Observable Optimization for Precision Theory: Machine Learning Energy Correlators

Arindam Bhattacharya

HARVARD
UNIVERSITY

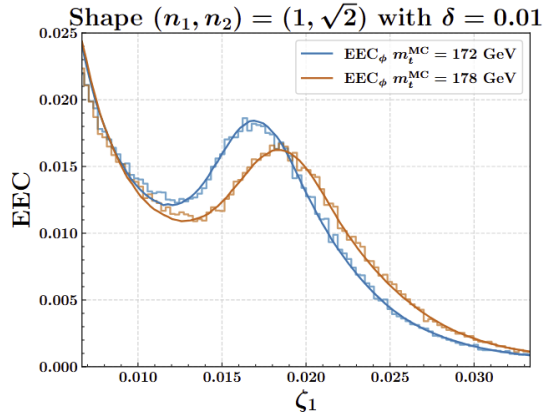
Based on work with Katie Fraser (UC Berkeley) & Matthew Schwartz (Harvard University)

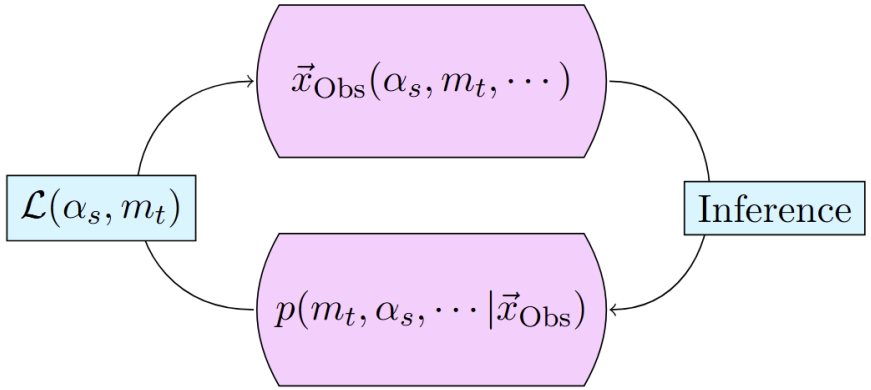
2508.10988

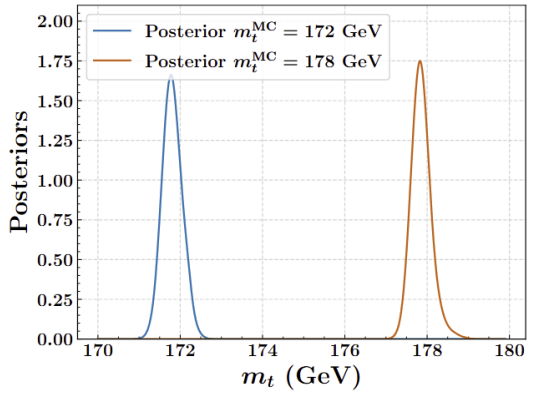
SYNERGIES TOWARDS
THE FUTURE STANDARD MODEL

DESY Theory Workshop

Outline







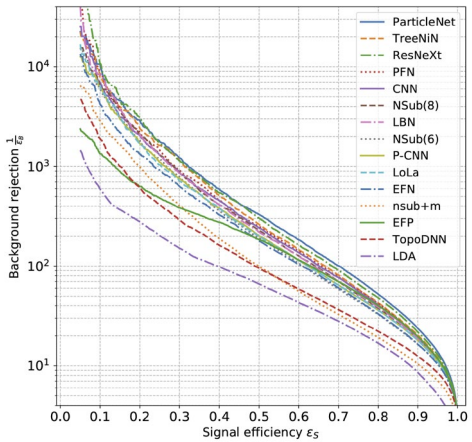
- Precision Theory meets Machine Learning
- Energy Correlators for Precision Top Physics
- Simulation Based Inference for Observable Optimization
- Conclusions

Precision Theory Meets Machine Learning

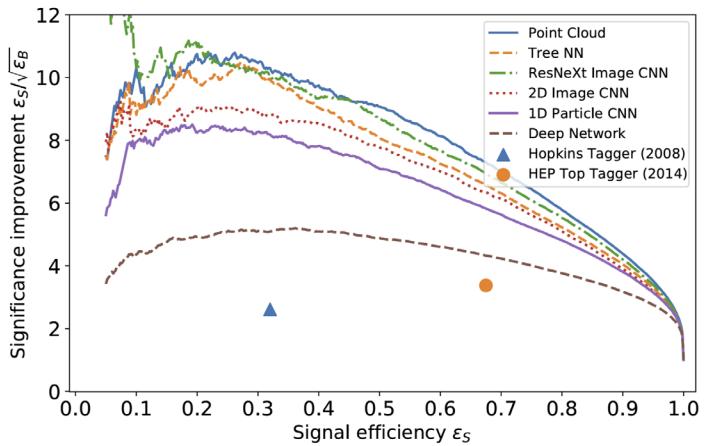
ML techniques are already beating classical methods at several HEP tasks

however

Ex : Top Tagging



Butter et.al. 1902.09914



M.D.Schwartz 2103.12226

Interpretability or computability of the observables parametrized by the neural nets still unclear

Precision theory needs observables that are *computable* from first principles in QFT

Simultaneously, they need to be useful for parameter estimation (mass, coupling,$\cdots$)

Use **ML** to *efficiently* and *quickly* **search** the space of **computable, precision observables** for **parameter sensitivity**

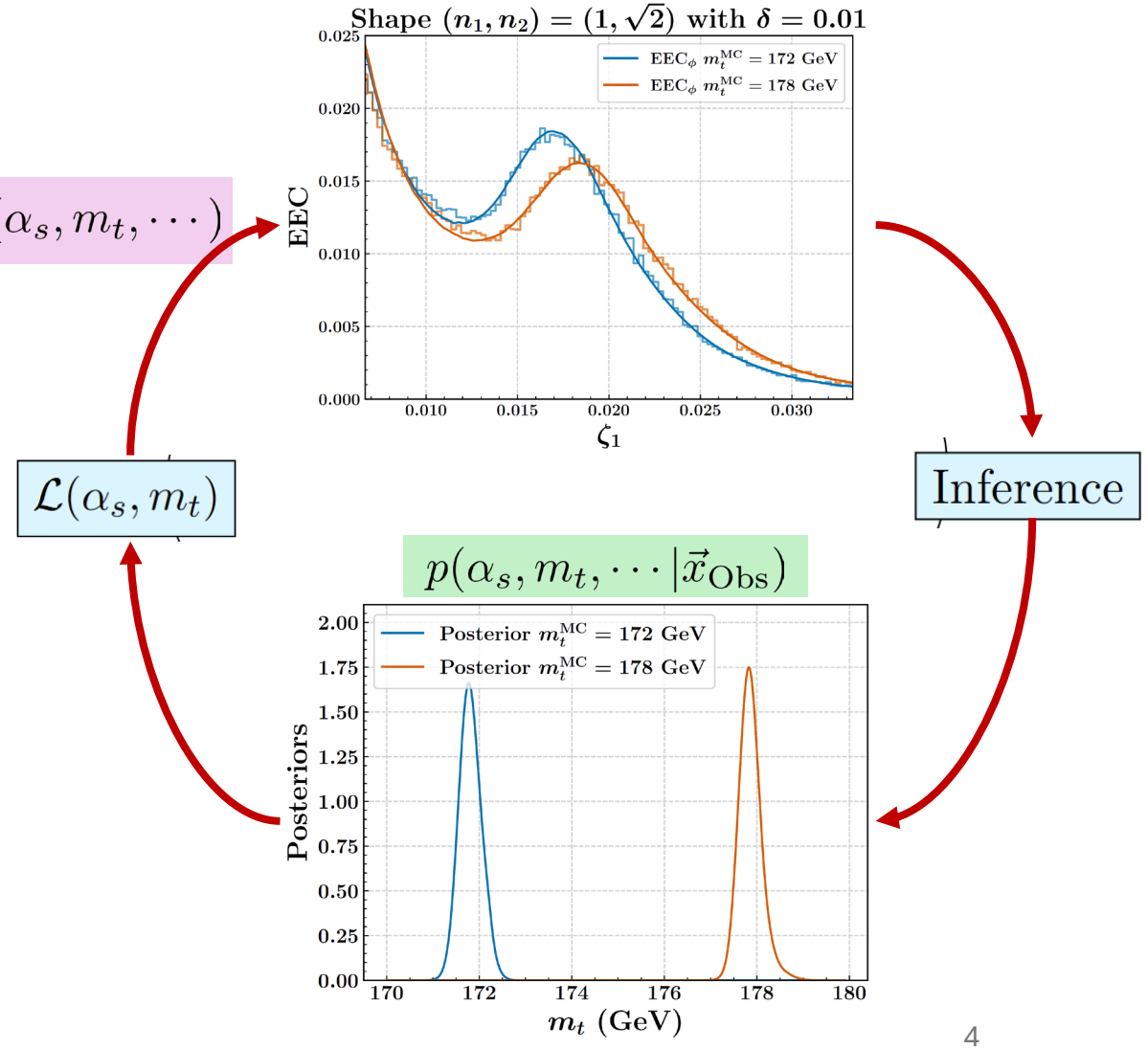
Precision Theory Meets Machine Learning

Use **ML** to *efficiently* and *quickly* **search** the space of **computable, precision observables** for **parameter sensitivity**

Crudely, two step process

- Learn the underlying multi-differential (master) distribution of precision observables

- Explore the space of marginals/observables for parameter sensitivity using inference



Energy Correlators for Precision Top Physics

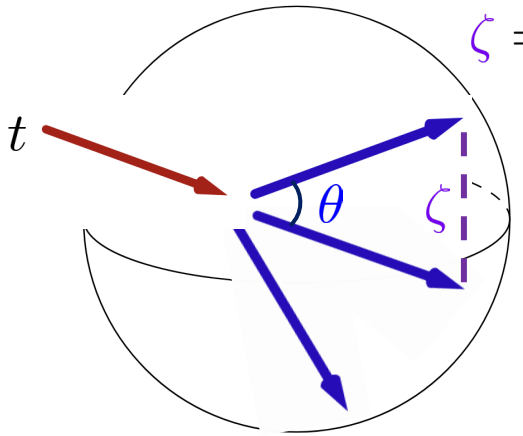
I'll illustrate this using **Energy Correlators** for determining **top quark mass**

OBSERVABLE PARAMETER

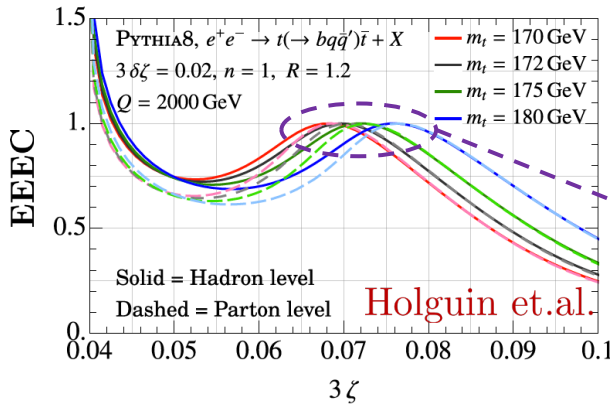
Energy correlators have seen a recent revival in precision HEP

Aditya's plenary on Tuesday

See review HX Zhu and I Moult 2506.09119



$$\zeta = (1 - \cos\theta)/2 \sim \theta^2$$



Holguin et.al. 2201.08393

k pt energy correlator measures angular correlation of energy flow at null infinity

$$\langle E(\vec{n}_1)E(\vec{n}_2)\cdots E(\vec{n}_k)\rangle = \frac{1}{\sigma}\int d\sigma \times E(\vec{n}_1) \times E(\vec{n}_2) \times \cdots \times E(\vec{n}_k)\ .$$

3-pronged decay of top quark gets embedded in 3 pt energy correlator

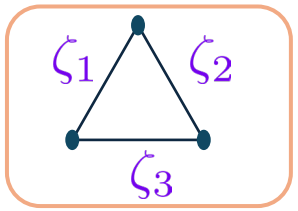
$$\zeta_t \sim \frac{m_{\rm Top}^2}{Q^2}$$

Holguin et.al. 2201.08393, 2311.02157, 2407.12900

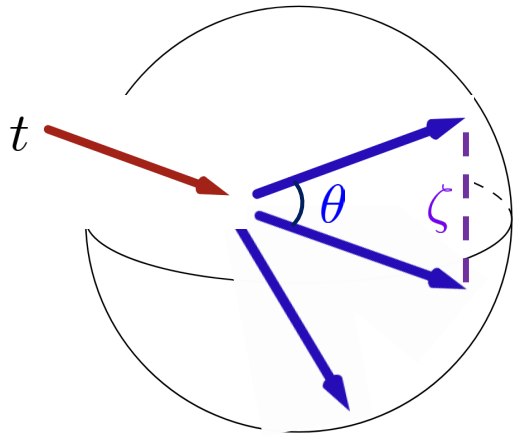
M Xiao, Y Ye, X Zhu 2405.20001

$$\mathrm{EEEC}(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{\sigma}\frac{d^3\sigma}{d\zeta_1 d\zeta_2 d\zeta_3}\ ,$$
$$= \frac{1}{\sigma}\frac{1}{2Q^2}\sum_{(i,j,k)}\int d\Pi_n\ \left|\mathcal{M}(e^+e^- \to B)\right|^2 \frac{E_i E_j E_k}{Q^3}\ \delta_{ijk}(\text{shape})$$

$$\delta_{ijk}(\text{shape}) = \delta\left(\zeta_1 - \frac{1-\cos\theta_{jk}}{2}\right)\ \delta\left(\zeta_2 - \frac{1-\cos\theta_{ki}}{2}\right)\ \delta\left(\zeta_3 - \frac{1-\cos\theta_{ij}}{2}\right)$$



Energy Correlators for Precision Top Physics

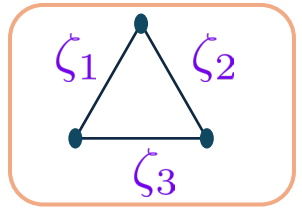


3-pronged decay of top quark gets embedded in 3 pt energy correlator

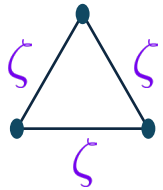
Holguin et.al. 2201.08393, 2311.02157, 2407.12900
M Xiao, Y Ye, X Zhu 2405.20001

$$\mathrm{EEEC}(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{\sigma}\frac{d^3\sigma}{d\zeta_1 d\zeta_2 d\zeta_3}\,,$$
$$= \frac{1}{\sigma}\frac{1}{2Q^2}\sum_{(i,j,k)}\int d\Pi_n \;\left|\mathcal{M}(e^+e^-\to B)\right|^2 \frac{E_iE_jE_k}{Q^3}\,\delta_{ijk}(\text{shape})$$

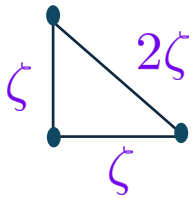
$$\delta_{ijk}(\text{shape}) = \delta\left(\zeta_1 - \tfrac{1-\cos\theta_{jk}}{2}\right)\,\delta\left(\zeta_2 - \tfrac{1-\cos\theta_{ki}}{2}\right)\,\delta\left(\zeta_3 - \tfrac{1-\cos\theta_{ij}}{2}\right)$$

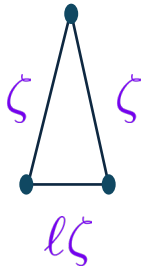


Considered particular shapes



Can other shapes regress m_{Top} better?





Can we find the *ideal* shape/marginal that encompasses the same amount of info as EEEC($\vec{\zeta}$)?

ML !

1. Make the machine learn EEEC
2. Quantify and compare shapes for m_{Top} sensitivity

Machine Learning Energy Correlators

$$\mathrm{EEEC}(\zeta_1, \zeta_2, \zeta_3) = \frac{1}{\sigma} \frac{d^3\sigma}{d\zeta_1 d\zeta_2 d\zeta_3}$$ ← We seek to learn the master distribution

Use MC simulation from PYTHIA
Hadronically decaying top jets from $e^+e^- \to t\bar{t}$ at $Q = 2$ TeV
$R = 1.2$ anti-k_t jets reclustered using $R = 0.1$ using Cambridge Aachen

PYTHIA produces tuples of form $\vec{x} = \left(\zeta_1, \zeta_2, \zeta_3, \frac{E_1 E_2 E_3}{Q^3}, m_t\right)$

Learn $p(\vec{x})$ using samples in an unbinned manner

Neural network surrogate $p_{\boldsymbol{\phi}}(\vec{x})$ that converges to $p(\vec{x})$ with loss function

$$\begin{aligned}\mathcal{L}_{\mathrm{KL}}(\boldsymbol{\phi}) &= D_{\mathrm{KL}}(p_{\mathrm{Data}}(\vec{x}) || p_{\boldsymbol{\phi}}(\vec{x})) \\ &= -\mathbb{E}_{p_{\mathrm{Data}}(\vec{x})}\left[\ln p_{\boldsymbol{\phi}}(\vec{x})\right] + \text{constant} \\ &\approx -\frac{1}{N}\sum_{i=1}^{N} \ln p_{\boldsymbol{\phi}}(\vec{x}_{i,\ \mathrm{Data}}) \ + \ \text{constant}\end{aligned}$$

→ Cumbersome, does not learn region of interest pertinent to m_{Top}

Use EEC observable definition inspired loss!

Machine Learning Energy Correlators

$$\mathrm{EEEC}(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{\sigma}\frac{d^3\sigma}{d\zeta_1 d\zeta_2 d\zeta_3}$$ ← We seek to learn the master distribution

PYTHIA produces tuples of form $\vec{x} = \left(\zeta_1,\zeta_2,\zeta_3,\frac{E_1E_2E_3}{Q^3},m_t\right)$

Use EEC observable definition inspired loss : Energy weight the loss!

Upto normalization, EEC is a positively weighted probability distribution itself

$$\mathrm{EEEC}(\vec{\zeta},m_t) = \int d\tilde{E}\ \tilde{E}\ p(\vec{\zeta},\tilde{E},m_t)$$

Neural network surrogate $\mathrm{EEEC}_{\boldsymbol{\phi}}$ that converges to true EEEC with loss function

$$\begin{aligned}
\mathcal{L}_{\mathrm{EEEC}}(\boldsymbol{\phi}) &= D_{\mathrm{KL}}(\mathrm{EEEC}_{\mathrm{Data}}(\vec{\zeta},m_t)||\mathrm{EEEC}_{\boldsymbol{\phi}}(\vec{\zeta},m_t)) \\
&= -\mathbb{E}_{\mathrm{EEEC}_{\mathrm{Data}}}[\ln \mathrm{EEEC}_{\boldsymbol{\phi}}(\vec{\zeta},m_t)] + \lambda\left|\ln\left(\int d^3\vec{\zeta}\,dm_t\ \mathrm{EEEC}_{\boldsymbol{\phi}}(\vec{\zeta},m_t)\right)\right| \\
&\approx -\frac{1}{N}\sum_{i=1}^{N}\tilde{E}_i \ln \mathrm{EEEC}_{\boldsymbol{\phi}}(\vec{\zeta}_i,m_{t,i}) + \lambda\left|\ln\left(\frac{1}{M}\sum_{j=1}^{M}\mathrm{EEEC}_{\boldsymbol{\phi}}(\vec{\zeta}_j,m_{t,j})\right)\right|,
\end{aligned}$$

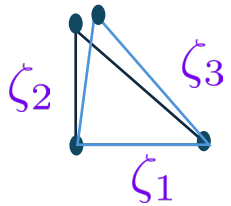
Term to enforce normalized surrogate
Needed for simple architectures like DNN, but not for Normalizing Flows

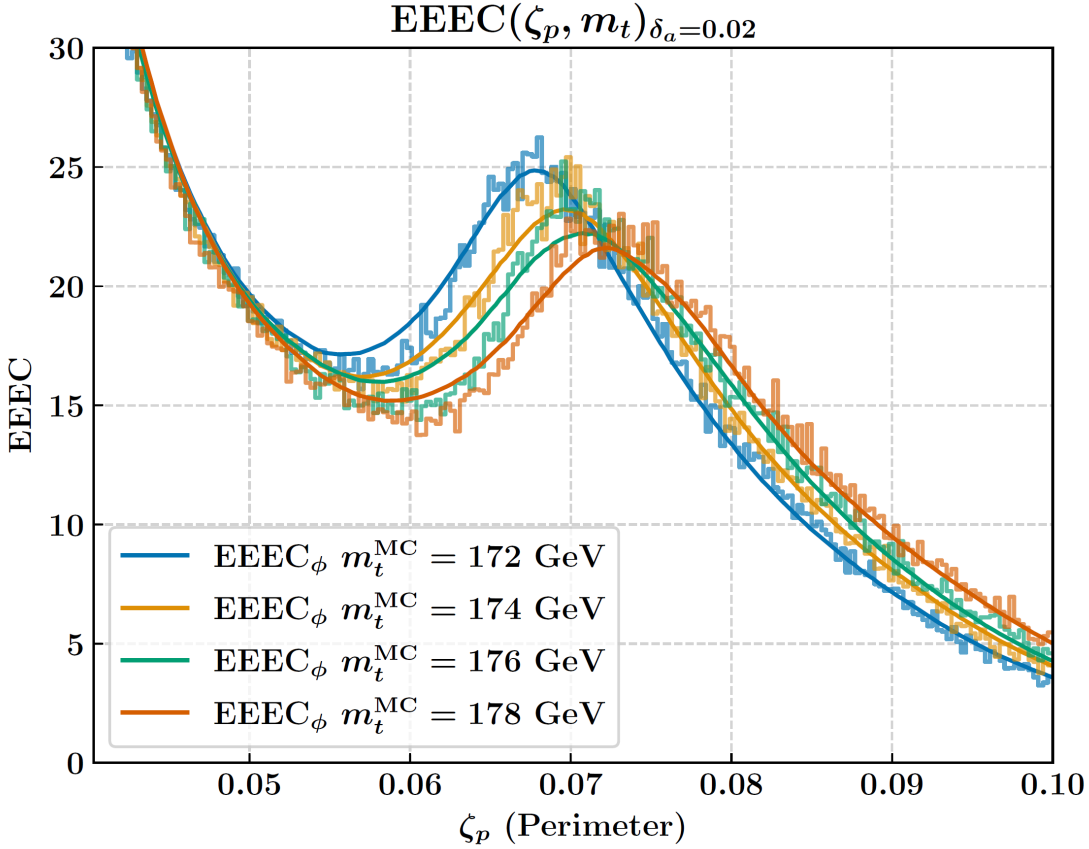
Machine Learning Energy Correlators

Find that the learnt surrogate interpolates well and covers all of the regions well

$$\mathrm{EEEC}(\zeta_p, m_t)_{\delta\zeta} = \int_0^1 d\zeta_1\, d\zeta_2\, d\zeta_3\, \mathrm{EEEC}(\zeta_1, \zeta_2, \zeta_3, m_t)\, \delta\left(\zeta_p - \sum_{i=1}^{3} \zeta_i\right)$$
$$\times\ \theta(\zeta_3 - \zeta_2)\, \theta(\zeta_2 - \zeta_1)\, \theta(\delta\zeta - \zeta_3 + \zeta_1)$$

Asymmetry

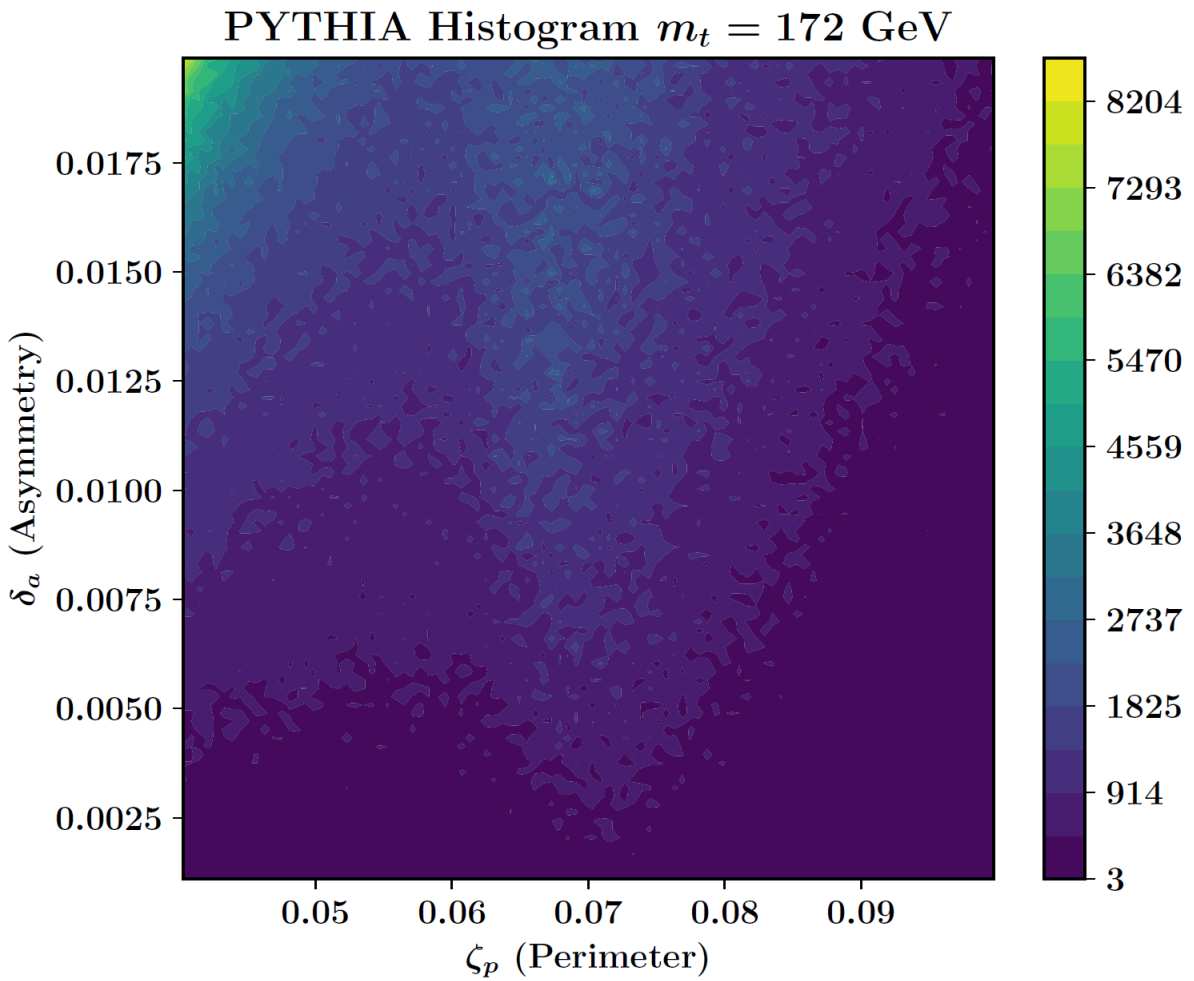


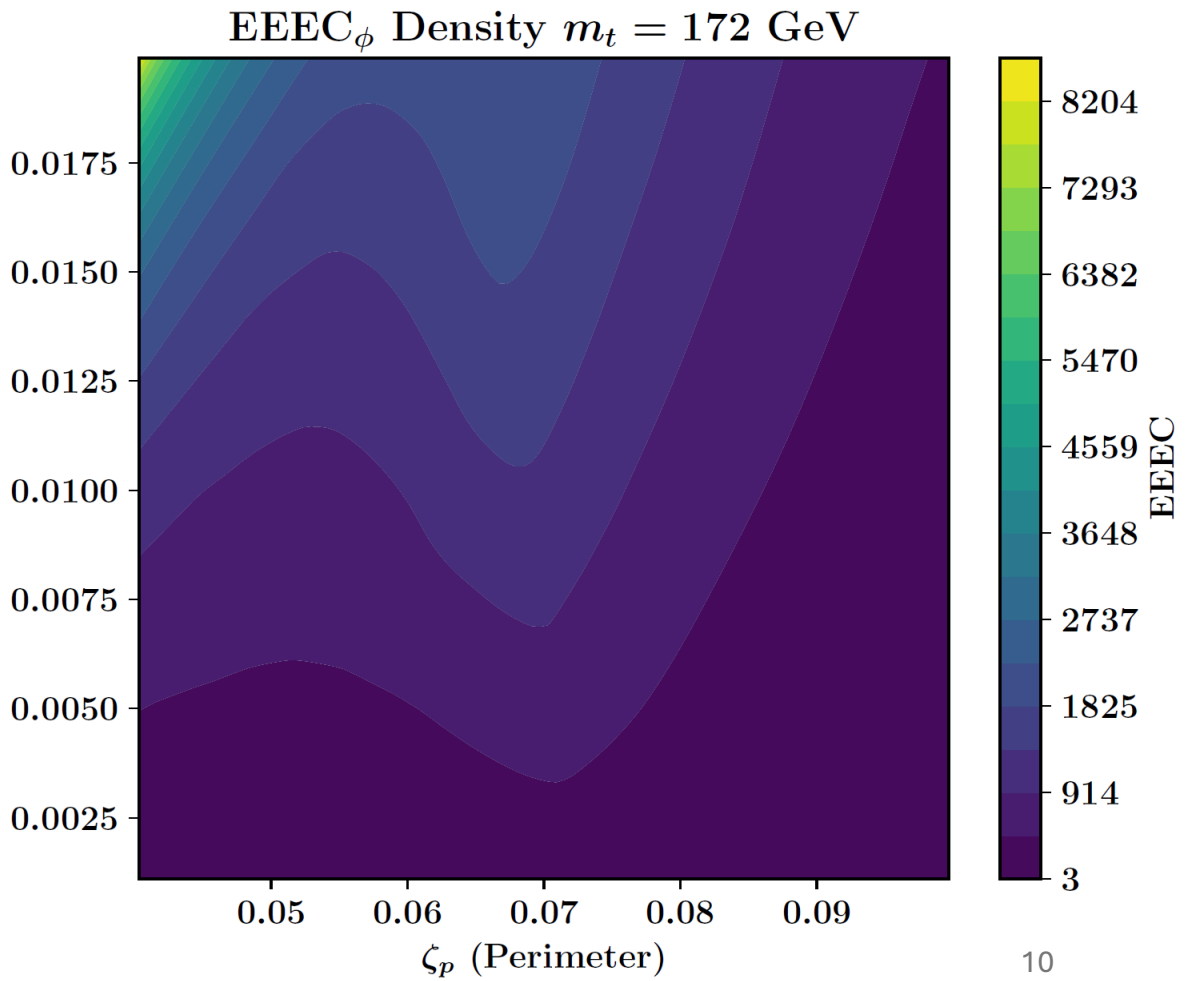


Machine Learning Energy Correlators

Find that the learnt surrogate interpolates well and covers all of the regions well

$$\mathrm{EEEC}(\zeta_p, \delta\zeta_a, m_t)_{\delta\zeta} = \int_0^1 d\zeta_1\ d\zeta_2\ d\zeta_3\ \mathrm{EEEC}(\zeta_1, \zeta_2, \zeta_3, m_t)\ \delta\left(\zeta_p - \sum_{i=1}^{3}\zeta_i\right)\ \delta(\delta\zeta_a - \zeta_3 + \zeta_1) \times\ \theta(\zeta_3 - \zeta_2)\ \theta(\zeta_2 - \zeta_1)$$



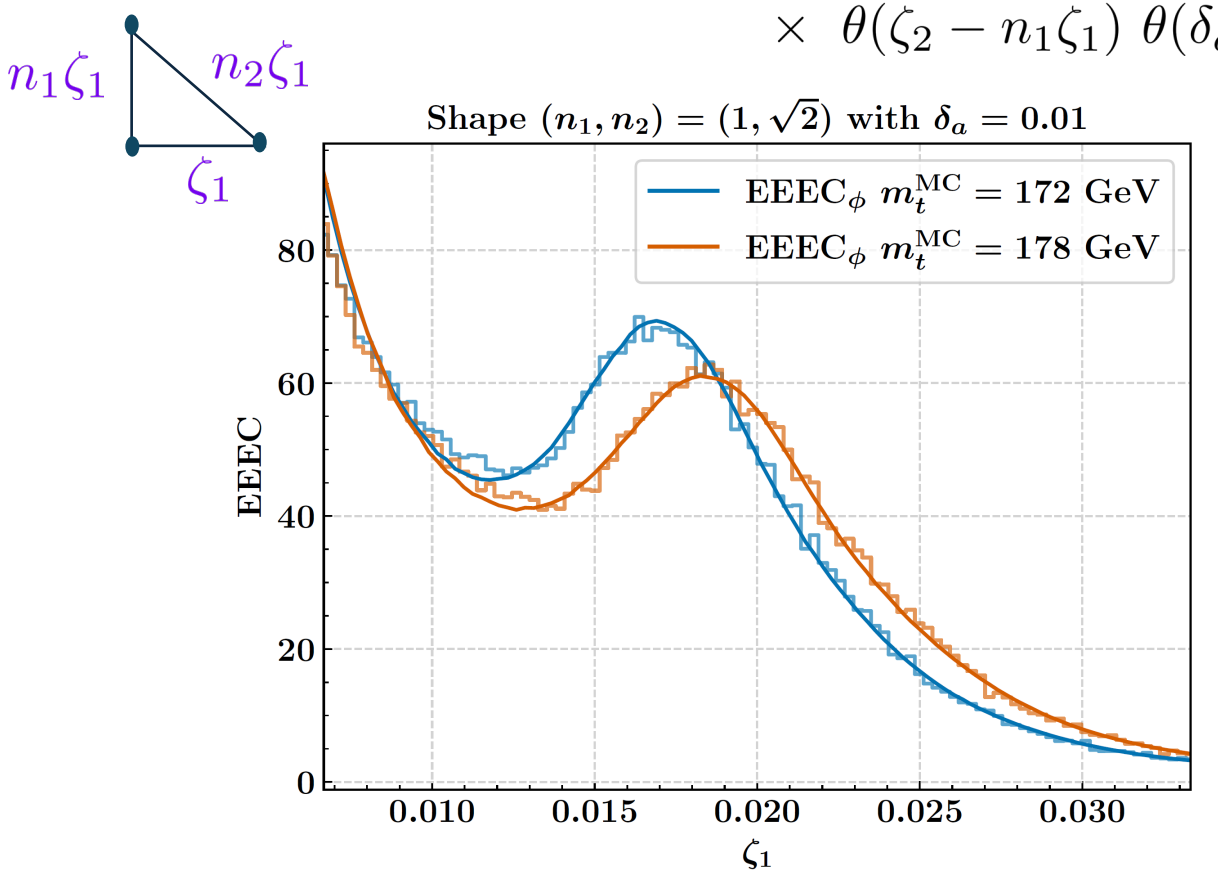


Machine Learning Energy Correlators

Find that the learnt surrogate interpolates well and covers all of the regions well

Triangles parametrized by ratio of sides and a asymmetry window

$$\mathrm{EEEC}(\zeta_1, n_1, n_2, m_t)_{\delta_a} = \int_0^1 d\zeta_2 \; d\zeta_3 \; \mathrm{EEEC}(\zeta_1, \zeta_2, \zeta_3, m_t)$$
$$\times \; \theta(\zeta_2 - n_1\zeta_1)\; \theta(\delta_a + n_1\zeta_1 - \zeta_2)\; \theta(\zeta_3 - n_2\zeta_1)\; \theta(\delta_a + n_2\zeta_1 - \zeta_3)$$

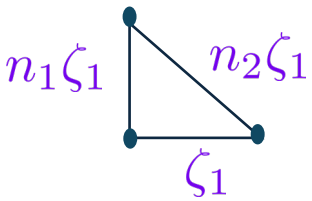


- Fast, continuous interpolator for EEEC distribution
- Access to multiple shapes, even if these are sparsely populated in samples

Observable Optimization : Neural Ratio Estimation

Triangles parametrized by ratio of sides and a asymmetry window

$$\mathrm{EEEC}(\zeta_1, n_1, n_2, m_t)_{\delta_a} = \int_0^1 d\zeta_2\ d\zeta_3\ \mathrm{EEEC}(\zeta_1, \zeta_2, \zeta_3, m_t)$$
$$\times\ \theta(\zeta_2 - n_1\zeta_1)\ \theta(\delta_a + n_1\zeta_1 - \zeta_2)\ \theta(\zeta_3 - n_2\zeta_1)\ \theta(\delta_a + n_2\zeta_1 - \zeta_3)$$



Onto step 2 : Searching the space of precision observables (shapes above)

Need a method that gives a parameter estimate (m_t) and an uncertainty

Neural Ratio Estimation : Compute a posterior given a shape ✓

J. Hermans, V. Begy, and G. Louppe, 1903.04057
A. Cole et.al. 2111.08030
B. K. Miller, C. Weniger, and P. Forré, 2210.06170

$$\frac{p(\vec{\zeta}_{\mathrm{shape}}, m_t)}{p(\vec{\zeta}_{\mathrm{shape}})p(m_t)} = \frac{p(\vec{\zeta}_{\mathrm{shape}}|m_t)}{p(\vec{\zeta}_{\mathrm{shape}})} = \frac{p(m_t|\vec{\zeta}_{\mathrm{shape}})}{p(m_t)}$$

Posterior
Prior

Use a classifier to learn the ratio

Observable Optimization : Neural Ratio Estimation

Neural Ratio Estimation : Compute a posterior given a shape

J. Hermans, V. Begy, and G. Louppe, 1903.04057
A. Cole et.al. 2111.08030
B. K. Miller, C. Weniger, and P. Forré, 2210.06170

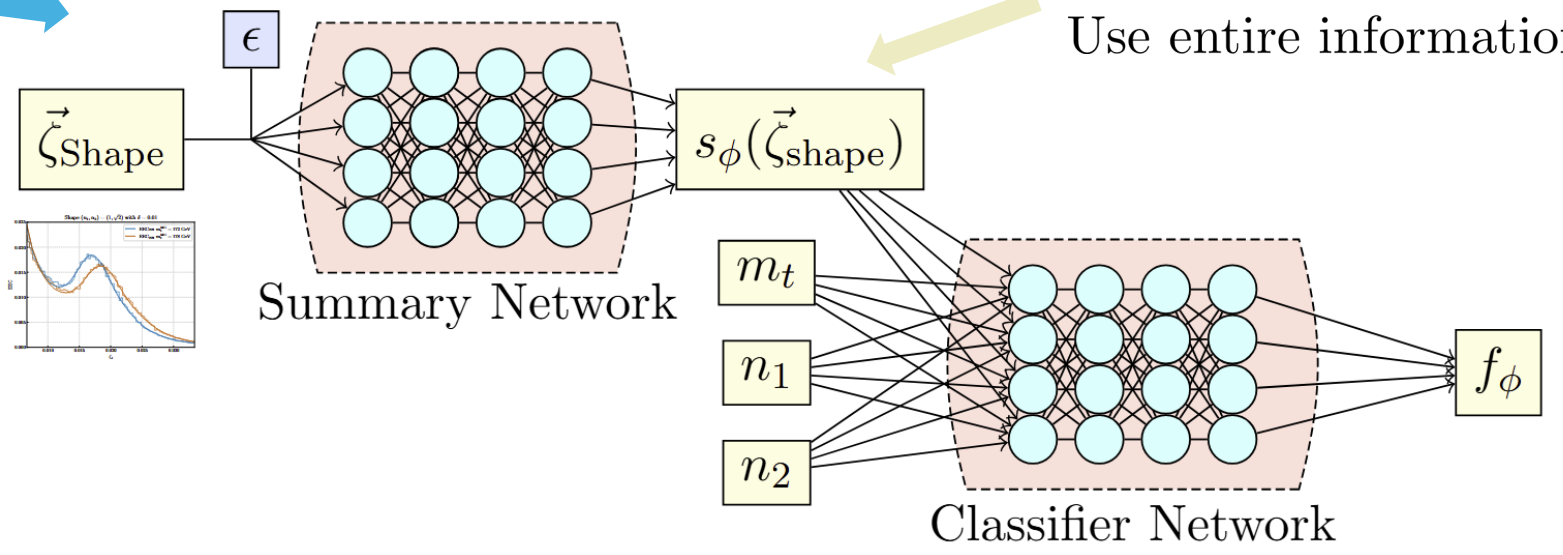
$$\frac{p(\vec{\zeta}_{\text{shape}}, m_t)}{p(\vec{\zeta}_{\text{shape}})p(m_t)} = \frac{p(\vec{\zeta}_{\text{shape}}|m_t)}{p(\vec{\zeta}_{\text{shape}})} = \frac{p(m_t|\vec{\zeta}_{\text{shape}})}{p(m_t)}$$

Posterior

Prior

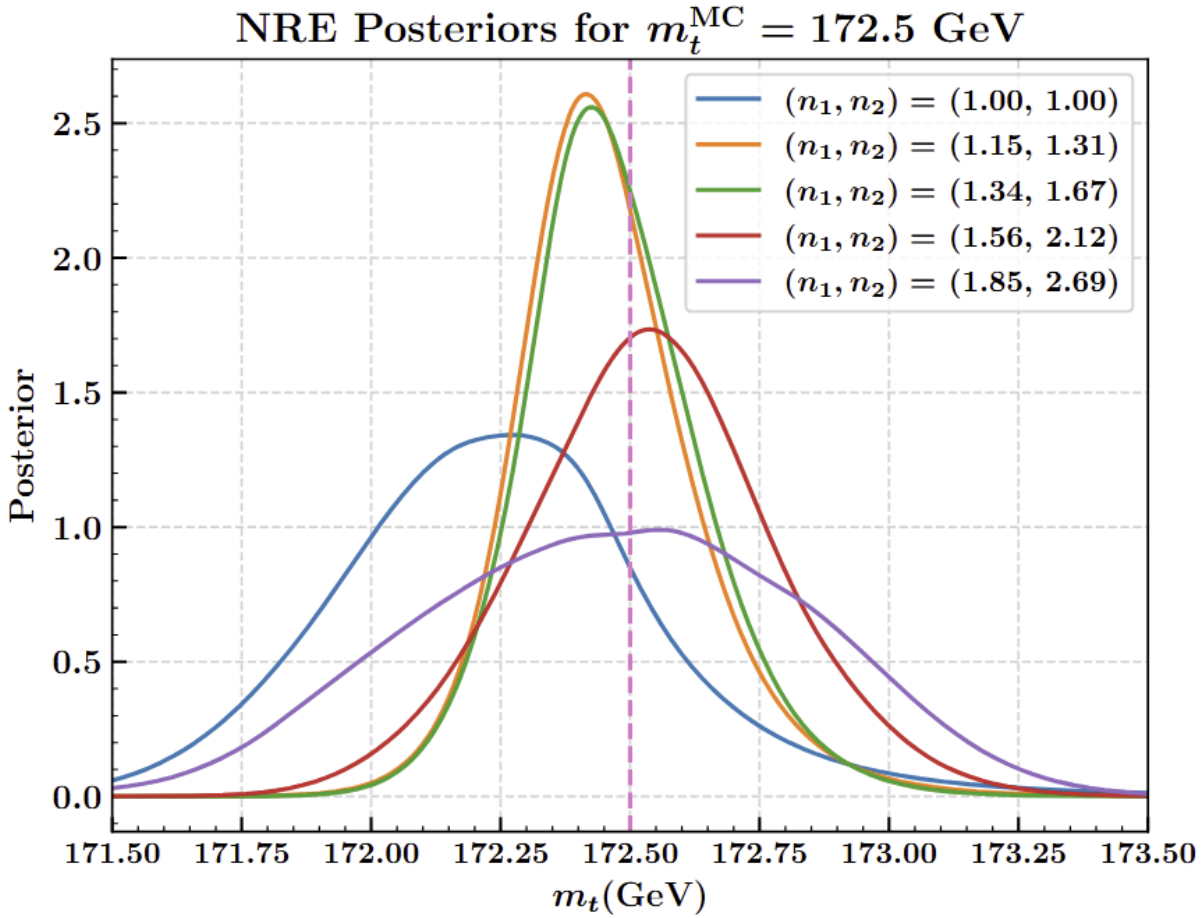
Use a classifier to learn the ratio

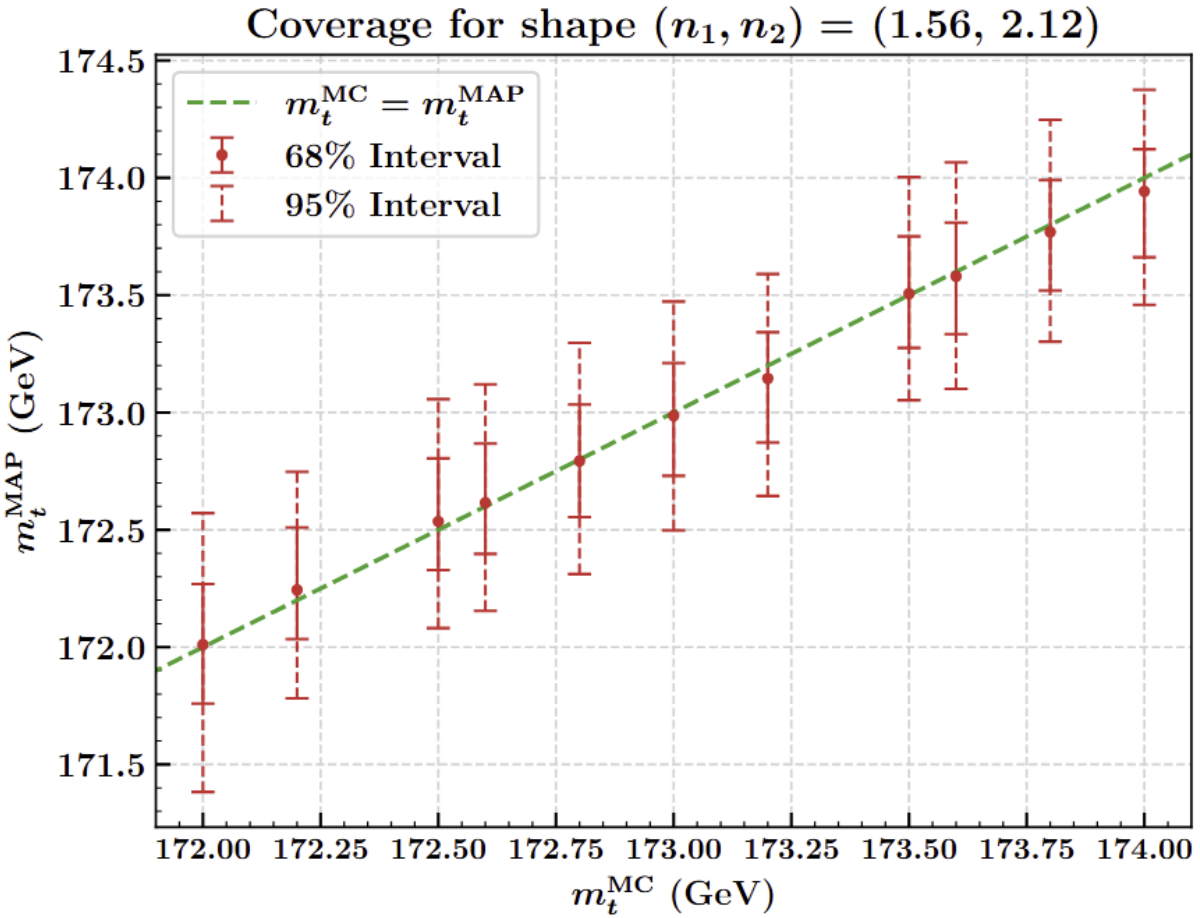
Generalizes peak position of shape.
Use entire information!



Observable Optimization : Neural Ratio Estimation

With posteriors, mode is the parameter estimate, and width is the uncertainty. Can compare shapes!





Observable Optimization : Neural Ratio Estimation

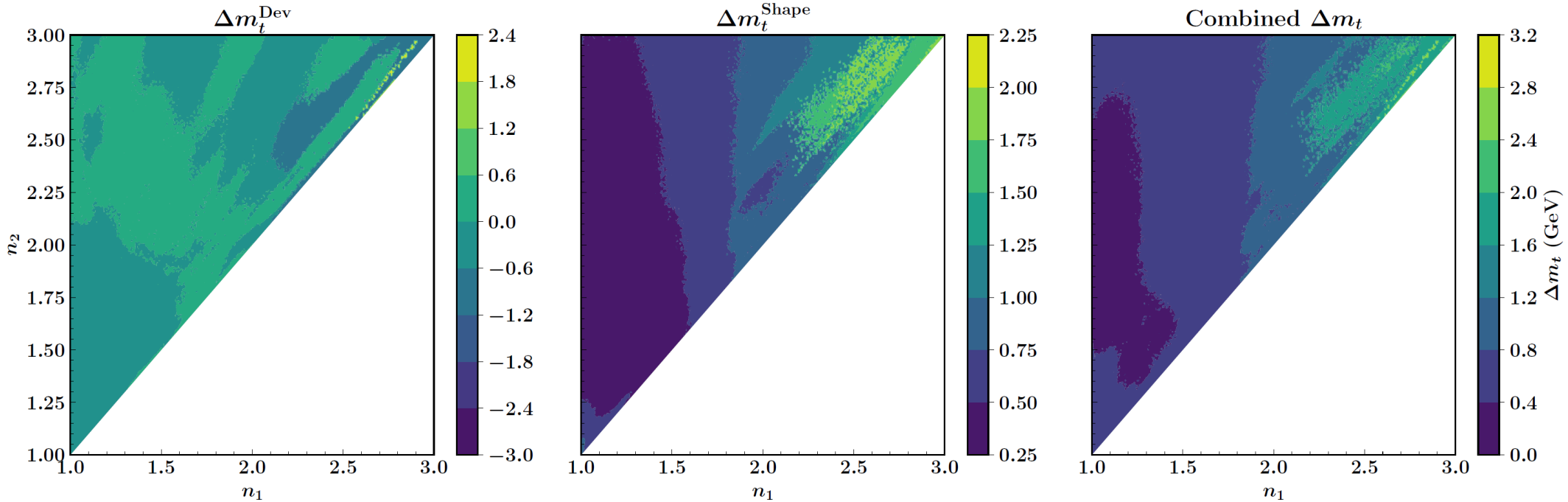
Compute highest posterior density $\Longrightarrow$ Coverage interval

$$\mathrm{HPD}_k = \{m_t \mid p(m_t|\vec{\zeta}_{\mathrm{Shape}}) \geq k\}$$

$$\Theta_\alpha = \mathrm{HPD}_k \text{ s.t. } \int_{\mathrm{HPD}_k} dm_t \; p(m_t|\vec{\zeta}_{\mathrm{Shape}}) = \alpha$$

$$\Delta m_t^{\mathrm{Dev}} = |m_t^{\mathrm{MAP}} - m_t^{\mathrm{MC}}| \qquad \Delta m_t^{\mathrm{Shape}} = \frac{\max \Theta_{\alpha=0.95} - \min \Theta_{\alpha=0.95}}{2} \; .$$

Grid search $(n_1, n_2) \in [1,3] \times [1,3]$ with $\delta_a = 0.01$



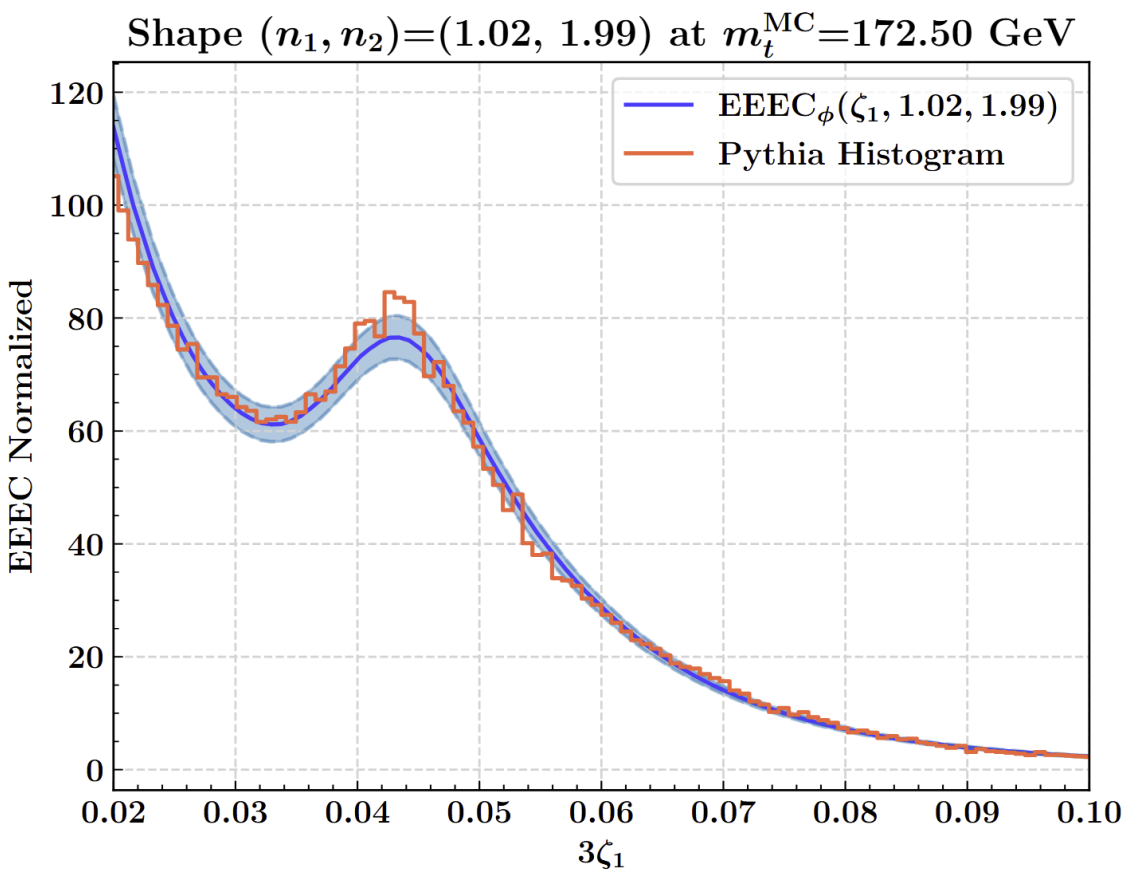
Observable Optimization : Neural Ratio Estimation

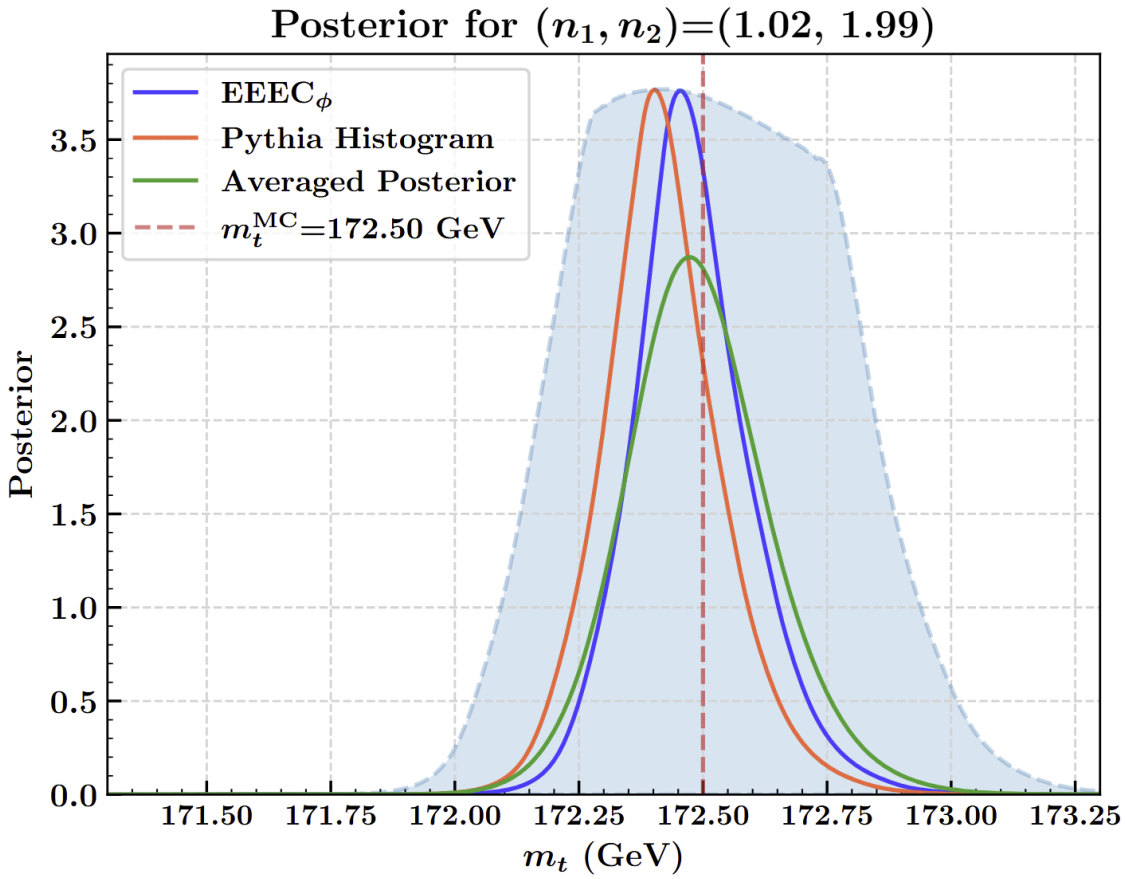
Grid search $(n_1, n_2) \in [1,3] \times [1,3]$ with $\delta_a = 0.01$

Optimal shape

$$(n_1^\star, n_2^\star)_{\delta_a=0.01} = (1.02, 1.99)$$

$$m_t^{\text{Inferred}} = 172.47_{\text{MAP}} \pm 0.31_{\Delta m_t^{\text{Shape}}} \text{ GeV}$$





Observable Optimization : Neural Ratio Estimation

Grid search $(n_1, n_2) \in [1,3] \times [1,3]$ with $\delta_a = 0.01$

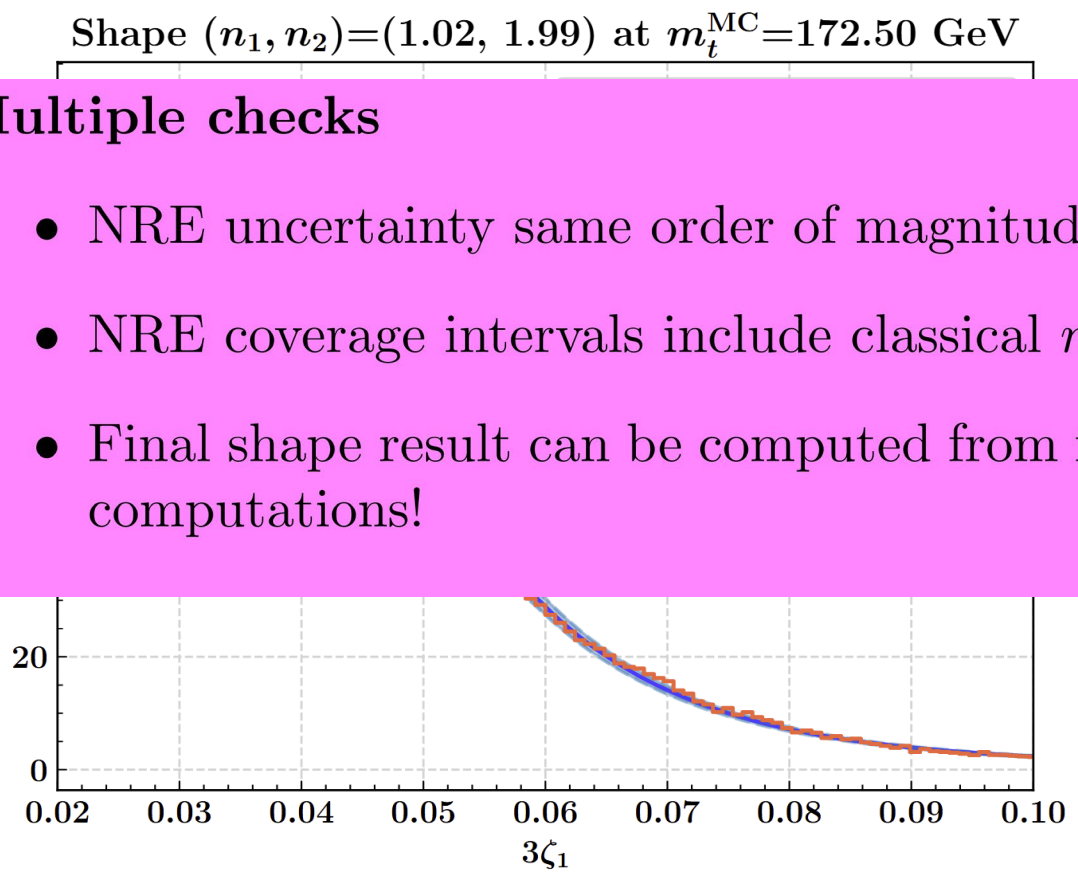
Optimal shape

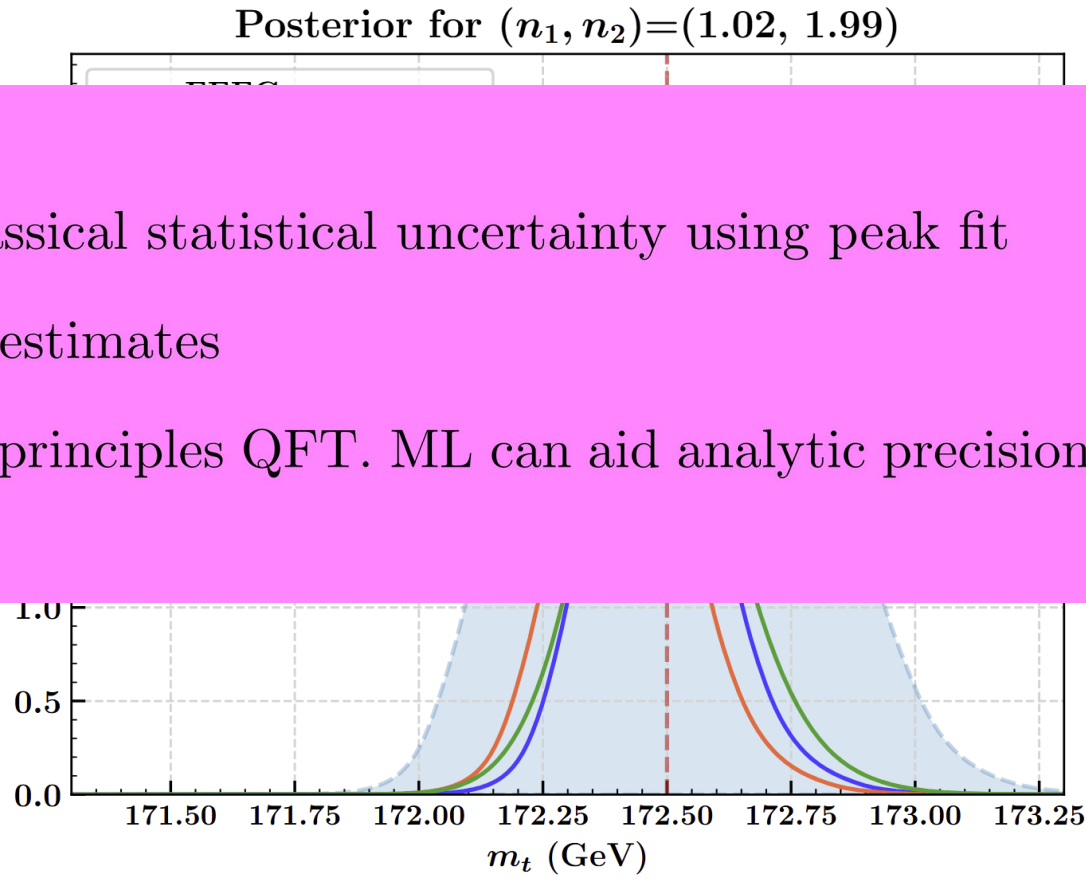
$$(n_1^\star, n_2^\star)_{\delta_a=0.01} = (1.02, 1.99)$$

$$m_t^{\text{Inferred}} = 172.47_{\text{MAP}} \pm 0.31_{\Delta m_t^{\text{Shape}}} \text{ GeV}$$

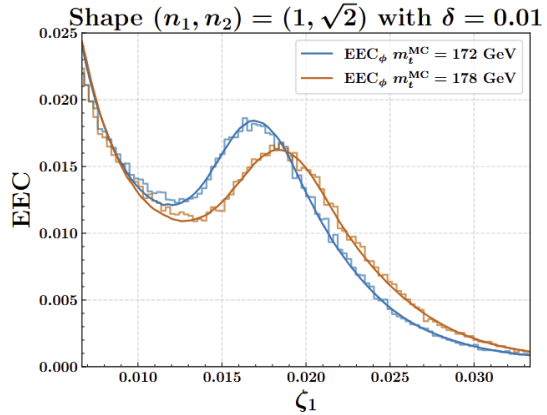
Multiple checks

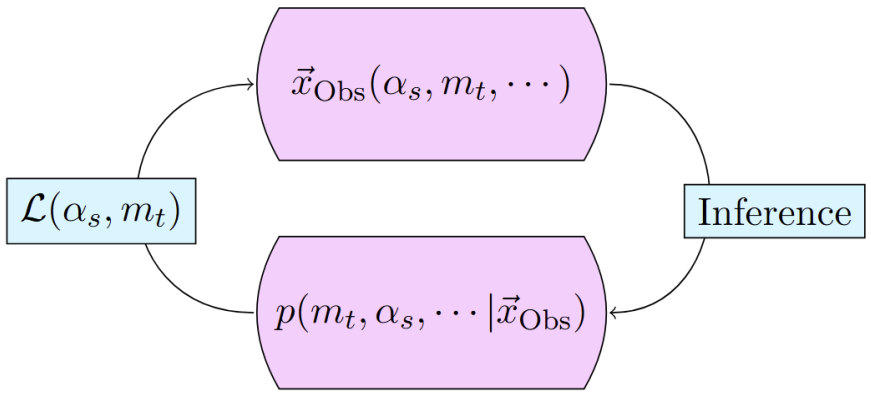
- NRE uncertainty same order of magnitude classical statistical uncertainty using peak fit
- NRE coverage intervals include classical m_t^{Fit} estimates
- Final shape result can be computed from first principles QFT. ML can aid analytic precision computations!

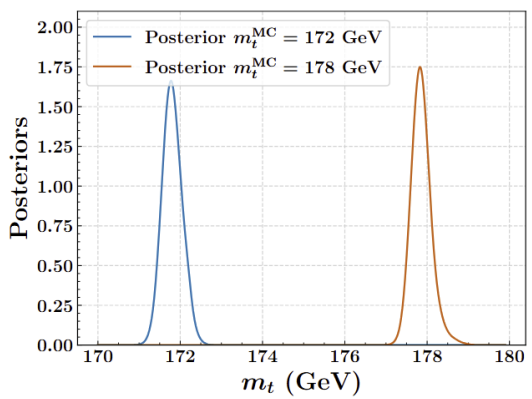


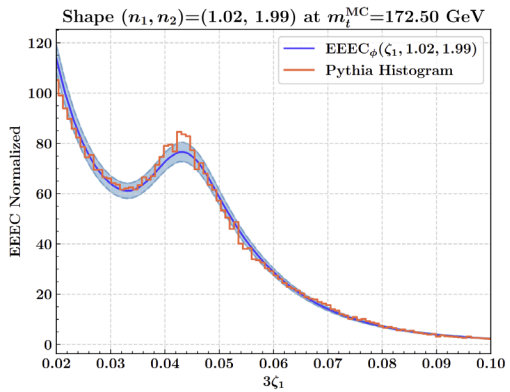


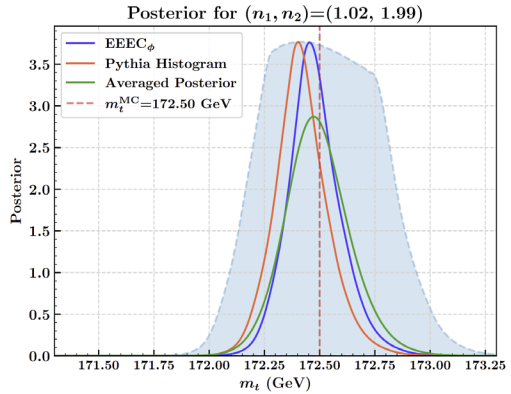
Conclusions











- Neural inference methods from ML can amortize precision computations for collider physics
- Several applications possible
  - Amortization of pipelines for parameter fitting in HEP
  - Develop ML based priors for computationally costly fixed order observable computation
  - Devise largely hadronization-insensitive observables
- Aid develop uncertainty-aware ML tool for precision HEP

Thanks for Listening! Questions?

BACKUP

Training Details for Density Estimation DNN

Learning Rate	8×10^{-5}
Optimizer	AdamW
Weight Decay	10^{-8}
Number of Epochs	100
Early Stop Patience	20
Learning Rate Drop Factor	0.5
Learning Rate Drop Epochs	10

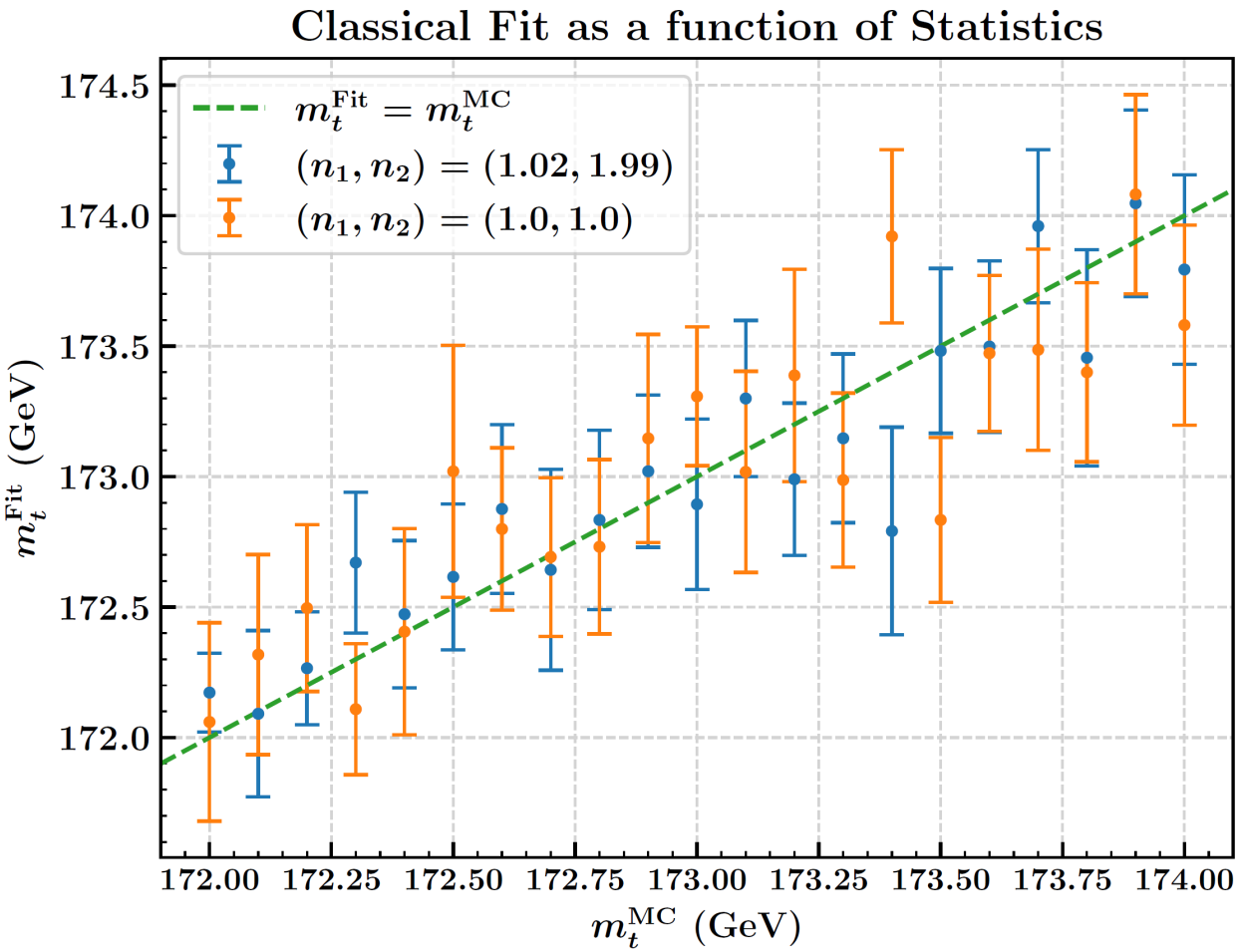
Table 1: Architecture and Training Details for the MLP used to learn the EEEC density from simulated PYTHIA data.

7 layer deep multi-layer-perceptron (MLP) network with 256 nodes each and ReLU activation, with the final layer outputting $\ln \mathrm{EEEC}_{\phi}$. The training set consisted of triplets from 10000 jets at each mass, using a 90:10 train-validation split. Masses drawn uniformly between 170 GeV and 180 GeV with step size of 0.1 GeV.

Training Details for Density Estimation Normalizing Flow

5 dim, 10 block deep flow implemented using `nflows`. Each block consists of a rational quadratic spline (RQS) with a batch norm layer, and a random permutation. For the RQS layers, we also use: ReLU activations, 40 bins, 200 hidden features, 2 context features, a tail bound of 14, min bin widths of 10^{-6}, linear tails, and turn off residual blocks. Base dist is taken to be Gaussian in the four dimensions corresponding to energy and ζ, and uniform prior between $[-1, 2]$ in m_t. Network was trained for 700 epochs, where each epoch consists of reading 500,000 tuples randomly sampled from each mass, and each training batch consists of 500,000 tuples randomly mixed between masses. The flow is trained using the Adam optimizer with initial learning rate 10^{-4}, with the learning rate reduced by a factor of 2 after 20 epochs without improvement, and early stopping after 50 epochs without improvement.

Classical Fits using Peaks



$$m_{t\ ,(1.02,1.99)_{0.01}}^{\text{Fit}} = 172.62 \pm 0.28 \text{ GeV}$$

$$\zeta_{\text{peak}} = a\left(\frac{m_t^{\text{MC}}}{Q}\right)^2 + b$$

$$m_t^{\text{Fit}} = Q\sqrt{\frac{\zeta_{\text{peak}} - b_{\text{Fit}}}{a_{\text{Fit}}}}$$

Compute EEEC on half the fraction of the tuples from 1M jets

Fit deg. 15 polynomial and compute ζ_{peak} analytically

Sampled 20 times for each m_t^{MC}

Shape Comparison