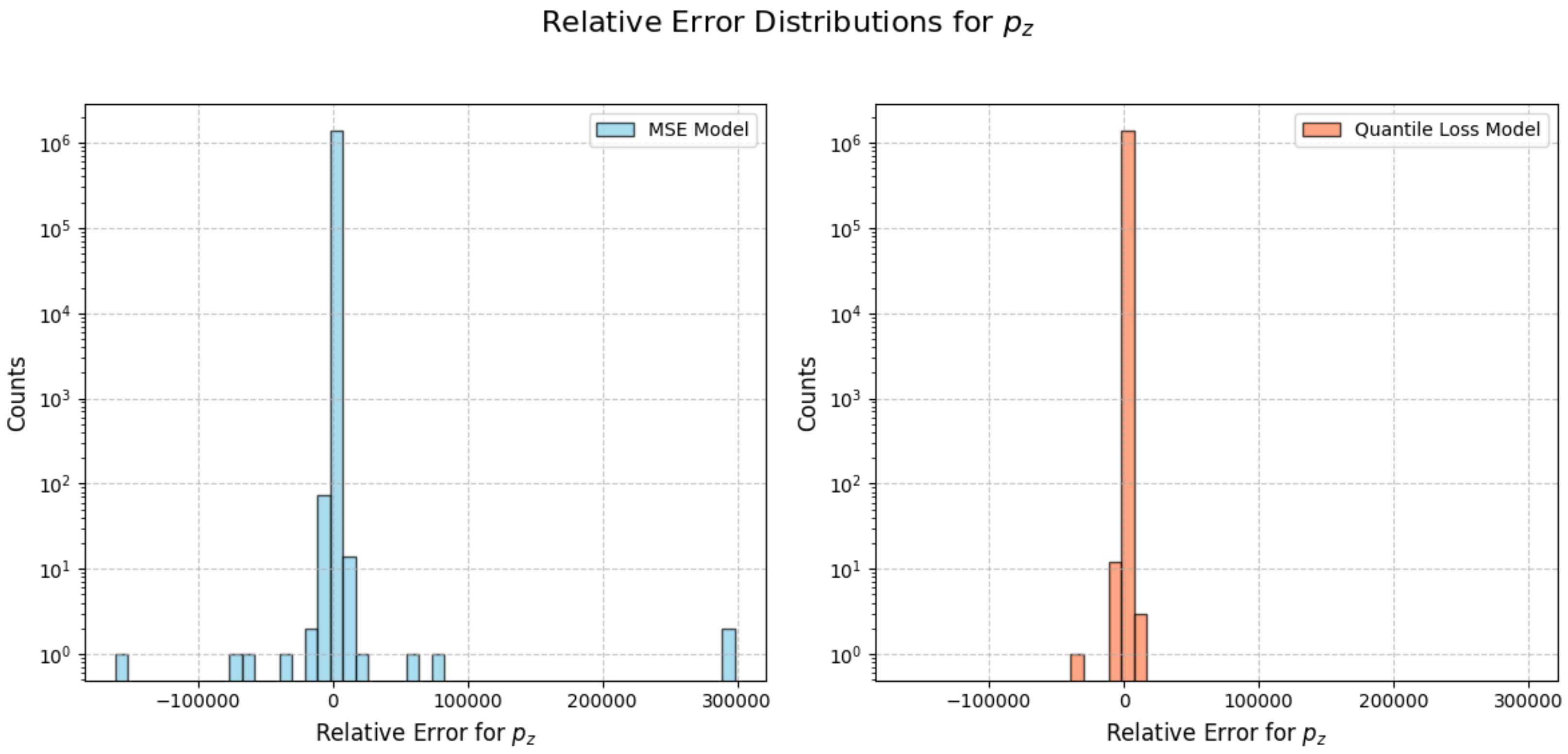
Quantile Loss



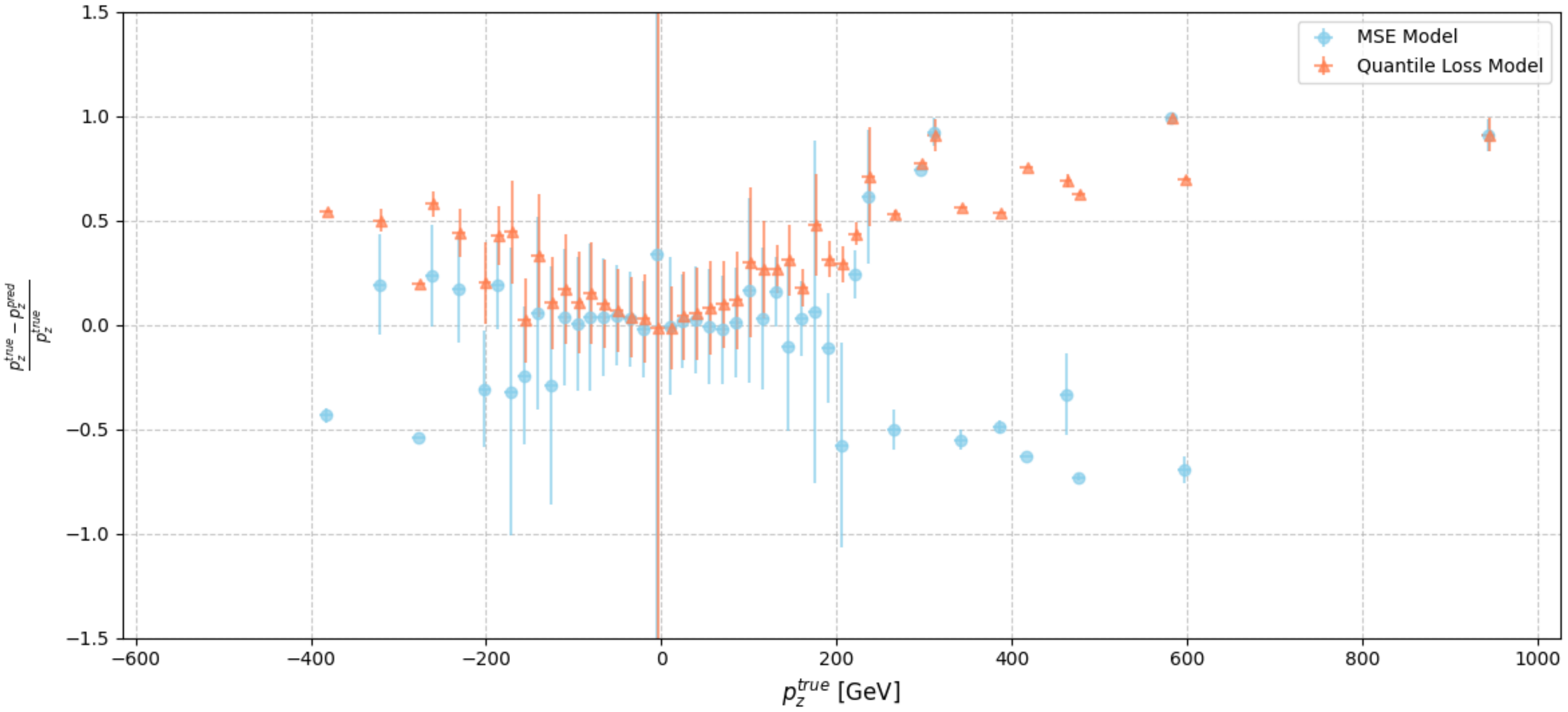
MSE Loss



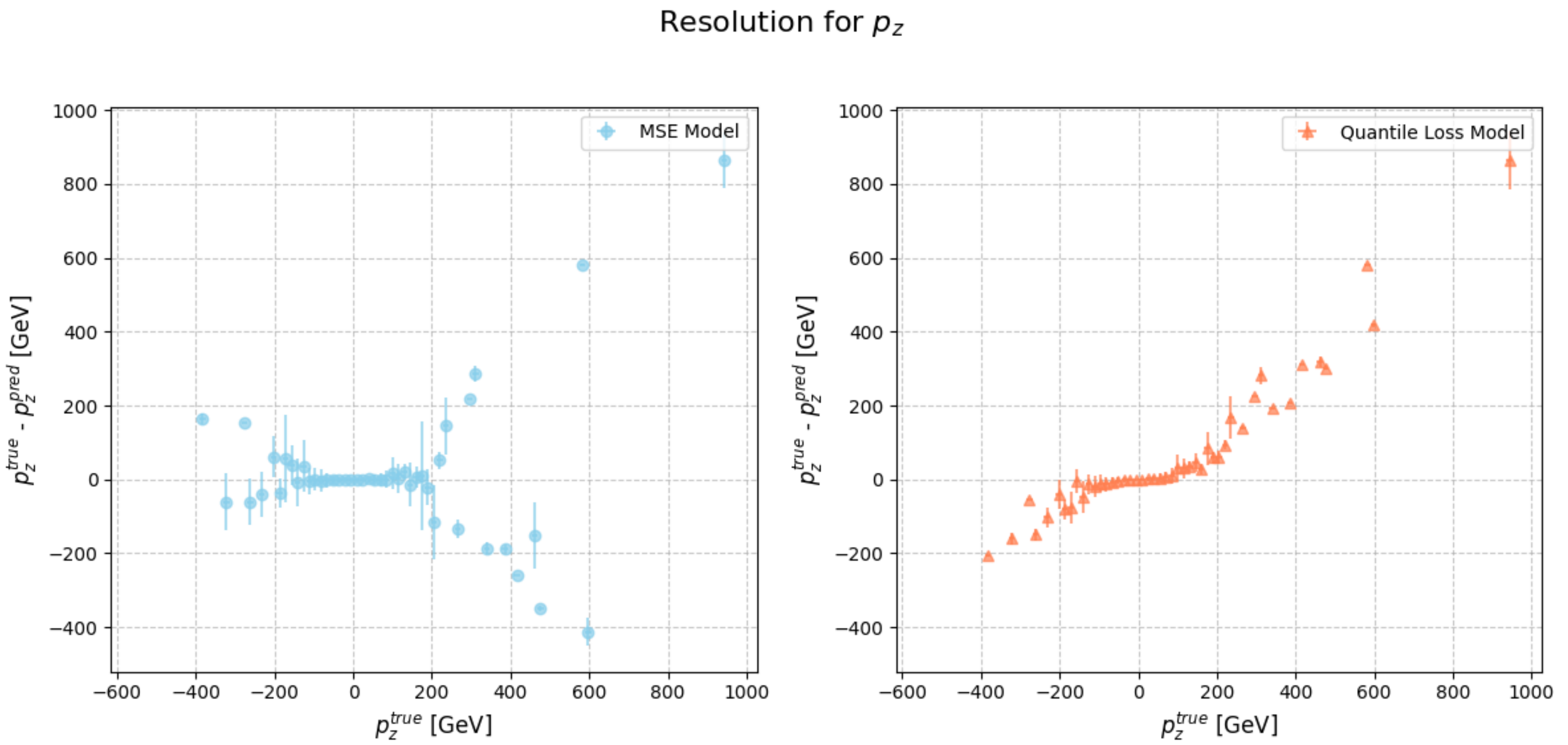
More results



More results



More results



BACKUP