FlashSim

End-to-end event simulation with Deep Learning

Physics in the AI era, September 2024

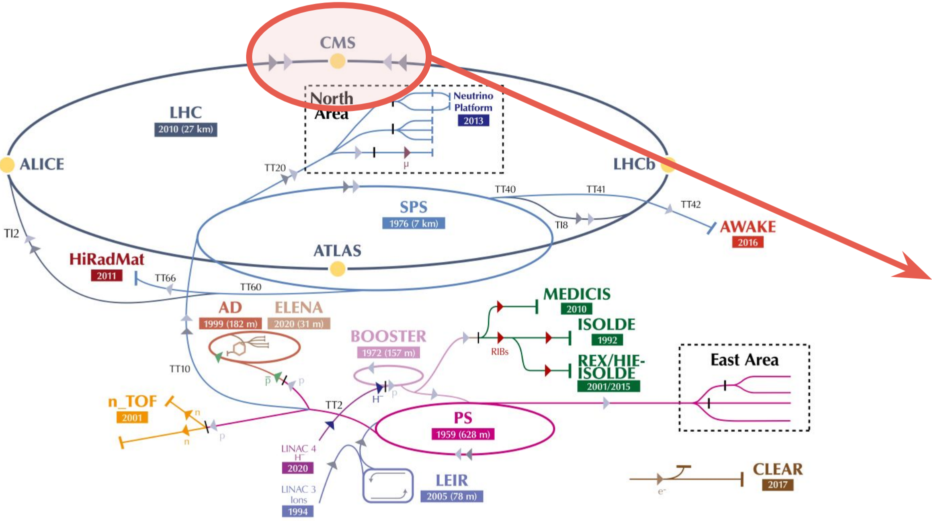
Filippo Cattafesta

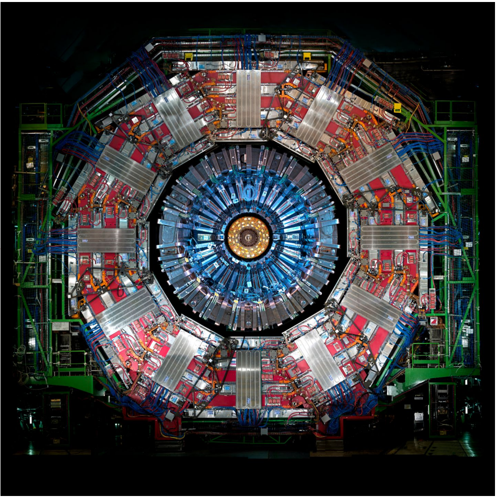
CMS
Compact Muon Solenoid

INFN
Istituto Nazionale di Fisica Nucleare

SCUOLA
NORMALE
SUPERIORE

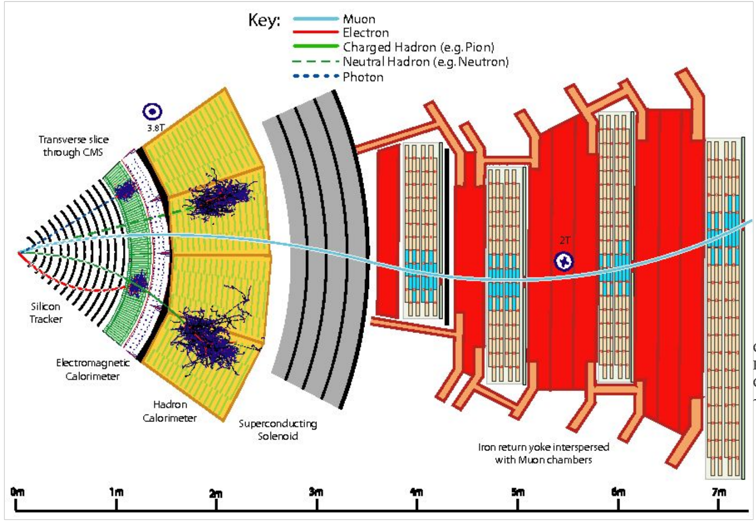
The Compact Muon Solenoid experiment

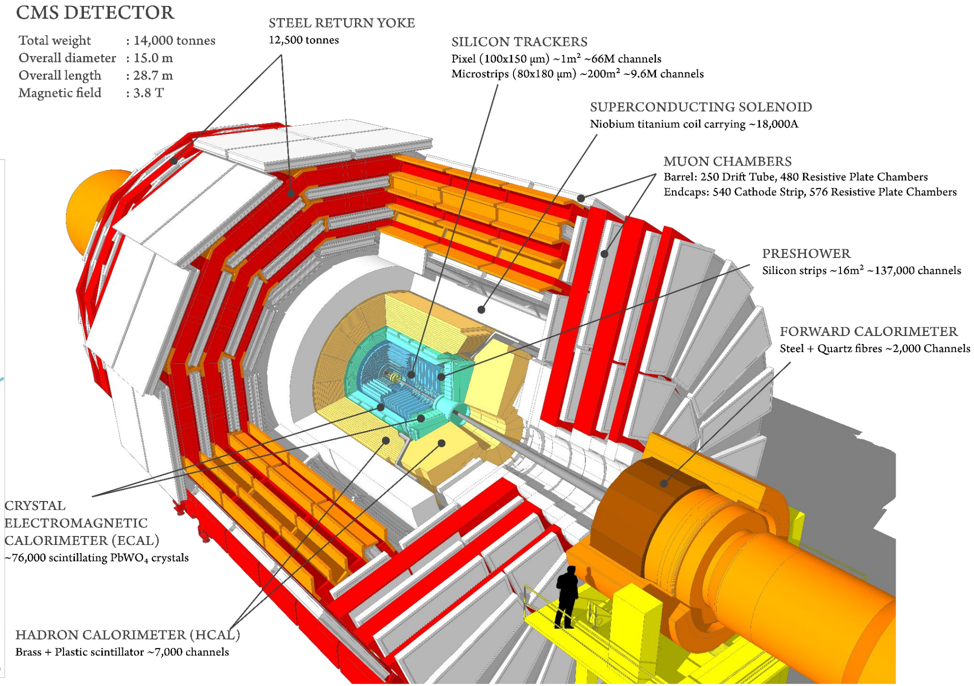




- General purpose experiment at the Large Hadron Collider

The CMS detector

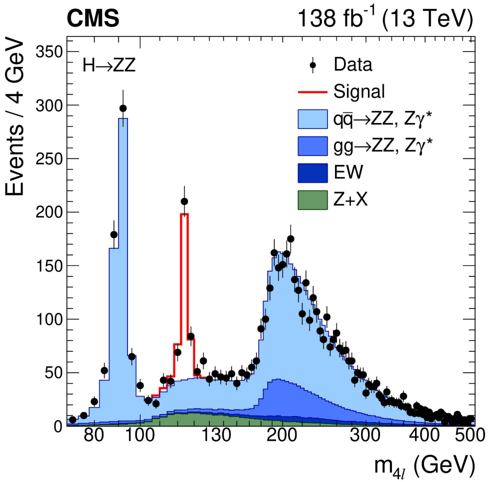

Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
3.8T
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
2T
Iron return yoke interspersed with Muon chambers
0m
1m
2m
3m
4m
5m
6m
7m

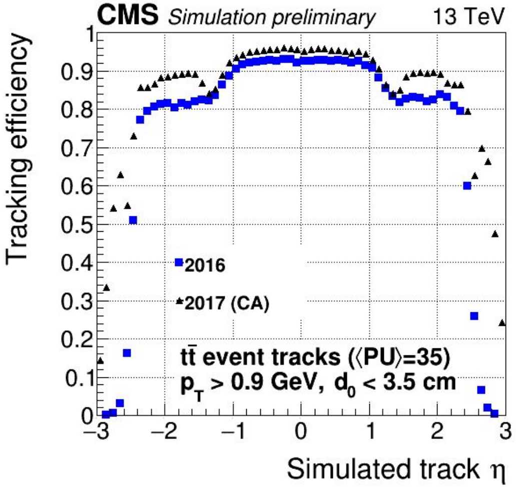

CMS DETECTOR
Total weight : 14,000 tonnes
Overall diameter : 15.0 m
Overall length : 28.7 m
Magnetic field : 3.8 T
STEEL RETURN YOKE
12,500 tonnes
SILICON TRACKERS
Pixel (100x150 μm) ~1m² ~66M channels
Microstrips (80x180 μm) ~200m² ~9.6M channels
SUPERCONDUCTING SOLENOID
Niobium titanium coil carrying ~18,000A
MUON CHAMBERS
Barrel: 250 Drift Tube, 480 Resistive Plate Chambers
Endcaps: 540 Cathode Strip, 576 Resistive Plate Chambers
PRESHOWER
Silicon strips ~16m² ~137,000 channels
FORWARD CALORIMETER
Steel + Quartz fibres ~2,000 Channels
CRYSTAL ELECTROMAGNETIC CALORIMETER (ECAL)
~76,000 scintillating PbWO₄ crystals
HADRON CALORIMETER (HCAL)
Brass + Plastic scintillator ~7,000 channels

- Multiple technologies to detect different particles
- Complex event reconstruction algorithms

Simulation is an indispensable tool

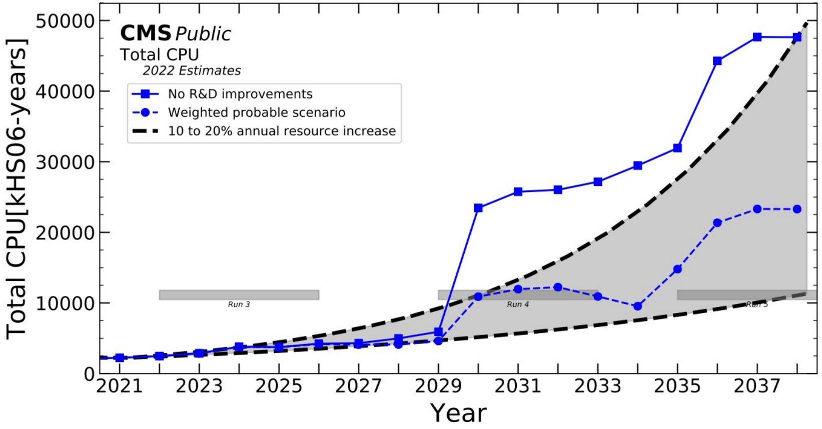
- "Event Simulation" is used:
  - to understand the **detector response** (particle-matter interaction)
  - to estimate the performances of **reconstruction algorithms**
  - in **analysis** (uncertainties, template fits, etc.)





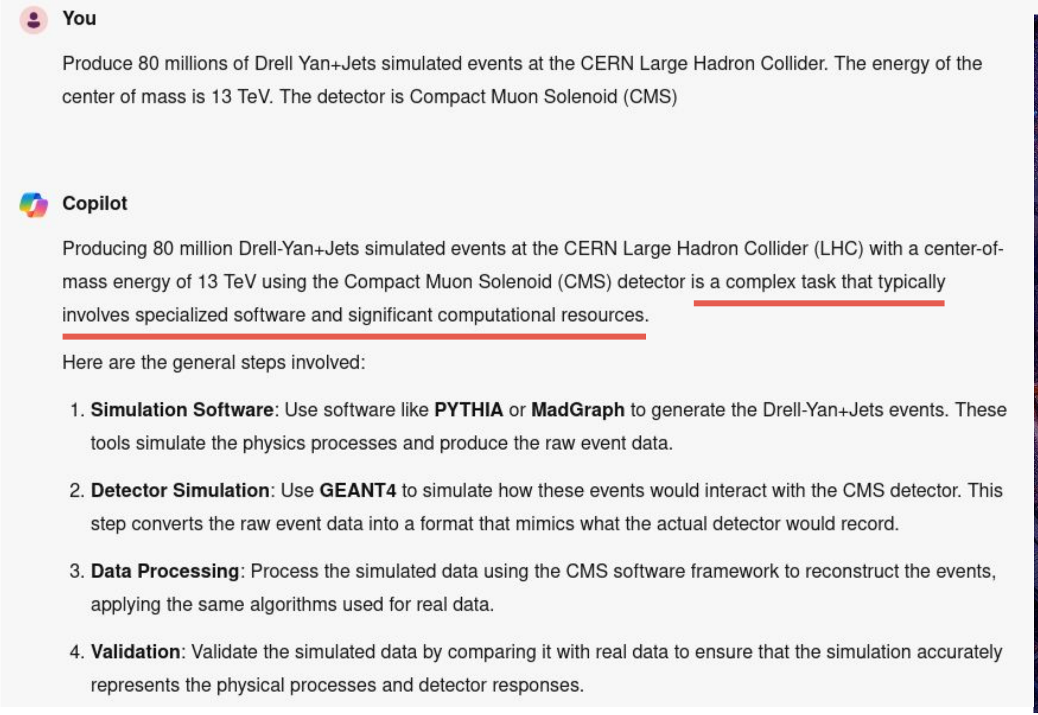
- Huge amount of computing resources needed

- Critical for *High Luminosity LHC* and *Phase 2* upgrade



We live in the AI era

For a summary: https://iml-wg.github.io/HEPML-LivingReview/



You

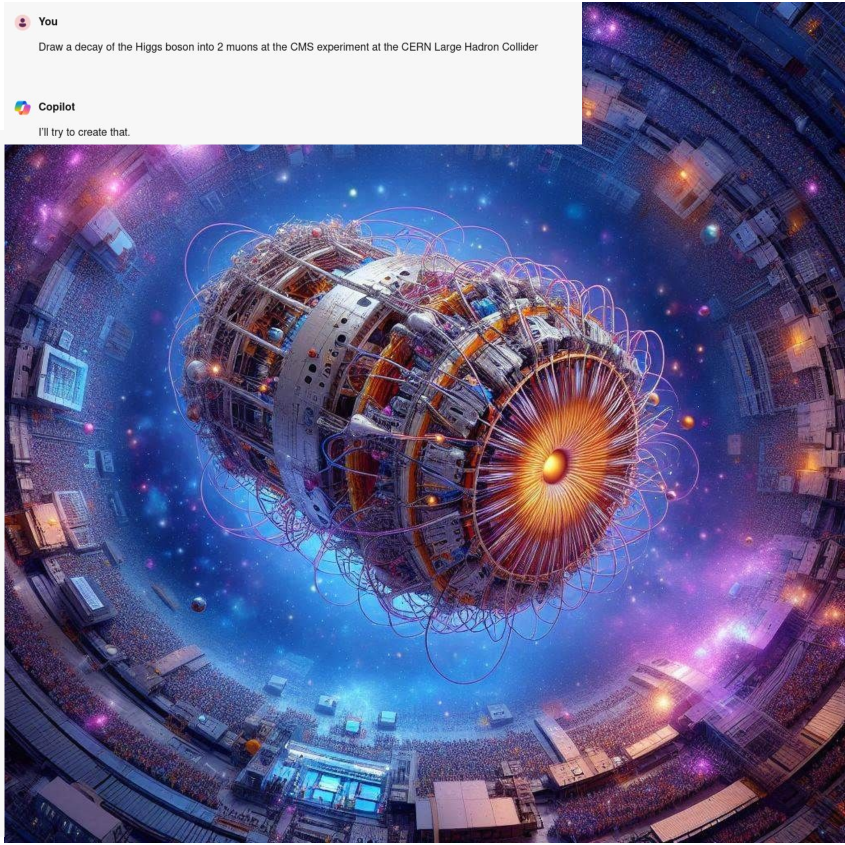
Produce 80 millions of Drell Yan+Jets simulated events at the CERN Large Hadron Collider. The energy of the center of mass is 13 TeV. The detector is Compact Muon Solenoid (CMS)

Copilot

Producing 80 million Drell-Yan+Jets simulated events at the CERN Large Hadron Collider (LHC) with a center-of-mass energy of 13 TeV using the Compact Muon Solenoid (CMS) detector is a complex task that typically involves specialized software and significant computational resources.

Here are the general steps involved:

1. **Simulation Software**: Use software like **PYTHIA** or **MadGraph** to generate the Drell-Yan+Jets events. These tools simulate the physics processes and produce the raw event data.
2. **Detector Simulation**: Use **GEANT4** to simulate how these events would interact with the CMS detector. This step converts the raw event data into a format that mimics what the actual detector would record.
3. **Data Processing**: Process the simulated data using the CMS software framework to reconstruct the events, applying the same algorithms used for real data.
4. **Validation**: Validate the simulated data by comparing it with real data to ensure that the simulation accurately represents the physical processes and detector responses.



You

Draw a decay of the Higgs boson into 2 muons at the CMS experiment at the CERN Large Hadron Collider

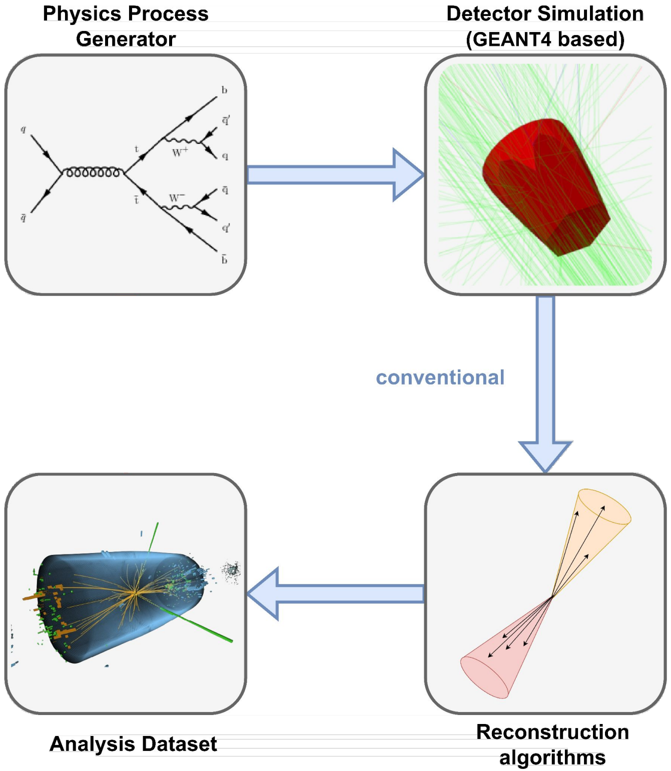
Copilot

I'll try to create that.

“Conventional” Simulation

- FullSim
    - **Generation**: production of particles using theoretical calculations (e.g. MadGraph)
    - **Detector simulation**: propagation through each element of the detector (GEANT4)
    - **Digitization** of the energy deposits and **reconstruction algorithms**
    - **Data processing** to build different data formats

~50% of available CPUs used for these steps (CMS)



Physics Process Generator

Detector Simulation (GEANT4 based)

conventional

Reconstruction algorithms

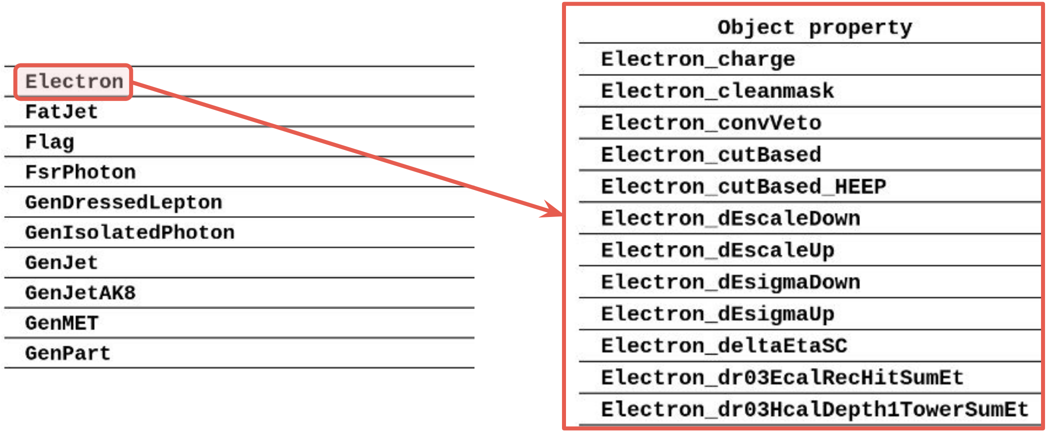
Analysis Dataset

From 2402.13684

CMS FlashSim

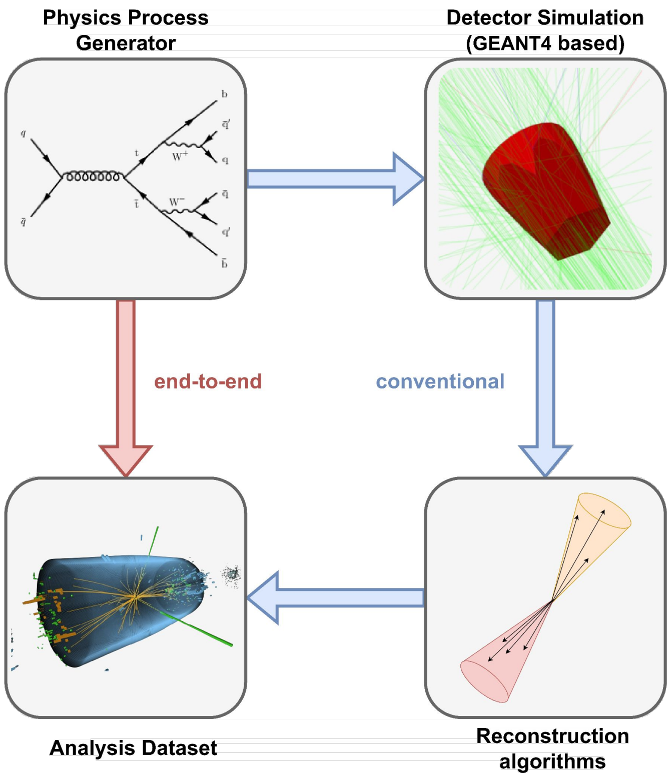
FlashSim — Universal very fast ML-based end-to-end simulation

- → targeting **directly** analysis-ready high-level variables (NANOAOD)
- → using **state-of-the-art** generative models
- → simulation speed ~**100 Hz**
- → analysis and sample **independent**



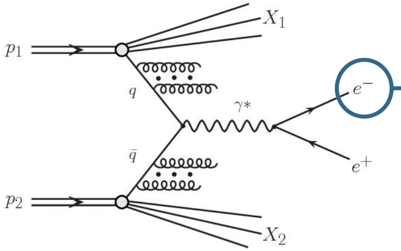
Electron
FatJet
Flag
FsrPhoton
GenDressedLepton
GenIsolatedPhoton
GenJet
GenJetAK8
GenMET
GenPart

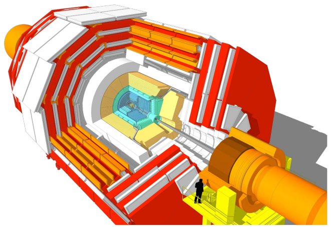
Object property
Electron_charge
Electron_cleanmask
Electron_convVeto
Electron_cutBased
Electron_cutBased_HEEP
Electron_dEscaleDown
Electron_dEscaleUp
Electron_dEsigmaDown
Electron_dEsigmaUp
Electron_deltaEtaSC
Electron_dr03EcalRecHitSumEt
Electron_dr03HcalDepth1TowerSumEt

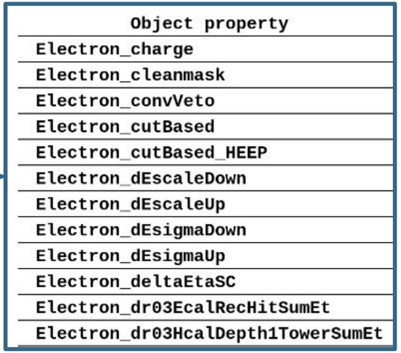

Physics Process Generator
Detector Simulation (GEANT4 based)
end-to-end
conventional
Analysis Dataset
Reconstruction algorithms

Conditioned detector response

- The goal is to learn a universal detector response
  - we must consider all the **information correlated to the reconstruction**







Object property
Electron_charge
Electron_cleanmask
Electron_convVeto
Electron_cutBased
Electron_cutBased_HEEP
Electron_dEscaleDown
Electron_dEscaleUp
Electron_dEsigmaDown
Electron_dEsigmaUp
Electron_deltaEtaSC
Electron_dr03EcalRecHitSumEt
Electron_dr03HcalDepth1TowerSumEt

Generator-level Electron

Reconstructed Electron (NANOAOD)

Output *pdf*

$$\boxed{P(\boldsymbol{x} \mid conditioning)}$$

Electron p_T,η,ϕ, …

Gen-level Electron p_T,η,ϕ, …

Multiple objects simulation

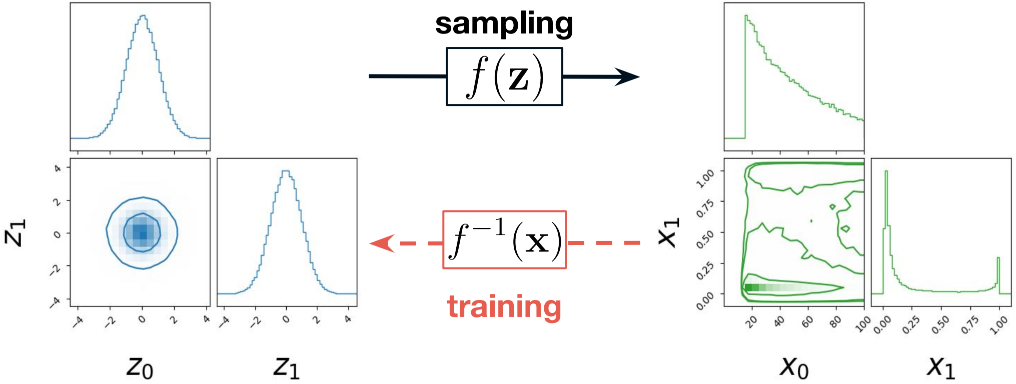
- Single model for each object
  - trained on existing FullSim dataset
  - smaller models (~1M parameters)
  - more control on the physical information used as conditioning

- We must consider all possible sources
  - because of errors and pileup, *fake objects* are reconstructed
  - e.g. electrons originated from energy deposits of particle jets

Physics objects	Sources (one NN model for each source)			Number of simulated attributes per object
Jets	Generator Jet	Fake from PU		39
Muons	Generator Muons	Fake from Jets/PU	Duplicates	53
Electrons	Generator Electrons	Generator Photons (prompt)	Fake from Jets/PU	48
Photons	Generator Photons (prompt)	Generator Electrons	Fake from Jets/PU	22
MET	GenMET and HT			25
FatJets	Generator AK8 Jets			53
SubJets	Generator AK8 SubJets			13
Tau	Reconstructed Jets with a Tau	RecoJets without a Tau		27
Secondary Vertices	Jets with Heavy Flavour	Light Jets	Taus	16
Non MET scalars (e.g. PV)	Various event level inputs			16
FSRPhotons	GenMuon/RecoMuon			6

Normalizing Flows as backbone

- We can get new samples from a complex multi-dimensional distribution starting from Gaussian noise
- Achieved by applying an **invertible transformation** to the Gaussian samples
- We learn the inverse transformation during the training process



$$\begin{cases} \mathbf{x} = f(\mathbf{z}) \\ p_x(\mathbf{x}) = p_z(\mathbf{z}) \det \left| \dfrac{d\mathbf{z}}{d\mathbf{x}} \right| \end{cases}$$

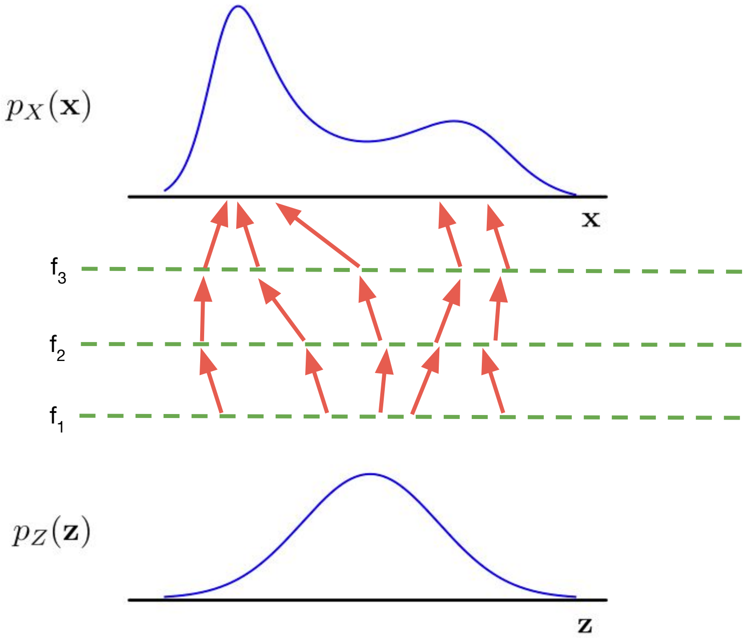
https://arxiv.org/abs/1912.02762

“Discrete” Flows

Build an (efficient) invertible transformation is not easy

Composition of **simple transformations**, correlated so that the jacobian is tractable

Affine transform: $\tau(z_i; \boldsymbol{h}_i) = \alpha_i z_i + \beta_i$



Adapted from https://ehoogeboom.github.io/post/en_flows/

Continuous Flows (and Flow Matching)

Continuous transformation (t∈[0, 1])

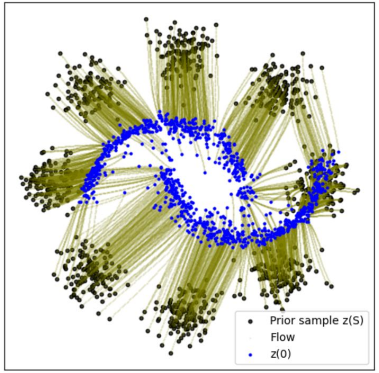
$$f(0; z) = z = Gaussian$$

$$f(1; z) = \text{target p.d.f.}$$

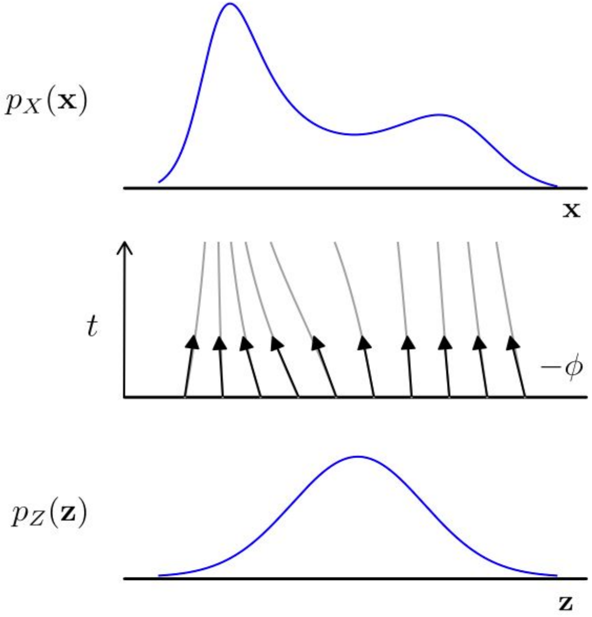
$$f(t + dt) = f(t) + v(t) \cdot dt$$

$$f(t + dt) = f(t) + DNN(f(t)) \cdot dt$$

Thanks to *Flow Matching*, we can learn the vector field v_t



From https://github.com/atong01/conditional-flow-matching

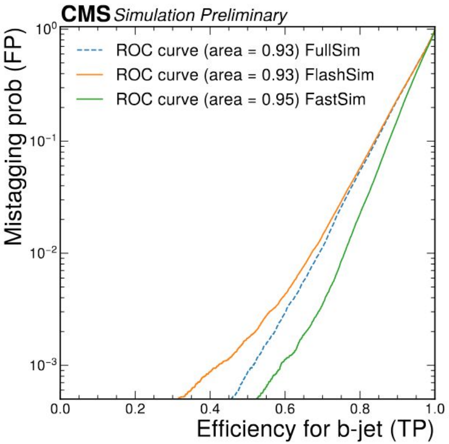


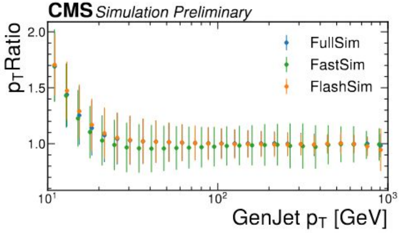
https://arxiv.org/abs/2210.02747 and https://arxiv.org/abs/2302.00482

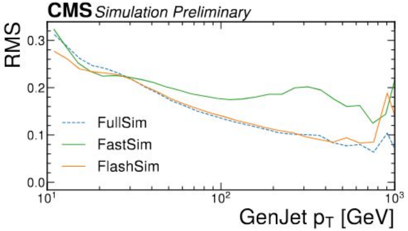
Object-level results: Jets

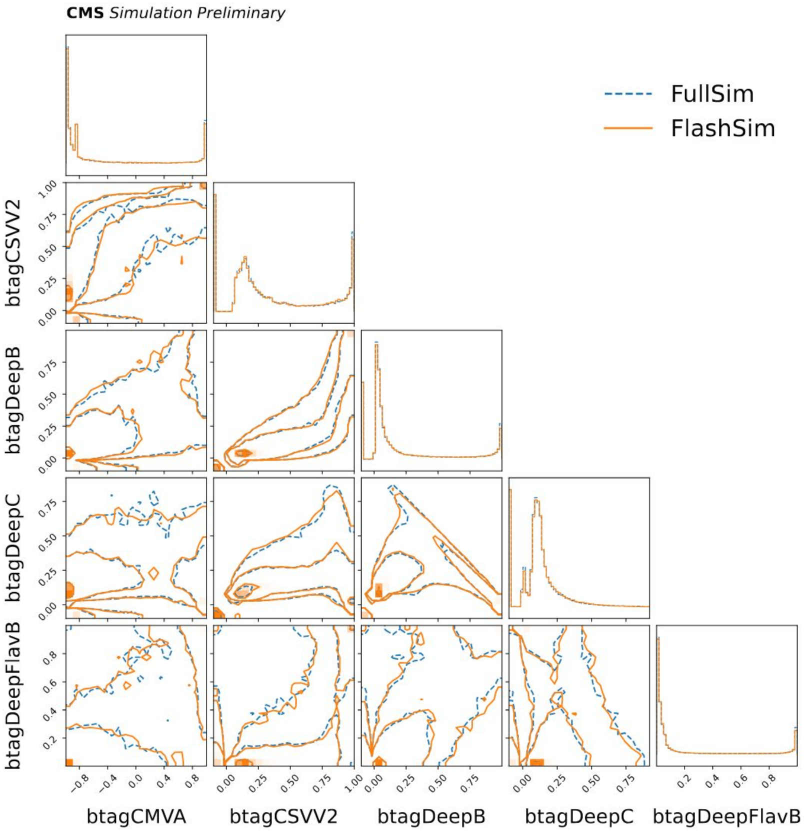
See CMS NOTE 2023 003

- **2D correlations** on b-tag scores
  - Deep Learning based discriminators
- Output correctly **influenced by the conditioning**
  - b-tag ROC compatible with FullSim
  - Reconstructed p_T (and resolution) as a function of the starting GenJet p_T

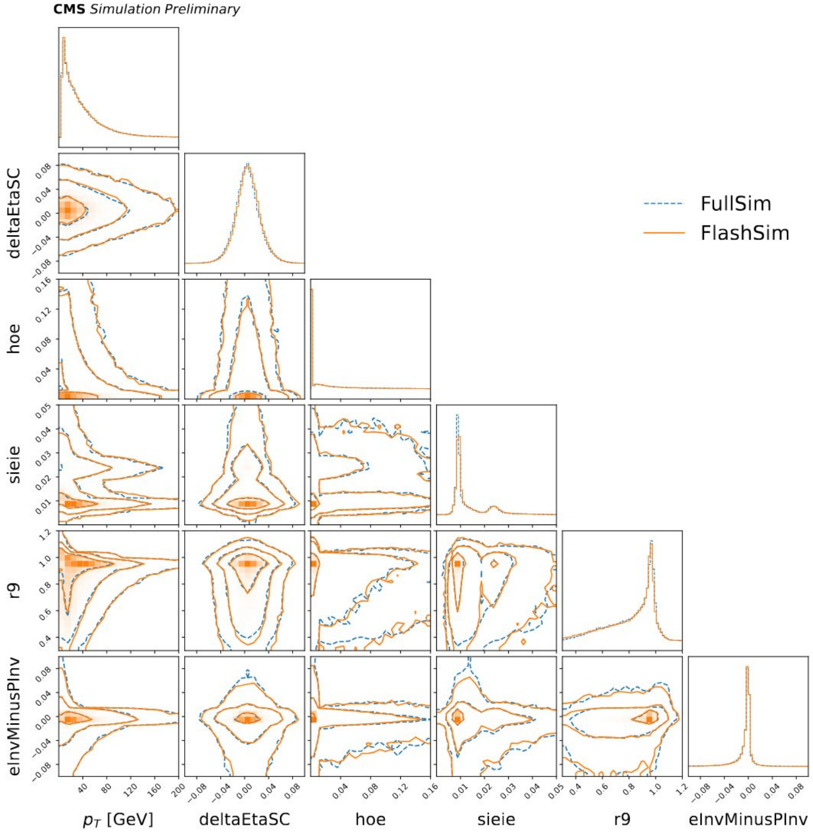


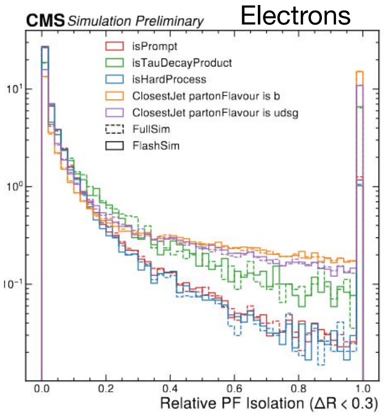


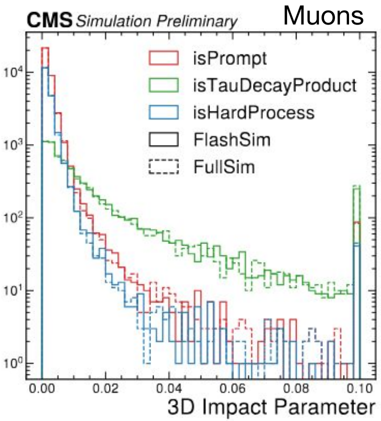




Results (object-level): Leptons



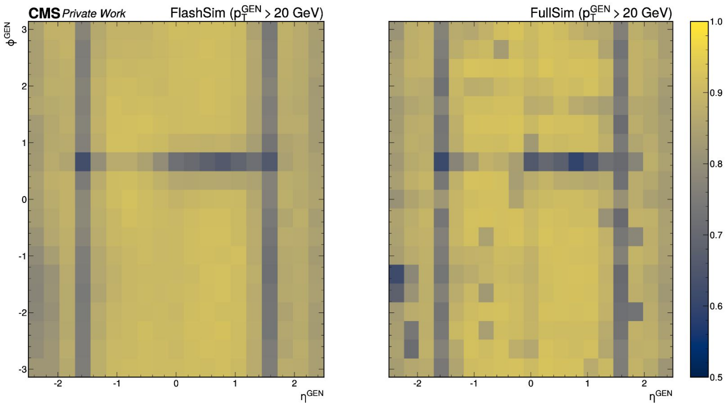




- Good correlations between variables associated with the energy deposit in the EM calorimeter
- Isolation and Impact Parameter distributions change with different conditioning properties

Event simulation

- We need to **merge all the models**
    - efficient I/O and model evaluation
- We must also consider the probability of an input object to be reconstructed (**Efficiency**)
    - e.g. "*Probability of a Generator-level Electron to be reconstructed as an Electron*"
    - Estimated using Neural Networks (MLP, ResNet)
- Remarkable event simulation rate
    - compared to FullSim 0.05 Hz

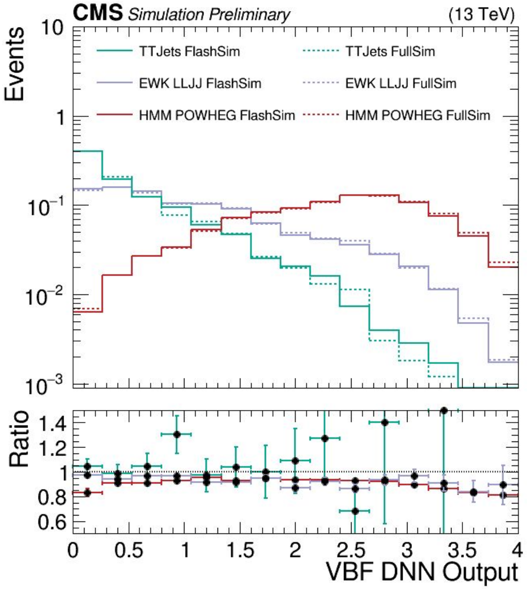


Number of samples of t [slide]

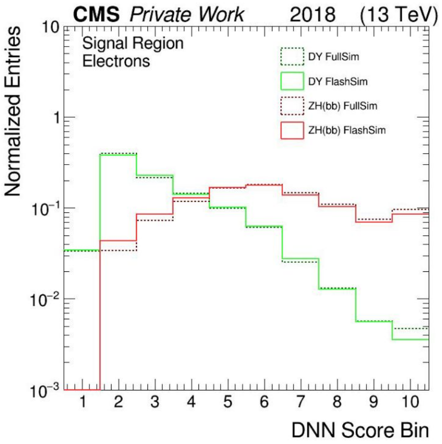
Processor	ODE accuracy (timesteps)	Event simulation rate
GPU 3050	100	325 Hz
GPU 3050	20	690 Hz
CPU 1-core	100	15 Hz
CPU 1-core	20	60 Hz
CPU 4-core	20	120 Hz

Analysis-level validation

- Important to validate the event simulation in pseudo-analysis
  - Higgs boson decay into muons [https://arxiv.org/pdf/2009.04363]
  - Higgs boson decay into b-quarks [https://arxiv.org/pdf/1808.08242]

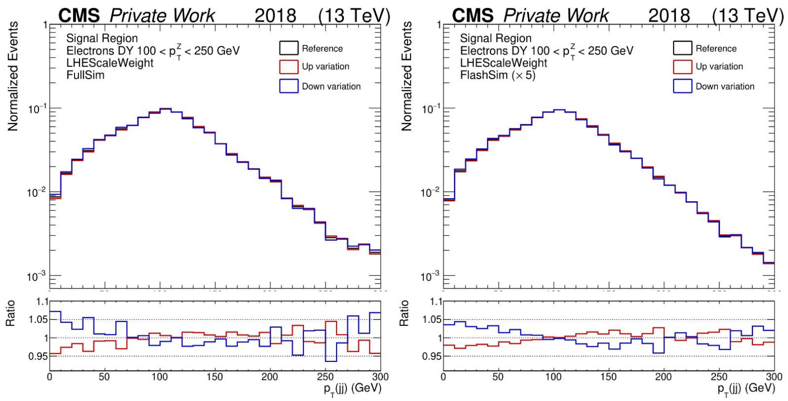


- Target accuracy within 10-20% (typical Data-Simulation agreement)

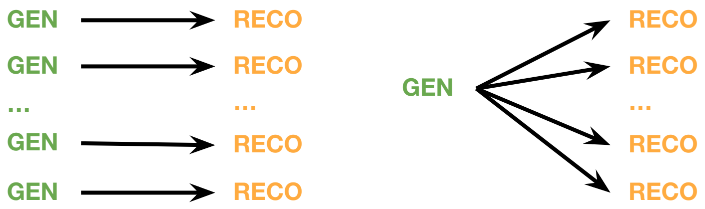


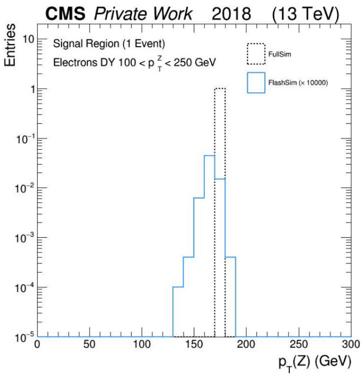
Implications and applications

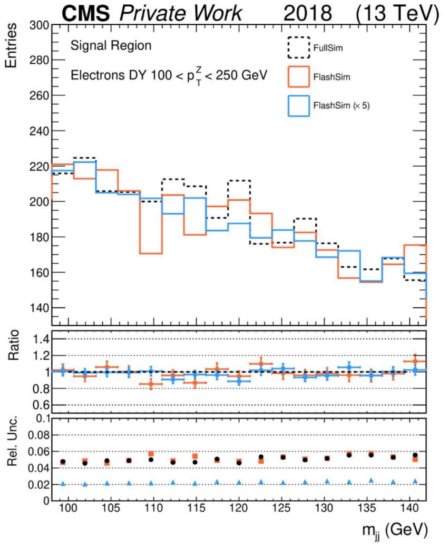
- We can produce a large number of events in a fraction of the time
  - more precise estimate of systematic uncertainties
  - more events in high rejection regions



- If the “generator” is slow, we can use **oversampling**
  - because of the stochastic detector response, we can simulate more reco events starting from the same GEN
  - we must consider the correlation







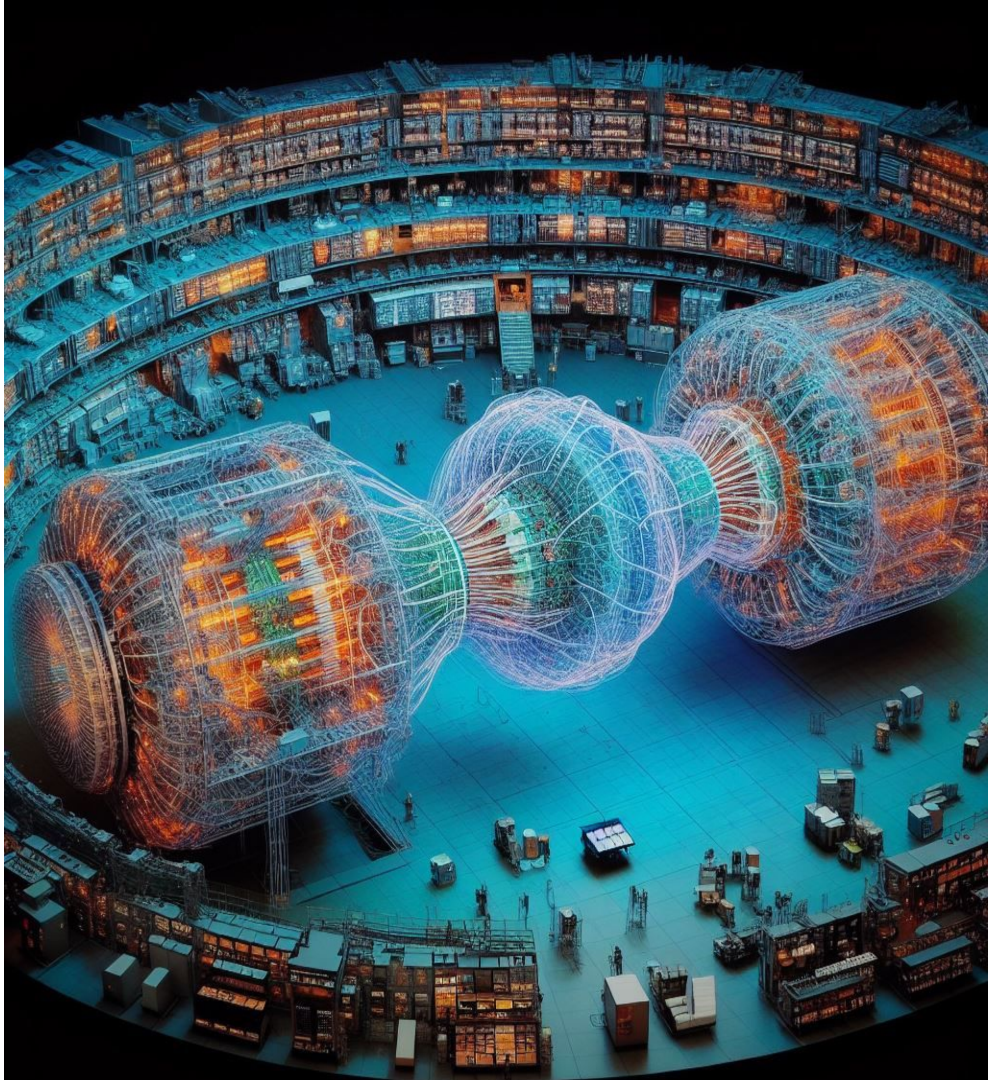
Future Developments

- We are working on a complete framework
    - most recent results under approval (talk at CHEP 2024)
- Testing the framework in real analysis scenario
    - to check the performances and the limitations
- Implement production at HPC nodes
    - large availability of GPUs can be beneficial

Conclusions

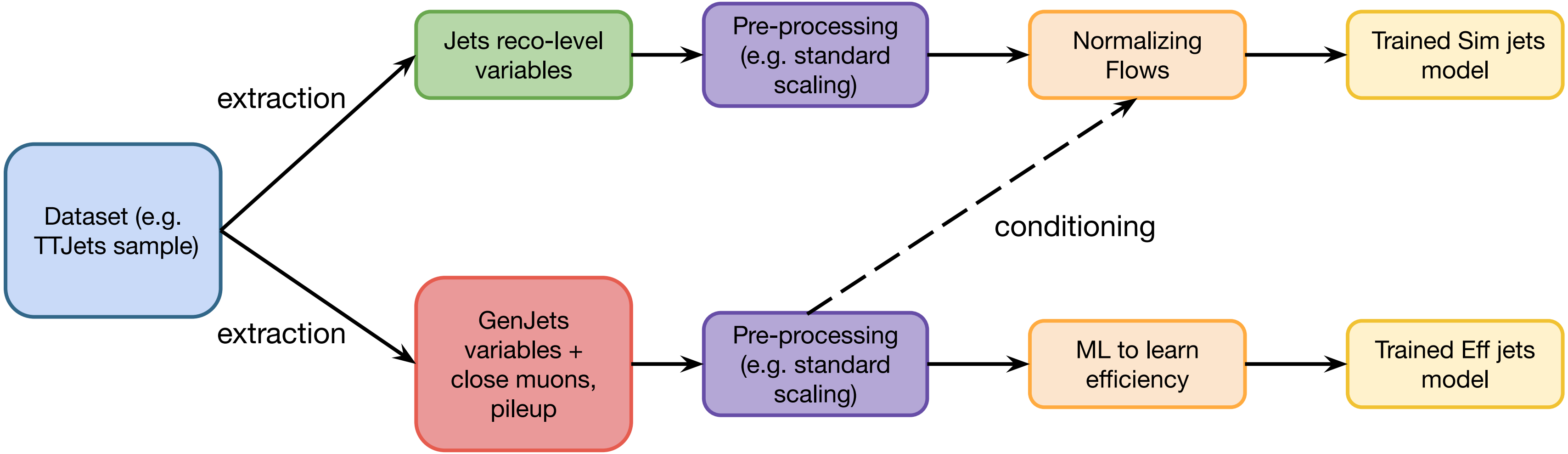
- End-to-end event simulation is being developed in CMS
  - using Normalizing Flows
  - much faster than FullSim and with acceptable accuracy

✉ filippo.cattafesta@sns.it

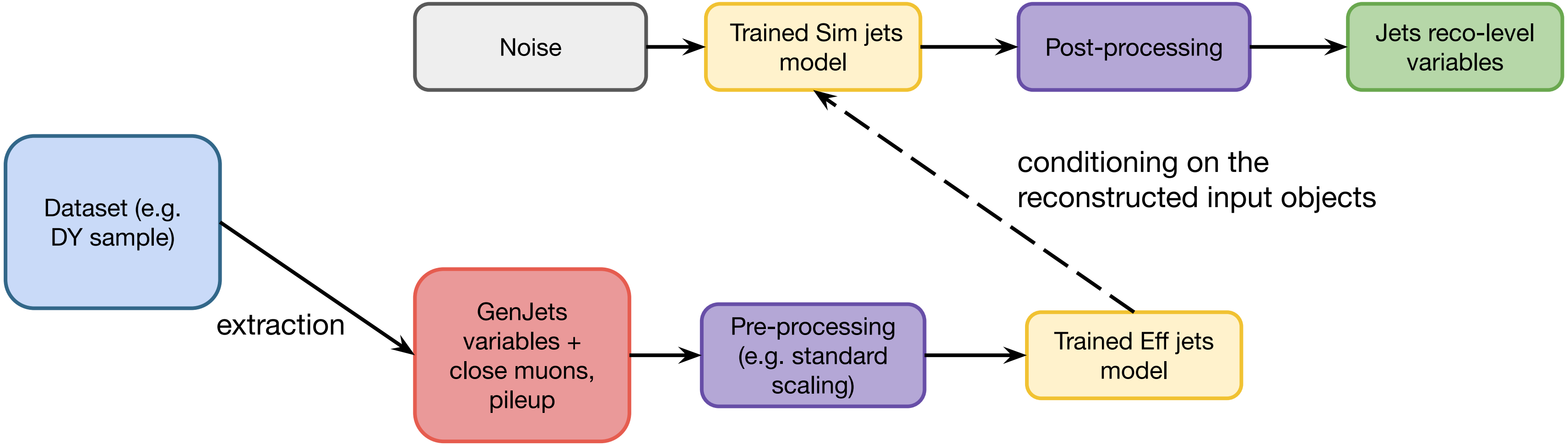


Backup

The basic idea – Training


Jets reco-level variables
Pre-processing (e.g. standard scaling)
Normalizing Flows
Trained Sim jets model
extraction
Dataset (e.g. TTJets sample)
conditioning
extraction
GenJets variables + close muons, pileup
Pre-processing (e.g. standard scaling)
ML to learn efficiency
Trained Eff jets model

The basic idea – Simulation



Noise

Trained Sim jets model

Post-processing

Jets reco-level variables

Dataset (e.g. DY sample)

extraction

GenJets variables + close muons, pileup

Pre-processing (e.g. standard scaling)

Trained Eff jets model

conditioning on the reconstructed input objects

Flow Matching as a solution

see https://arxiv.org/abs/2210.02747
and https://arxiv.org/abs/2302.00482

t=0 - - - - - - - - → p(*z*) = N(0,1)

t=1 - - - - - - - - → p(*z*) = N(*x*, $\sigma_{\min}$)

Learn vector field *u*, approximation of *v*

u is the field going from noise to data
under a Gaussian assumption

$$p_t(z|x) = \mathcal{N}(z|tx, (t\sigma_{\min} - t + 1)^2),$$

$$u_t(z|x) = \frac{x - (1 - \sigma_{\min})z}{1 - (1 - \sigma_{\min})t},$$

y = NN(*x*)
Loss = (*u* - y)**2
Simple regression!

Differential Scale Factors for systematic variations

VAR.(FullSim) = REF.(FullSim) × RATIO(FlashSim)

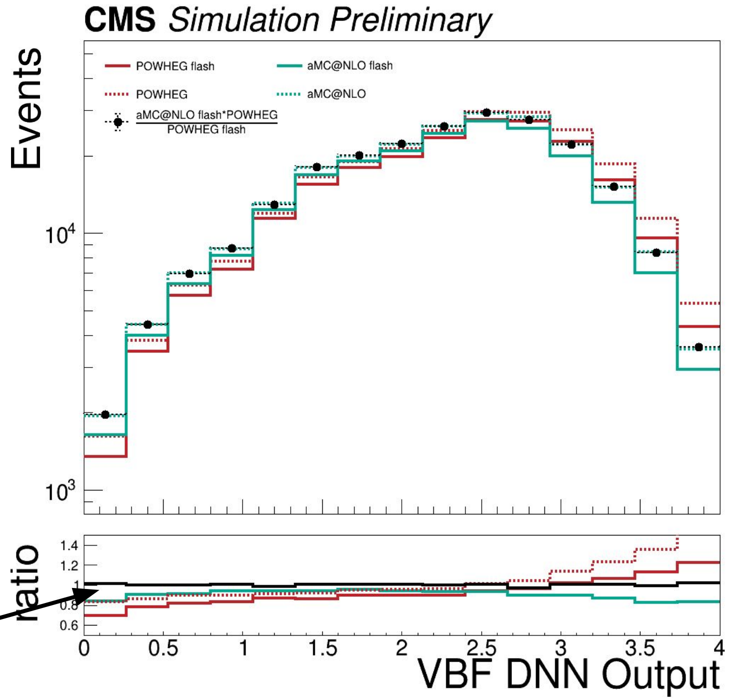
Variation at FullSim accuracy

FlashSim to FullSim differences cancel out

Given a (FullSim) sample made from a generator (e.g. POWHEG)

→ we can get a sample from another generator (e.g. aMC@NLO)
  - with **FullSim accuracy**
  - **in very short time!**

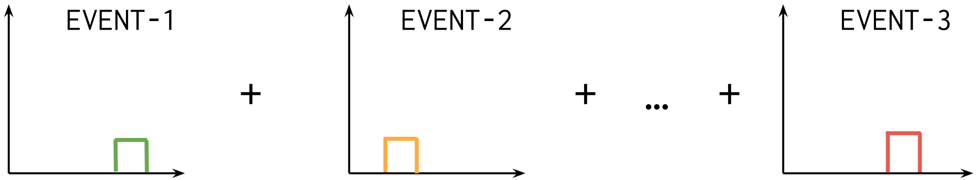
$$\text{aMC@NLO(FullSim)} = \text{POWHEG(FullSim)} \times \frac{\text{aMC@NLO(FlashSim)}}{\text{POWHEG(FlashSim)}}$$



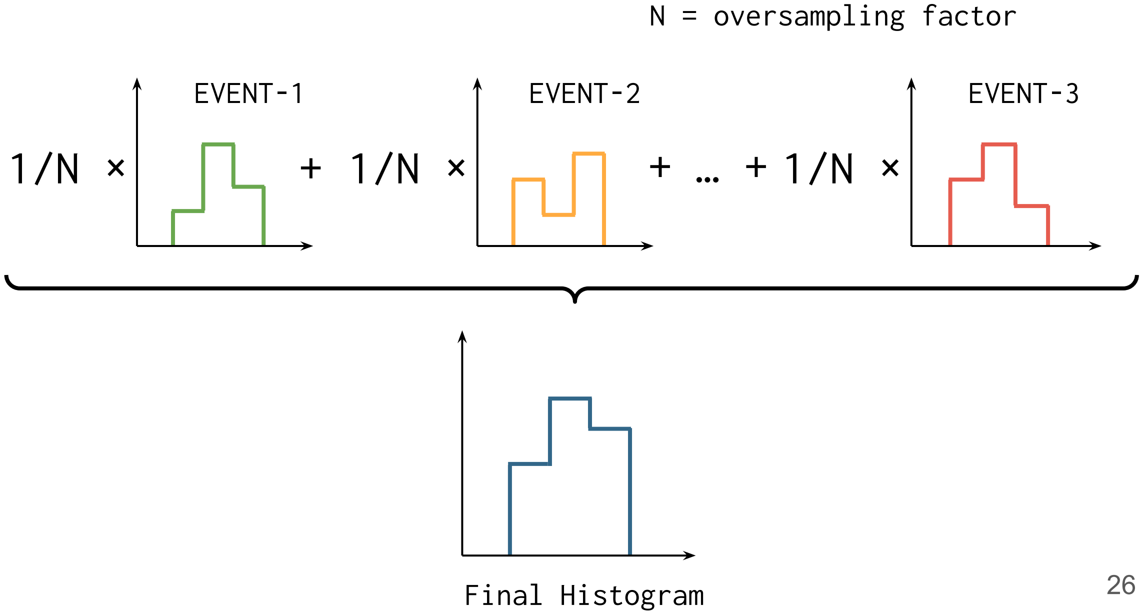
True FullSim aMC@NLO sample for the comparison

Oversampling: statistical treatment

Usually, a histogram is filled with events (and their weights)



Oversampling → the final histogram is given by the weighted sum of *sub-histograms* filled with the **distributions of events sharing the same GEN**



Note: the final uncertainty is larger than just calling `TH1::Fill()`

Oversampling

Prototype `OversampledTH` within ROOT RDataFrame (**to be improved!**) (https://github.com/cms-flashsim/OversampledTH)

- Non-oversampled case
  - w statistical weight associated with the MC event
  - For the *i*-th bin of an histogram, the probability of being in this bin and the associated uncertainty are

$$p_i = \frac{\sum_{j \in \text{bin}} w_j}{\sum_{k \in \text{sample}} w_k} \qquad \sigma_i = \frac{\sqrt{\sum_{j \in \text{bin}} w_j^2}}{\sum_{k \in \text{sample}} w_k}$$

- Oversampled case
  - A *fold* is the set of RECO events sharing the same GEN

$$p_i = \frac{\sum_{j \in \text{bin}} \sum_{l \in \text{fold} \in \text{bin}} w_{jl}}{N \sum_{k \in \text{sample}} w_k} = \frac{\sum_{j \in \text{bin}} \sum_{l \in \text{fold} \in \text{bin}} w_{jl}/N}{\sum_{k \in \text{sample}} w_k} \equiv \frac{\sum_{j \in \text{bin}} w_j p_j^{\text{fold}}}{\sum_{k \in \text{sample}} w_k}$$

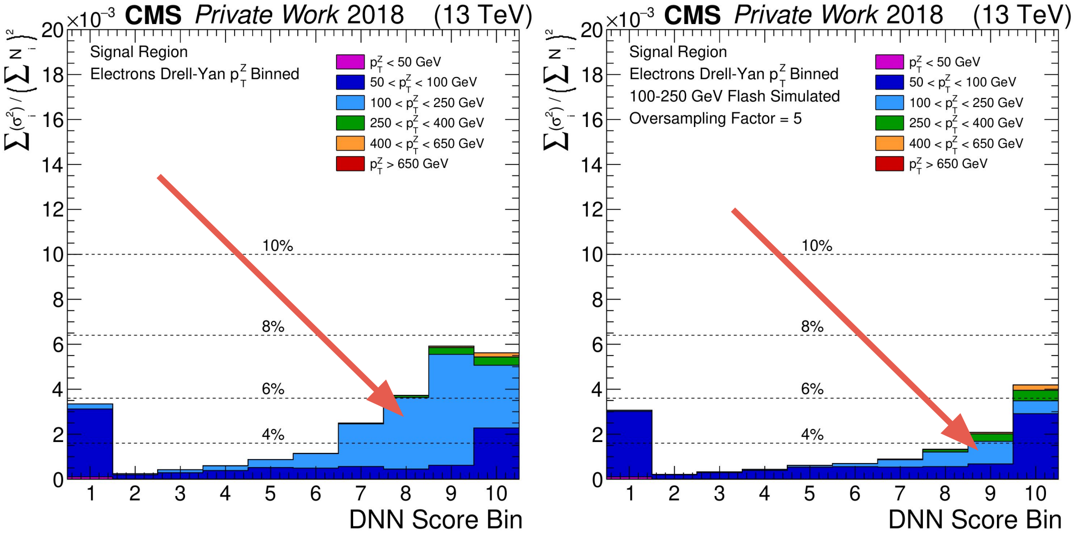
$$\sigma_i = \frac{\sqrt{\sum_{j \in \text{bin}} (w_j p_j^{\text{fold}})^2}}{\sum_{k \in \text{sample}} w_k}$$

Statistical uncertainty reduction $Z(\ell^+\ell^-)H(bb)$

DY+jets with $100 < p_T(Z) < 250$ GeV **oversampled** by a factor 5

~80M ×5 simulated events

□ **Statistical uncertainty is reduced by a factor 2-3 in the signal-enriched region**



without oversampling

oversampling ×5

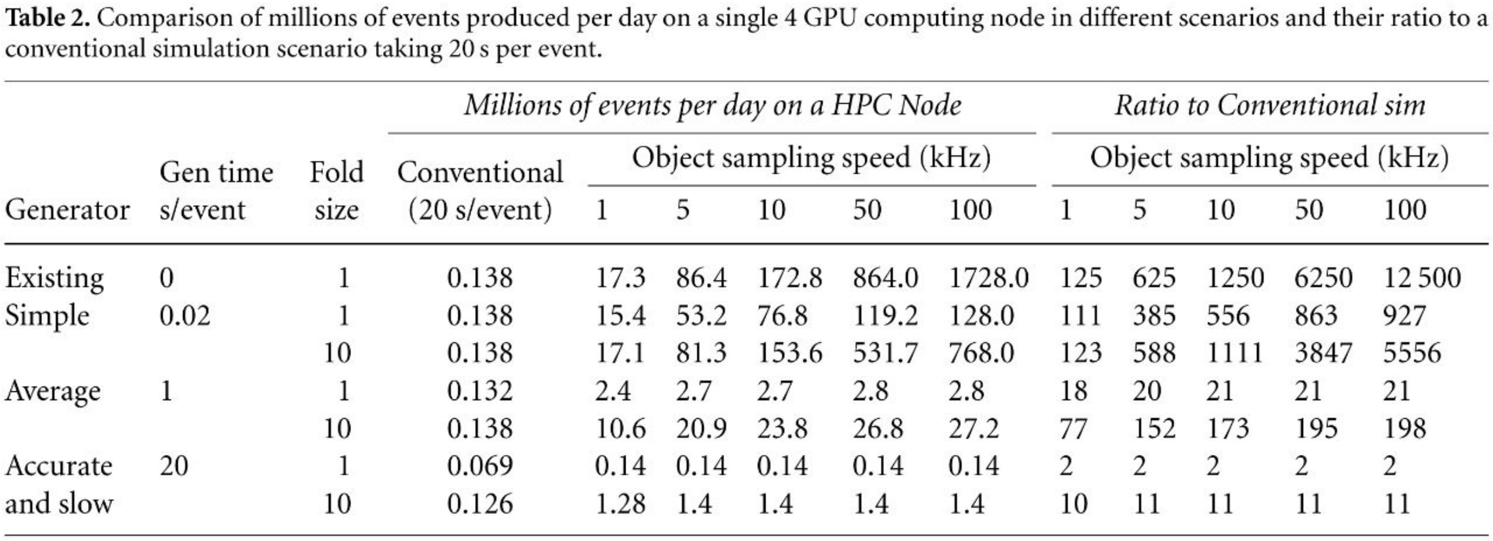


Table 2. Comparison of millions of events produced per day on a single 4 GPU computing node in different scenarios and their ratio to a conventional simulation scenario taking 20 s per event.

			Millions of events per day on a HPC Node						*Ratio to Conventional sim*				
				Object sampling speed (kHz)					Object sampling speed (kHz)				
Generator	Gen time s/event	Fold size	Conventional (20 s/event)	1	5	10	50	100	1	5	10	50	100
Existing	0	1	0.138	17.3	86.4	172.8	864.0	1728.0	125	625	1250	6250	12 500
Simple	0.02	1	0.138	15.4	53.2	76.8	119.2	128.0	111	385	556	863	927
		10	0.138	17.1	81.3	153.6	531.7	768.0	123	588	1111	3847	5556
Average	1	1	0.132	2.4	2.7	2.7	2.8	2.8	18	20	21	21	21
		10	0.138	10.6	20.9	23.8	26.8	27.2	77	152	173	195	198
Accurate and slow	20	1	0.069	0.14	0.14	0.14	0.14	0.14	2	2	2	2	2
		10	0.126	1.28	1.4	1.4	1.4	1.4	10	11	11	11	11