Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CLUSTER OF EXCELLENCE
QUANTUM UNIVERSE

FSP CMS
Erforschung von
Universum und Materie

gefördert durch das
Bundesministerium
für Bildung
und Forschung

CMS

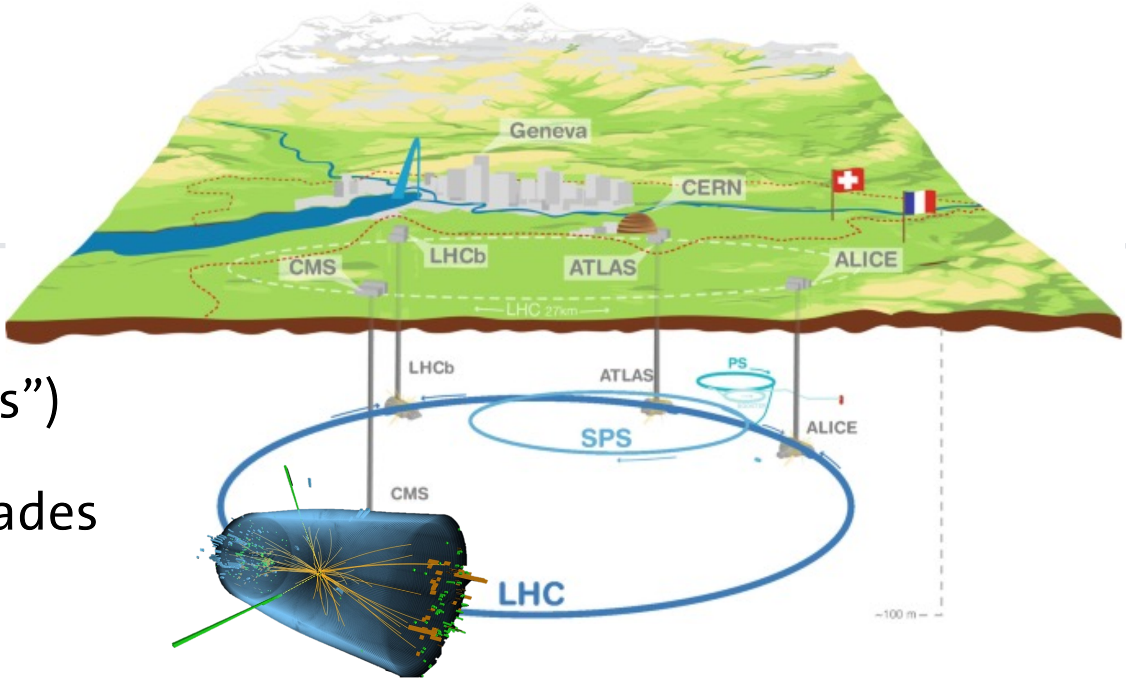
NOVEMBER 29, 2022
15TH ANNUAL MEETING OF THE HELMHOLTZ ALLIANCE "PHYSICS AT THE TERASCALE"

REFINING FAST SIMULATION USING MACHINE LEARNING

SAMUEL BEIN[1], PATRICK CONNOR[1], KEVIN PEDRO[2], PETER SCHLEPER[1], MORITZ WOLF[1]

[1]UNIVERSITÄT HAMBURG, [2]FERMI NATIONAL ACCELERATOR LABORATORY

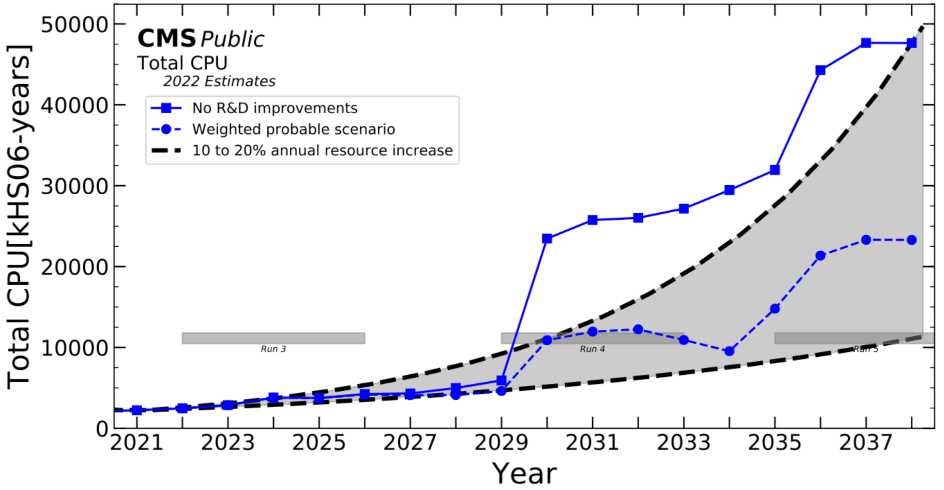
INTRODUCTION



- Most high energy physics analyses rely on a large number of **simulated** proton-proton collisions (= "events")
- Higher **LHC luminosity** (= more events) & detector upgrades (= more complex data) → **fast simulation** techniques and R&D needed to stay within **computing budget**
- In **CMS**, two simulation chains (FullSim/FastSim) are used that produce output of same dimensionality/structure

	GEN → Event generation	**SIM** → Detector simulation	**DIGI** → Digitization	**RECO** Reconstruction
FullSim	same e.g., MadGraph	GEANT4	same	analyze as if data
FastSim ≈ 15% of sim. events		parametrized energy loss x100 faster		use GEN info x2.5 faster

➢ FullSim: ≈ 100 s/event, FastSim: ≈ 10 s/event



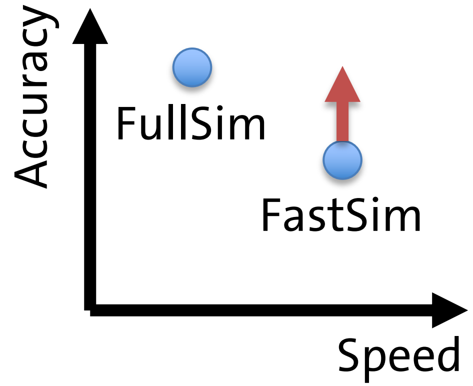
twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults

GEN →	SIM →	DIGI →	RECO →	AOD →	NanoAOD
Event generation	Detector simulation	Digitization	Reconstruction	“Analysis Object Data”	further processed analysis format

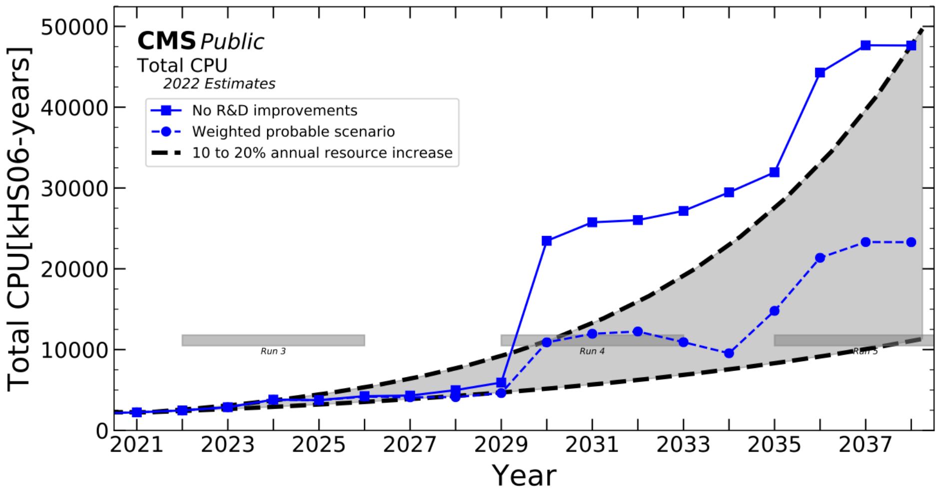
INTRODUCTION

- FastSim advantage in speed comes at the price of **decreased accuracy** in some of the final analysis observables

➢ **Aim**: increase FastSim accuracy to promote its wider usage



- Possible FastSim **tuning** approaches:
  - Internal tuning of functions/parameters (within **SIM/RECO**)
    e.g., denoising showers using ML github.com/cms-denoising/SimDenoising
  - Post-hoc tuning (after **NanoAOD**)
    - **Reweighting** = defining weights for individual events/jets/…
      e.g., DCTR introduced in arXiv:1907.08209
    - **Refining** = changing (high-level) observables
      e.g., Wasserstein-GAN for air showers in arXiv:1802.03325

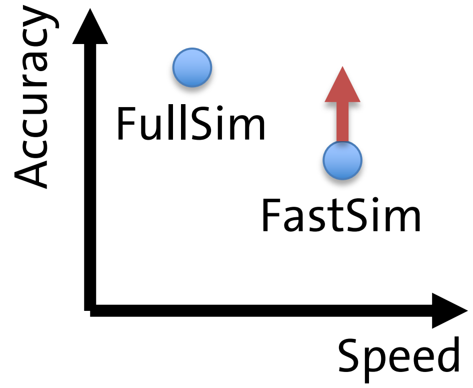


twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults

INTRODUCTION

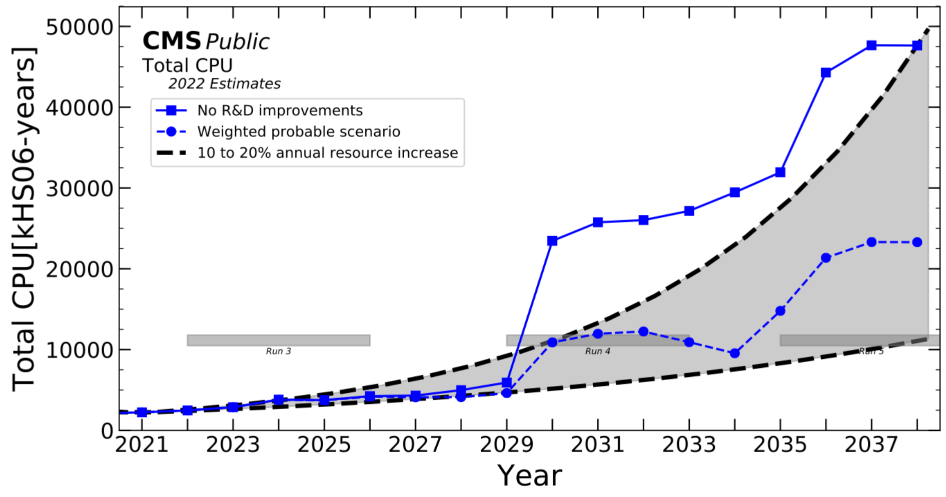
GEN →	SIM →	DIGI →	RECO →	AOD →	NanoAOD
Event generation	Detector simulation	Digitization	Reconstruction	“Analysis Object Data”	further processed analysis format

- FastSim advantage in speed comes at the price of **decreased accuracy** in some of the final analysis observables



➢ **Aim**: increase FastSim accuracy to promote its wider usage

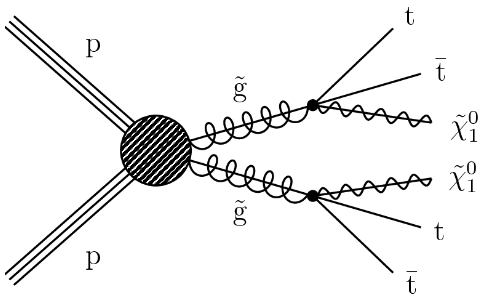
- Possible FastSim **tuning** approaches:
  - Internal tuning of functions/parameters (within **SIM/RECO**)
    e.g., denoising showers using ML github.com/cms-denoising/SimDenoising
  - Post-hoc tuning (after **NanoAOD**)
    - **Reweighting** = defining weights for individual events/jets/…
      e.g., DCTR introduced in arXiv:1907.08209
    - **Refining** = changing (high-level) observables
      e.g., Wasserstein-GAN for air showers in arXiv:1802.03325

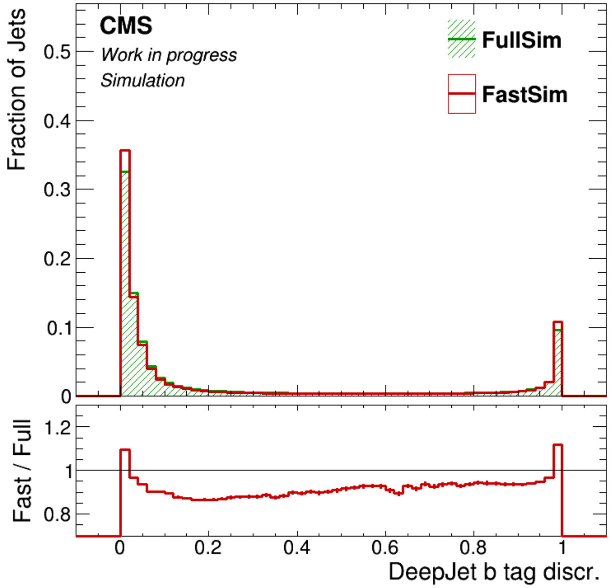


twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults

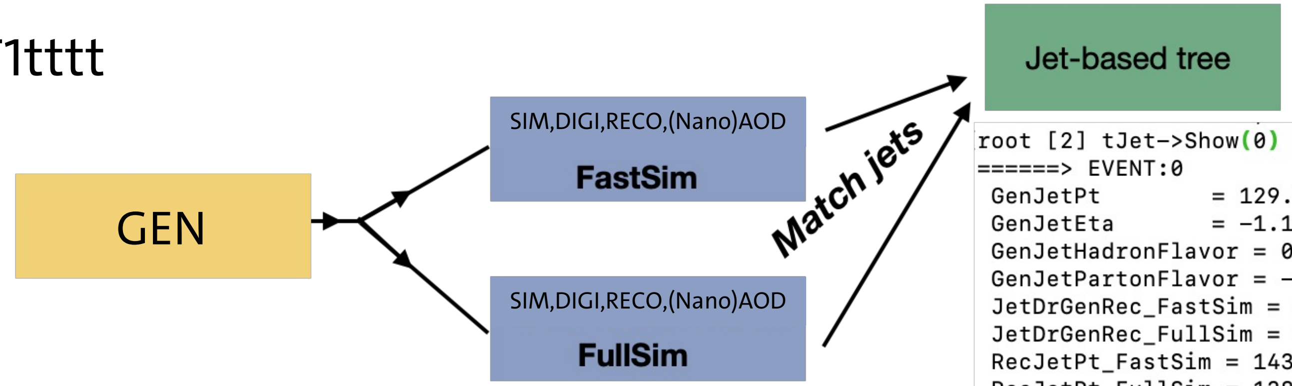
INTRODUCTION

- Focus on refinement of **jet observables** e.g.,
  - Magnitude and direction of momentum:
    $(p_x, p_y, p_z) \rightarrow (p_T, \eta, \varphi)$
  - Jet (sub)structure e.g.,
    - Flavour tagging, here: **DeepJet** algorithm (DNN, arXiv:2008.10519)
    - ➢ 4 output nodes: b, c, uds, g (softmax activated)
    - ➢ 4 discriminators in NanoAOD: b, c vs. b, c vs. uds+g („CvL“), uds vs. g („QG“)





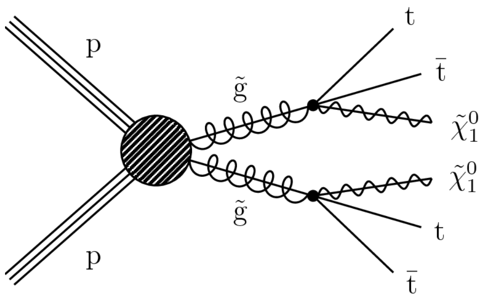
- **Training sample**: SUSY simplified model T1tttt simulated with FastSim and FullSim (same GEN events, 0 PU)
  - ➢ Match jets using ΔR angular criterion
  - ➢ Jet **triplets**: (GEN, FastSim, FullSim)

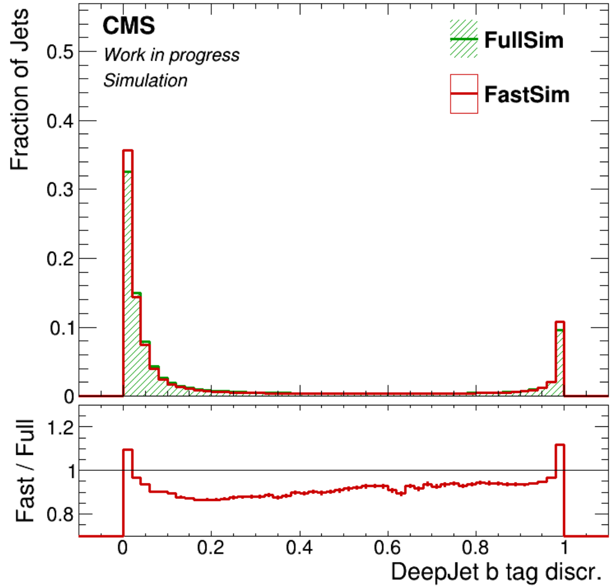



```
root [2] tJet->Show(0)
======> EVENT:0
 GenJetPt          = 129.967
 GenJetEta         = -1.19493
 GenJetHadronFlavor = 0
 GenJetPartonFlavor = -2
 JetDrGenRec_FastSim = 0.021565
 JetDrGenRec_FullSim = 0.0141163
 RecJetPt_FastSim = 143.902
 RecJetPt_FullSim = 138.24
 RecJetEta_FastSim = -1.177
 RecJetEta_FullSim = -1.1809
```

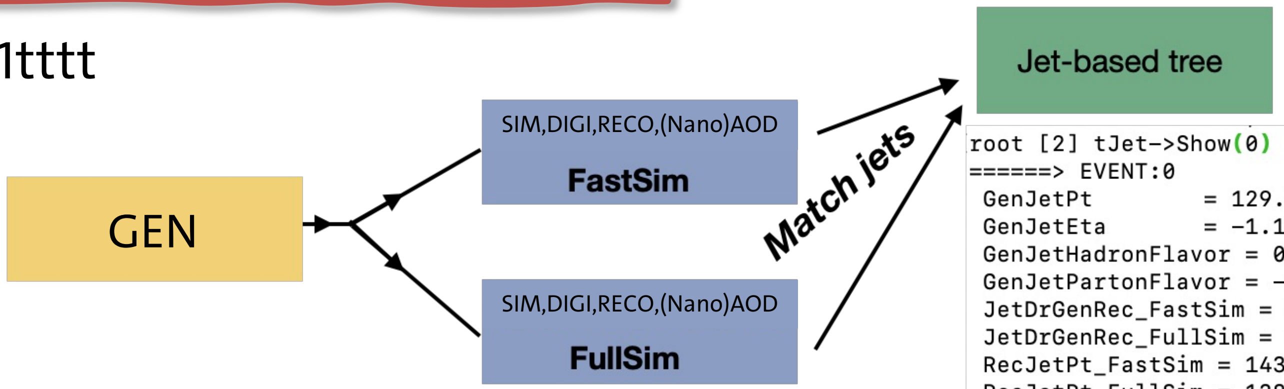
INTRODUCTION

- Focus on refinement of **jet observables** e.g.,
  - Magnitude and direction of momentum: $(p_x, p_y, p_z) \rightarrow (p_T, \eta, \varphi)$
  - Jet (sub)structure e.g.,
    - Flavour tagging, here: **DeepJet** algorithm (DNN, arXiv:2008.10519)
  - ➢ 4 output nodes: b, c, uds, g (softmax activated)
  - ➢ 4 discriminators in NanoAOD: b, c vs. b, c vs. uds+g („CvL“), uds vs. g („QG“)



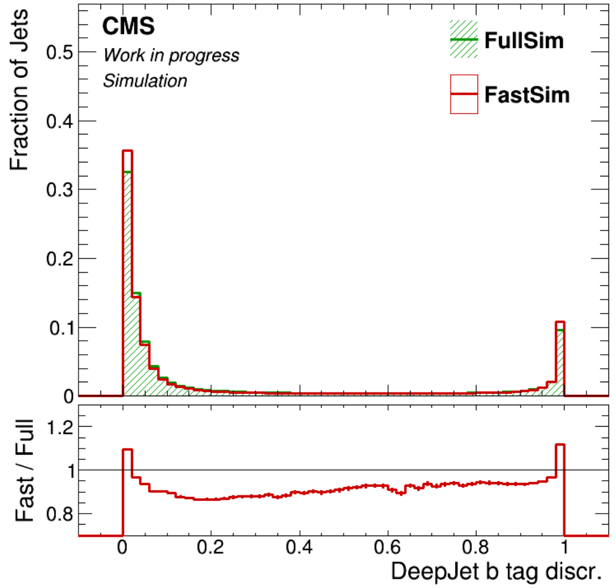


- **Training sample**: SUSY simplified model T1tttt simulated with FastSim and FullSim (same GEN events, 0 PU)
  - ➢ Match jets using ΔR angular criterion
  - ➢ Jet **triplets**: (GEN, FastSim, FullSim)




```
root [2] tJet->Show(0)
======> EVENT:0
 GenJetPt          = 129.967
 GenJetEta         = -1.19493
 GenJetHadronFlavor = 0
 GenJetPartonFlavor = -2
 JetDrGenRec_FastSim = 0.021565
 JetDrGenRec_FullSim = 0.0141163
 RecJetPt_FastSim = 143.902
 RecJetPt_FullSim = 138.24
 RecJetEta_FastSim = -1.177
 RecJetEta_FullSim = -1.1809
```

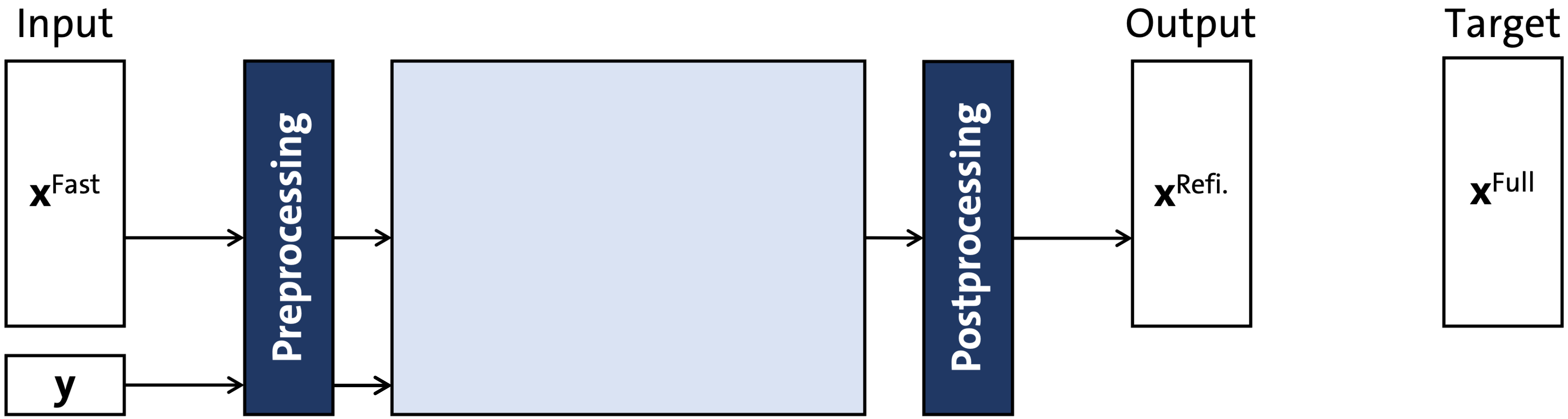
REFINING (REGRESSION) – GENERAL IDEA



Employ **regression neural network** to refine FastSim:

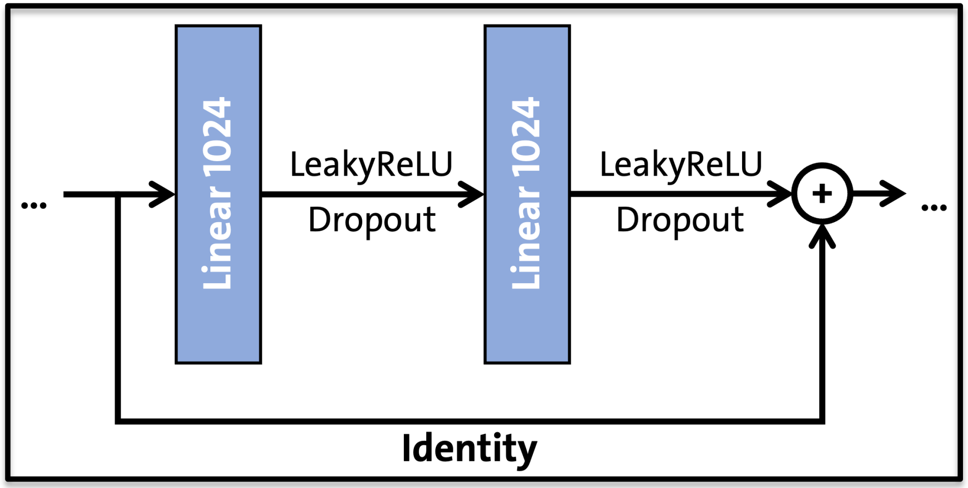
- Input: **FastSim** variables $\mathbf{x}^{\mathrm{Fast}}$ = 4 DeepJet discriminators

  Parameters $\mathbf{y}$ = p_T^{GEN}, η^{GEN}, true hadron flavor (b, c, or light quark/gluon)
- Output: **Refined** variables $\mathbf{x}^{\mathrm{Refined}}$ = 4 DeepJet discriminators
- Target: **FullSim** variables $\mathbf{x}^{\mathrm{Full}}$ = 4 DeepJet discriminators

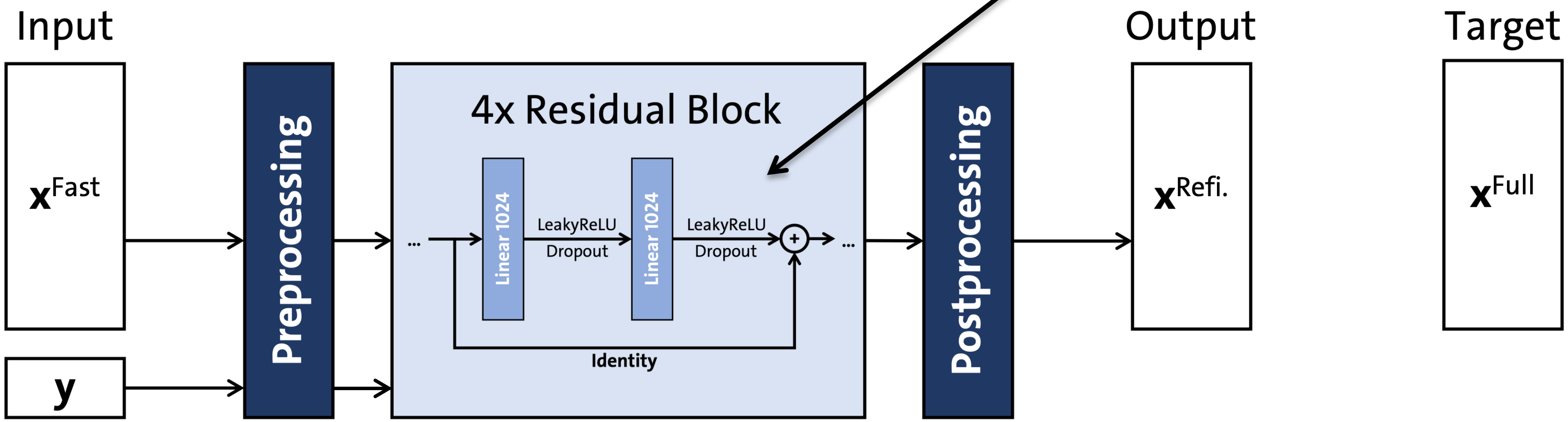


REFINING (REGRESSION) – NN ARCHITECTURE

ResNet paper arXiv:1512.03385



- ResNet-like **skip connections** → learn only residual corrections
- **Preprocessing**: transform input variables/parameters (logit-transform in order to center around 0, transform to original DeepJet output values, one-hot-encode true hadron flavour)
- **Postprocessing**: DeepJet softmax constraint + backtransf.



MAE: mean absolute error (jet-jet pairs)
MSE: mean squared error (jet-jet pairs)
MMD: maximum mean discrepancy (ensembles)

REFINING (REGRESSION) – LOSS TERMS

- **MSE**: output-target pair-based
  - Correct for deterministic FastSim biases

use MSE/MAE combination („Huber loss"), less sensitive to outliers

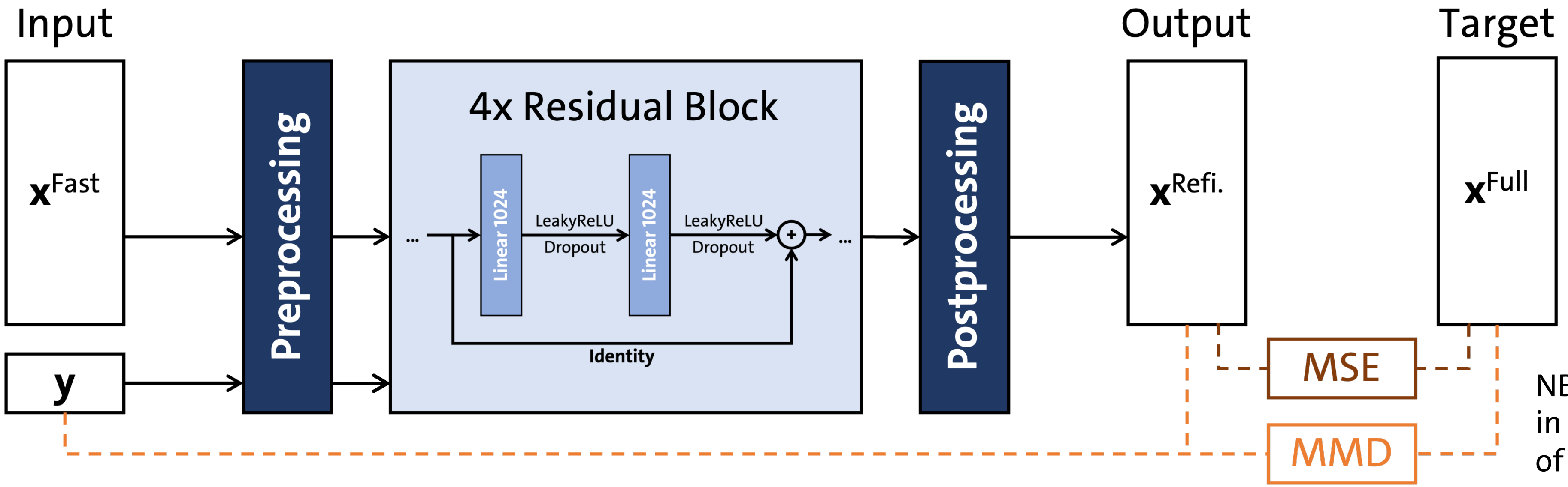
$$l_n = \begin{cases} 0.5(x_n - y_n)^2, & \text{if } |x_n - y_n| < delta \\ delta * (|x_n - y_n| - 0.5 * delta), & \text{otherwise} \end{cases}$$

- **MMD**: distribution-based (**primary loss**)
  - Correct for stochastic FastSim biases

given two samples from P(X) and Q(Y):

$$\widehat{\mathrm{MMD}}(P,Q) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i,x_j) + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} k(y_i,y_j) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k(x_i,y_j)$$

n = m = batch size = 4096,
k: Gaussian kernel (adaptive σ)

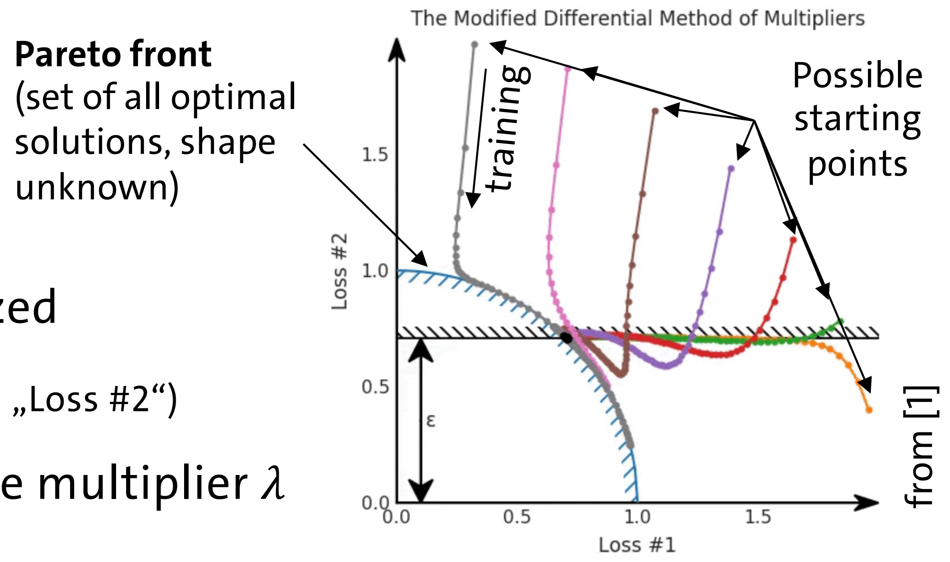


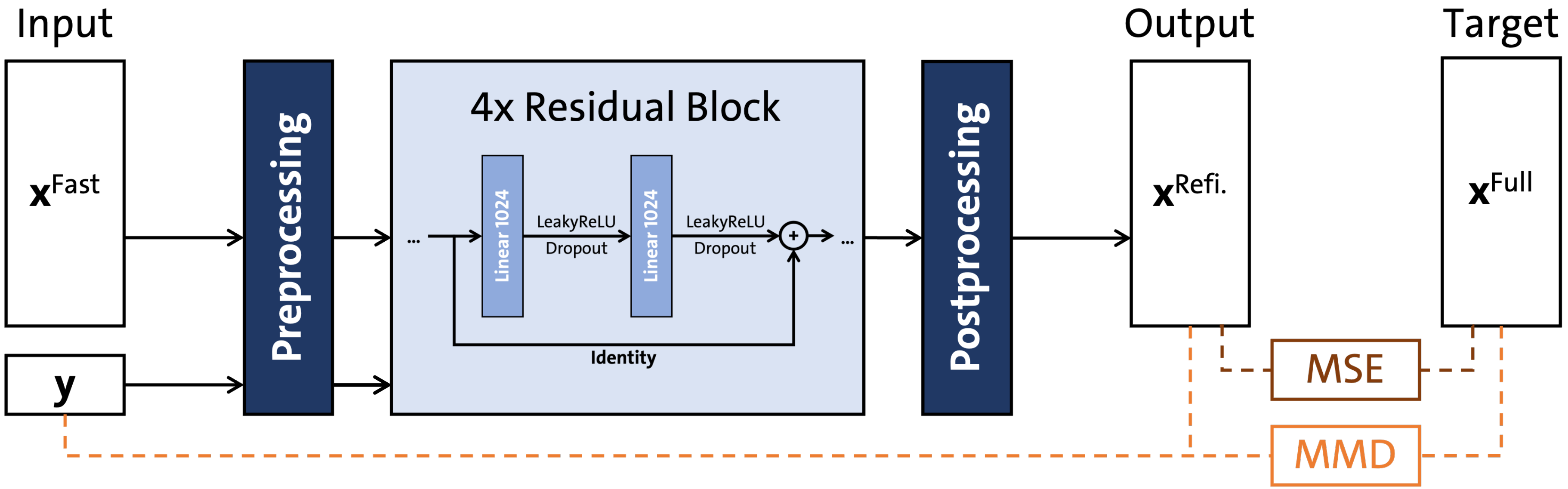
NB: include parameters in MMD to take account of correlations

REFINING (REGRESSION) – LOSS TERMS

Combine loss terms via **MDMM algorithm**:

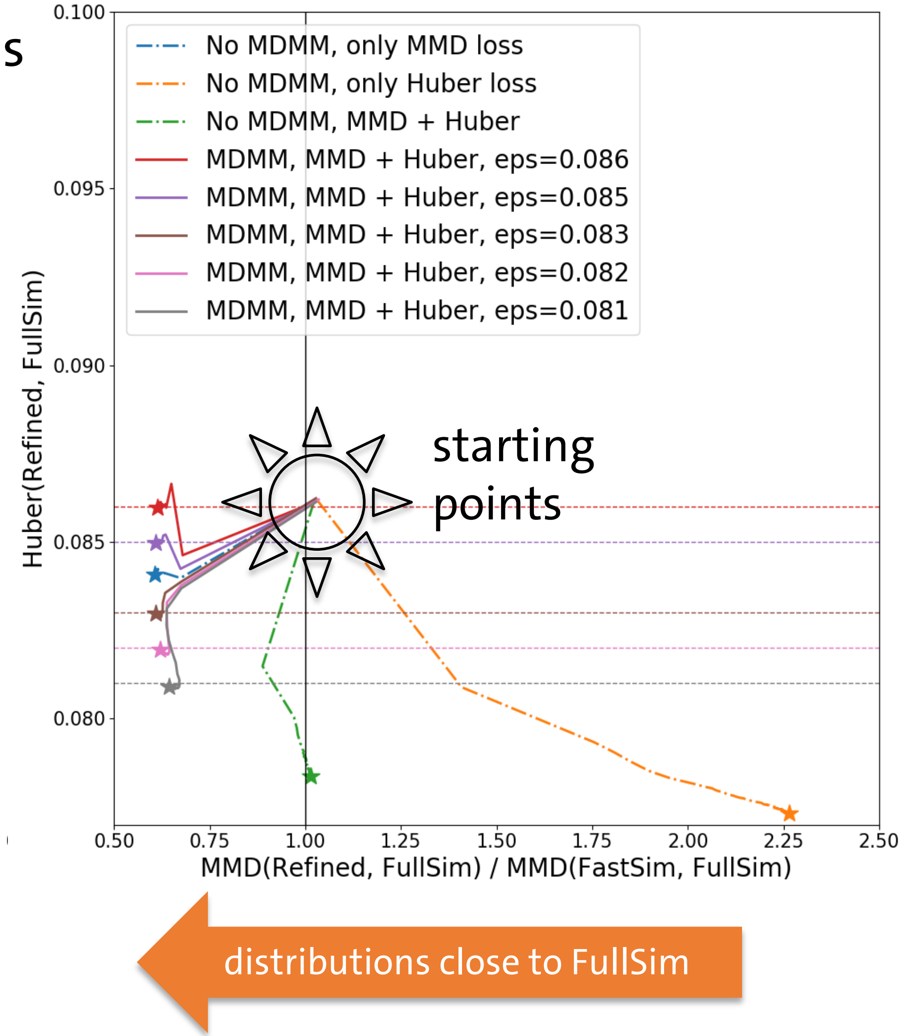
- Reframe problem as **constrained optimization** using **Lagrangian**:
  $\mathcal{L} = f(\theta) - \lambda * (\varepsilon - g(\theta))$ → convergence mathematically formalized
- Minimize $f(\theta)$ (primary loss, „Loss #1") subject to $g(\theta) = \varepsilon$ (additional loss, „Loss #2")
- Gradient descent for NN parameters θ, gradient ascent for Lagrange multiplier λ





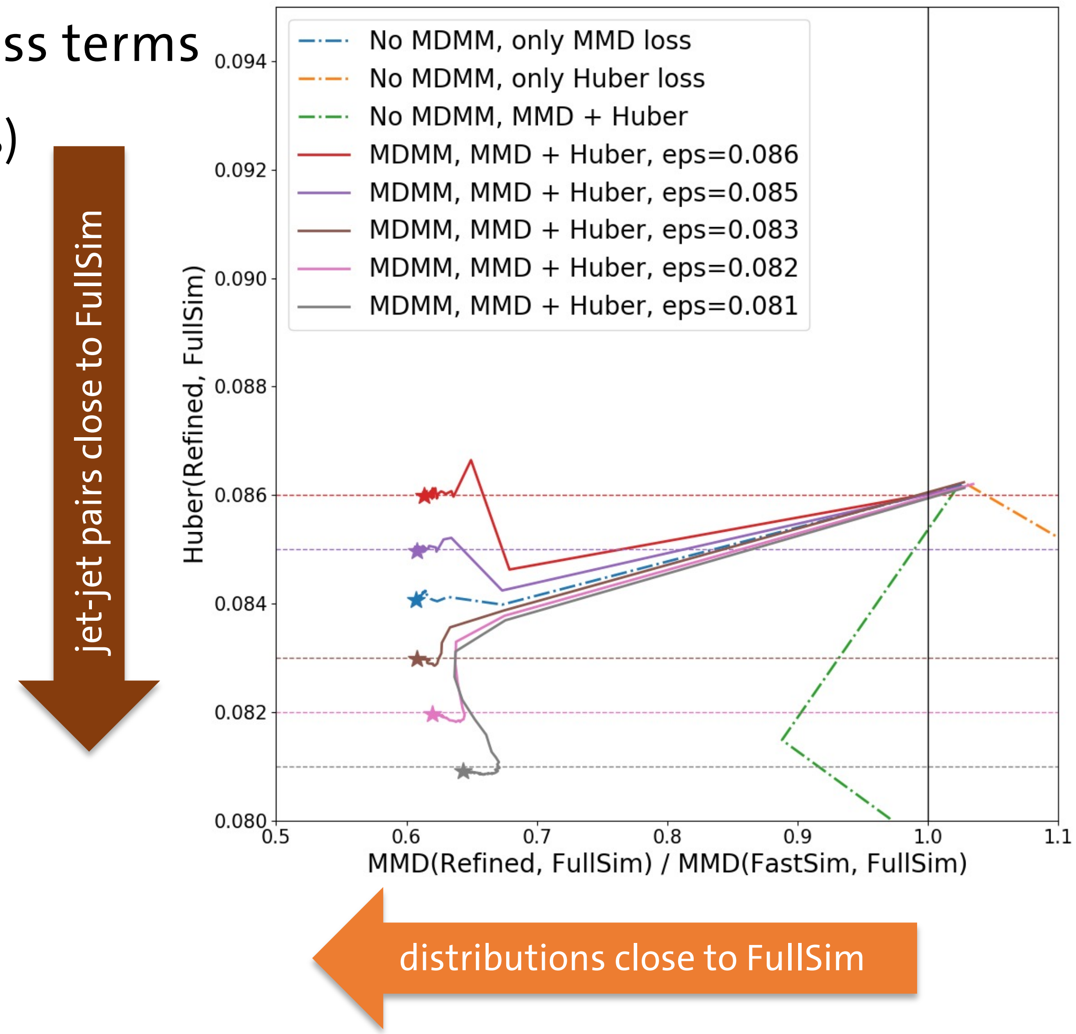
REFINING (REGRESSION) – LOSS TERMS

- **Pareto plot**: training history in plane of the two loss terms
  - Primary loss: **MMD(Refined, FullSim)** (horizontal axis)
    normalized to MMD(FastSim, FullSim)
  - Constraint: **Huber(Refined, FullSim)** (vertical axis)
- Trained for 100 epochs with 500 batches
- Simple addition **not optimal**
- **MDMM**: scan of different ε values
  - ➢ If ε is too small it can only be reached by disregarding MMD
  - ➢ If ε is too large MMD also gets worse because jet-jet pairs *have to* change too much
  - ➢ Choose **ε = 0.083**

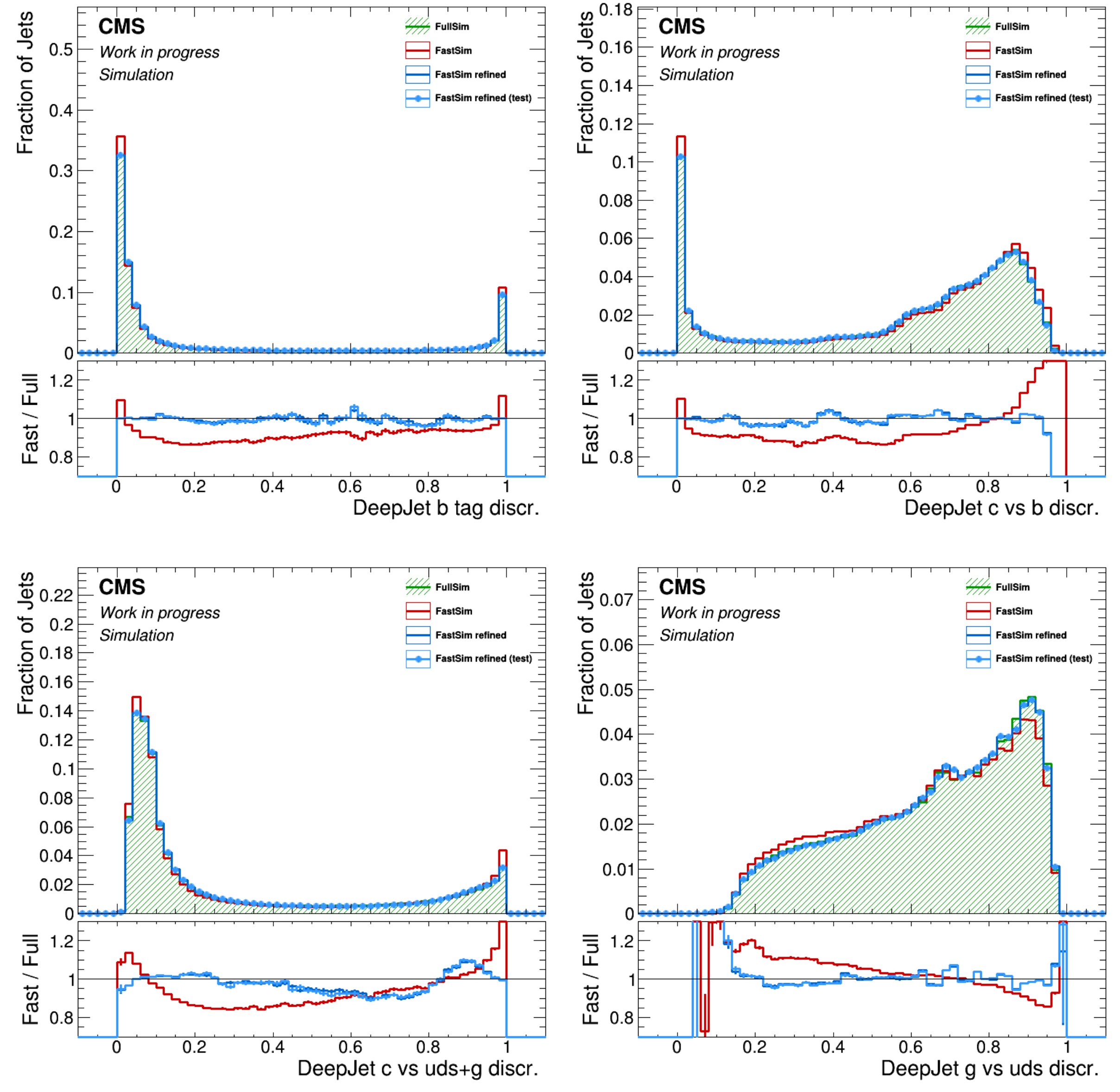


REFINING (REGRESSION) – LOSS TERMS

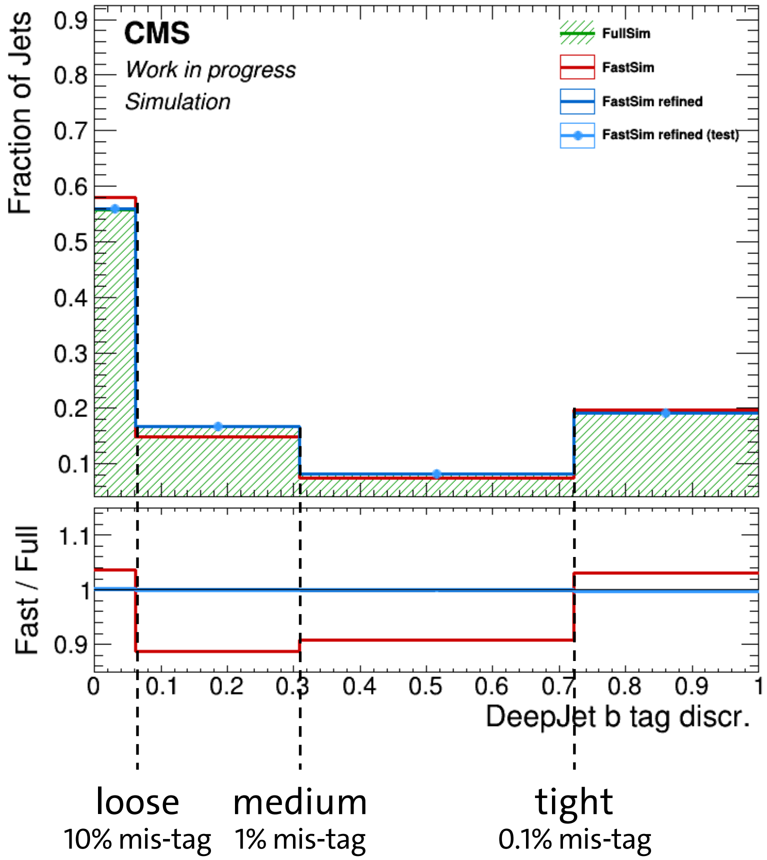
- **Pareto plot**: training history in plane of the two loss terms
  - Primary loss: **MMD(Refined, FullSim)** (horizontal axis) normalized to MMD(FastSim, FullSim)
  - Constraint: **Huber(Refined, FullSim)** (vertical axis)
- Trained for 100 epochs with 500 batches
- Simple addition **not optimal**
- **MDMM**: scan of different ε values
  - ➢ If ε is too small it can only be reached by disregarding MMD
  - ➢ If ε is too large MMD also gets worse because jet-jet pairs *have to* change too much
  - ➢ Choose **ε = 0.083**



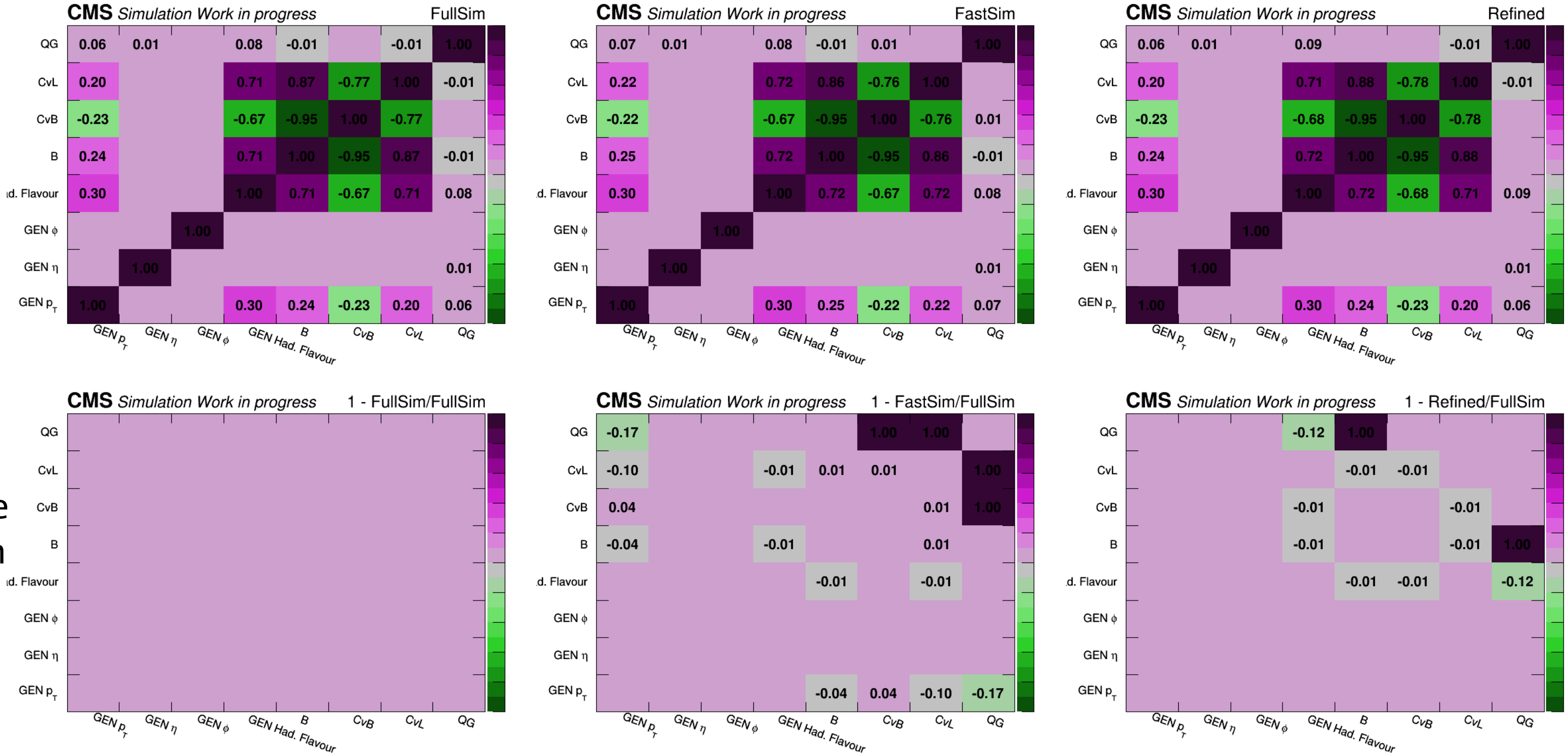
REFINING (REGRESSION) – RESULTS



b tag binned by working point

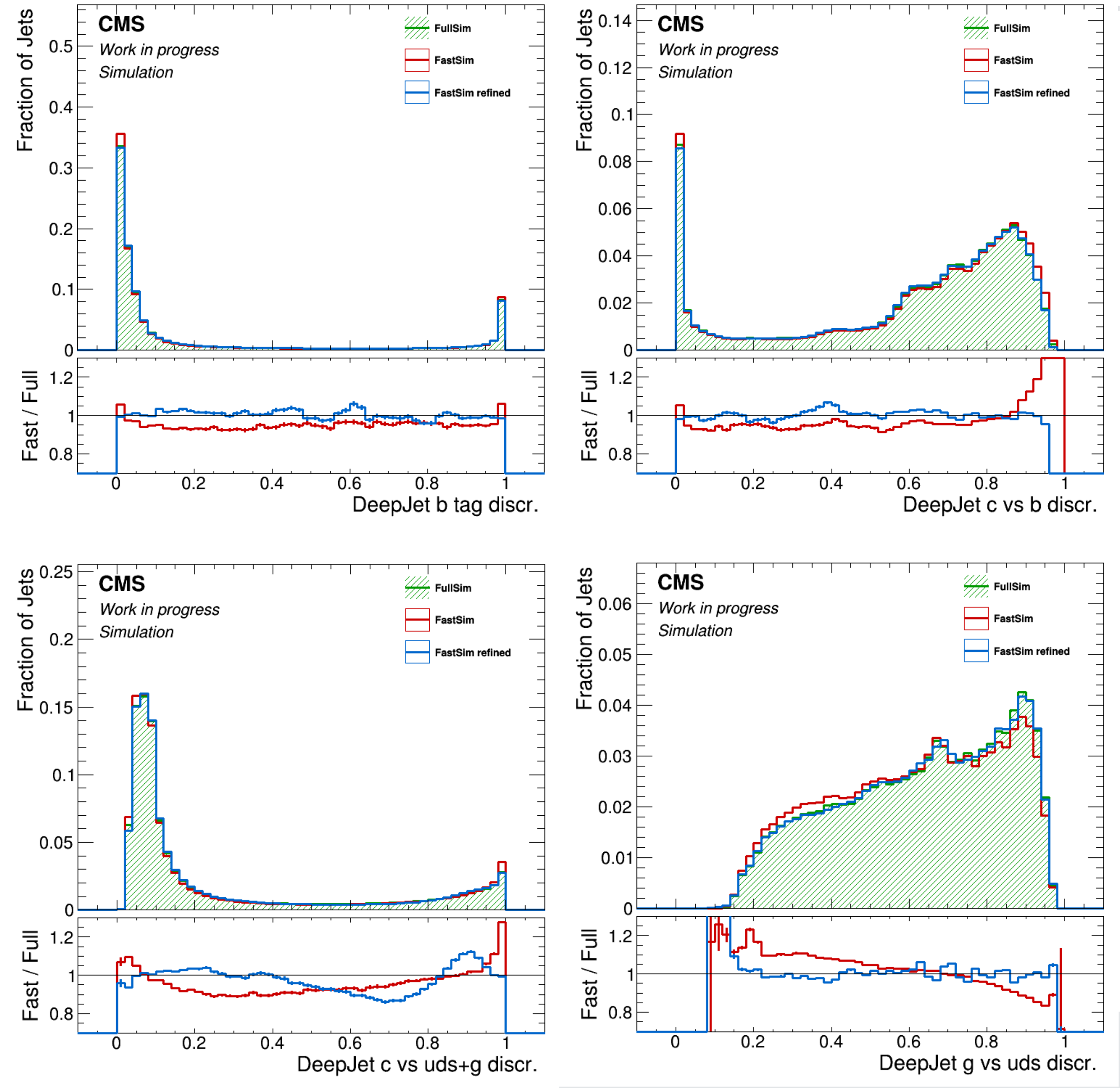


REFINING (REGRESSION) – LINEAR CORRELATION COEFFICIENTS



difference to FullSim

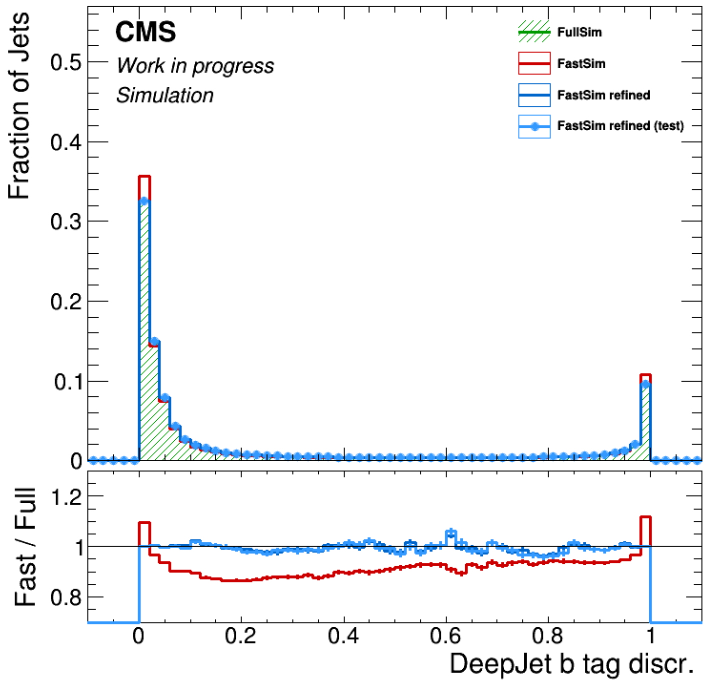
REFINING (REGRESSION) – VALIDATION IN TTBAR



- Good performance for evaluation on **TTbar** (NN trained on T1tttt SUSY dataset)

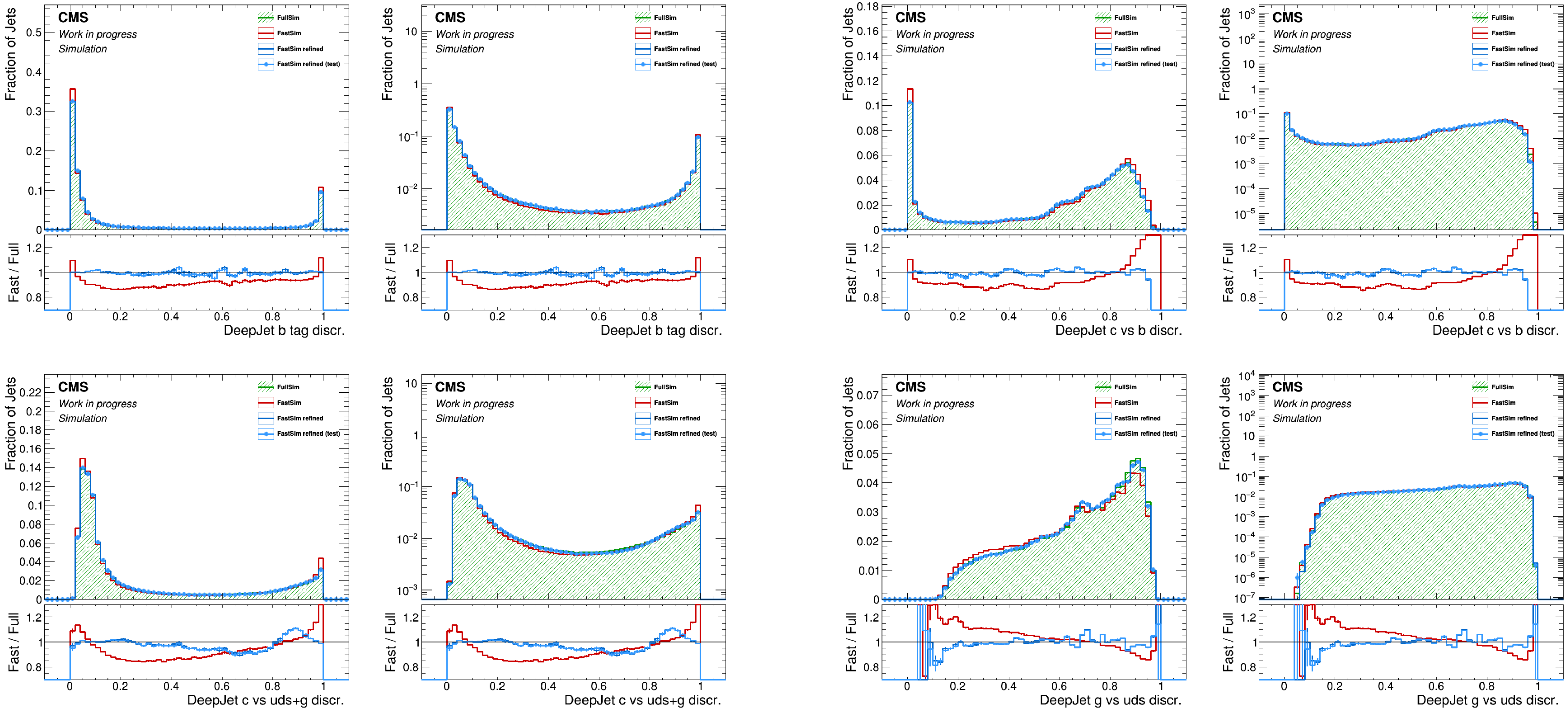
SUMMARY

- **Fast simulation** techniques needed to face computing challenges
- High **accuracy** of simulation required for most physics analyses
- Applying ML-based **post-hoc refinement** to CMS FastSim output (ResNet-like regression NN)
  - ➢ Considerably improved agreement with FullSim output
  - ➢ Improvement in correlations among output observables and external parameters
- Integration into CMS software framework (CMSSW) in the works (via **ONNX** format)
- Possible extension to other variables/physics objects (e.g., FatJet substructure)

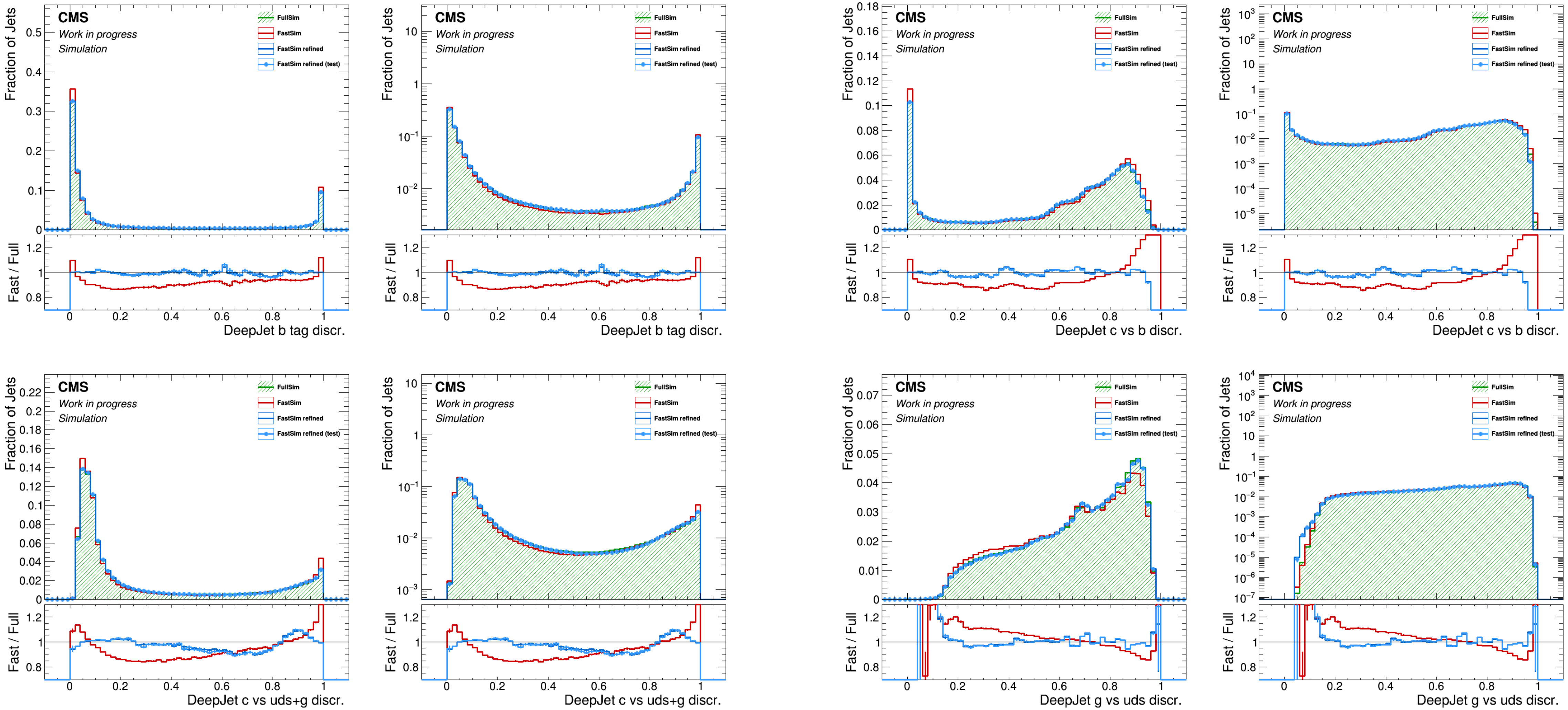


BACKUP

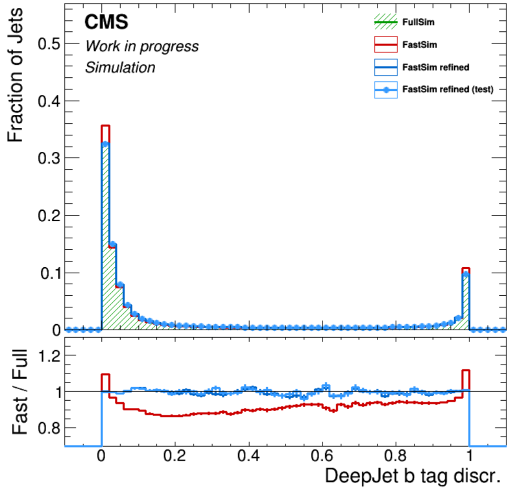
REFINING (REGRESSION) – ONLY MMD

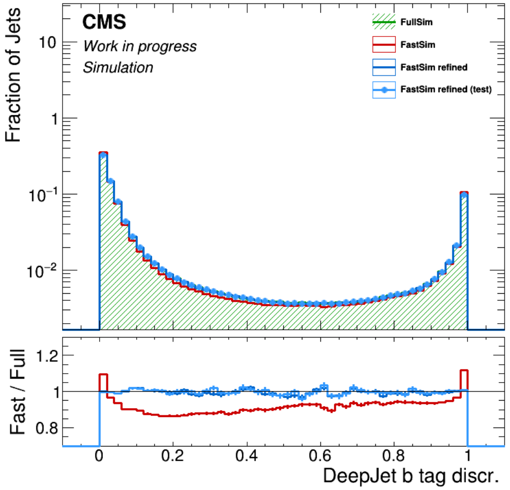


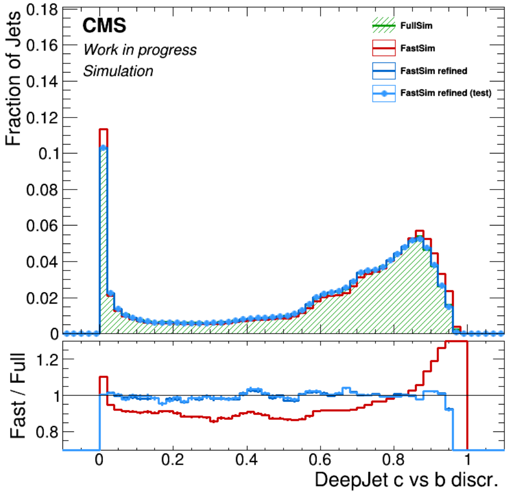
REFINING (REGRESSION) – EPS=0.083

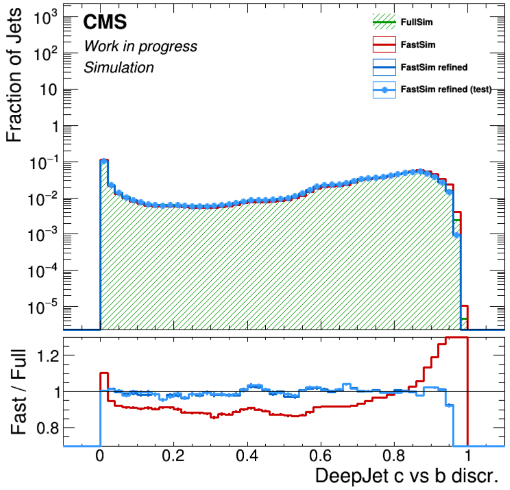


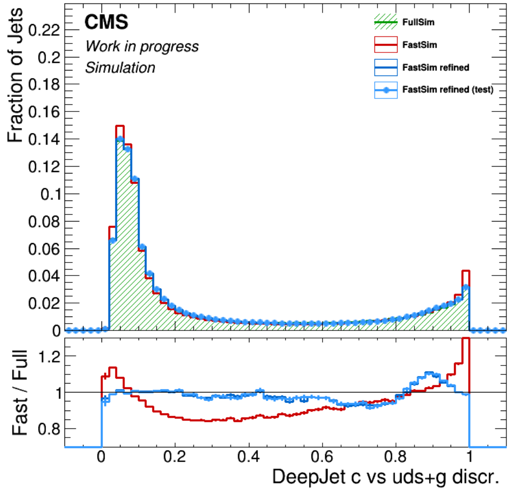
REFINING (REGRESSION) – EPS=0.085

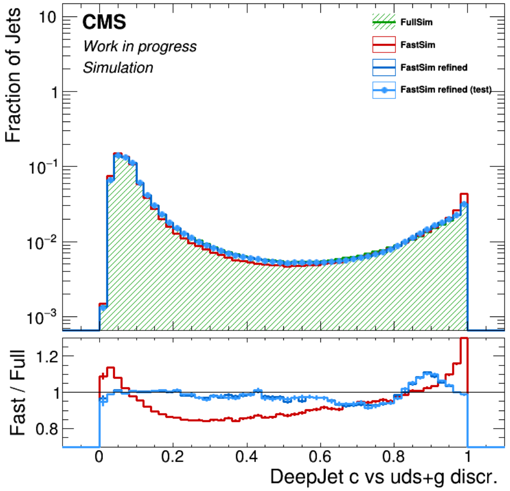


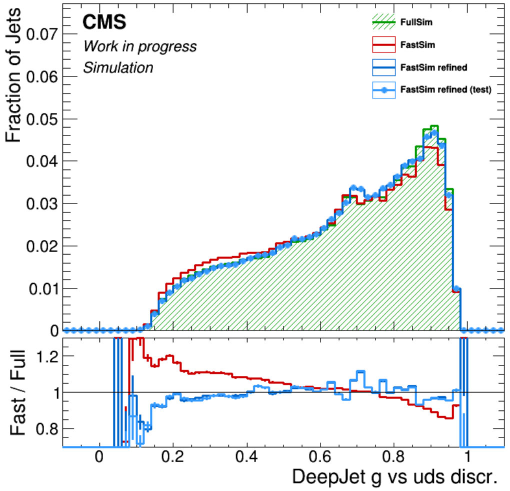


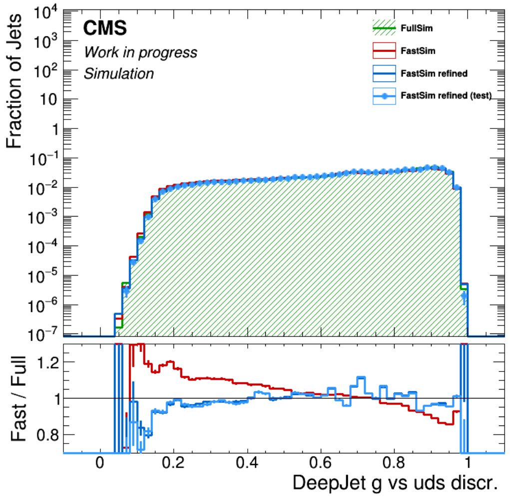




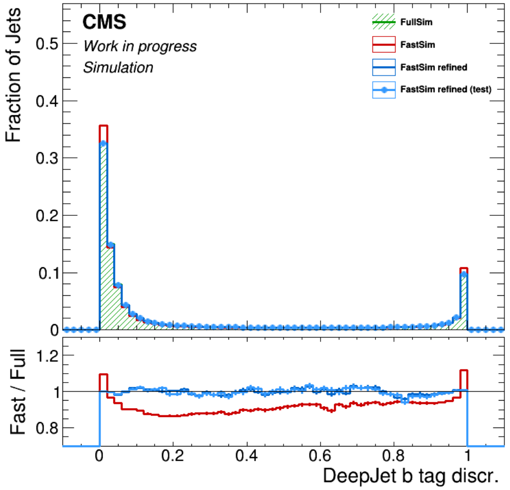


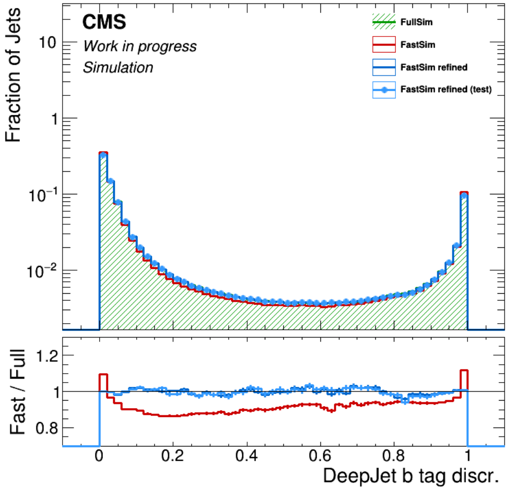


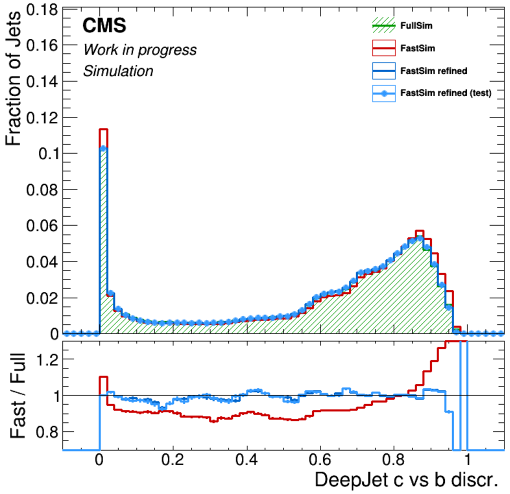


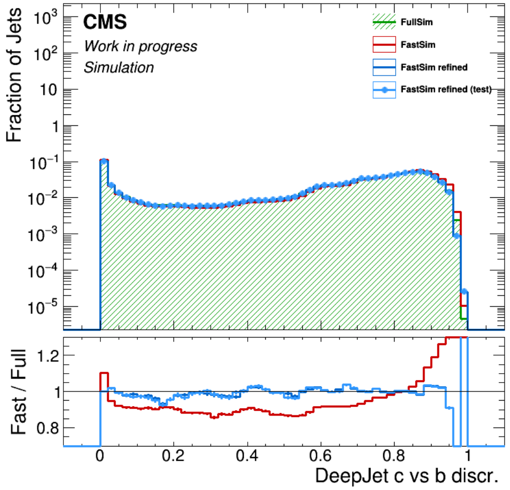


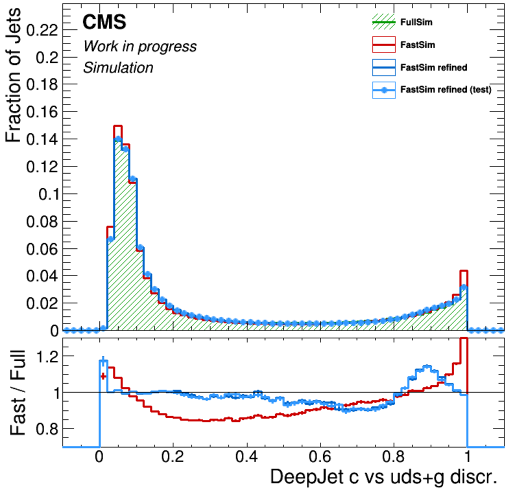
REFINING (REGRESSION) – EPS=0.086

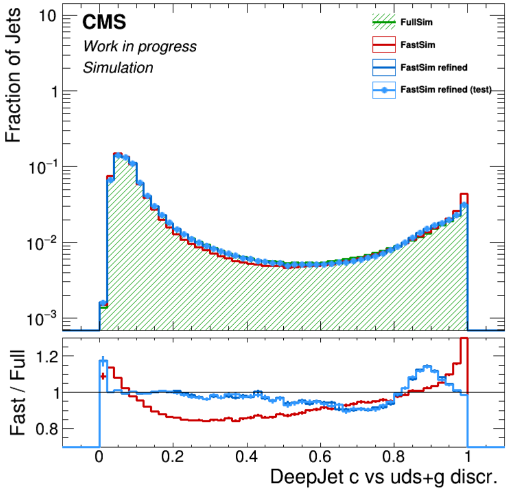


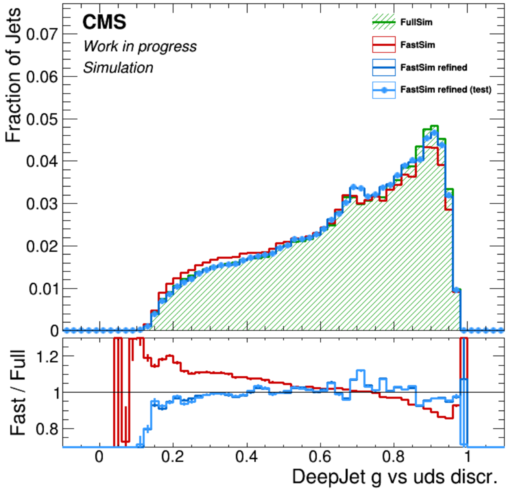


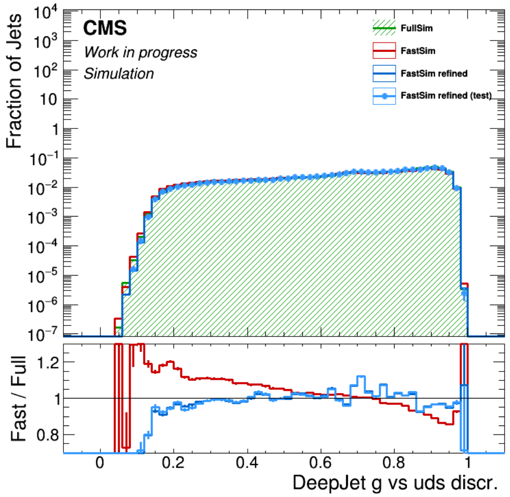




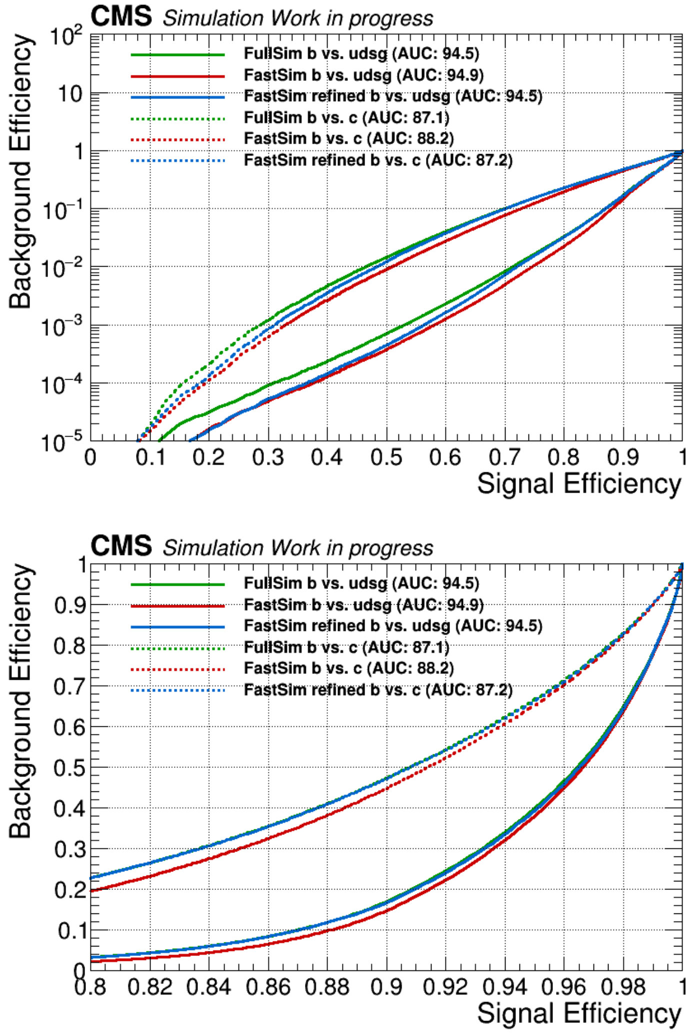


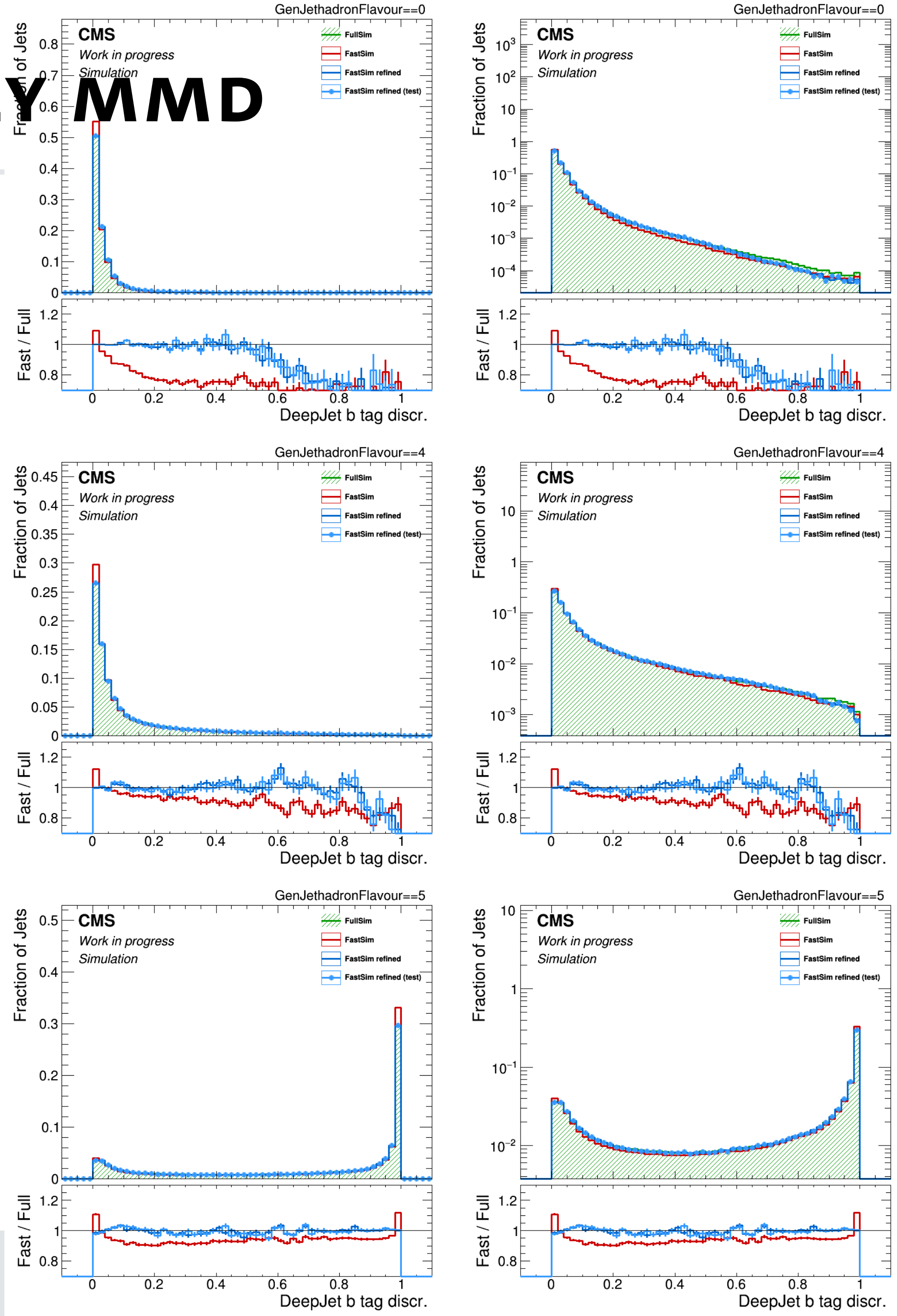




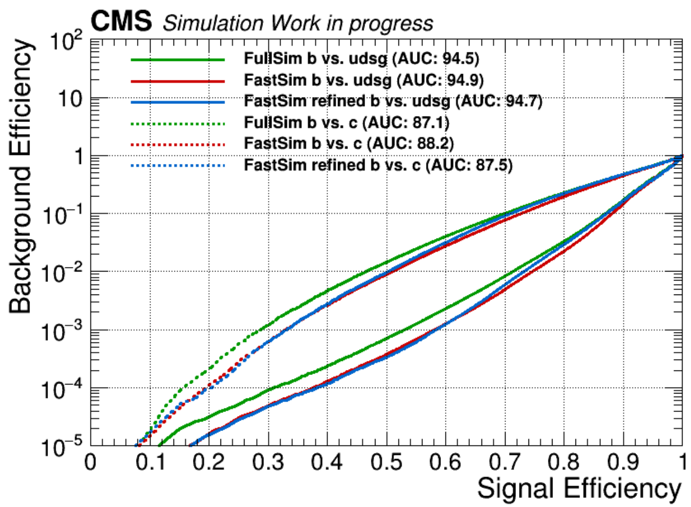


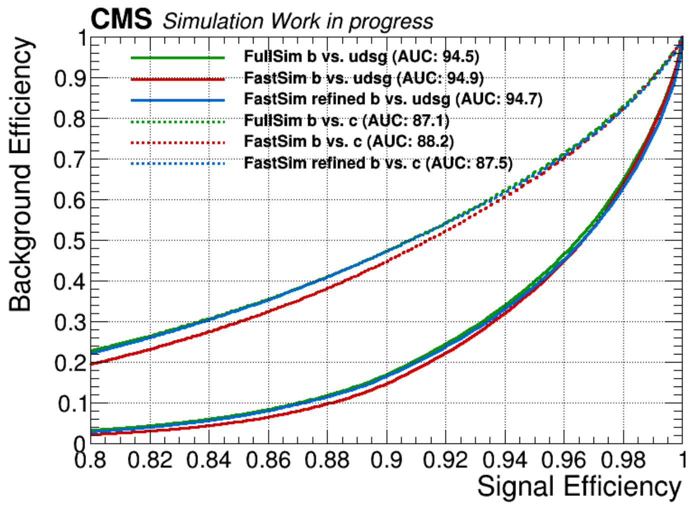
REFINING (REGRESSION) – ONLY MMD

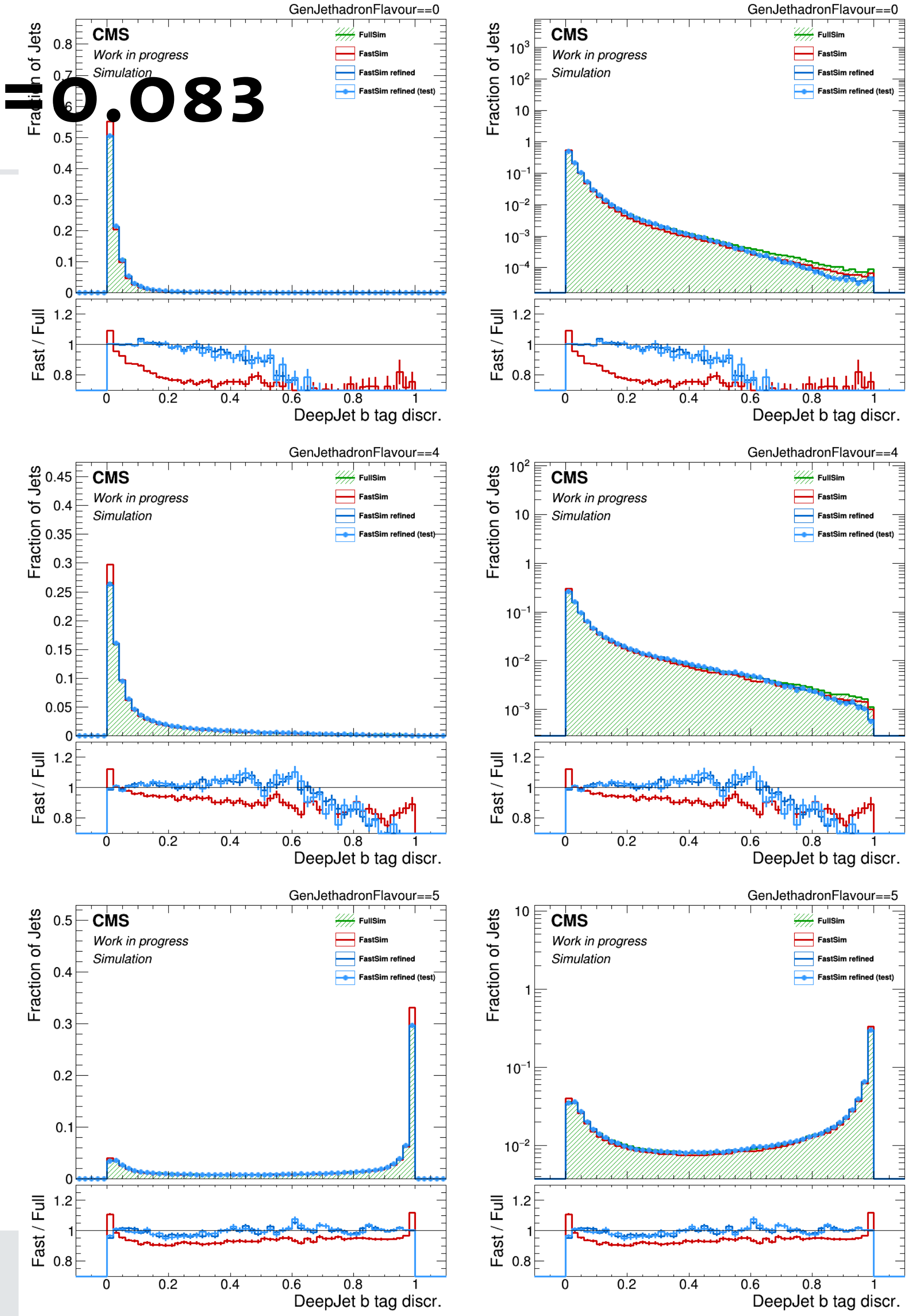




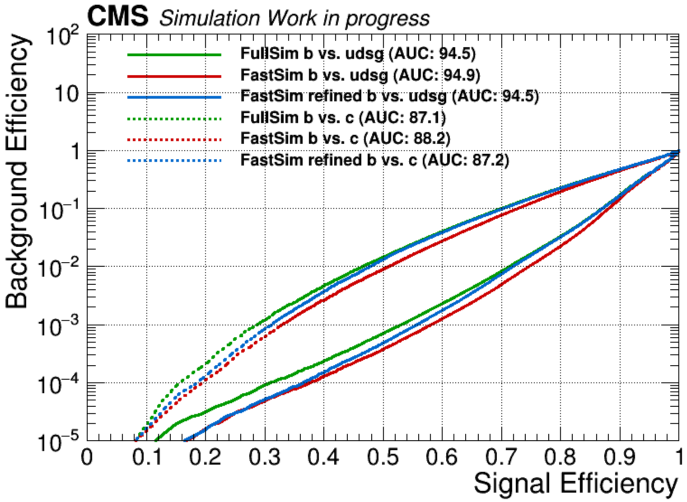
REFINING (REGRESSION) – EPS=0.083

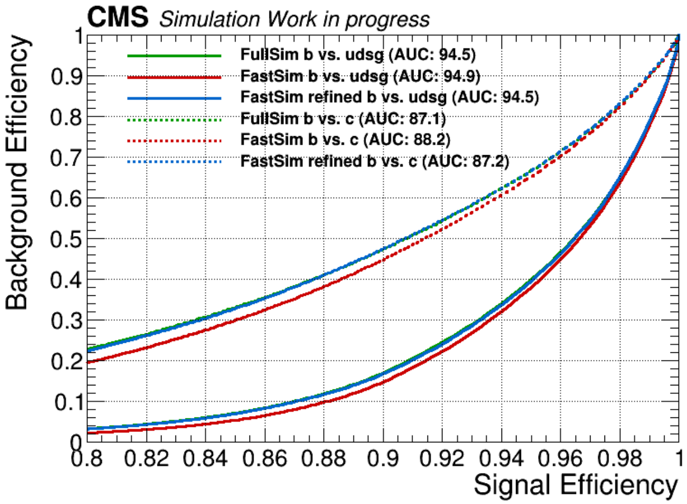


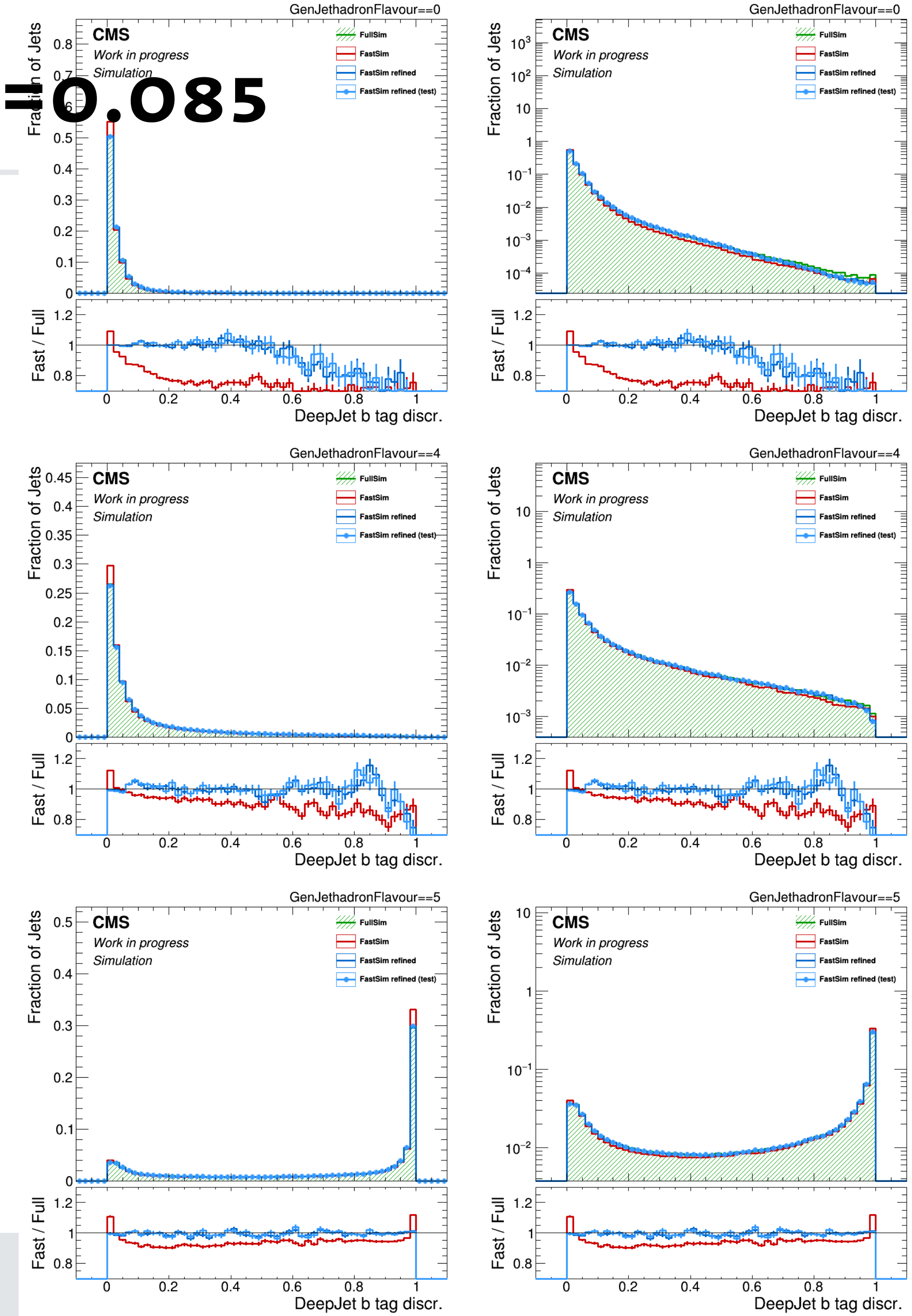




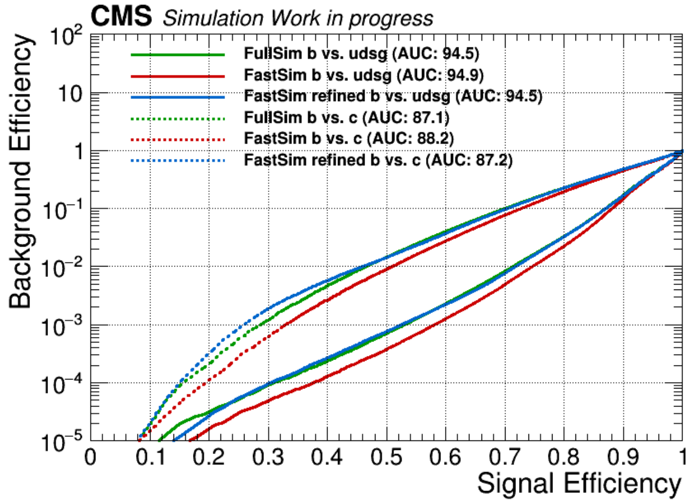
Refining (regression) – EPS=0.085

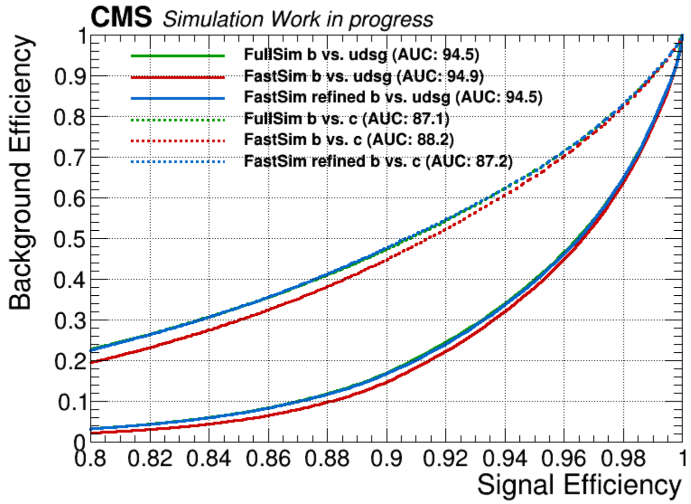


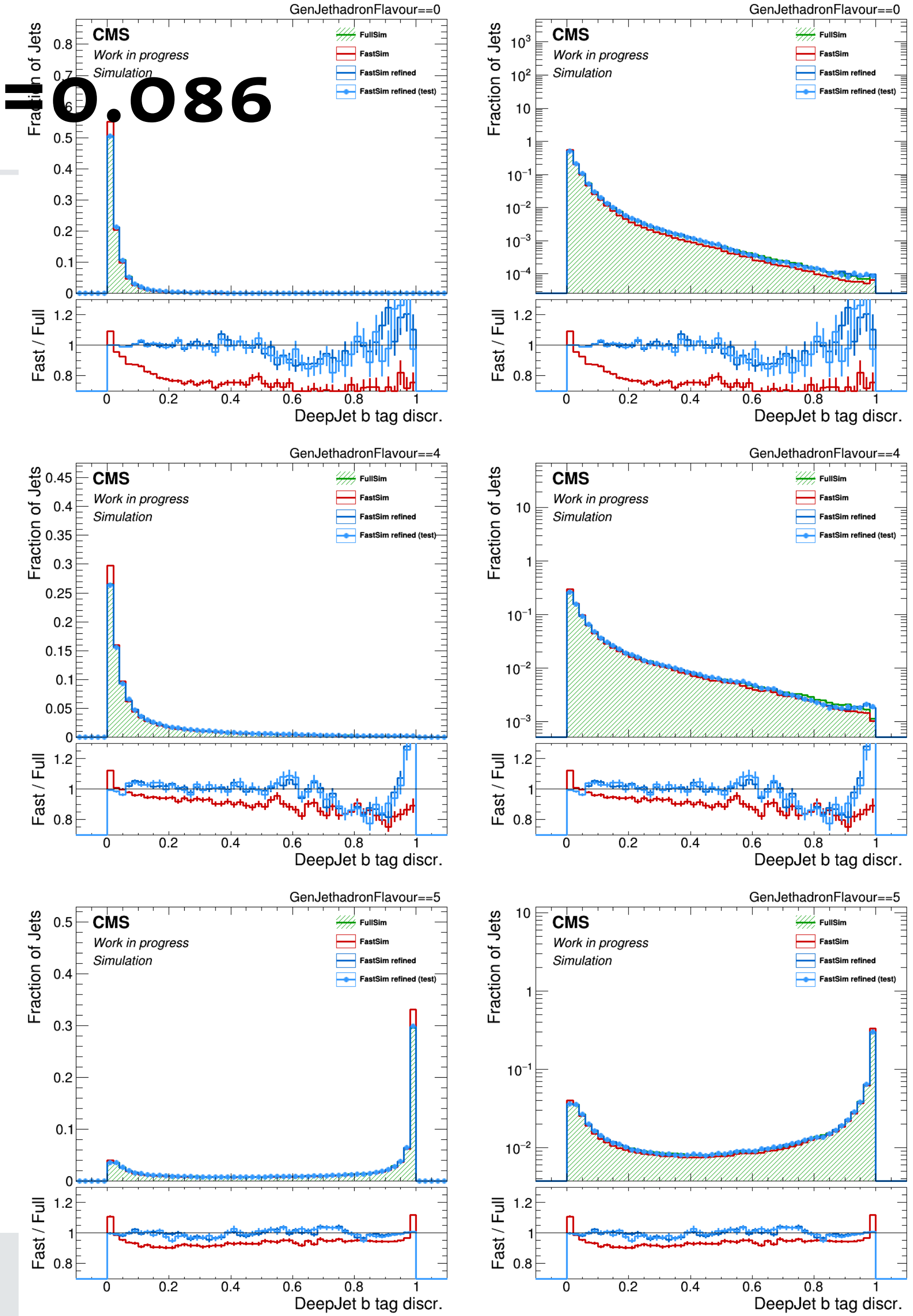




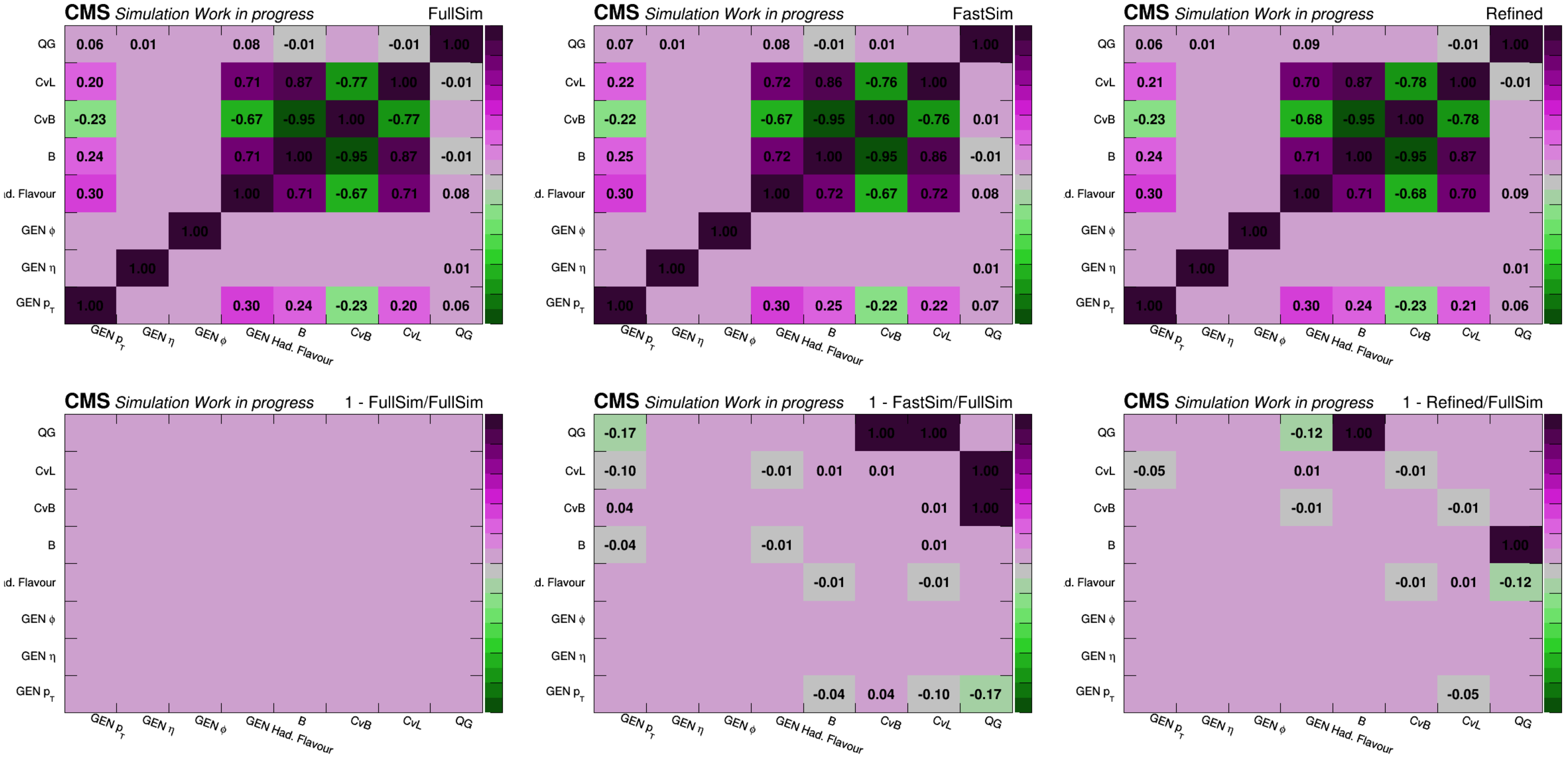
Refining (regression) – EPS=0.086



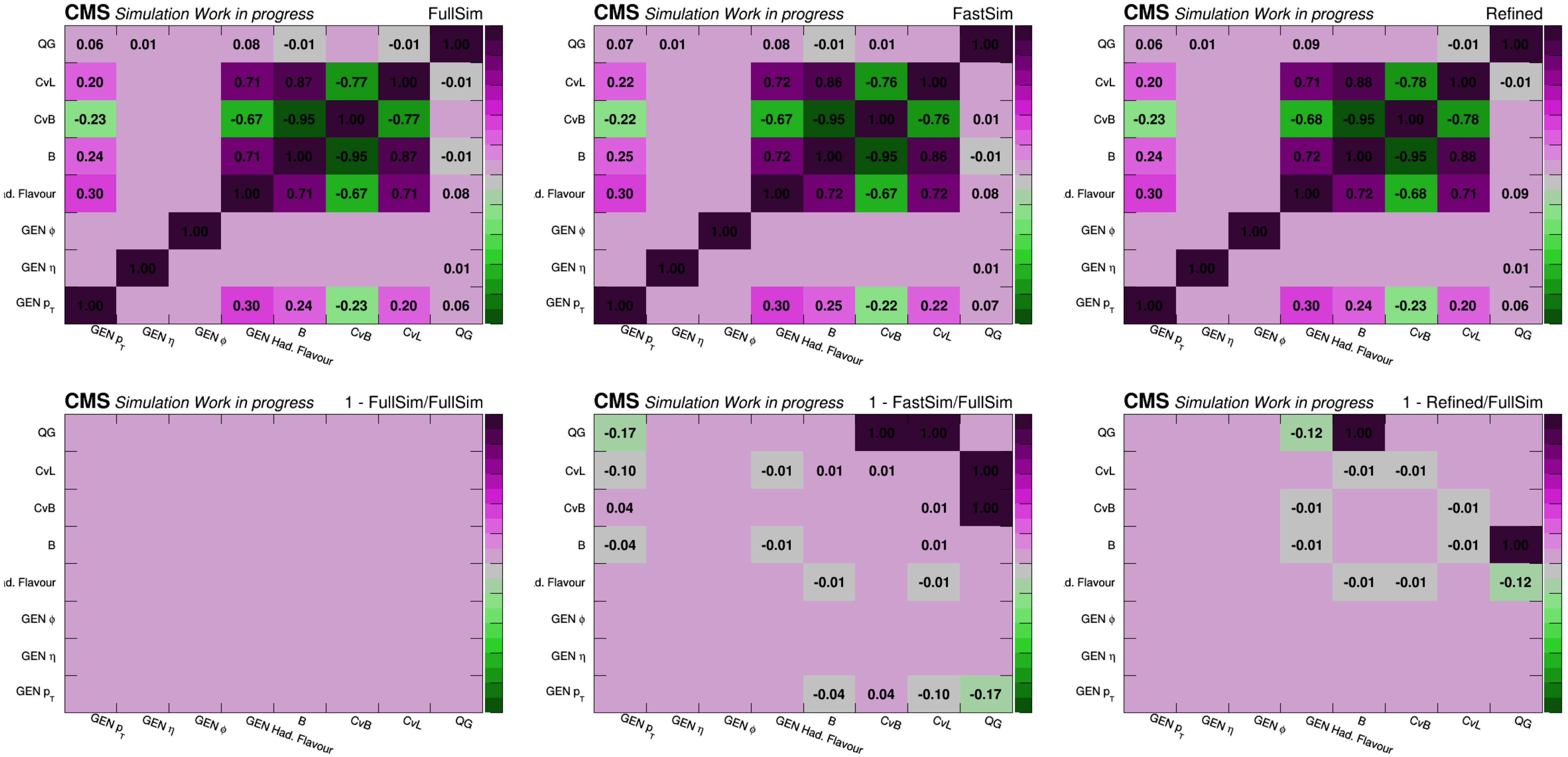




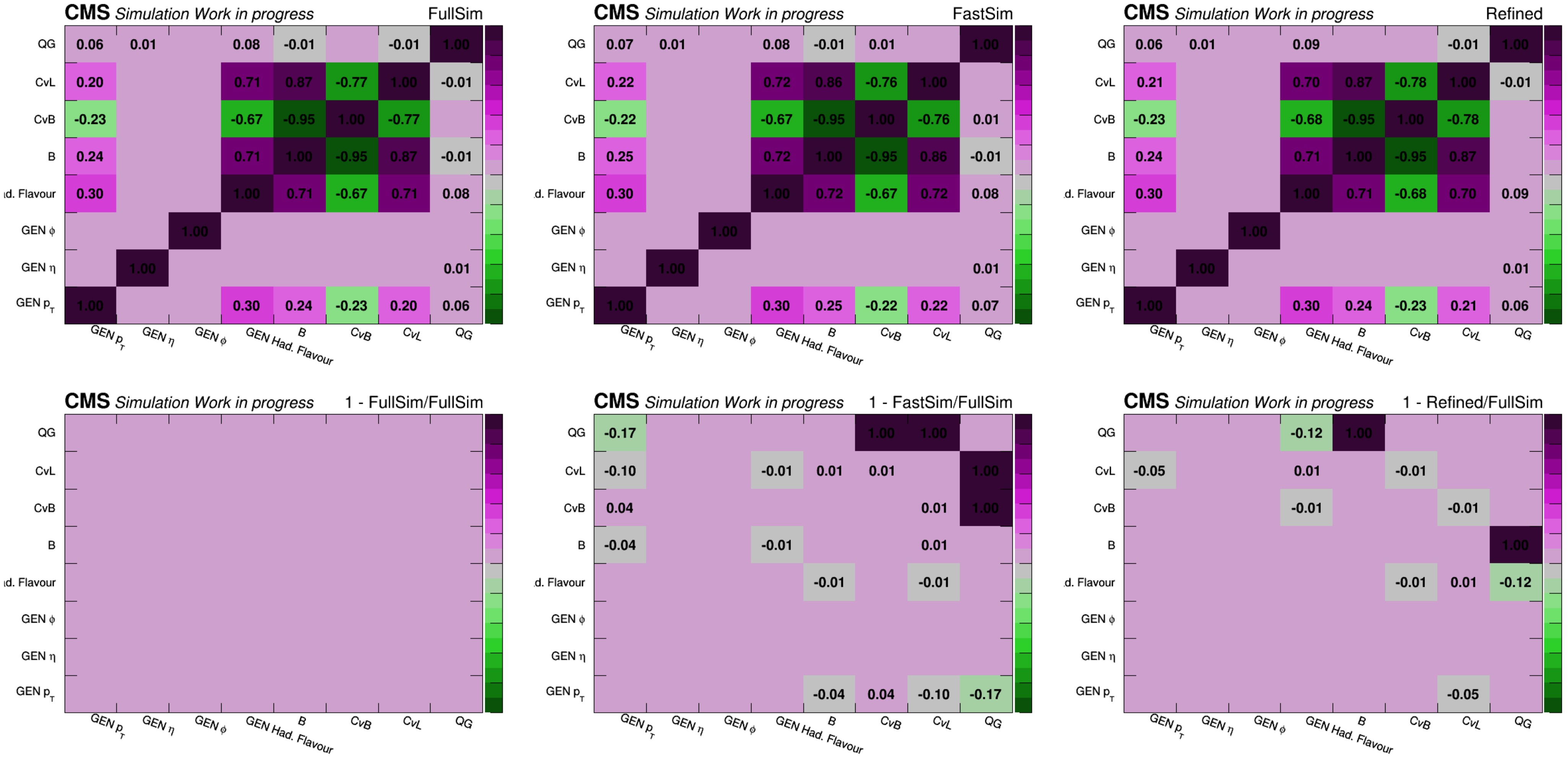
REFINING (REGRESSION) – ONLY MMD



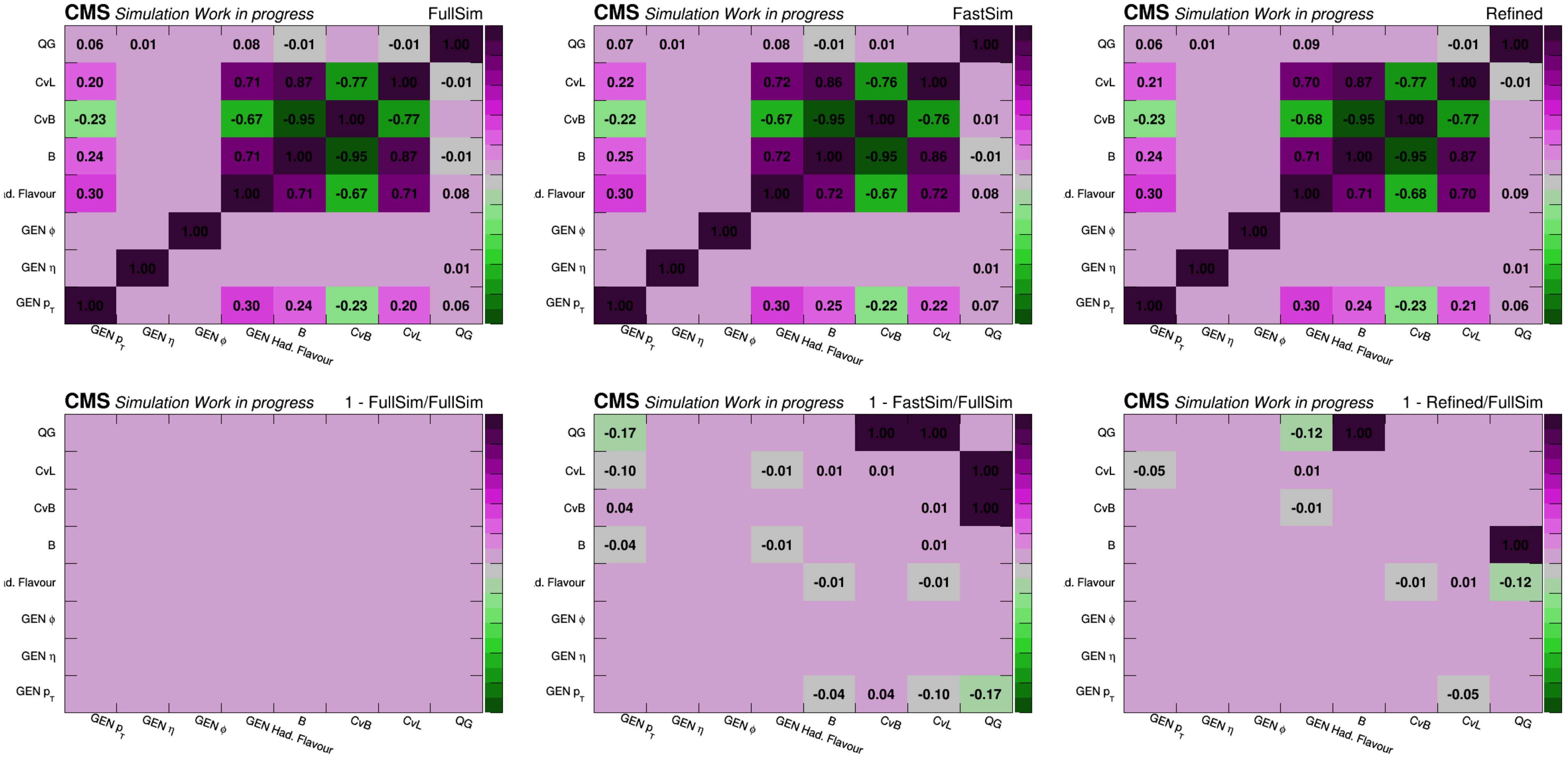
REFINING (REGRESSION) – EPS=0.083



Refining (regression) – eps=0.085



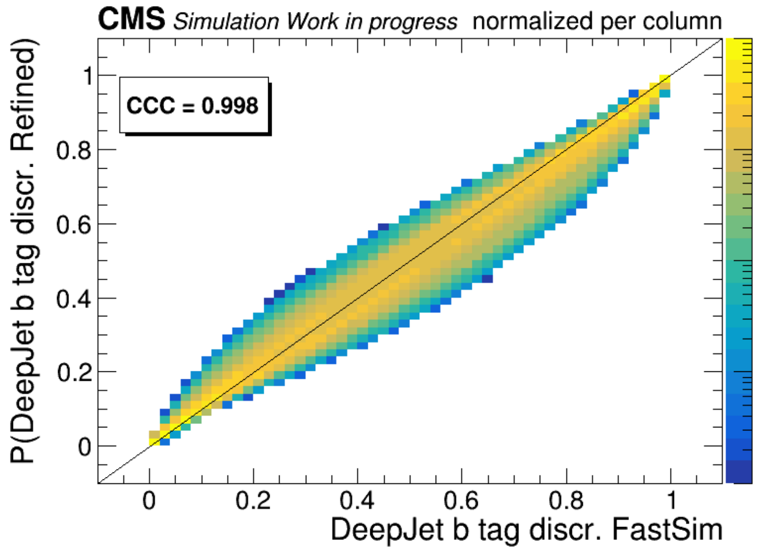
REFINING (REGRESSION) – EPS=0.086

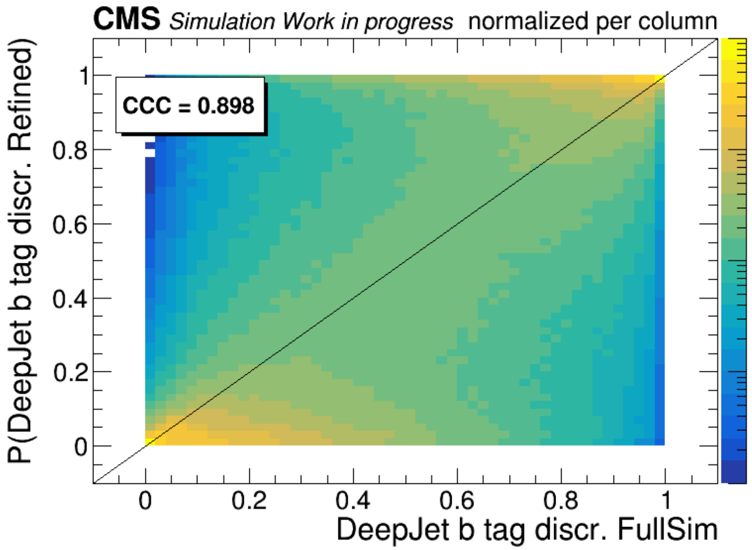


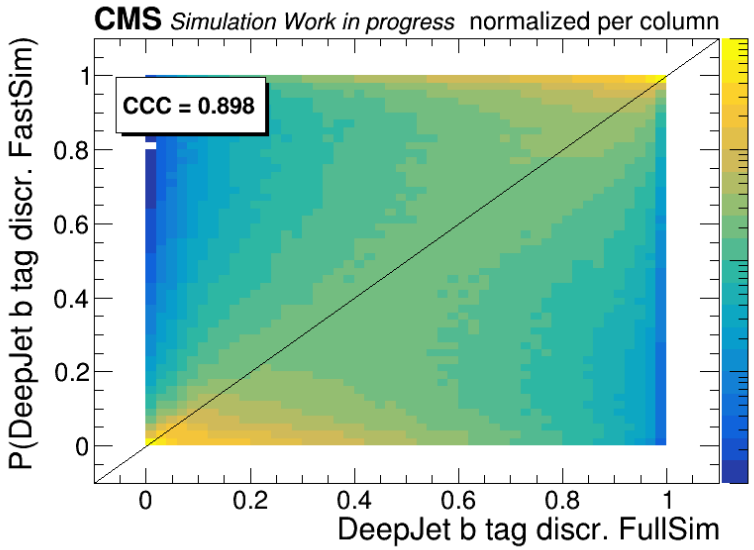
CCC = concordance correlation coefficient
“measures the agreement between two variables”

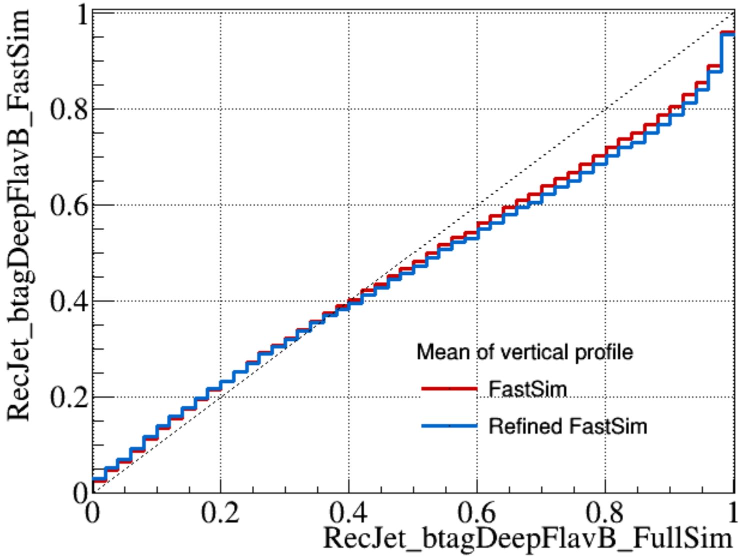
$$\rho_c = \frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2}$$

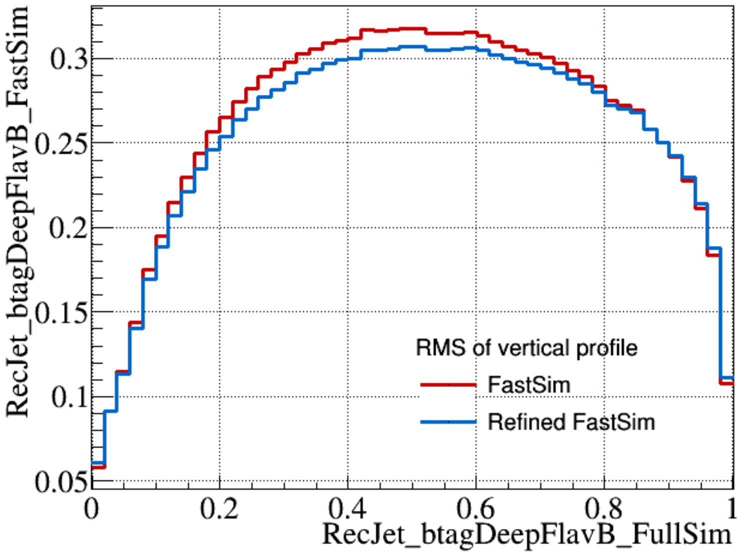
REFINING (REGRESSION) – ONLY MMD







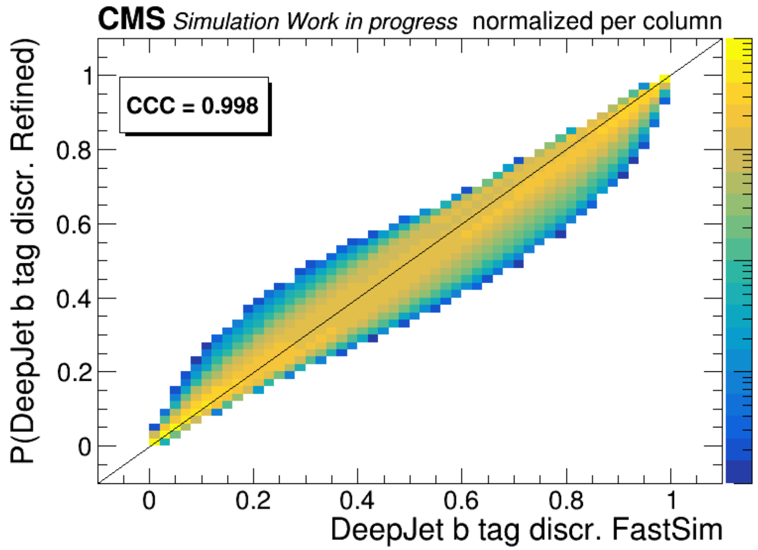


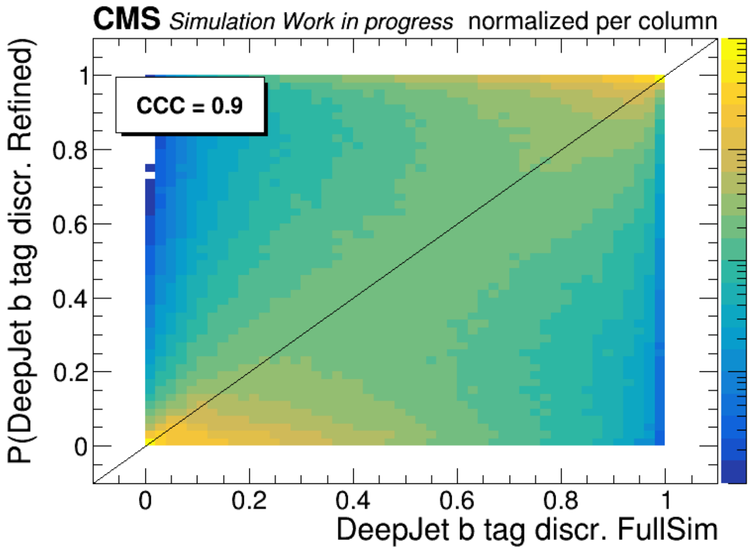


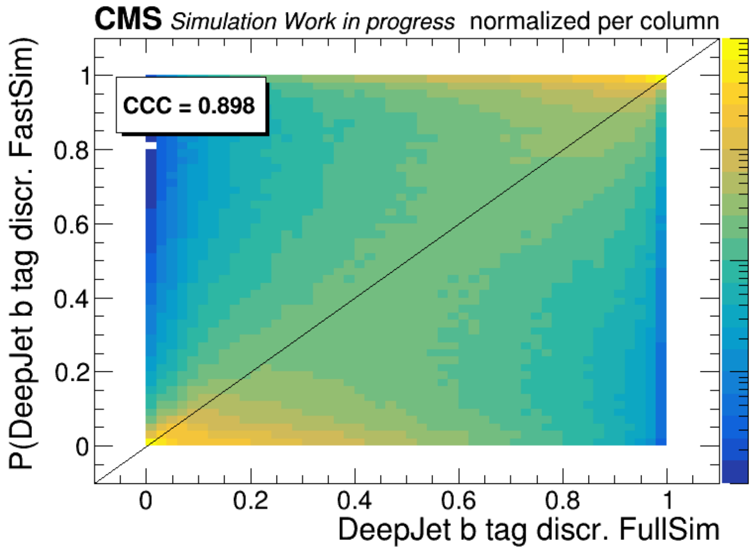
CCC = concordance correlation coefficient
“measures the agreement between two variables”

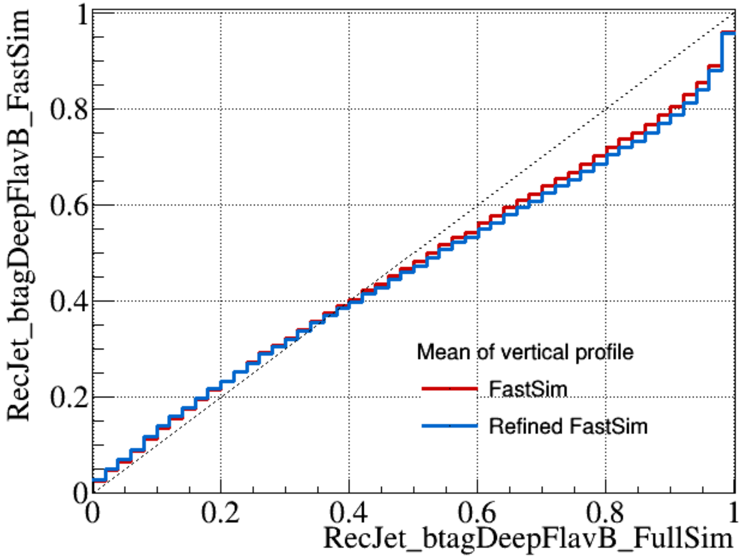
$$\rho_c = \frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2}$$

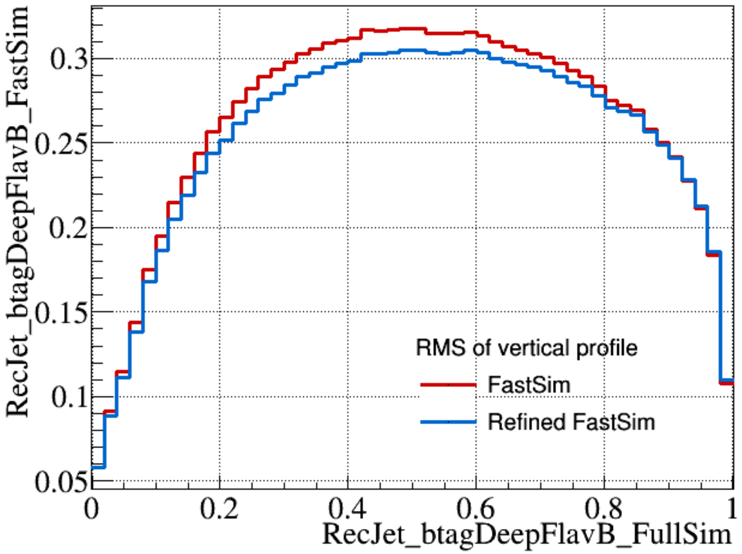
REFINING (REGRESSION) – EPS=0.083







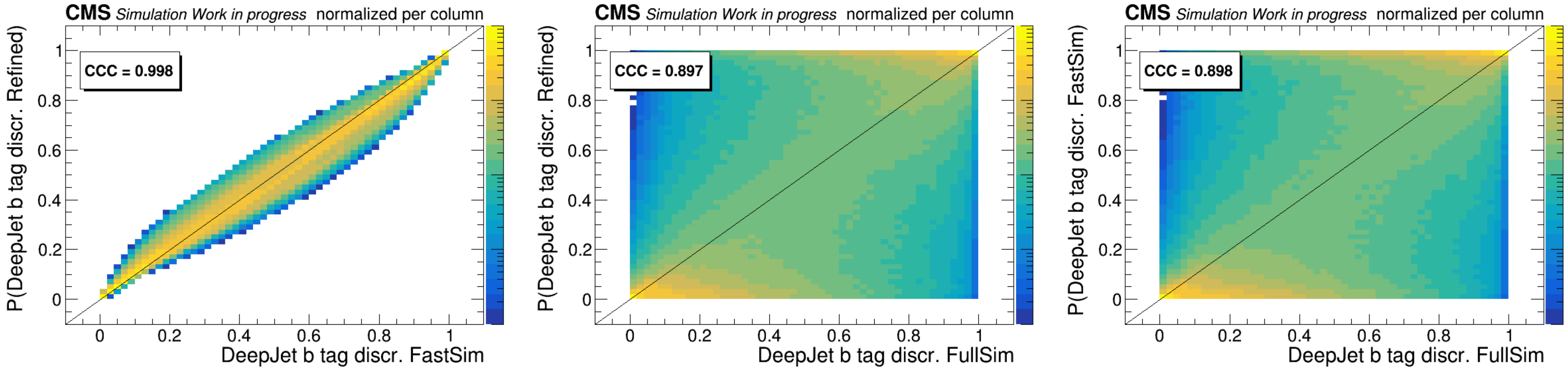


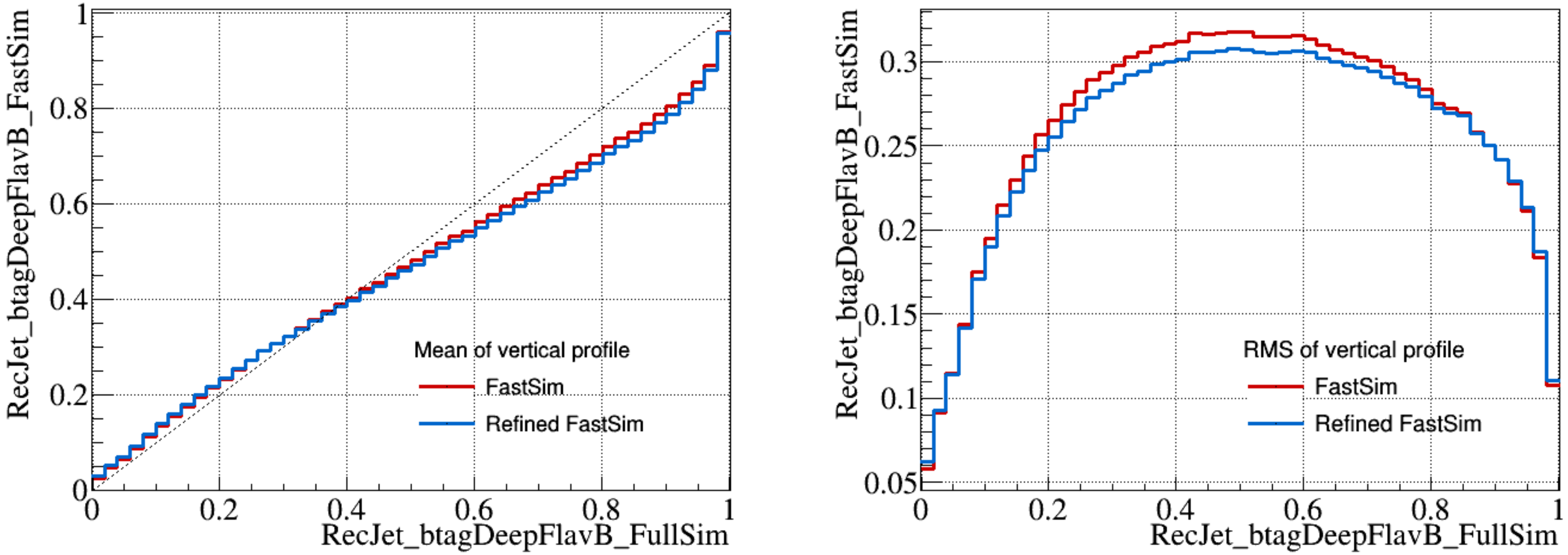


CCC = concordance correlation coefficient
“measures the agreement between two variables”

$$\rho_c = \frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2}$$

REFINING (REGRESSION) – EPS=0.085

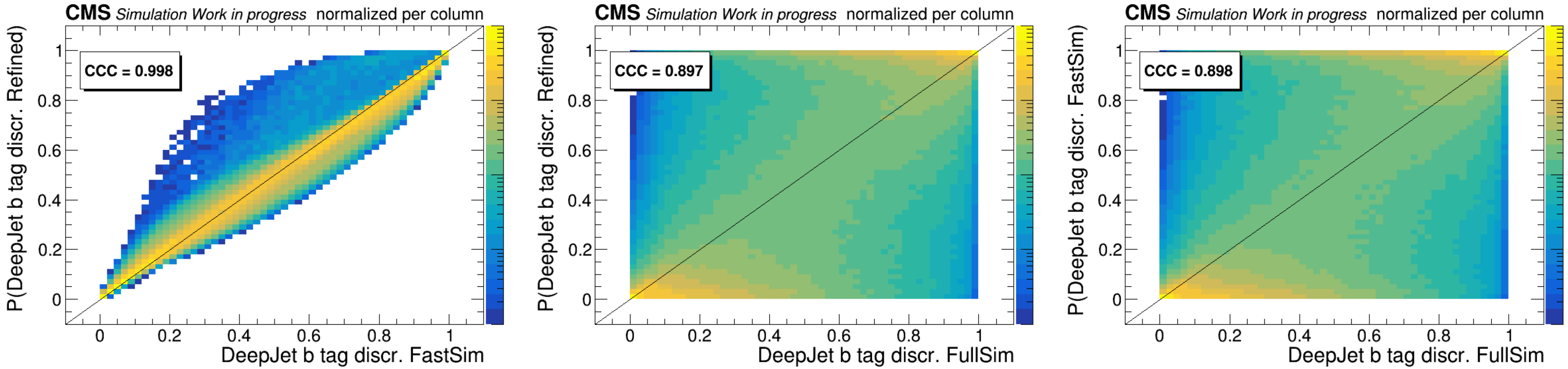


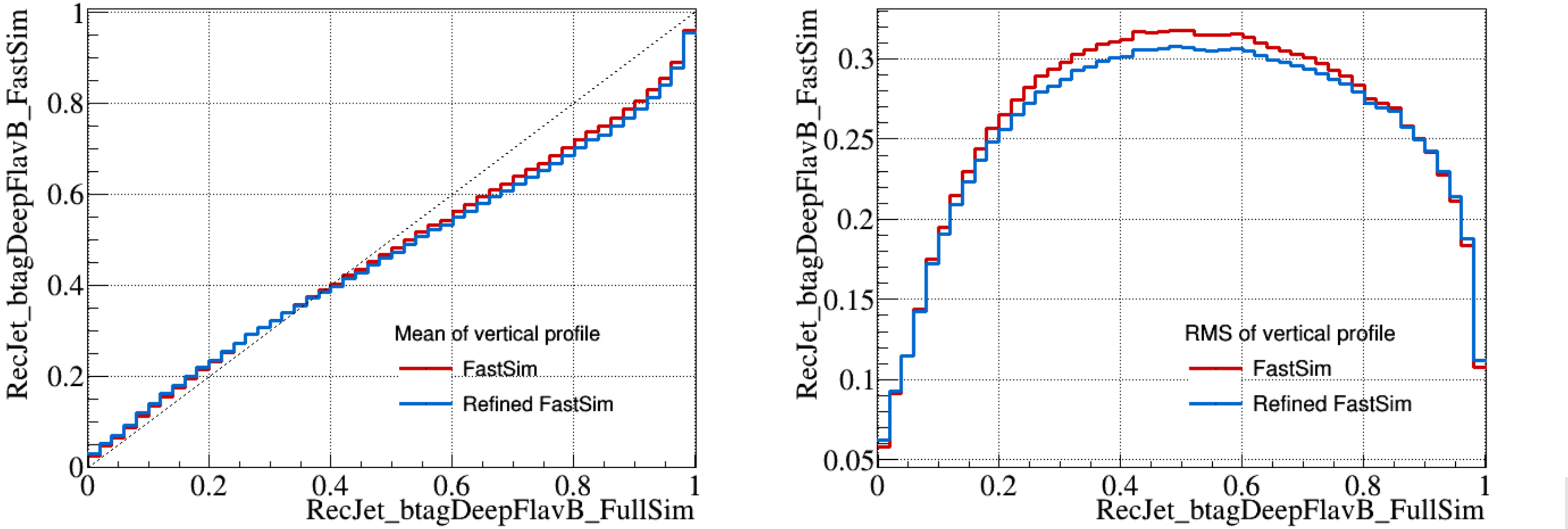


CCC = concordance correlation coefficient
“measures the agreement between two variables”

$$\rho_c = \frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2}$$

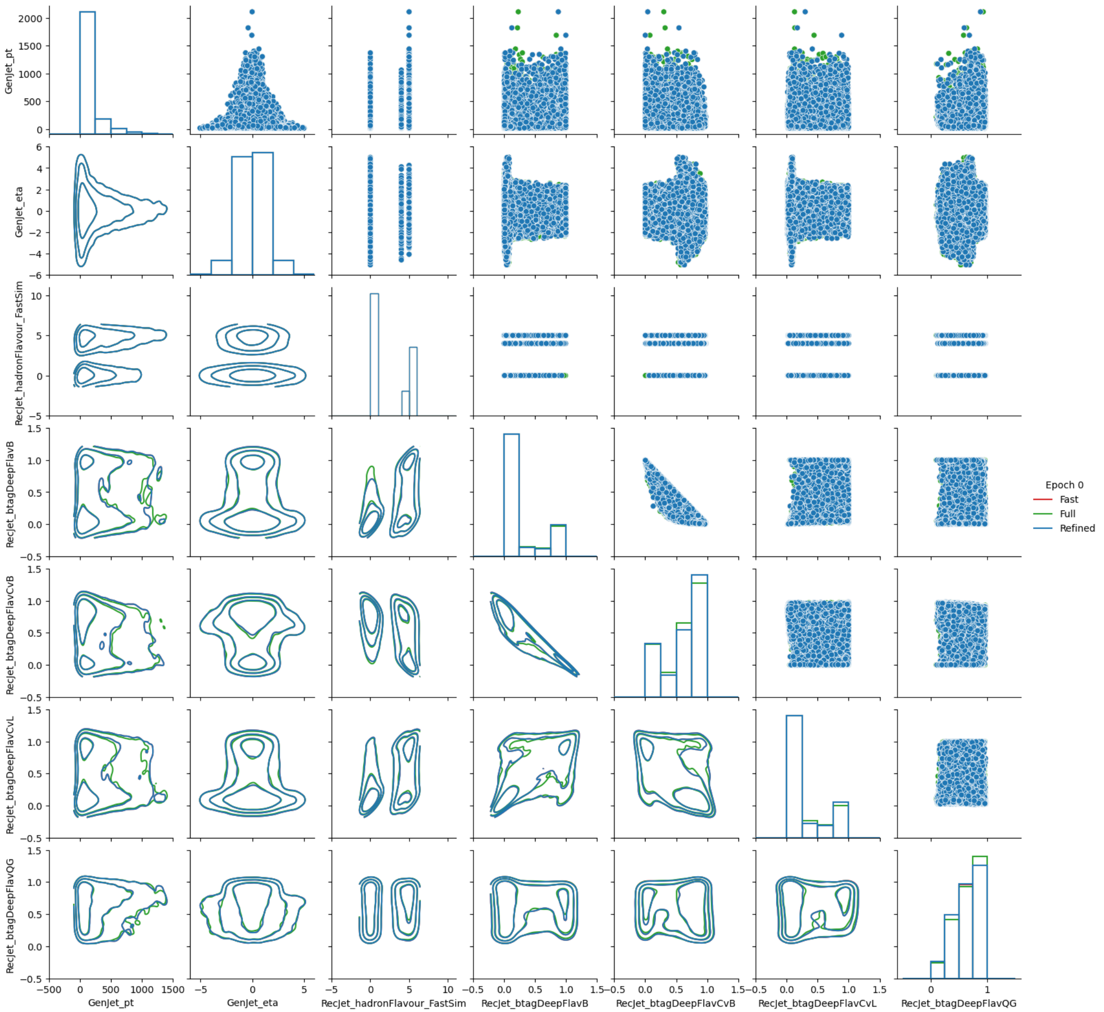
REFINING (REGRESSION) – EPS=0.086





REFINING (REGRESSION) – TRAINING SHAPSHOTS (EPS=0.084)

- Starting point
- Normal space

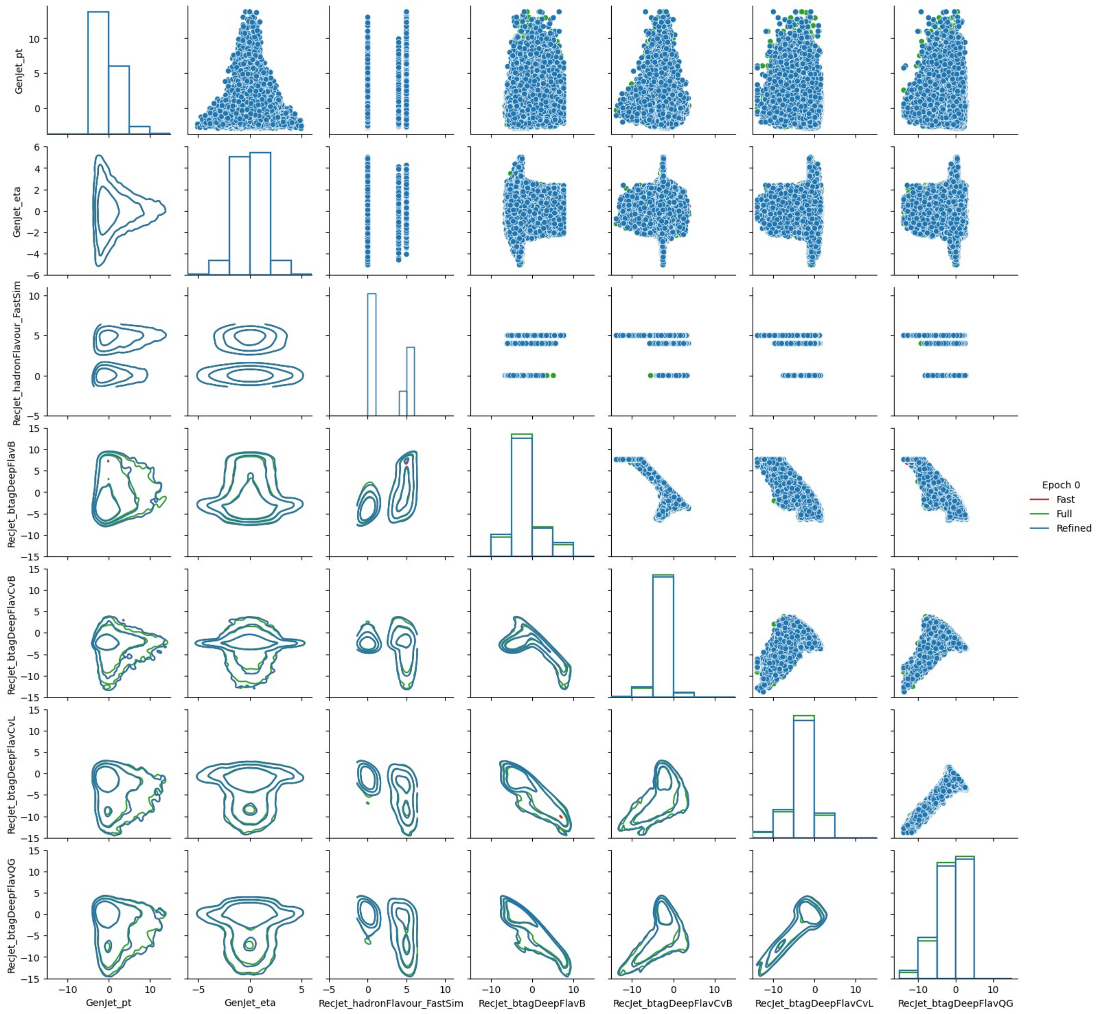


REFINING (REGRESSION) – TRAINING SHAPSHOTS (EPS=0.084)

- After epoch 100
- Normal space

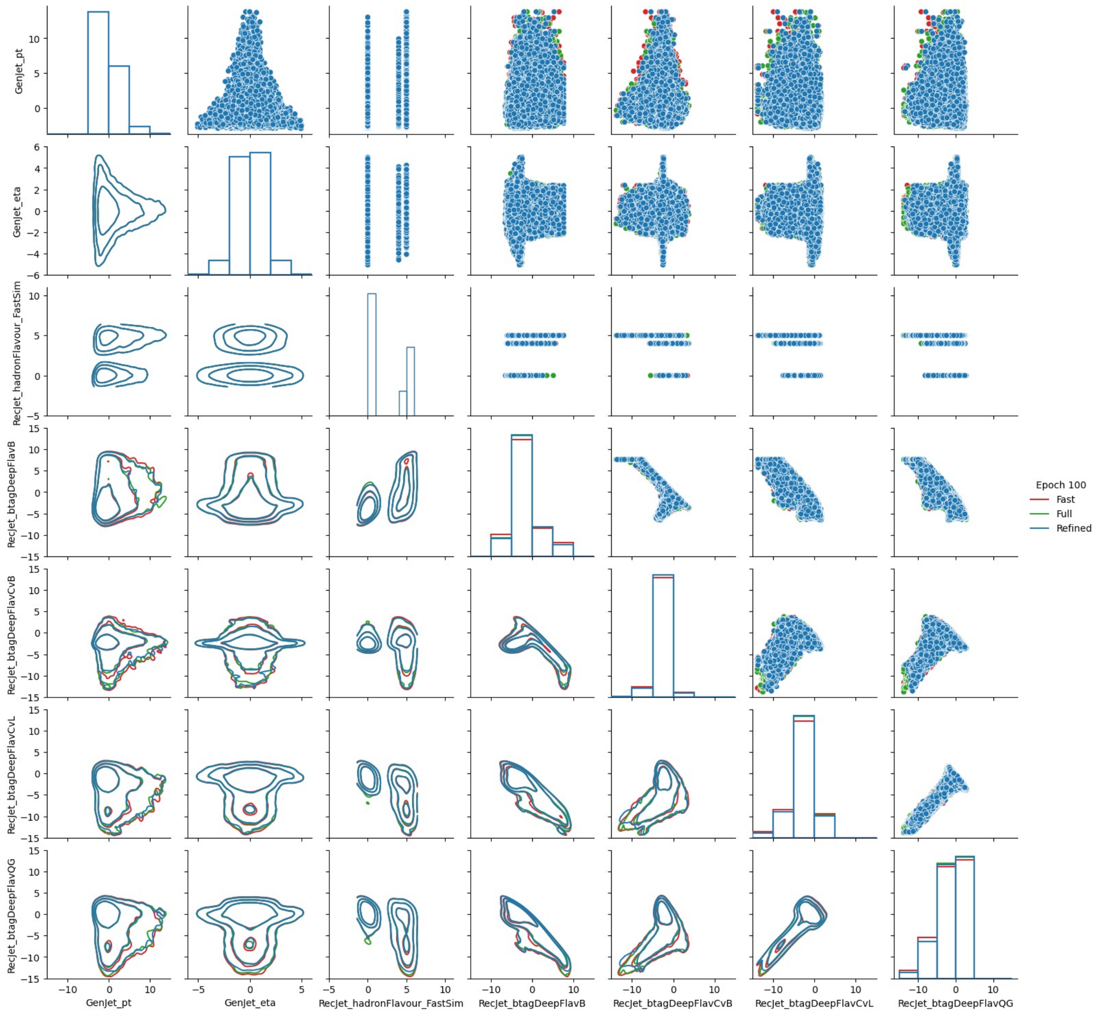
Refining (Regression) – Training shapshots (eps=0.084)

- Starting point
- Transformed space

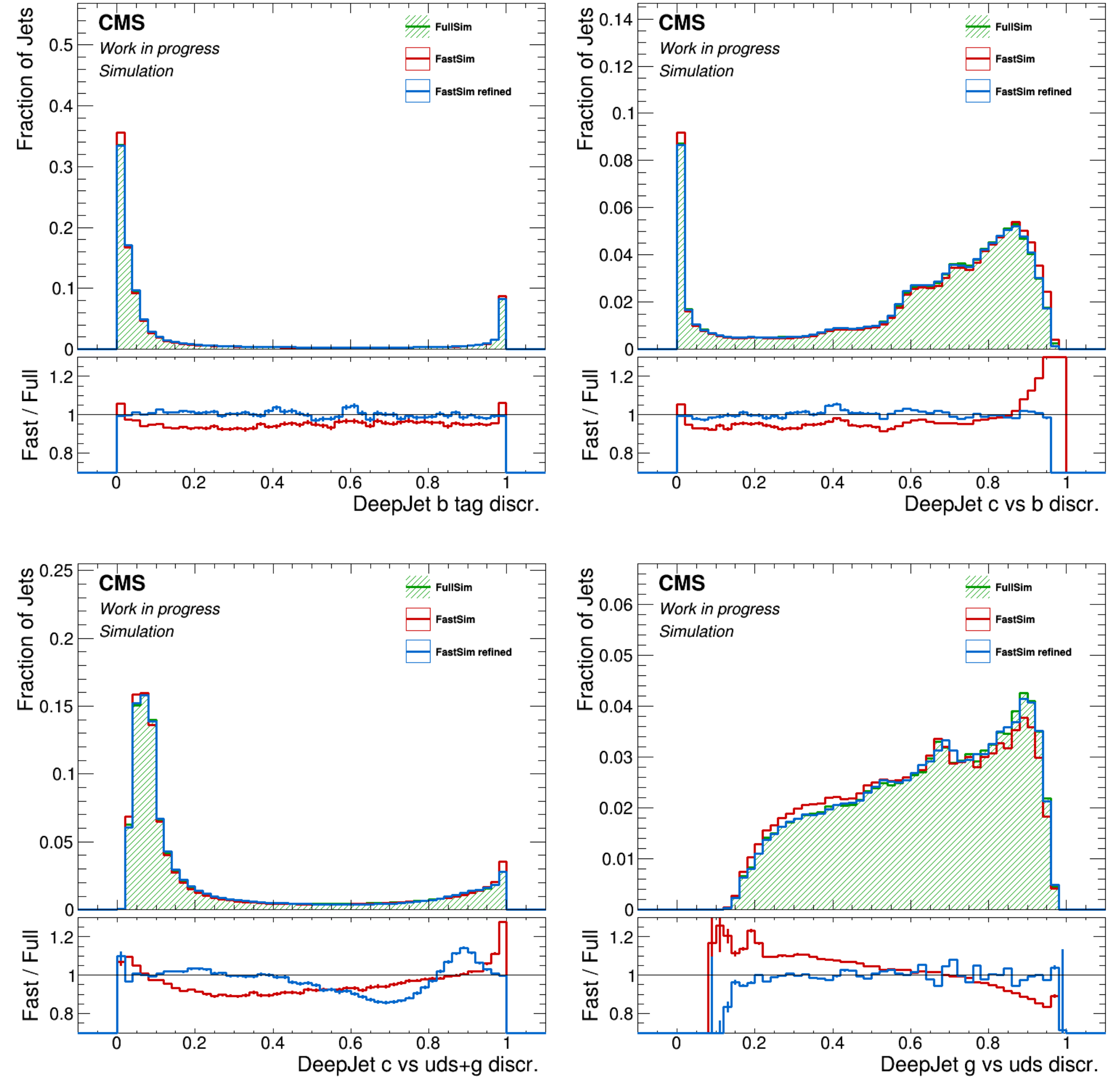


Refining (regression) – training shapshots (eps=0.084)

- After epoch 100
- Transformed space

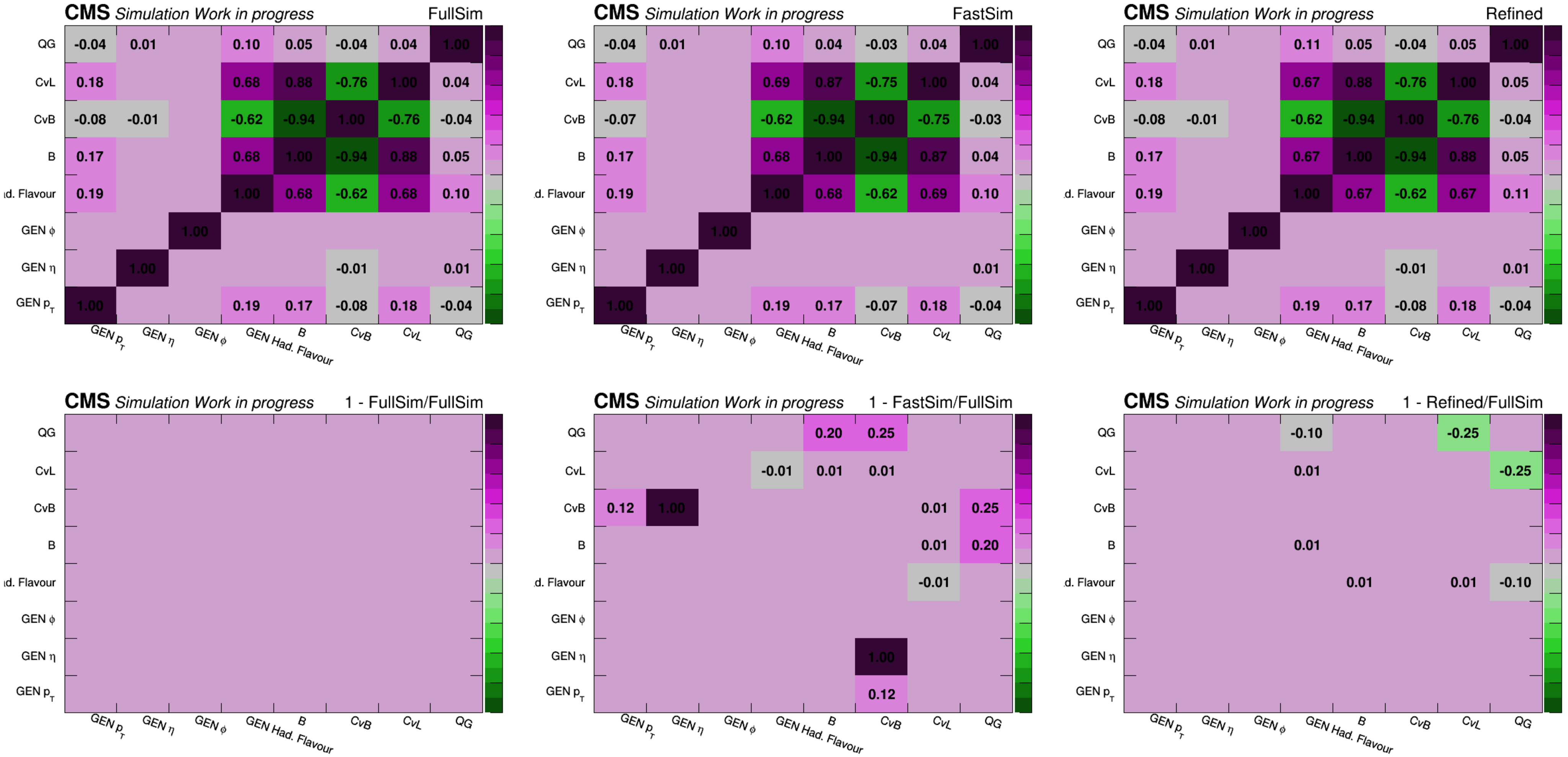


REFINING (REGRESSION) – VALIDATION IN TTBAR (ONLY MMD)

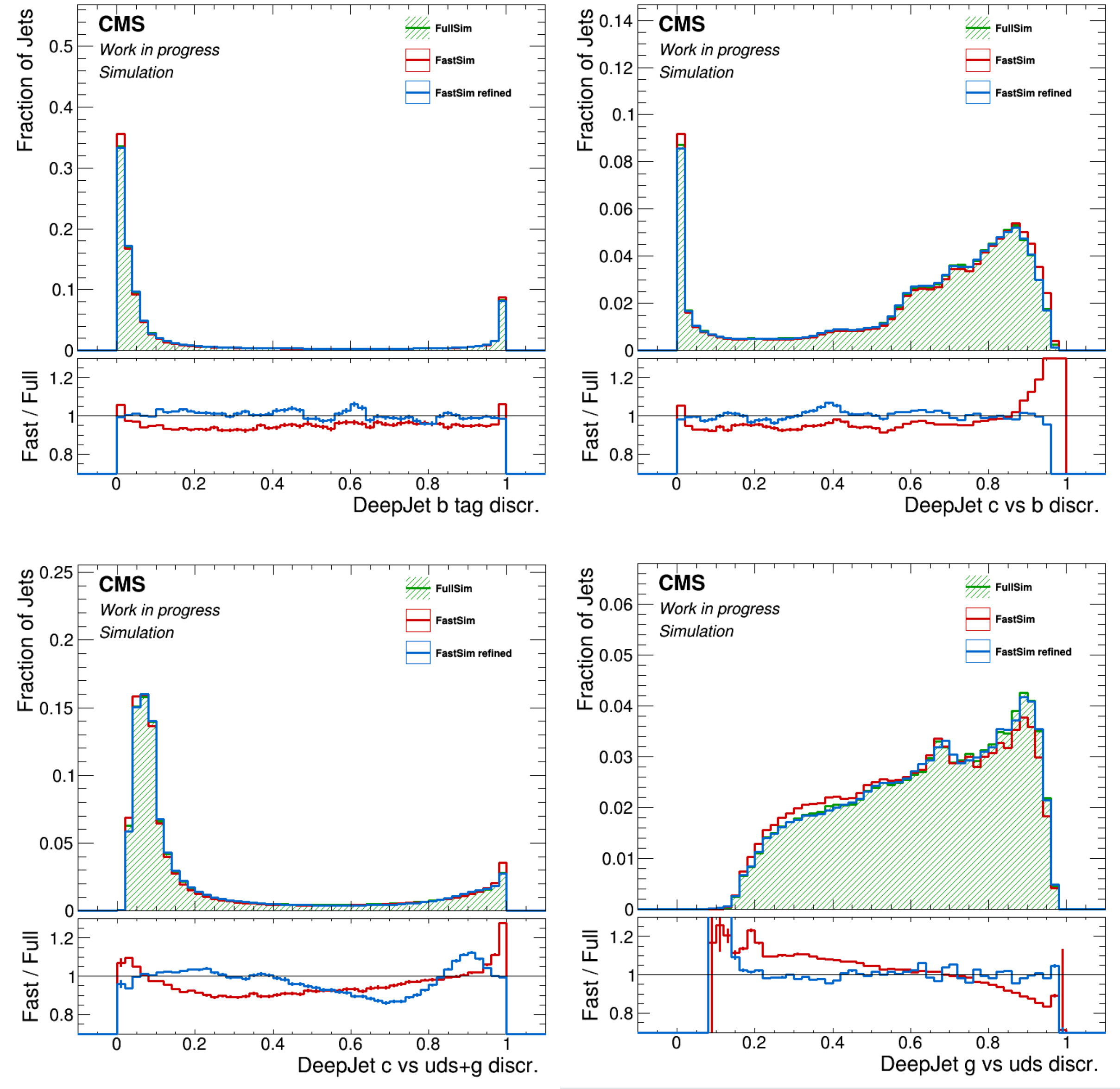


- Good performance for evaluation on **TTbar** (NN trained on T1tttt SUSY dataset)

REFINING (REGRESSION) – VALIDATION IN TTBAR (ONLY MMD)

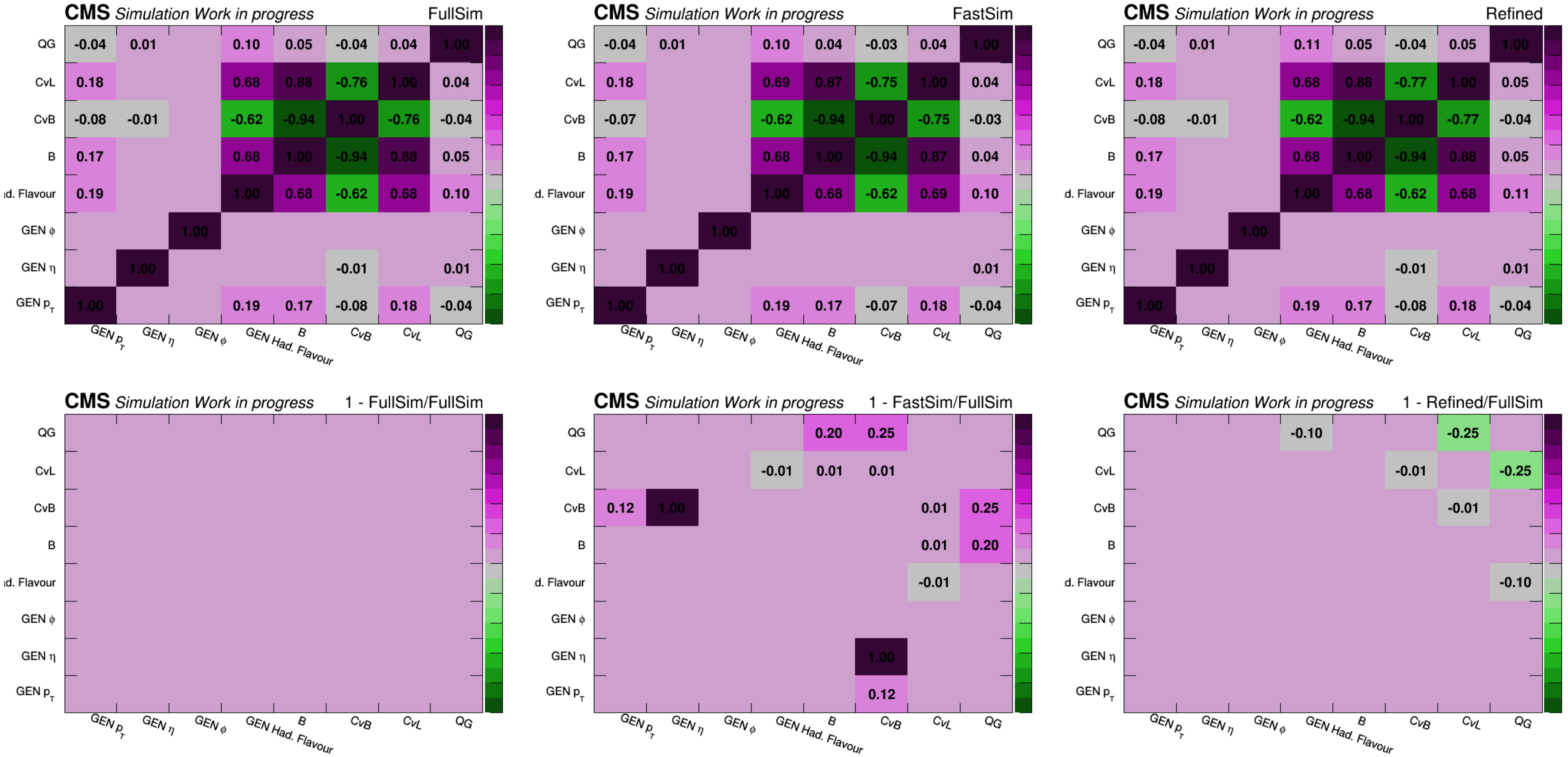


REFINING (REGRESSION) – VALIDATION IN TTBAR (EPS=0.083)

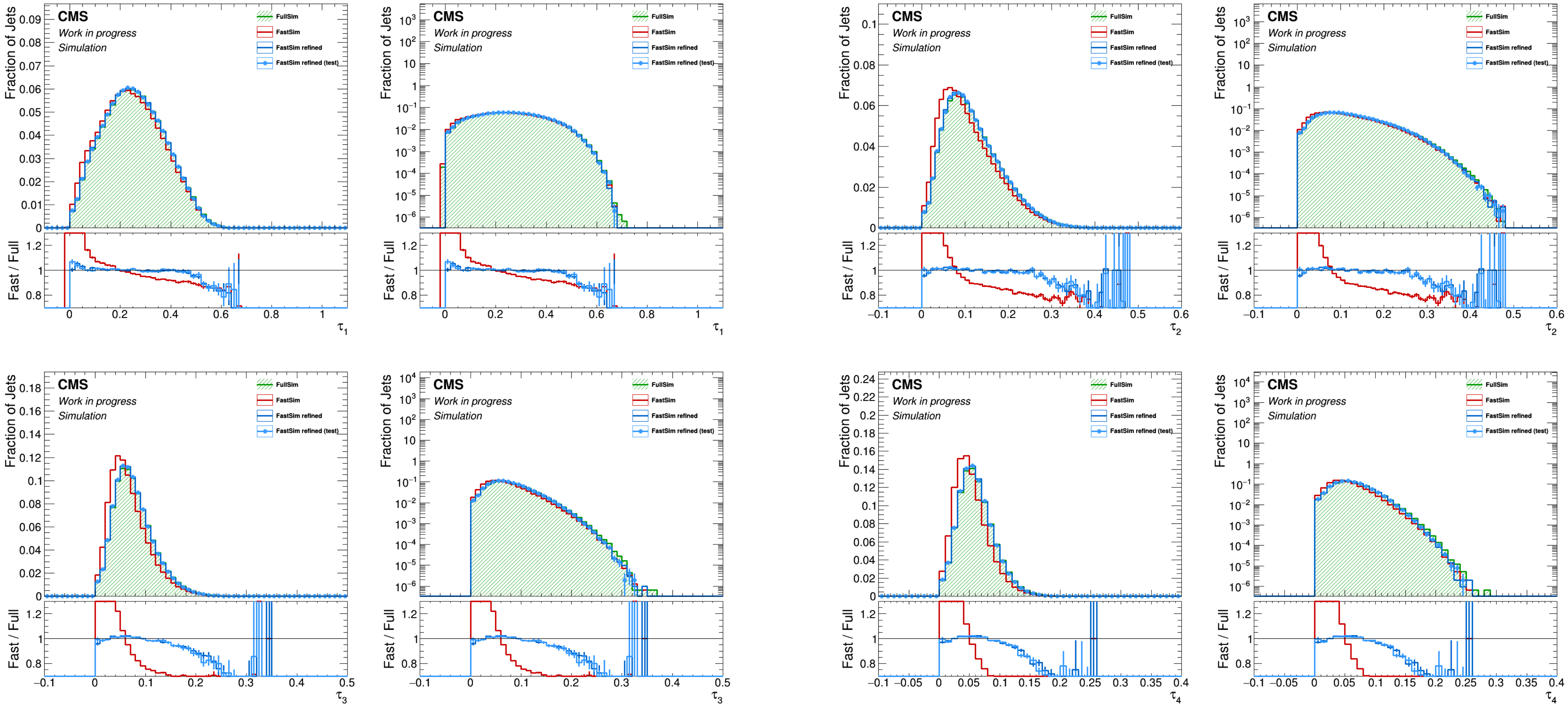


- Good performance for evaluation on **TTbar** (NN trained on T1tttt SUSY dataset)

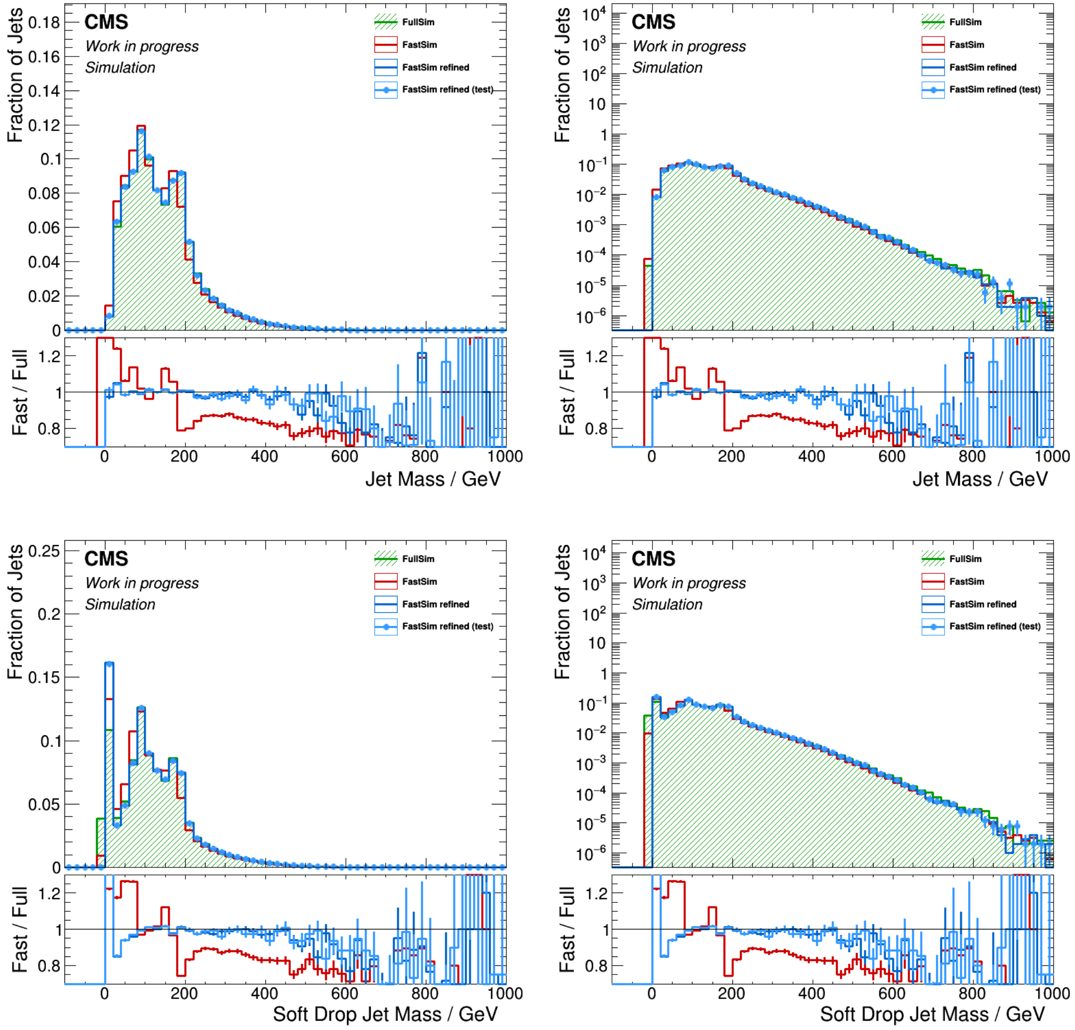
REFINING (REGRESSION) – VALIDATION IN TTBAR (EPS=0.083)



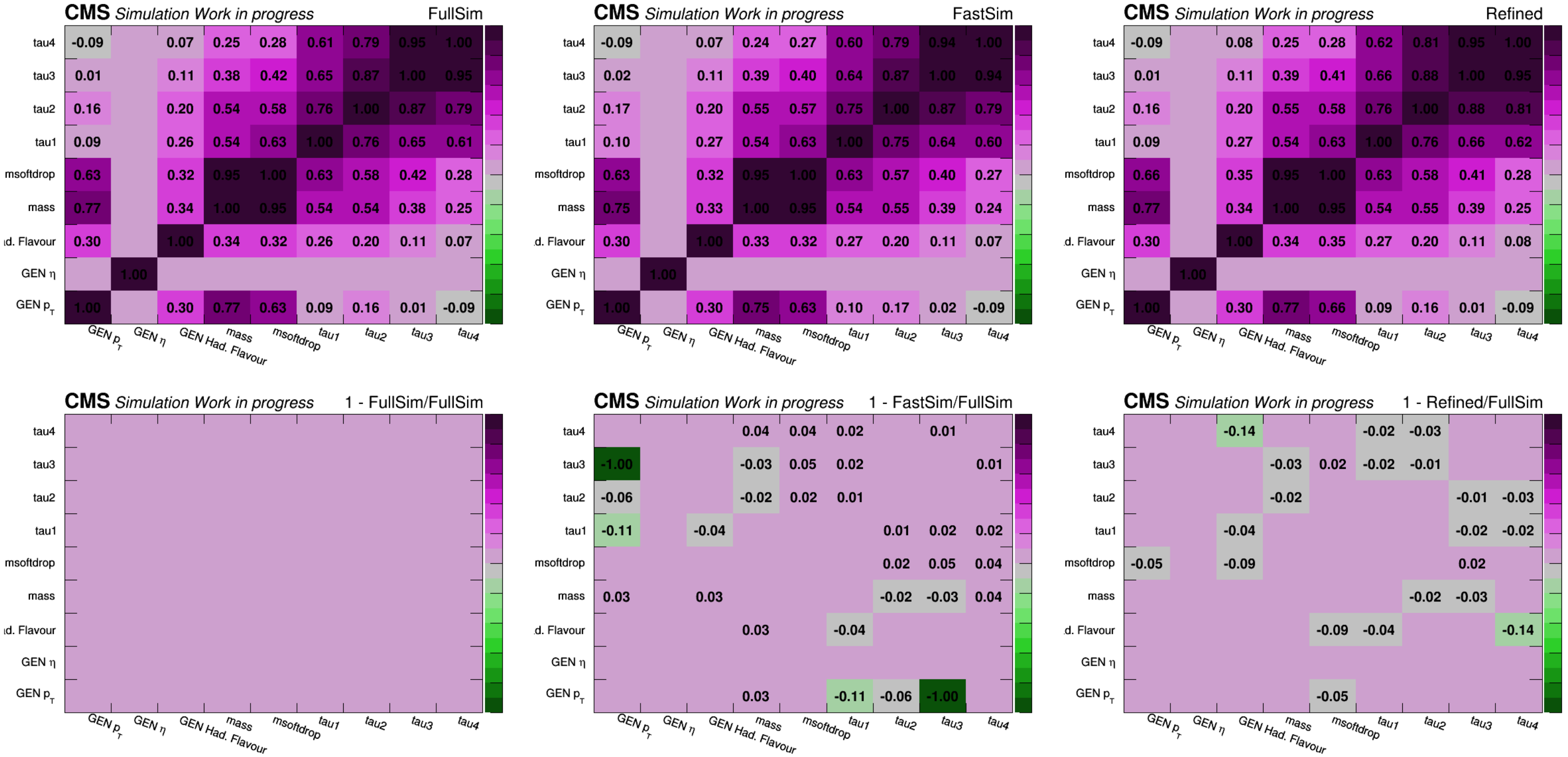
REFINING (REGRESSION) – FATJETS (ONLY MMD)



REFINING (REGRESSION) – FATJETS (ONLY MMD)



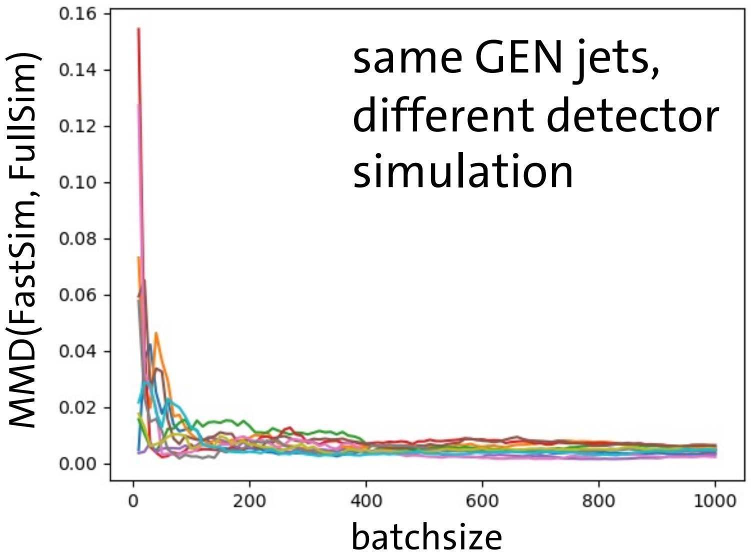
REFINING (REGRESSION) – FATJETS (ONLY MMD)



Refining (regression) – network details

- Jet preselection dR(jet, closest jet) > 0.5 (for GEN, FastSim & FullSim jet individually)
- NN architecture
  - LeakyReLU negative slope = 0.01
  - Dropout rate = 0.5
- Loss terms
  - Huber delta = 0.1
  - MMD calculated individually for each class of hadron flavour: $\mathrm{MMD} = \sum_{i\,\epsilon\,had.\,flav.} MMD_i$
  - MMD Gaussian kernel bandwidth adaptive: $\mathrm{MMD} = \sum_{\sigma\,\epsilon\,\{\frac{\sigma_0}{100},\frac{\sigma_0}{10},\sigma_0,10\sigma_0,100\sigma_0\}} MMD_\sigma$, $\sigma_0 = \overline{\mathrm{L2\ dist.}}$
- Training
  - Learning rate = 1e-4 (exponential decay with $\gamma = 0.96$ each epoch)
  - Adamax optimizer

REFINING (REGRESSION) – MMD



- Training on jet$^{\text{FastSim}}$-jet$^{\text{FullSim}}$ pairs problematic (stochasticity in simulations)
  → instead use measure that compares **distributions**
- From Wikipedia:

Kernel two-sample test [edit]

Given *n* training examples from $P(X)$ and *m* samples from $Q(Y)$, one can formulate a test statistic based on the empirical estimate of the MMD

$$\widehat{\mathrm{MMD}}(P,Q) = \left\| \frac{1}{n}\sum_{i=1}^{n}\varphi(x_i) - \frac{1}{m}\sum_{i=1}^{m}\varphi(y_i) \right\|_{\mathcal{H}}^2$$

$$= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}k(x_i,x_j) + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}k(y_i,y_j) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}k(x_i,y_j)$$

to obtain a **two-sample test** [12] of the null hypothesis that both samples stem from the same distribution (i.e. $P=Q$) against the broad alternative $P \neq Q$.

- Gaussian kernel: $k(x,x') = \exp\left(-\frac{1}{2\sigma^2}\|x-x'\|^2\right)$
- Plot: MMD of different FastSim/FullSim jet batches with varying size (jets defined by p_T, η, DeepCSV)
  - **Batchsize** must be large enough to get a good estimate

REFINING (REGRESSION) – MDMM ALGORITHM

- Simple approach: add with constant weights
- Better: **Modified Differential Method of Multipliers** algorithm: [1] original paper, [2] explanatory blog post
  reframe problem as **constrained optimization** using a **Lagrangian**: $\mathcal{L} = f(\theta) - \lambda * (\varepsilon - g(\theta))$
  - Minimize $f(\theta)$ („primary loss") subject to $g(\theta) = \varepsilon$ („additional loss")
  - Convergence mathematically formalized, steady states are saddle points
  - Lagrange multiplier λ is adapted in the training along with NN parameters θ
  - $f(\theta)$ and $g(\theta)$ depend on each other
    → both can't be arbitrarily small at the same time
    → have to choose ε (point along Pareto front)

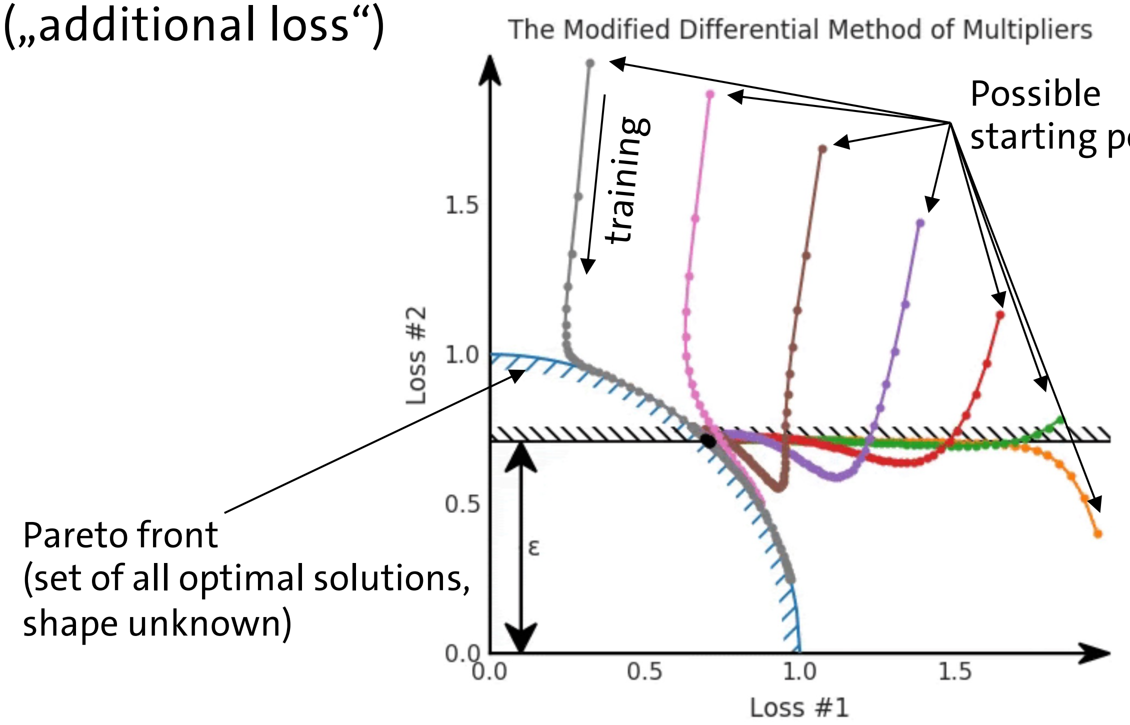
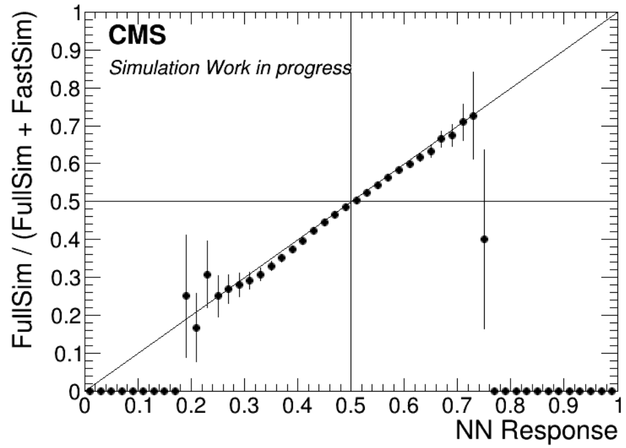


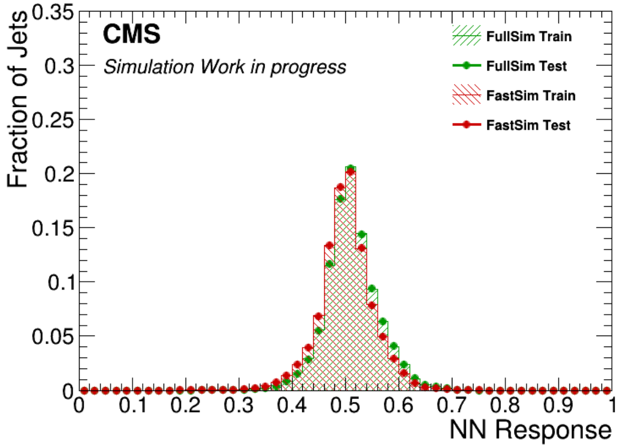
image from [2]

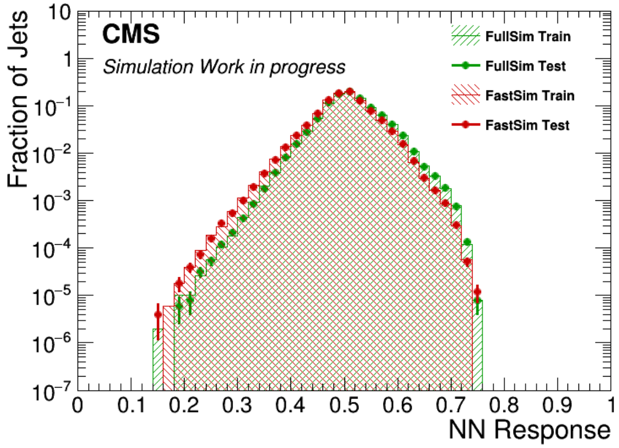
REWEIGHTING (DCTR)

- **DCTR** approach (*Deep neural networks using Classification for Tuning and Reweighting,* arXiv:1907.08209)
  1. Train calibrated NN classifier f(x) to distinguish FastSim from FullSim
  2. Define weight $w(x) = f(x) / (1 – f(x)) \approx p_{\text{FullSim}}(x) / p_{\text{FastSim}}(x)$
  3. Reweight FastSim by w(x)

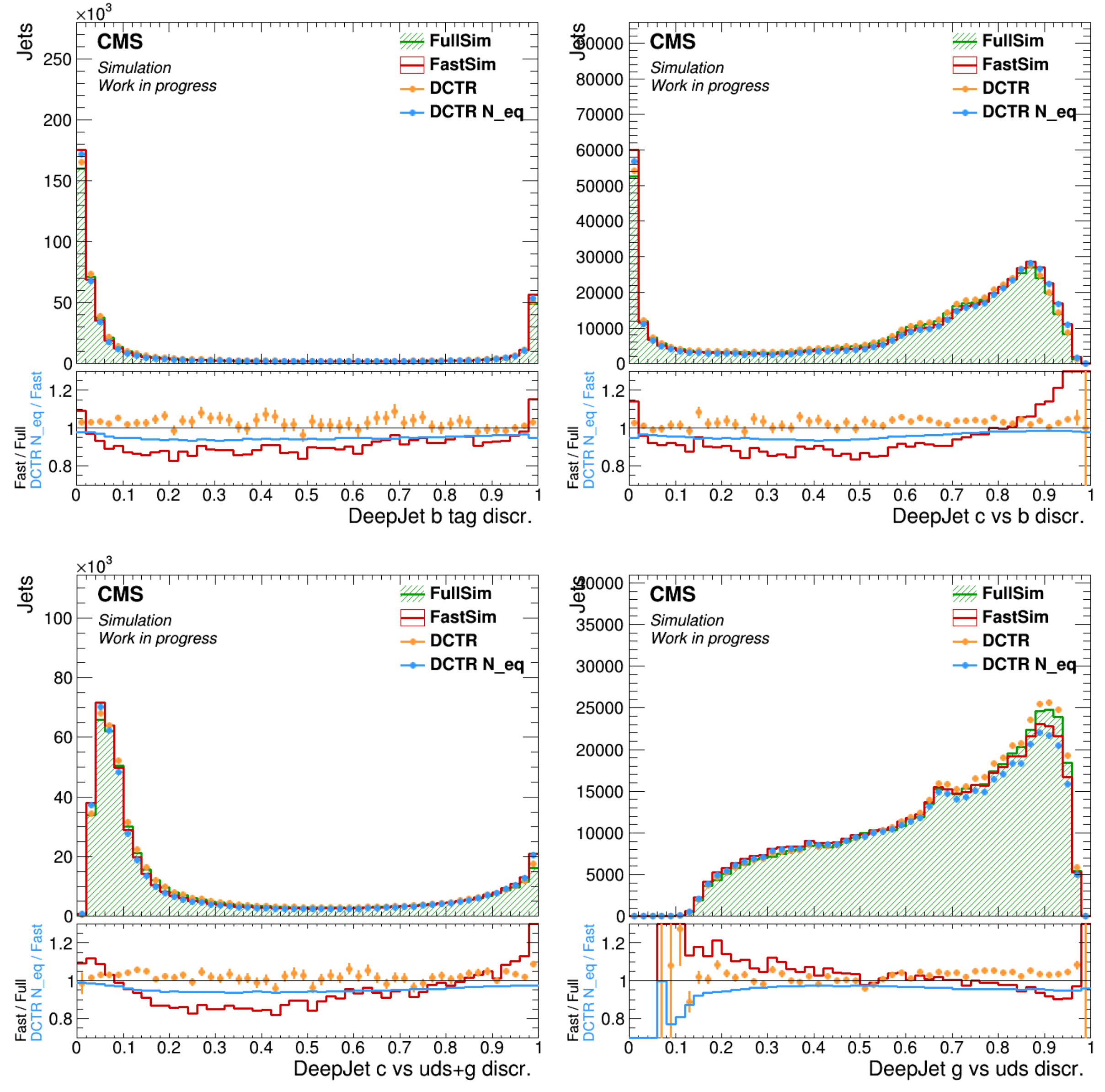


- Input **variables**: 4 DeepJet discriminator values
- GEN parameters: p_T^{GEN}, η^{GEN}, true hadron flavor
- Simple **fully-connected NN**:
  - 3 hidden layers with 64 nodes
  - Binary cross entropy loss
- Training with 1M jet triplets





REWEIGHTING (DCTR)



- **Good agreement** after reweighting
- Weights limit statistical power of FastSim

  → How to quantify?
- For a sample with N jets with weights $\sum_{i=1}^{N} w_i$
  - Error: $\delta Fast_{DCTR} = \sqrt{\sum_i w_i^2} = \sqrt{N * (RMS^2 + \overline{w}^2)}$

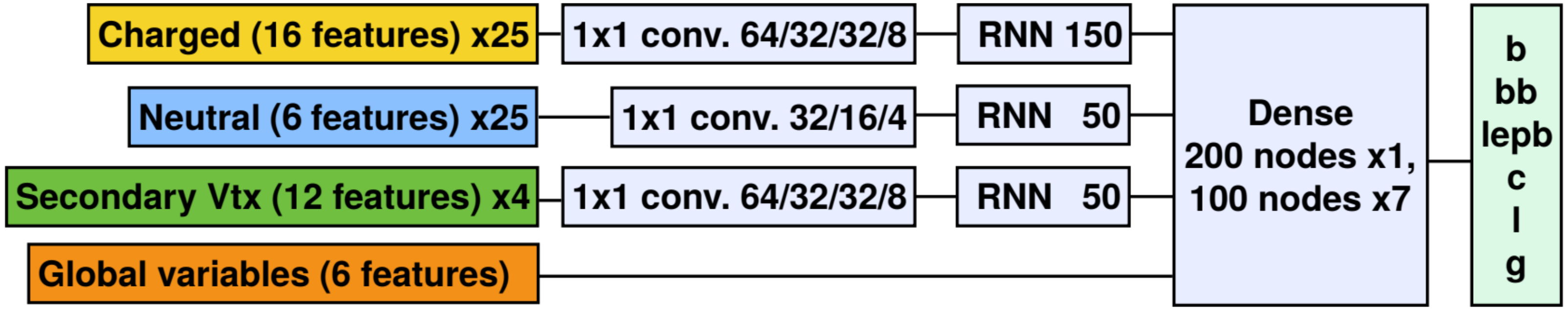
    with $RMS = \frac{1}{N} \sum_{i=1}^{N} (w_i - \overline{w})^2$
  - Equivalent events: $N_{eq} = \frac{(\sum_i w_i)^2}{\sum_i w_i^2} = \frac{N}{1+\left(\frac{RMS}{\overline{w}}\right)^2}$

    → N_{eq} events with weight=1 would have same relative statistical fluctuation

➢ Loss of **max. 5 – 10 %**

DEEPJET

- DeepJet architecture: 6 output nodes with **softmax** activation function → sum = 1 (arXiv:2008.10519)



- However, in NanoAOD: 4 (composite) discriminators:
  - btagDeepFlavB := b + bb + lepb
  - btagDeepFlavCvB := c / (c + b + bb + lepb)
  - btagDeepFlavCvL := c / (c + l + g)
  - btagDeepFlavQG := g / (g + l)

→ sum != 1