### INTRODUCTION

- Almost all 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 needed to stay within computing budget

- In **CMS**, two simulation chains (FullSim/FastSim) are used that produce output of same dimensionality/structure:

<table>
<thead>
<tr>
<th></th>
<th>GEN: Event generation</th>
<th>SIM: Detector simulation</th>
<th>DIGI: Digitization</th>
<th>RECO: Reconstruction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FullSim</strong></td>
<td>same e.g., MadGraph</td>
<td>GEANT4</td>
<td>same</td>
<td>analyze as if data</td>
</tr>
<tr>
<td><strong>FastSim</strong> ≈ 15% of sim. events</td>
<td>parametrized energy loss 100x faster</td>
<td>same</td>
<td>use GEN info 2.5x faster</td>
<td></td>
</tr>
</tbody>
</table>

- In total: FastSim ≈ 10x faster than FullSim

---

Refining fast simulation using machine learning – Moritz Wolf

---

[ CMS-NOTE-2022-008 ]

[ twiki.cern.ch/twiki/bin/view/CMSPublic/CMSSoftware/CMSOFFlineComputingResults ]
FastSim advantage in speed comes at the price of decreased accuracy in some of the final observables (N.B.: standard analyses use further processed data formats e.g., AOD or NanoAOD)

Aim: increase FastSim accuracy to promote its wider usage

Possible FastSim tuning approaches:

- Internal tuning of functions/parameters (within SIM/RECO)
- Post-hoc tuning (after NanoAOD)
  - Reweighting = defining weights for individual events/objects/… e.g., DCTR introduced in arXiv:1907.08209
  - Refining = changing (high-level) observables e.g., Wasserstein-GAN for air showers in arXiv:1802.03325
**INTRODUCTION**

- **FastSim advantage in speed** comes at the price of **decreased accuracy** in some of the final observables (N.B.: standard analyses use further processed data formats e.g., AOD or NanoAOD)

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

- Possible **FastSim tuning** approaches:
  - Internal tuning of functions/parameters (within SIM/RECO)
  - Post-hoc tuning (after NanoAOD)
    - **Reweighting** = defining weights for individual events/objects/… e.g., DCTR introduced in [arXiv:1907.08209](http://arxiv.org/abs/1907.08209)
    - **Refining** = changing (high-level) observables e.g., Wasserstein-GAN for air showers in [arXiv:1802.03325](http://arxiv.org/abs/1802.03325)
INTRODUCTION

Refining fast simulation using machine learning – Moritz Wolf
Focus on jet flavour tagging: 4 NanoAOD DeepJet discriminators: b+bb+lepb, c vs b+bb+lepb, c vs uds+g, g vs uds „B“ „CvB“ „CvL“ „QG“

(DeepJet NN softmax output nodes: b, bb, lepb, c, uds, g; arXiv:2008.10519)

FastSim/FullSim discrepancies O(10 %)

Training sample: SUSY simplified model „T1tttt“ simulated with FastSim and FullSim
(same GEN events, no pile-up)

Match jets using ΔR angular criterion
Jet triplets: (GEN, FastSim, FullSim)
REFINING (REGRESSION) — GENERAL IDEA

Employ regression neural network to refine FastSim:

- **Input**: FastSim variables $\mathbf{x}^{\text{Fast}} = 4$ DeepJet discriminators
  
  Parameters $\mathbf{y} = p_T^{\text{GEN}}, \eta^{\text{GEN}}$, true hadron flavor (b, c, or light quark/gluon)

- **Output**: Refined variables $\mathbf{x}^{\text{Refi.}} = 4$ DeepJet discriminators

- **Target**: FullSim variables $\mathbf{x}^{\text{Full}} = 4$ DeepJet discriminators

Refining fast simulation using machine learning – Moritz Wolf
**REFINING (REGRESSION) — NN ARCHITECTURE**

- **ResNet-like skip connections**: learn only residual corrections

- **Preprocessing**: transform input variables/parameters e.g., logit-transform $\text{logit}(x) = \ln \left( \frac{x}{1-x} \right)$ to map from $(0, 1)$ to $(-\infty, \infty)$

- **Postprocessing**: transform back & enforce DeepJet constraint (original DeepJet softmax output nodes need to sum to 1)
**REFINING (REGRESSION) — LOSS TERMS**

- **Primary loss:** MMD (distribution-based)
  - Comparing ensembles of jets not jet-jet pairs
  - To cope with independent stochasticity in both simulation chains

- **Additional loss:** MSE/Huber (output-target pair-based) \(\rightarrow\) correct for deterministic FastSim biases
  - Use Huber loss: \(l_n = \begin{cases} 0.5(x_n - y_n)^2, & \text{if } |x_n - y_n| < \delta \\ \delta |x_n - y_n| - 0.5 \delta, & \text{otherwise} \end{cases} \) (combination of MSE/MAE, less sensitive to outliers)

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

- **MMD:** Maximum Mean Discrepancy (ensembles)
- **MSE:** Mean Squared Error (jet-jet pairs)
- **MAE:** Mean Absolute Error (jet-jet pairs)

\[n = m = \text{batch size} = 4096\]

\[k: \text{Gaussian kernel (adaptive } \sigma)\]
**REFINING (REGRESSION) – LOSS TERMS**

Combine loss terms via **MDMM algorithm**:

- Reframe problem as **constrained optimization** using **Lagrangian**:
  \[ \mathcal{L} = f(\theta) - \lambda \cdot (\varepsilon - g(\theta)) \rightarrow \text{convergence mathematically formalized} \]
- Minimize \( f(\theta) \) (primary loss, „Loss #1“) subject to \( g(\theta) = \varepsilon \) (additional loss, „Loss #2“)
- Gradient descent for NN parameters \( \theta \), gradient ascent for Lagrange multiplier \( \lambda + \) damping

---

**Input**

\( \mathbf{x}^{\text{Fast}} \)

\( \mathbf{y} \)

**Preprocessing**

**4x Residual Block**

**Postprocessing**

\( \mathbf{x}^{\text{Refi.}} \)

**Target**

\( \mathbf{x}^{\text{Full}} \)

---

**REFINING (REGRESSION) — TRAINING**

- **Plot**: training convergence in plane of the two loss terms
  - Primary loss: MMD(Refined, FullSim) (horizontal axis)
  - Constraint: Huber(Refined, FullSim) (vertical axis)
- **No MDMM** (dash-dotted lines)
  - Only one loss or constant weighted addition
  - Convergence might not be optimal
- **With MDMM** (solid lines)
  - Scan of different $\epsilon$ values (horizontal dashed lines)
  - Convergence to desired point on Pareto front
  - Choose $\epsilon = 0.084$
REFINING (REGRESSION) – RESULTS

b tag binned by working point

- Loose: 10% mis-tag
- Medium: 1% mis-tag
- Tight: 0.1% mis-tag
Refining fast simulation using machine learning – Moritz Wolf

REFINING (REGRESSION) — RESULTS

Pearson correlation coefficients

relative difference to FullSim

FastSim

CMS Simulation Preliminary (13 TeV)

FullSim

CMS Simulation Preliminary (13 TeV)

FastSim Refined

CMS Simulation Preliminary (13 TeV)
SUMMARY & OUTLOOK

✓ ResNet-like regression NN can be used as **post-hoc refinement layer** to FastSim output
  - Considerably improved agreement with FullSim output
  - Improvement in correlations among output observables and external parameters

✓ Validation in TTbar events (different event topology)

✓ Corresponding pull request merged into CMSSW
  - Extend to other variables/objects (e.g., jet substructure)
  - Tune directly to data?