CMS FlashSim:

how ML powers end-to-end simulation in HEP

Istituto Nazionale di Fisica Nucleare

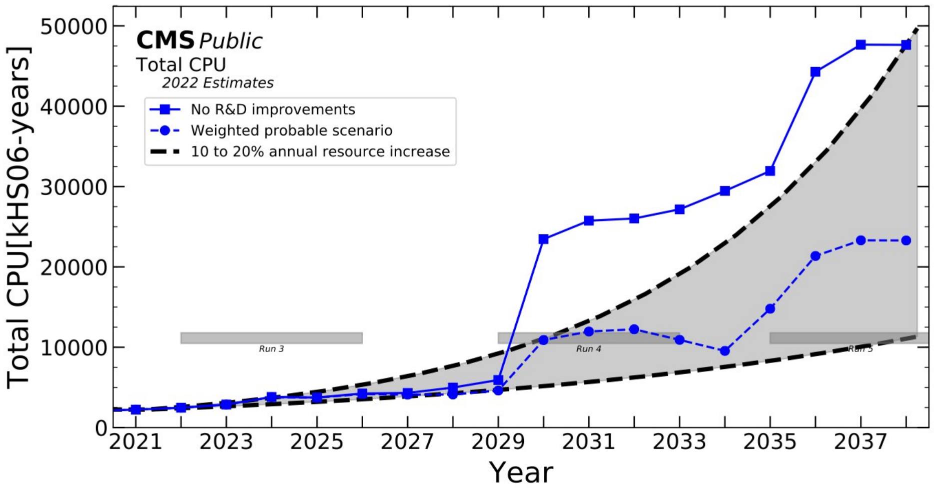
SCUOLA NORMALE SUPERIORE

Francesco Vaselli
On behalf of the CMS Collaboration

The future demands for simulations pose new challenges

Already in Run3, some analyses are limited by the statistical uncertainties due to limited simulated samples

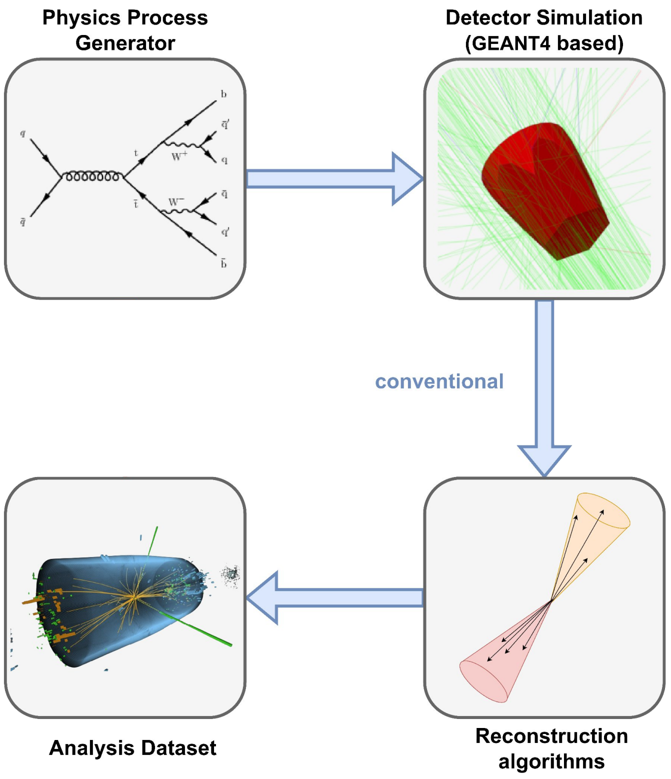
This will be a major issue for High-Lumi LHC; on top of these upgrades: increased granularity of CMS Phase 2 Detector



Conventional CMS Simulation

- **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)

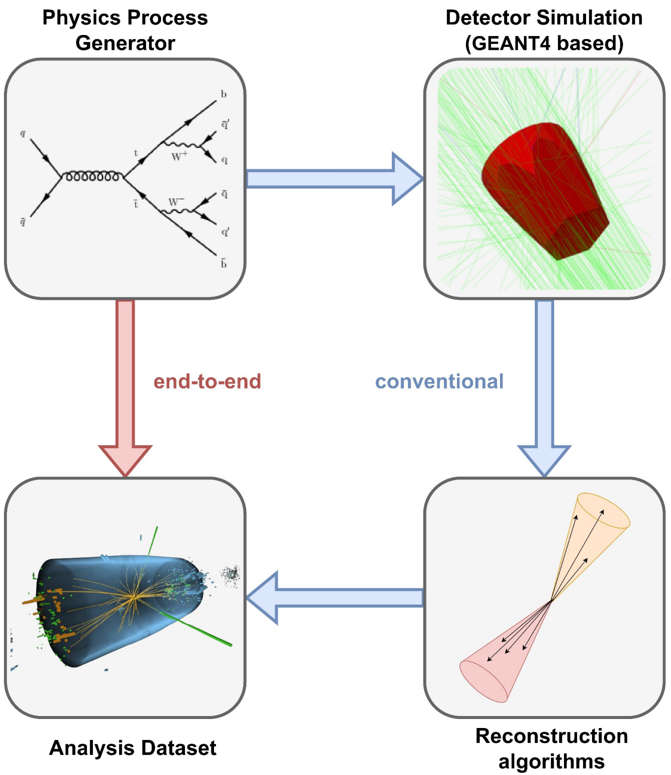


From 2402.13684

CMS FlashSim

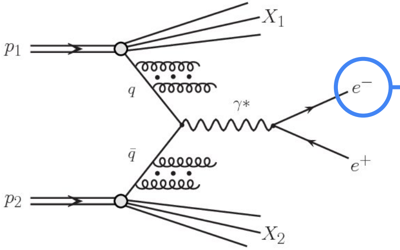
FlashSim — Universal, ML-based, end-to-end simulation framework

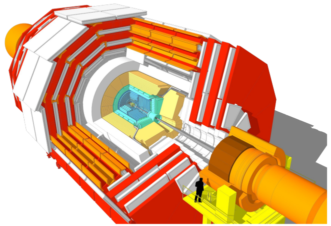
- targeting directly analysis-ready high-level variables (NANOAOD)
- using state-of-the-art generative models
- simulation speed ~100 Hz: x(100/1000) faster than FullSim
- analysis and sample independent



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*

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

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 (~2M 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

The simulation of objects is informed by what we simulated before, the output of a module can be the input of the next one (e.g. egamma)

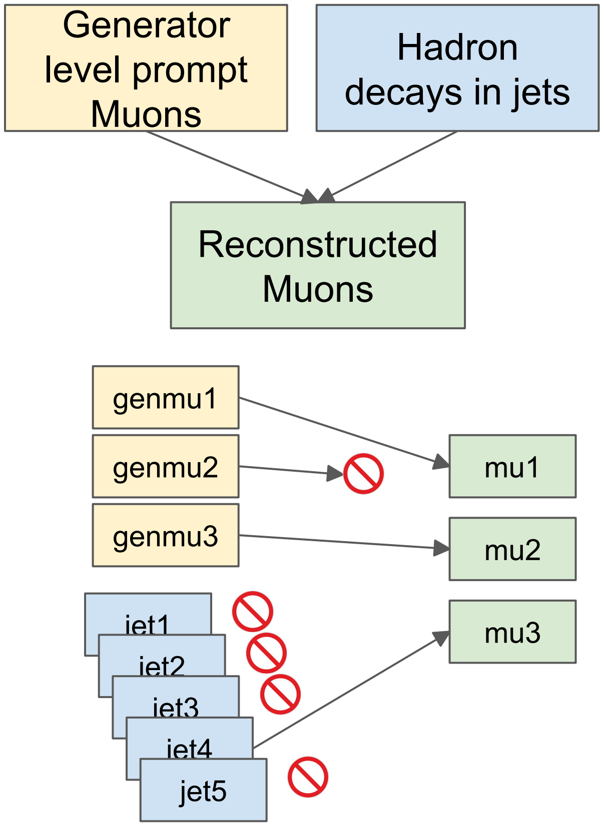
Physics objects	Sources (one NN model for each source)			Number of simulated attributes per object
Jets	Generator Jet	Fake from PU		39
Muons	Generator Muons	Fake from Jets/PU	Duplicates	53
Electrons	Generator Electrons	Generator Photons (prompt)	Fake from Jets/PU	48
Photons	Generator Photons (prompt)	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

The final structure combines two modules

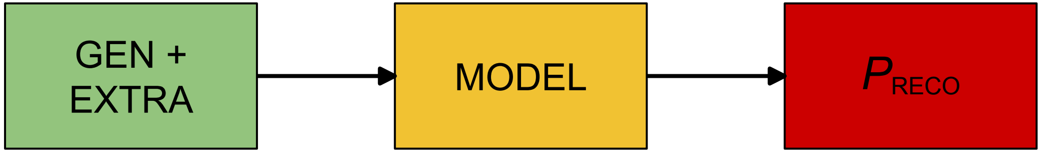
Trained on 4M events samples soup

Each object is handled by FlashSim with the various models:

- An efficiency model for each source
- A properties/simulation model for each source



We model the efficiencies with a basic NN

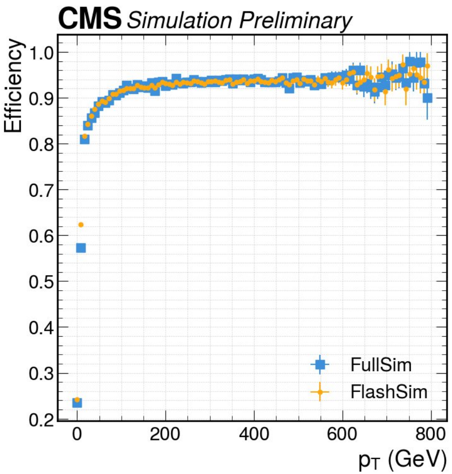


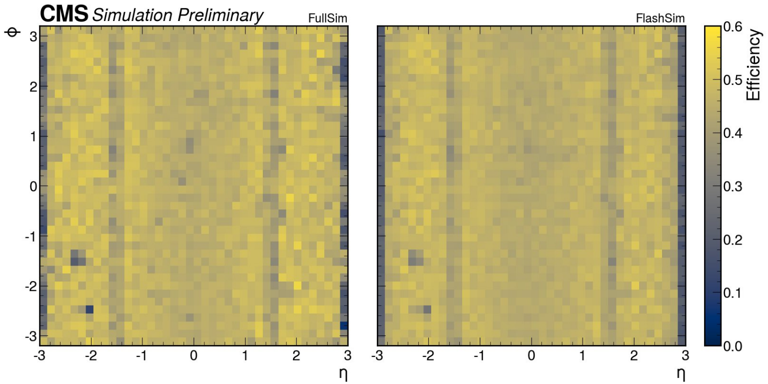
Efficiency = $P_{\text{RECO}}\,(p_T,\eta,\phi,\ldots)$

We must decide whether to simulate a given object!

$y \sim unif([0,1))$

isReco = DNN(inputs) > y



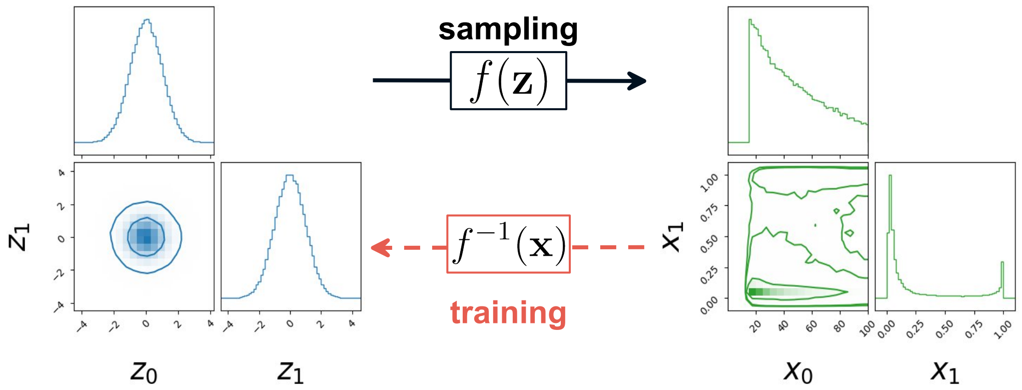


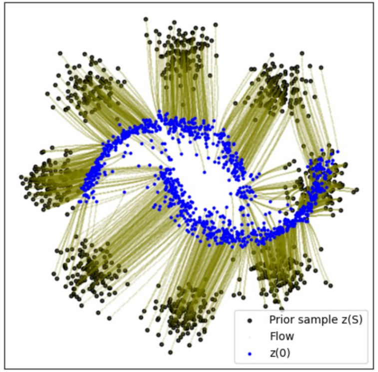
Normalizing Flows as backbone

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

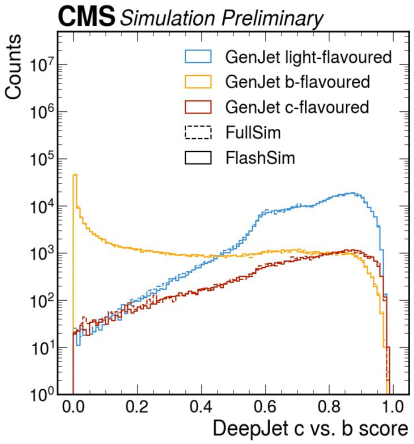
We use Continuous Flows trained with Flow Matching for optimal performances

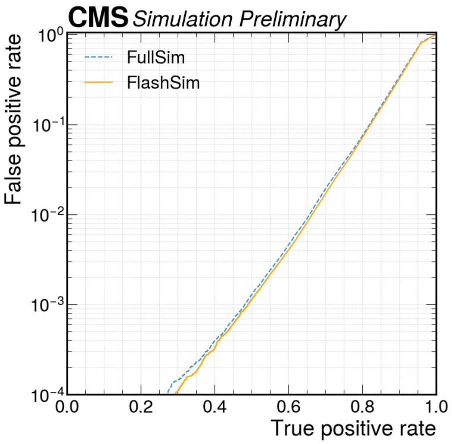


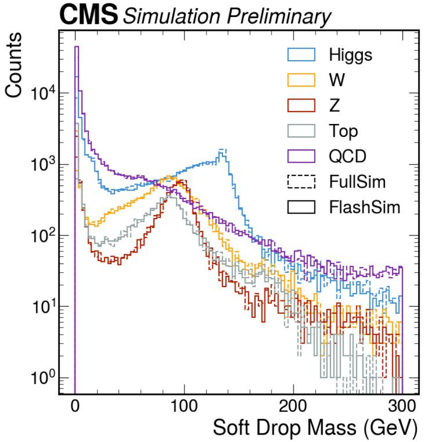


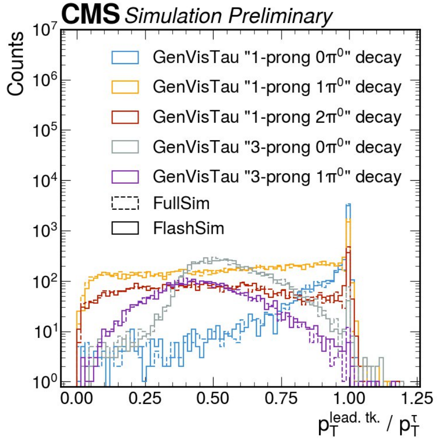
Good 1d performance on different plots

The same model should learn to produce *different distributions for different conditioning* values (momentum of a particle, flavour of the quark producing a jet, decay mode of a particle, etc…)

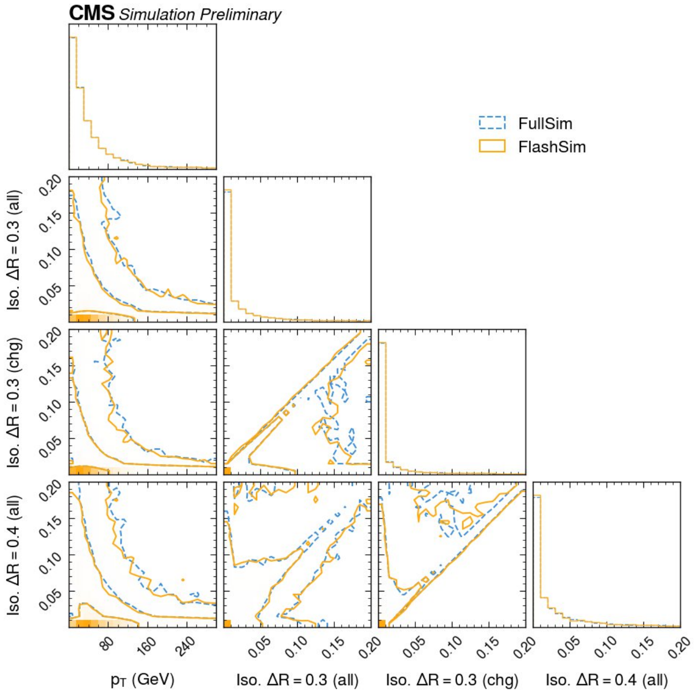


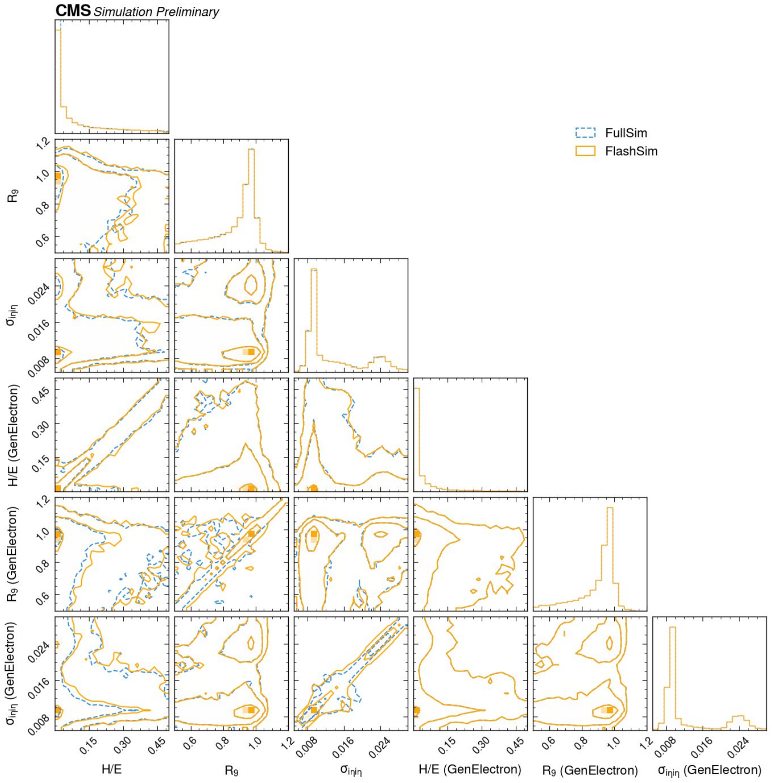






Good capturing of correlations between different variables





Analysis level performance

Once full NANOAOD events are available we can compare derived quantities and implement some analyses

Two toy analyses corresponding to VBF Higgs to muons search and ZH$\rightarrow$ llbb have been tested comparing flashsim with fullsim

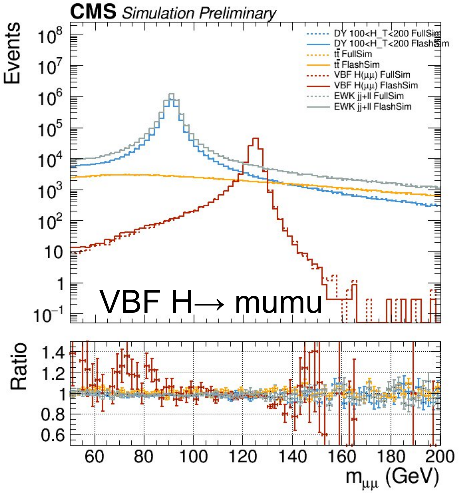
Analyses tested all the way down to the final DNN output, comparing different samples, some never seen during training

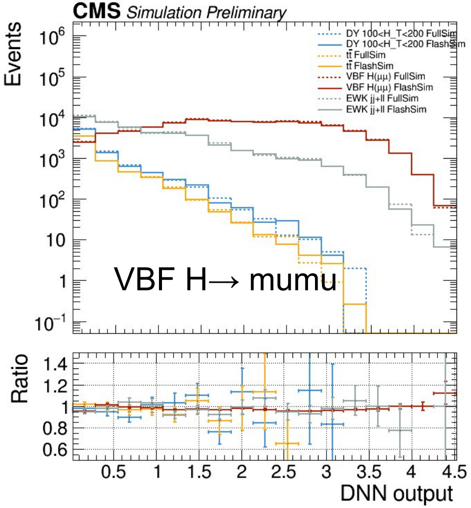
Analysis level performance

Once full NANOAOD events are available we can compare derived quantities and implement some analyses

Two toy analyses corresponding to VBF Higgs to muons search and ZH→ llbb have been tested comparing flashsim with fullsim

Analyses tested all the way down to the final DNN output, comparing different samples, some never seen during training





VBF H→ mumu

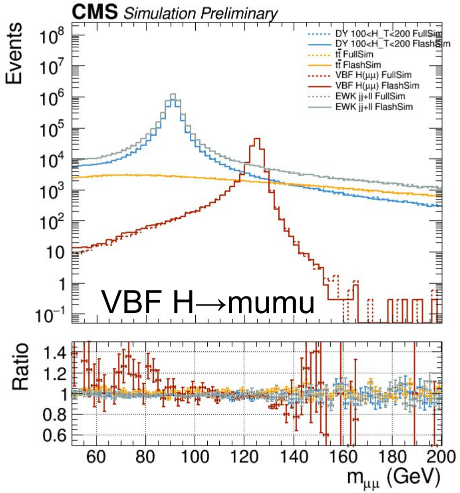
	Selection
Muons	$p_T > 20$ GeV, $\|\eta\| < 2.4$, Iso < 0.25, MediumID
Jets	$p_T > 25$ GeV, $\|\eta\| < 4.7$, puId > 0, jetId > 0
Signal Region	115 GeV < m(ll) < 135 GeV $p_T^{j1} > 35$ GeV , $p_T^{j2} > 25$ GeV , m(jj) > 150 GeV , $\|\Delta\eta(jj)\| > 2$

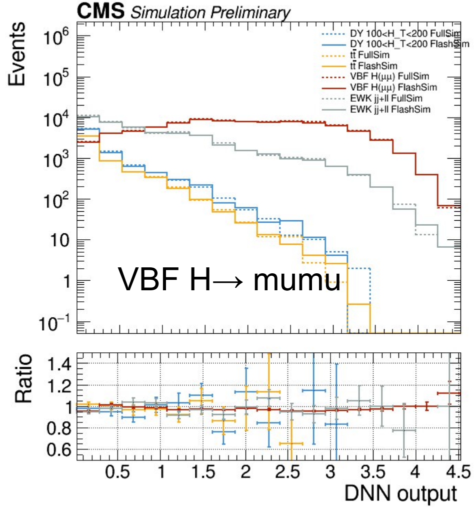
Analysis level performance

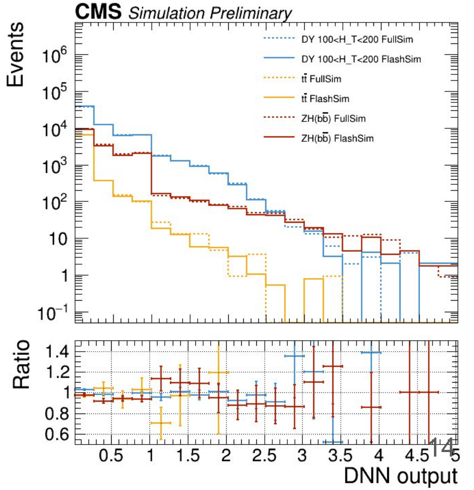
Once full NANOAOD events are available we can compare derived quantities and implement some analyses

Two toy analyses corresponding to VBF Higgs to muons search and ZH→ llbb have been tested comparing flashsim with fullsim

Analyses tested all the way down to the final DNN output, comparing different samples, some never seen during training







VBF H→ mumu	Selection
Muons	p_T > 20 GeV, |η| < 2.4, Iso < 0.25, MediumID
Jets	p_T > 25 GeV, |η| < 4.7, puId > 0, jetId > 0
Signal Region	115 GeV < m(ll) < 135 GeV p_T^{j1} > 35 GeV , p_T^{j2} > 25 GeV , m(jj) > 150 GeV , |Δη(jj)| > 2

ZH→ llbb	Selection
Muons	p_T > 20 GeV, |η| < 2.4, Iso < 0.25, MediumID
Jets	p_T > 20 GeV, |η| < 2.5, puId > 0, jetId > 0
Mediu m b-tag	DeepFlavour btag > 0.27
Signal Region	75 GeV ≤ m(Z) < 105 GeV, 90 GeV < m(jj) < 150 GeV, Medium b-tag (lead. jet)

The speed depends on the approach

The current prototype with ~20 properties model and ~20 efficiency models, starting from existing generated samples runs between 10Hz and 1KHz

If the generator is very slow, we are easily in the shadow of the generator

What if we can avoid being generator-speed limited by **reusing** generated events? **Oversampling**!

Processor	ODE accuracy (timesteps)	Event simulation rate
GPU 3060	100	325 Hz
GPU 3060	20	690 Hz
CPU 1-core	100	15 Hz
CPU 1-core	20	60 Hz
CPU 4-core	20	120 Hz

		Event generation speed				Ratio to Geant4-based		
Generator speed (Hz)	Oversample factor	0.1Hz Geant4 based sim	10Hz Flashsim	100Hz Flashsim	1KHz Flashsim	*10Hz Flashsim*	*100Hz Flashsim*	*1KHz Flashsim*
available	1x	0.10 Hz	10.00 Hz	100.00 Hz	1000.00 Hz	*100.0x*	*1000.0x*	*10000.0x*
50.00 Hz	1x	0.10 Hz	8.33 Hz	33.33 Hz	47.62 Hz	*83.5x*	*334.0x*	*477.1x*
50.00 Hz	10x	0.10 Hz	9.80 Hz	83.33 Hz	333.33 Hz	*98.1x*	*833.5x*	*3334.0x*
1.00 Hz	1x	0.09 Hz	0.91 Hz	0.99 Hz	1.00 Hz	*10.0x*	*10.9x*	*11.0x*
1.00 Hz	10x	0.10 Hz	5.00 Hz	9.09 Hz	9.90 Hz	*50.5x*	*91.8x*	*100.0x*
0.05 Hz	1x	0.03 Hz	0.05 Hz	0.05 Hz	0.05 Hz	*1.5x*	*1.5x*	*1.5x*
0.05 Hz	10x	0.08 Hz	0.48 Hz	0.50 Hz	0.50 Hz	*5.7x*	*6.0x*	*6.0x*

Conclusions

CMS is investigating FlashSim as the next approach of simulation during Run3/High-Lumi

- A complete working prototype for end-to-end simulation of CMS NANOAOD format: if you are in CMS you can use this TODAY
- Tests on toy analyses show a good accuracy also for derived quantities, next tests should be on real analysis, possibly already in Run3
- We can use the oversampling technique to maximize the exploitation of generator level MC event

contact: francesco.vaselli@cern.ch

For more FlashSim, see also:

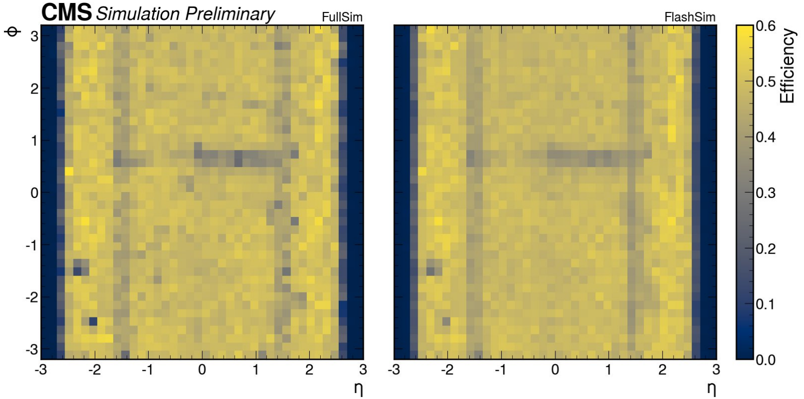
- CHEP24 Plenary talk
- CMS DPS Note
- CMS NOTE 2023 003 (old prototype with discrete flows)
- Technical paper: 2402.13684 (DOI)
- REPO: https://gitlab.cern.ch/cms-flashsim/cms-flashsim

Backup

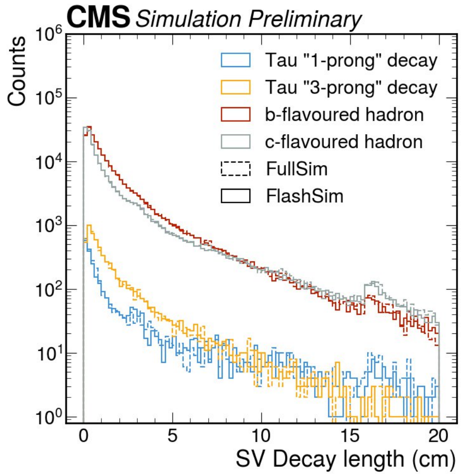
We train on a 4M events cocktail

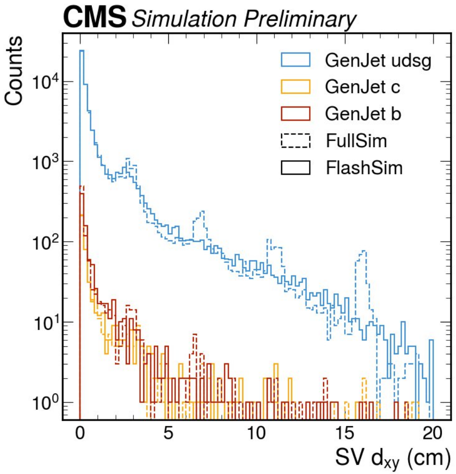
Trained on a cocktail of different processes, covering various signatures in the detector response

Likely a suboptimal choice, dedicated QCD/Particle Gun samples can be considered



Sample	Events
$t\bar{t}$	800k
DY HT [100, 200], 2J MLL [200-1400]	930k
HH → bb bb	840k
X(3000) → Y(500) H(125) → (bb) (WW → 2q 2lν)	147k
X → HH → qq qq (M_X 900, 1200, 1800; M_H 365, 400, 18)	90k
SMS TchiZH mNLSP200-1500	300k
X(1200) → Y(300) H(125) → bb $\gamma\gamma$	400k
VBF H → $\tau\tau$	270k
bbA → ZH → ll $\tau\tau$ (M = 900)	33k





Testing the power consumption of FlashSim

Using CERN IT machine

- 2x Silver 4110 (8 cores, 16 threads each)
- 4x NVIDIA T4 16 GB GDDR6 for the GPUs
- 194 GB of Memory,
- ~2Tb of storage

hep-benchmark-suite used to monitor the power of the server and the gpu stats as well through

- `ipmitool dcmi power reading`
- `nvidia-smi`.

For more see “Giordano, D. et al., HEPScore: A new CPU benchmark for the WLCG (2024), https://doi.org/10.1051/epjconf/202429507024 “, see also the previous talk “The Role of the HEP Benchmark Suite[...]”

Estimating the cost of a training run: extraction + training

Extraction of training data on CPU from ~ 4M events

~30 mins for the extraction with Effective Power Consumptions of 154W: 1.54 kWh for the extraction of all 20 objects

Training on 4 threads, 1 GPU (similar conditions to the training nodes on HTCondor)

average power ~211W with GPU util ~40%: assuming average of 16h training runs for each simulation model ~68 kWh

Considering efficiency models as well, we estimate ~100kWh for a full training run!

	Total server power W	Idle power W	Final consumption W
Extraction	194	40 (4 GPUs)	154
Training	241	30 (3 GPUs)	211

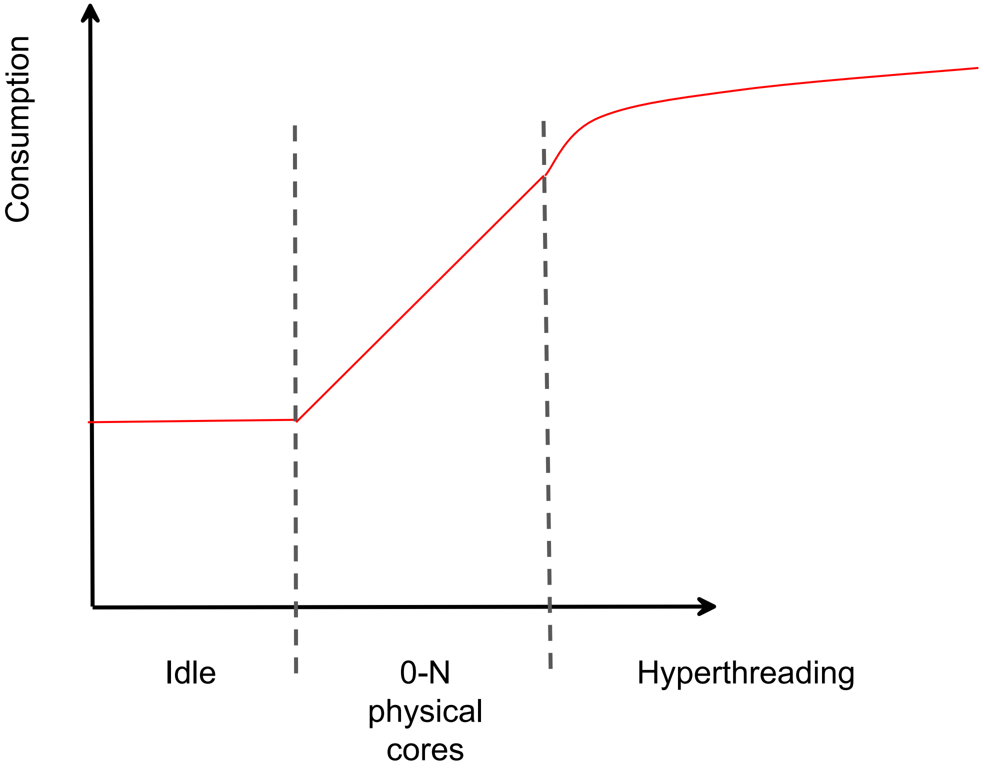
How to measure the FullSim power consumption fairly

Using again hep-benchmark-suite

We saturate the CPU and run multiple 4 threads copies, but we want to consider the consumption of just one!

We divide by the copies on “physical” cores since the scaling of consumption with hyperthreading is different

In our case 16 physical cores, 4 threads jobs -> consider just ¼ of the consumption vs idle



Current speed brings a reduction in simulation costs

Process	Total server power W	Idle power (to subtract) W	Final consumption W	Throughput (ev/s)	kWh/ev
FlashSim on GPU	253	30 (3 GPUs)	223	~163 Hz	3,80E-07
FlashSim on CPU	200	40 (4 GPUs)	160	~1 Hz	4,40E-05
FullSim	256	40 (4 GPUs)+ 72 (other copies running)=112	144	~0.07 Hz	5,00E-04

Both tested on RunII TTbar simulation, using 4 threads (and optionally 1 GPU)

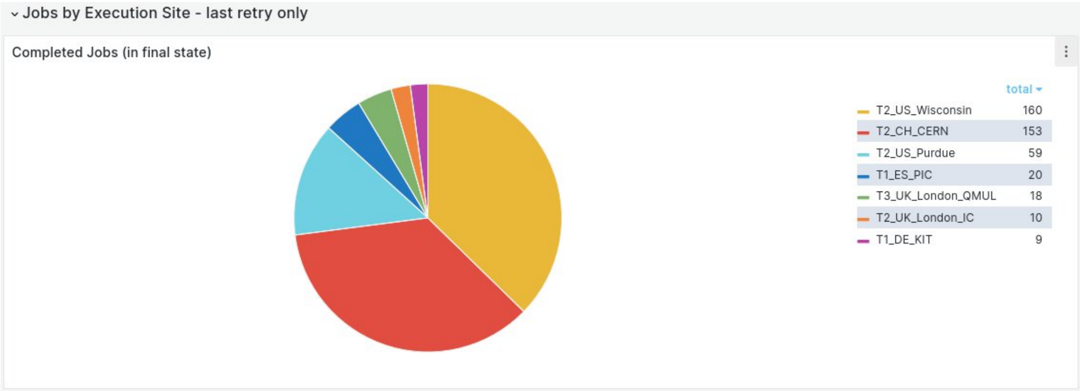
Caveat: CMS FullSim running gen-sim and reco. Best comparison would be FlashSim vs sim-digi-reco; however the consumption data and the throughput allow to extrapolate a reasonable estimate

FlashSim on GPU has a 3 orders of magnitude reduction in the cost of energy measured as kWh/ev!

FlashSim is already well integrated in the CMS Computing infrastructure

We can use the CMS Analysis Remote Builder tool (CRAB) to submit the simulation of large samples directly to gpu-enabled nodes of the grid

The dataset is automatically published on DAS and rucio at the end of the simulation (already simulated >300M of samples in a few hours).Training/inference scripts on HTCondor available as well



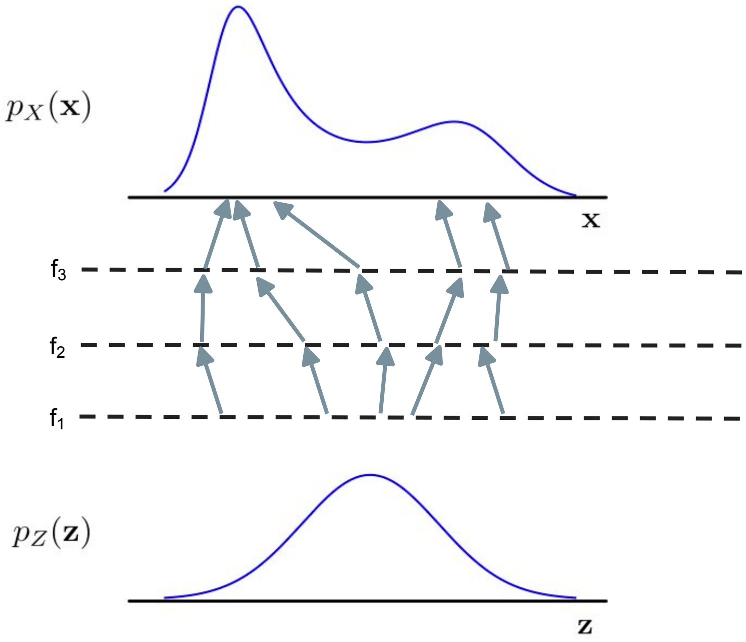
“Discrete” Flows

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])

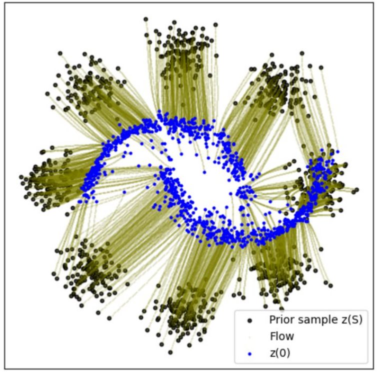
$$f(0; z) = z = Gaussian$$

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

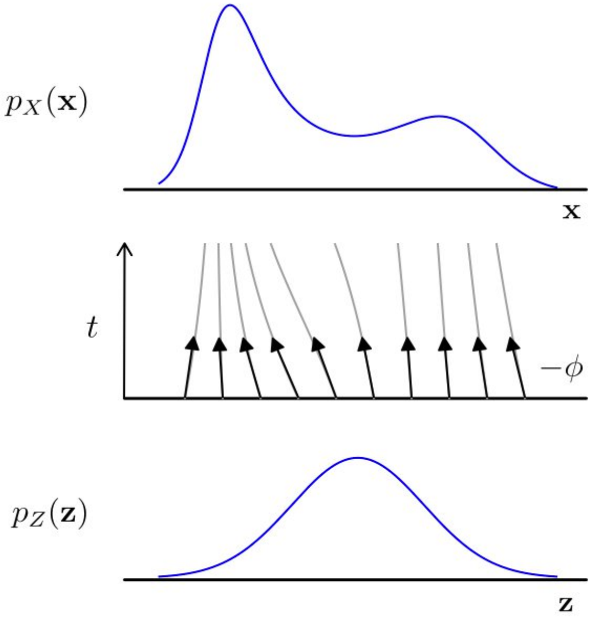
$$f(t + dt) = f(t) + v(t) \cdot dt$$

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

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

Flow Matching: basic idea

Main idea:

Learn vector field u, approximation of v

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

t=0: p(z) = N(0,1)

t=1: p(z) = N(x, sigma_min)

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

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

y = NN(x)

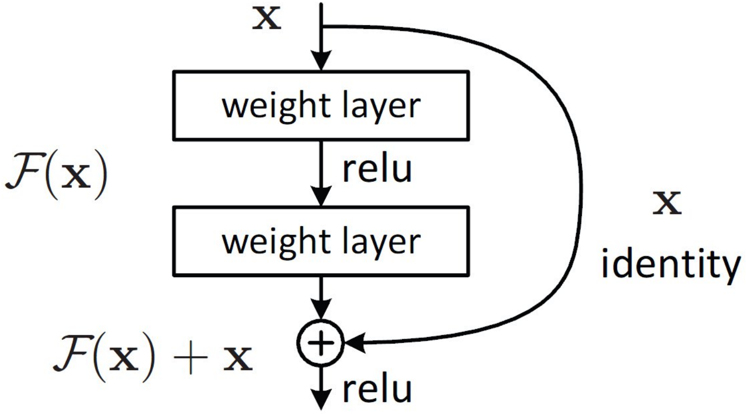
Loss = || u - y ||, simple regression!

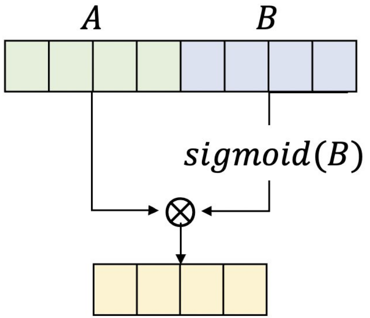
Model architecture and libraries

We use PyTorch as Deep Learning library

The architecture being used is a ResNet with some additional Gating (GLU layers) to improve the response to conditioning

~2M parameters, around 1-2 days of training on HTCondor (data is the bottleneck)

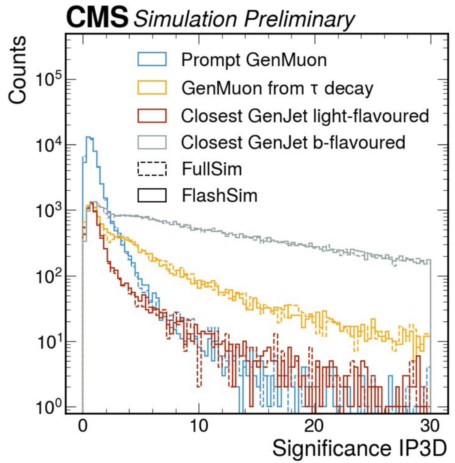




Conditioning and preprocessing are crucial

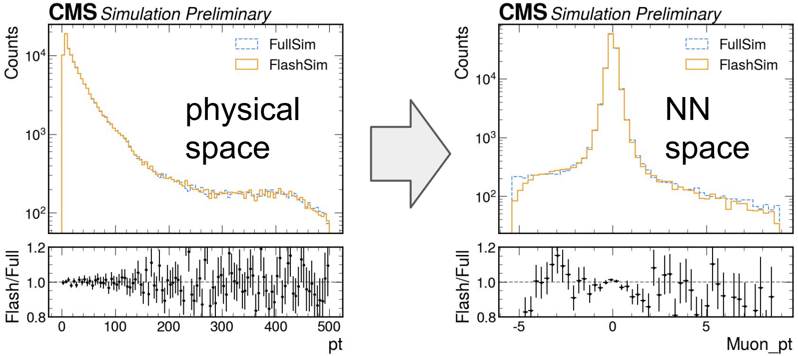
Some properties have obvious correlations with generator level information

- generated vs reconstructed four-momentum
- MC flavour with tagging variables



Two crucial points to reproduce correlations

- Conditioning:
  - e.g. is it b-quark jet?
- Transformations:
  - standard scaling
  - better learn P_T^{reco} or P_T^{reco}/P_T^{gen} ?
  - tails matter for physics (apply logs when needed)



Training resources and where to train

After optimizing the different modules, we can submit a series of train-all scripts to HTCondor

Need to train ~
20 Models + Efficiencies

Training on GPU, it takes about 1-2 days

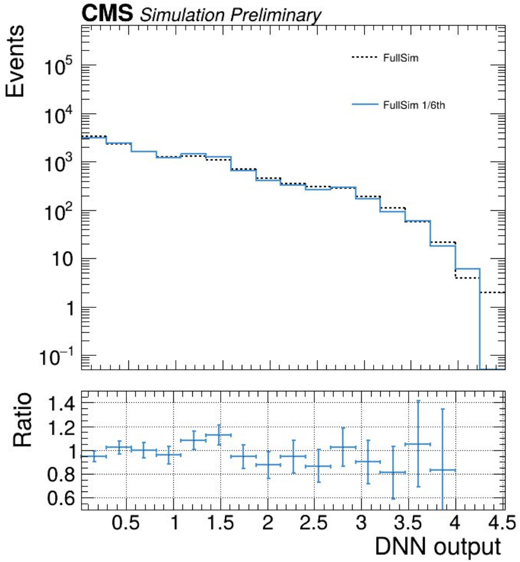
Convenient for retrain campaigns on new NanoAOD versions!

Oversampling

- Typical LHC MC samples are randomly sampled “twice”
  - in the generator
  - in simulating the detector response
- In many cases a large part of the uncertainty originates from the detector response
  - generator information can be reused

We call “**oversampling**” the repeated usage of the same generator event for multiple simulations

- Proper statistical treatment is needed for events originating from “same gen”
  - count events that end up in the same bin of a histogram as correlated
  - consider events in different bins as uncorrelated



Is oversampling introducing biases?

Let’s test it against full sim

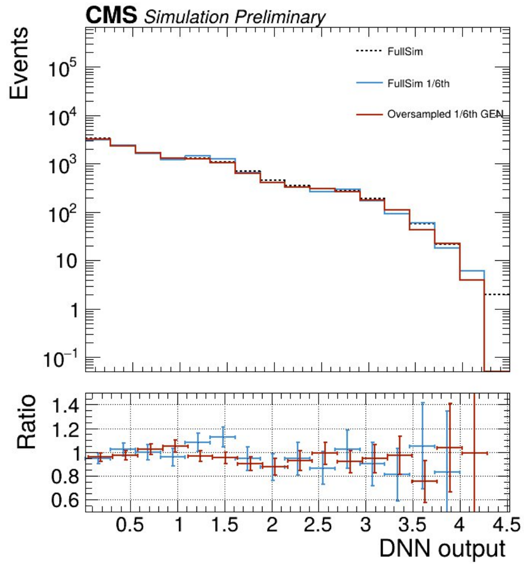
- We start from a sample for which we have 8M full sim events
- We take a fraction (1/6th, 1.3M events) of the full sim events and we can check how oversampling (6x or 10x) it would compare to the full sim sample

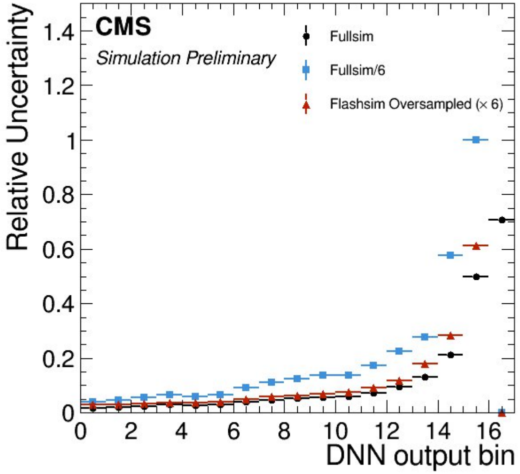
Oversampling

- Typical LHC MC samples are randomly sampled “twice”
  - in the generator
  - in simulating the detector response
- In many cases a large part of the uncertainty originates from the detector response
  - generator information can be reused

We call “**oversampling**” the repeated usage of the same generator event for multiple simulations

- Proper statistical treatment is needed for events originating from “same gen”
  - count events that end up in the same bin of a histogram as correlated
  - consider events in different bins as uncorrelated



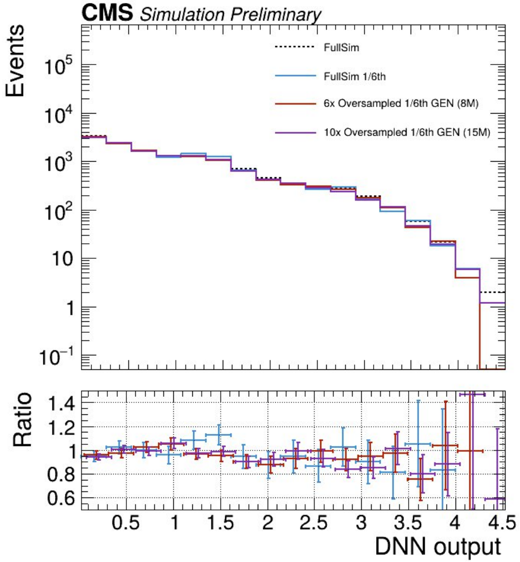


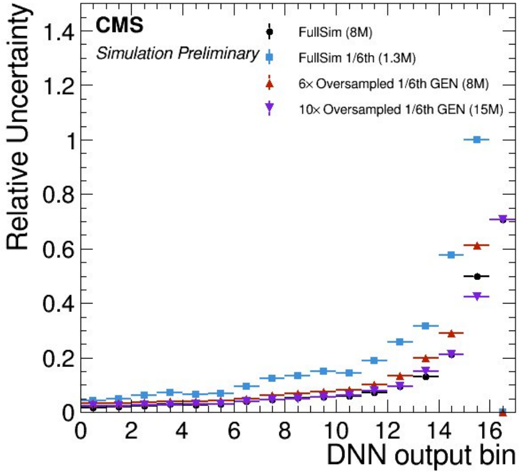
Oversampling

- Typical LHC MC samples are randomly sampled “twice”
  - in the generator
  - in simulating the detector response
- In many cases a large part of the uncertainty originates from the detector response
  - generator information can be reused

We call “**oversampling**” the repeated usage of the same generator event for multiple simulations

- Proper statistical treatment is needed for events originating from “same gen”
  - count events that end up in the same bin of a histogram as correlated
  - consider events in different bins as uncorrelated





Training samples vs flash-simulated samples

Samples used in training

Sample	Events
tt	800k
DY HT [100, 200], 2J MLL [200-1400]	930k
HH → bb bb	840k
X(3000) → Y(500) H(125) → (bb) (WW → 2q 2l)	147k
X → HH → qq qq (M_X 900, 1200, 1800; M_H 365, 400, 18)	90k
SMS TchiZH mNLSP200-1500	300k
X(1200) → Y(300) H(125) → bb	400k
VBF H →	270k
bbA → ZH → ll (M = 900)	33k

Samples simulated for event validation

Sample	Events
tt	100M
DY HT [100, 200]	25M
H →	1M
ZH	300k
jj + ll (ewk)	8M

About 4M events have been used to train FlashSim models while more than 100M events have been generated to make the plots of the event level validation. Some simulated samples, such as H → , were not used in training.

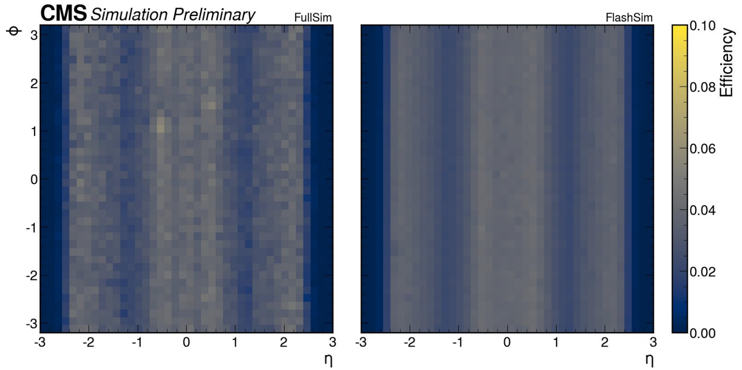
For samples used in training, such as tt, the event validation showed a remarkable agreement between FlashSim and FullSim even if only a fraction of less than 1%, of the 100M events available, was used for training.

Efficiency models

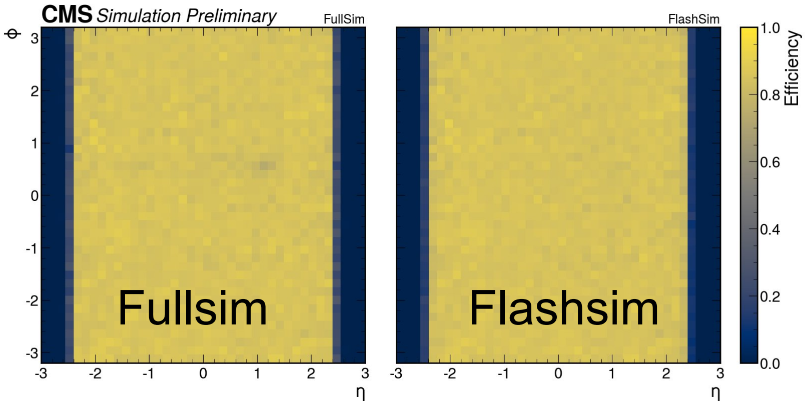
Given a source object to we get a reconstructed one?

- Efficiency models are **trained as simple classifiers** with binary cross-entropy loss
    - output can be interpreted as a probability!
- At inference time we just **toss in [0,1] and compare with model probability**

Probability of a jet producing a mu

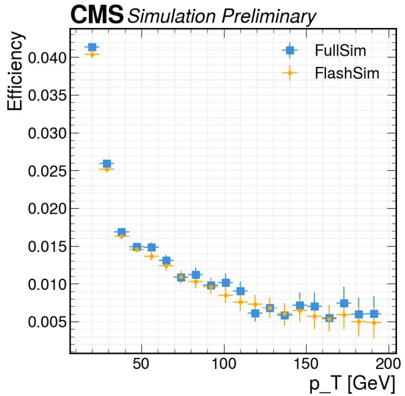


Prompt muon efficiency

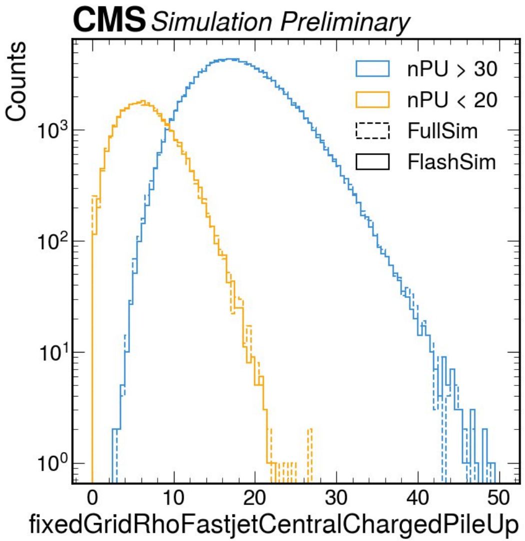


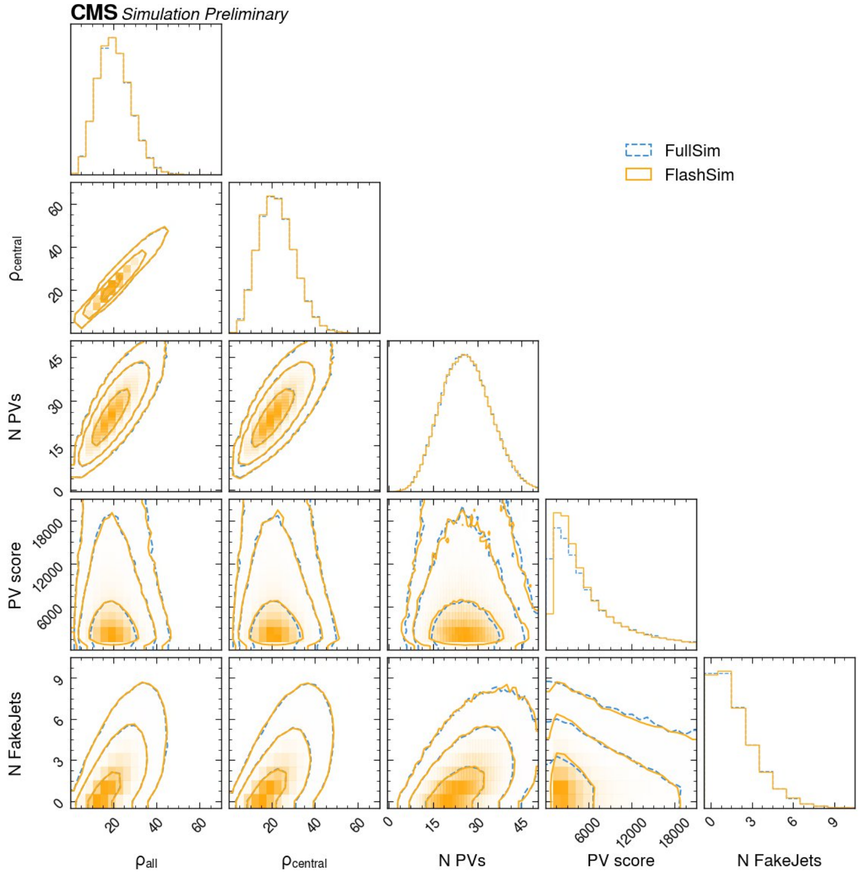
Prompt muon duplicate probability

Duplicates can be handled by training a second classifier to predict when a second copy is produced

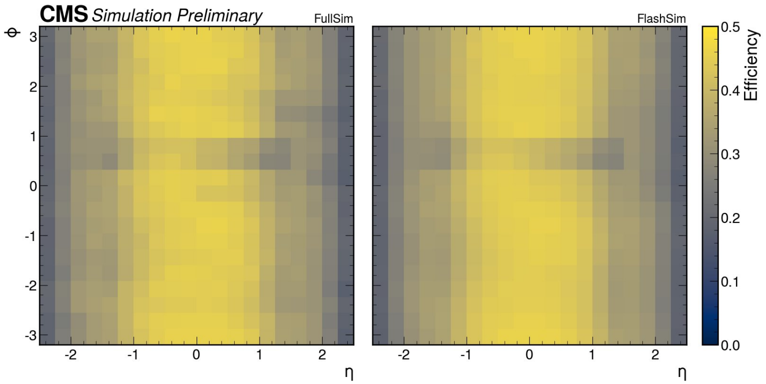


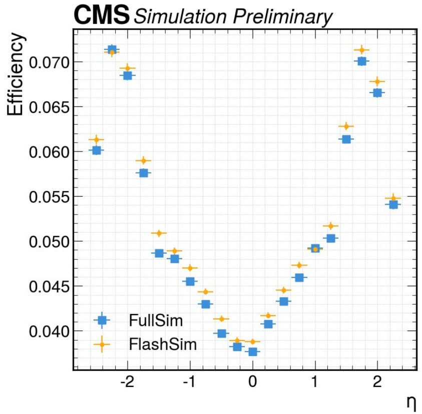
Vertex and Pileup

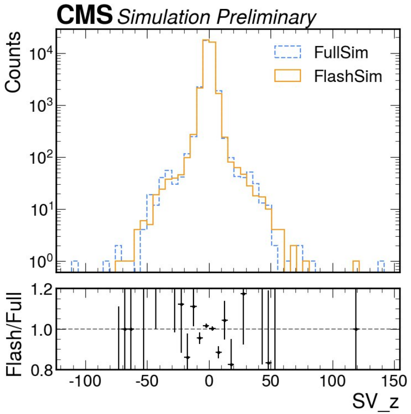




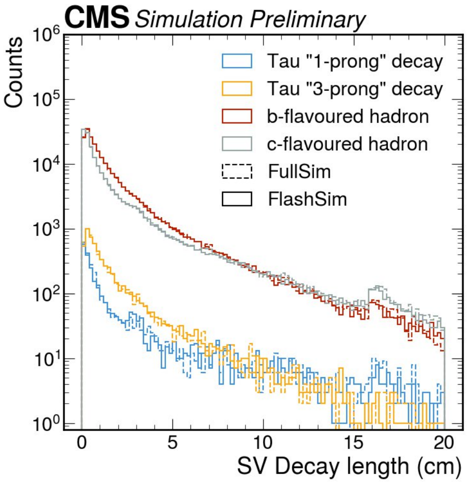
Secondary Vertices

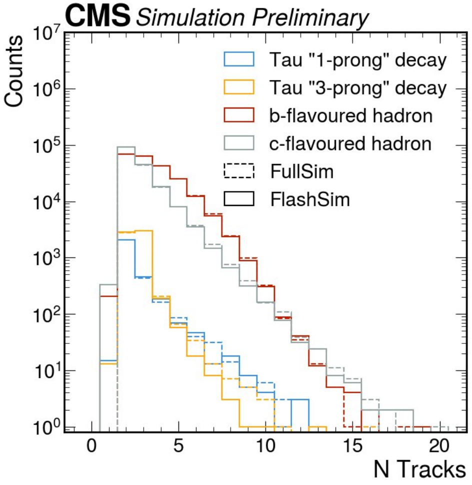


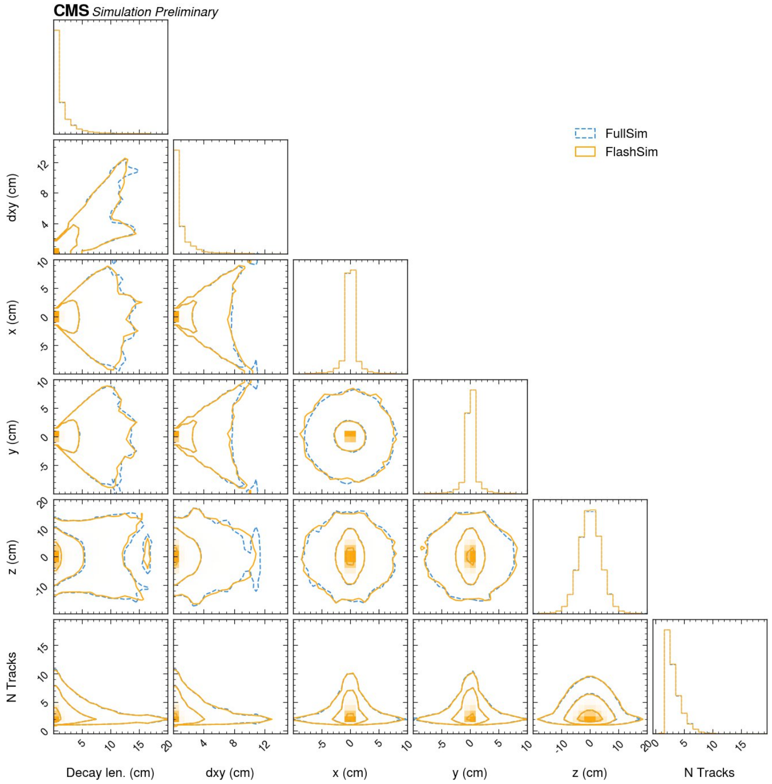




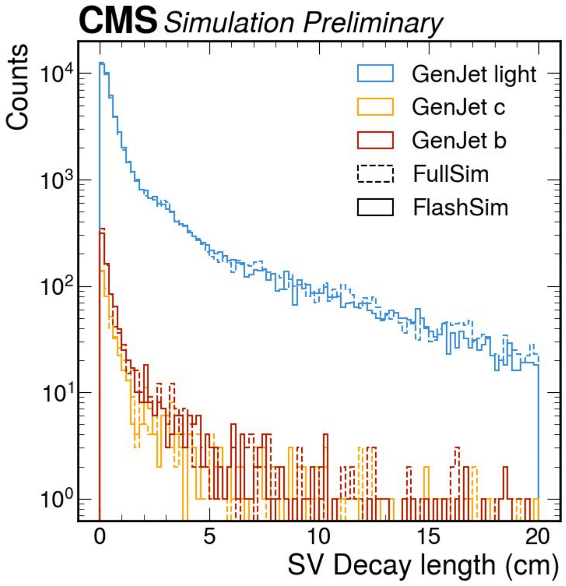
Secondary Vertex from Taus and Heavy Flavour

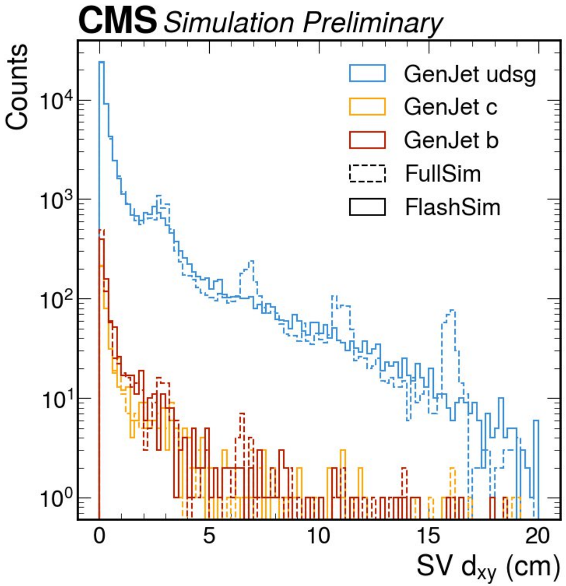


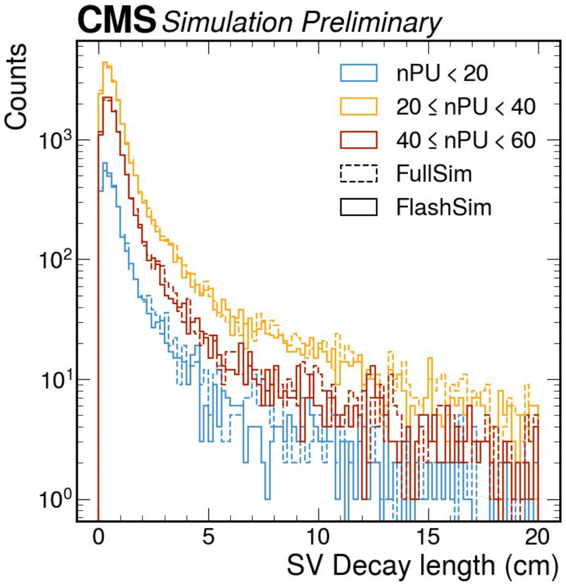




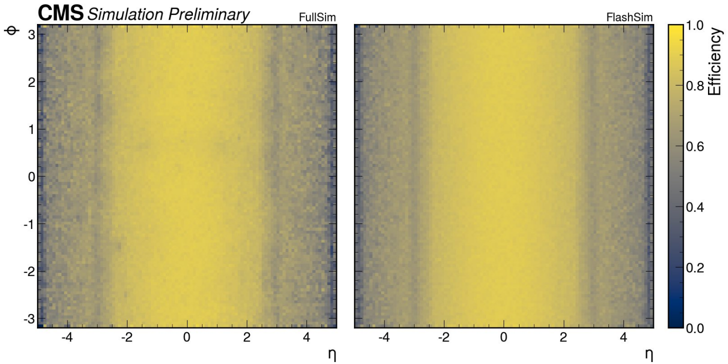
SV from GenJets

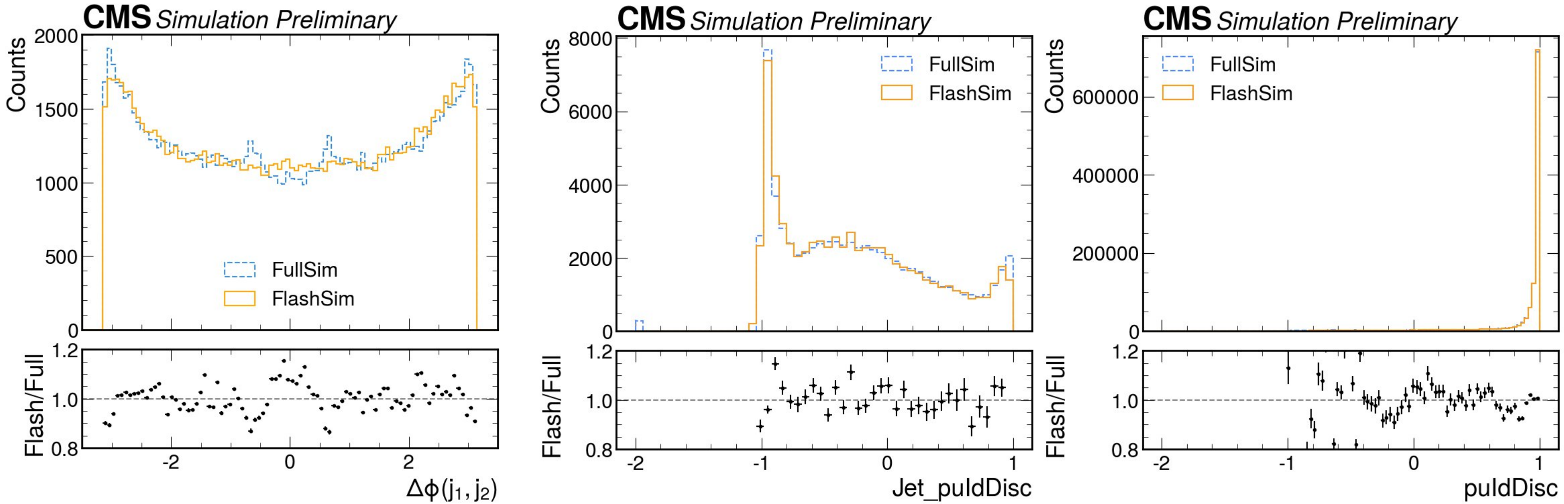




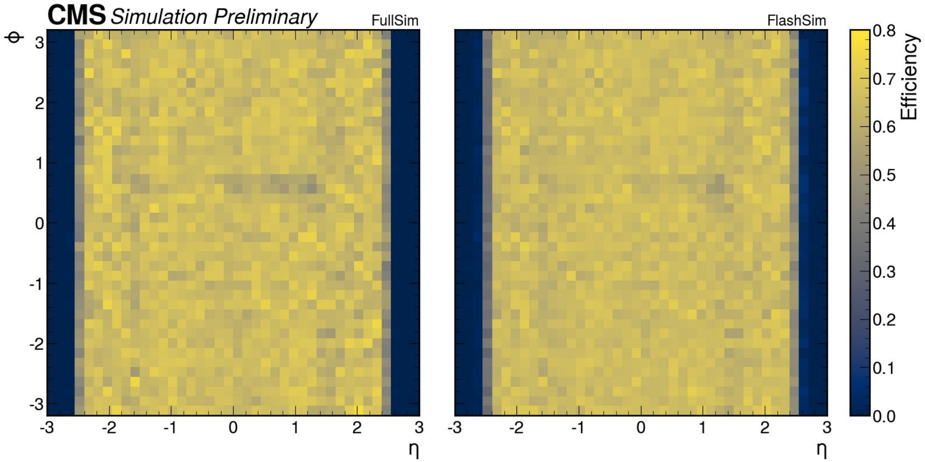


Jets and Fake Jets

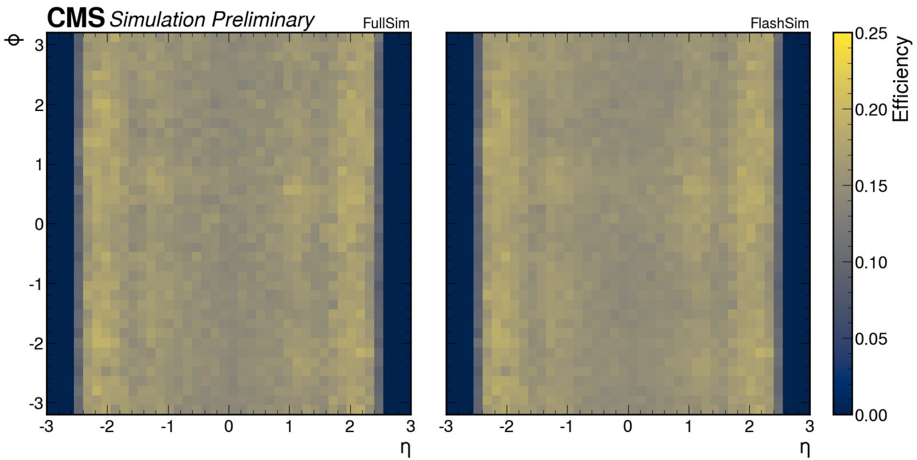




Tau

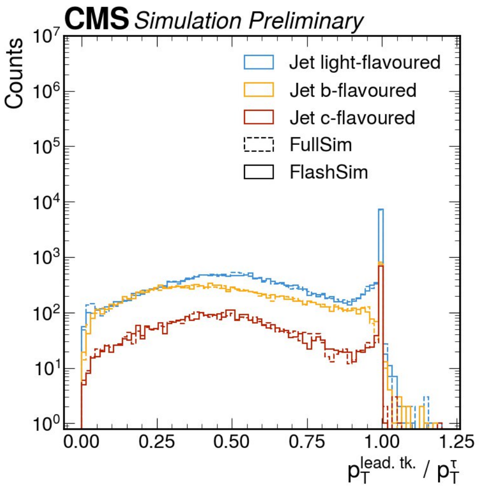


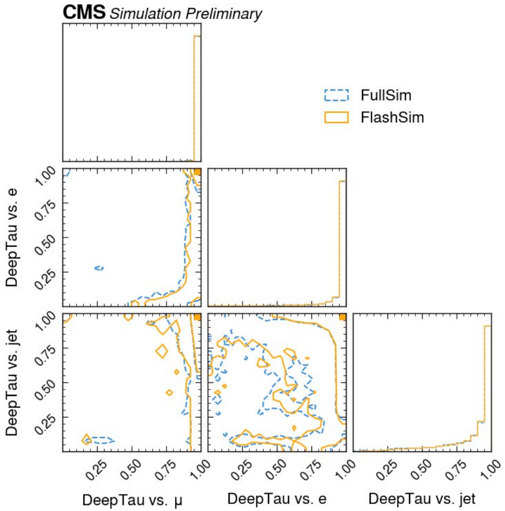
Signal

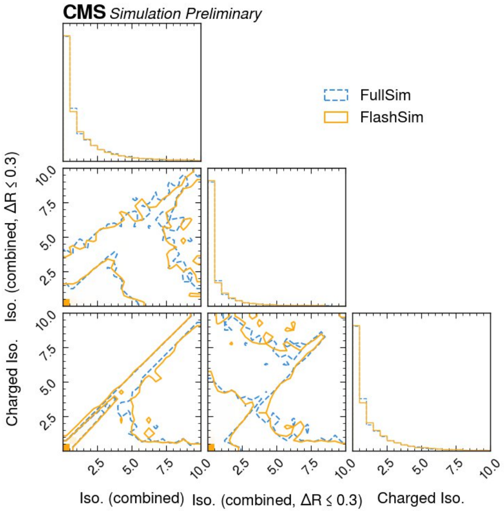


Background

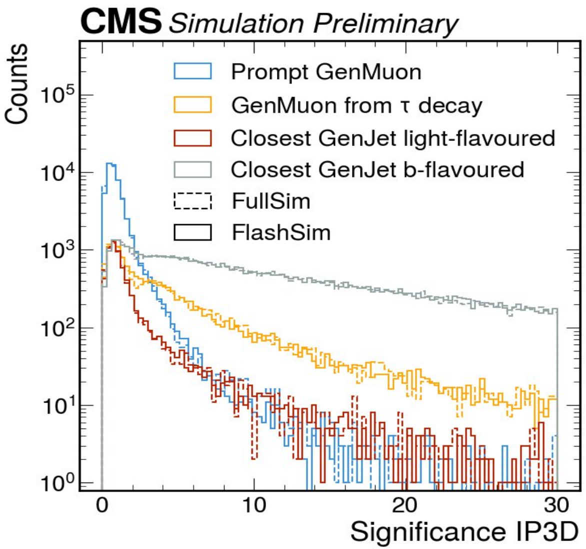
Tau properties

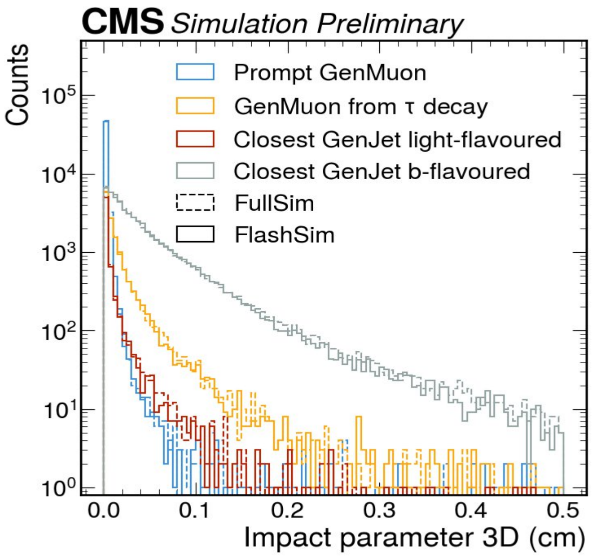


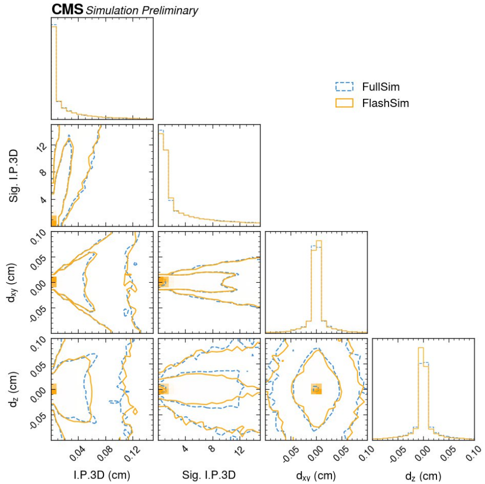


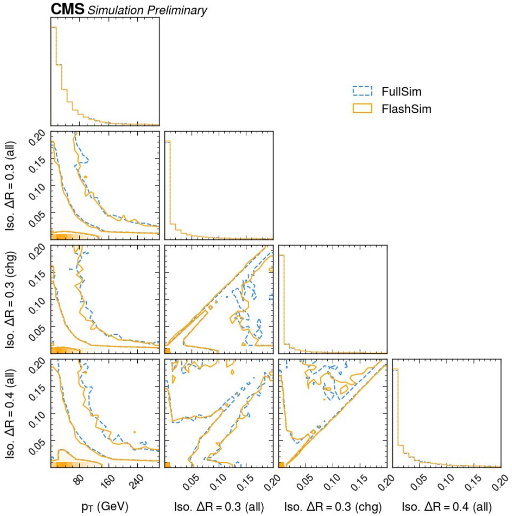


Muon features

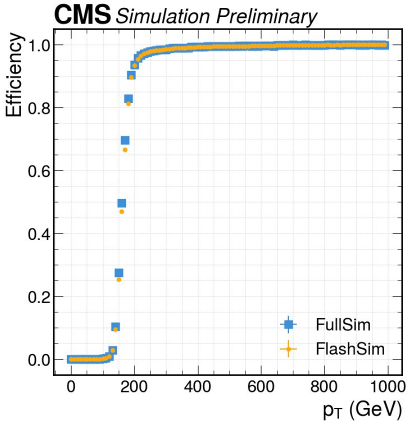




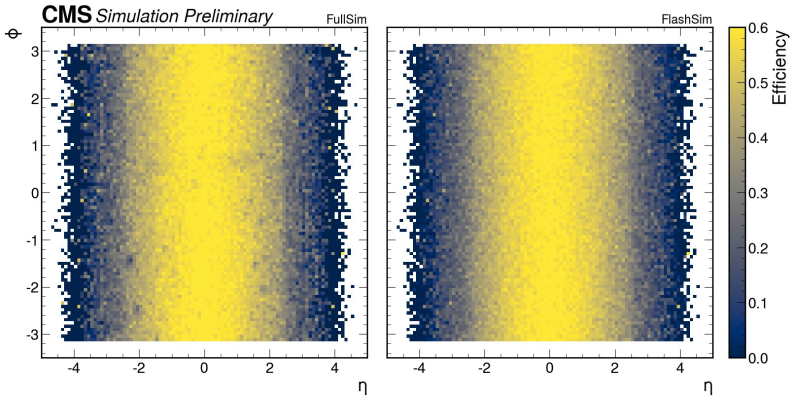




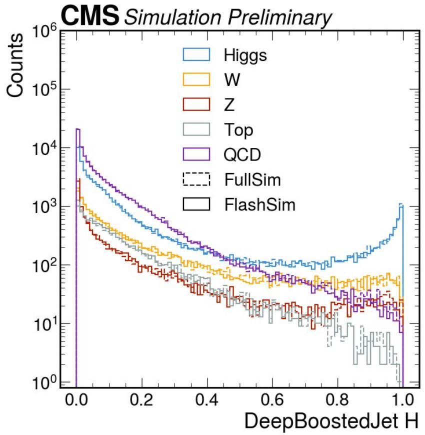
FatJets



CMS Simulation Preliminary
Efficiency
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
p_T (GeV)
FullSim
FlashSim

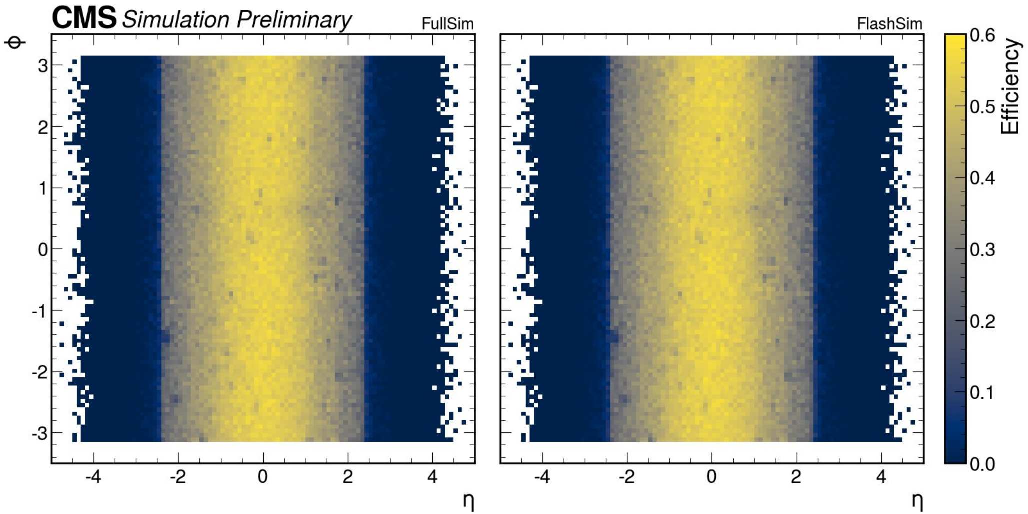


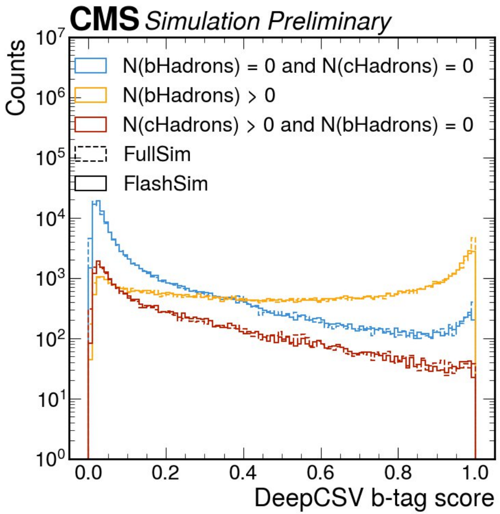
CMS Simulation Preliminary
FullSim
FlashSim
ϕ
η
Efficiency
0.0
0.1
0.2
0.3
0.4
0.5
0.6



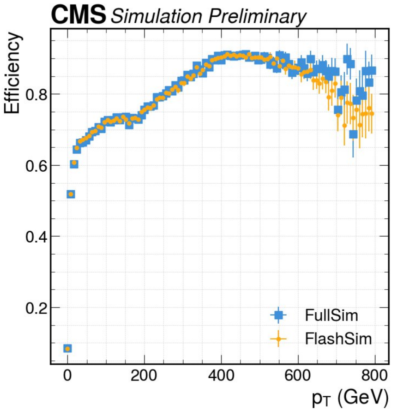
CMS Simulation Preliminary
Counts
Higgs
W
Z
Top
QCD
FullSim
FlashSim
DeepBoostedJet H

SubJets

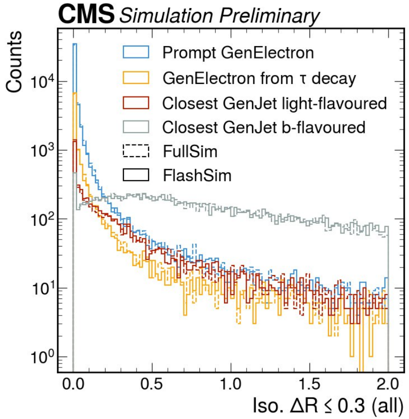

CMS Simulation Preliminary
FullSim
FlashSim
ϕ
η
Efficiency


CMS Simulation Preliminary
Counts
N(bHadrons) = 0 and N(cHadrons) = 0
N(bHadrons) > 0
N(cHadrons) > 0 and N(bHadrons) = 0
FullSim
FlashSim
DeepCSV b-tag score

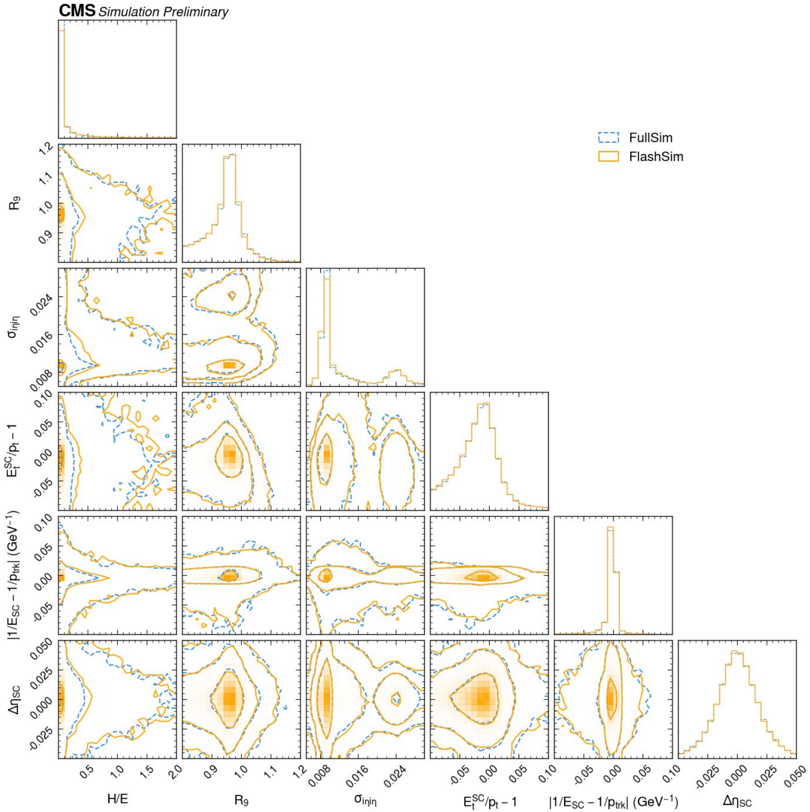
Electrons



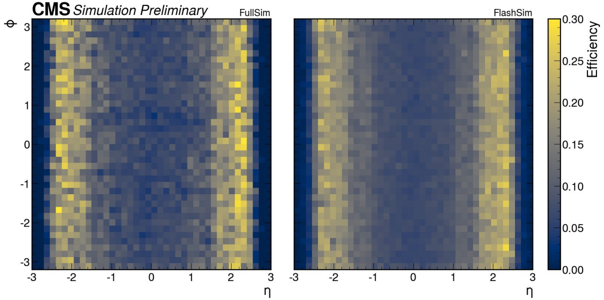
CMS Simulation Preliminary
Efficiency
pT (GeV)
FullSim
FlashSim



CMS Simulation Preliminary
Counts
Prompt GenElectron
GenElectron from τ decay
Closest GenJet light-flavoured
Closest GenJet b-flavoured
FullSim
FlashSim
Iso. ΔR ≤ 0.3 (all)

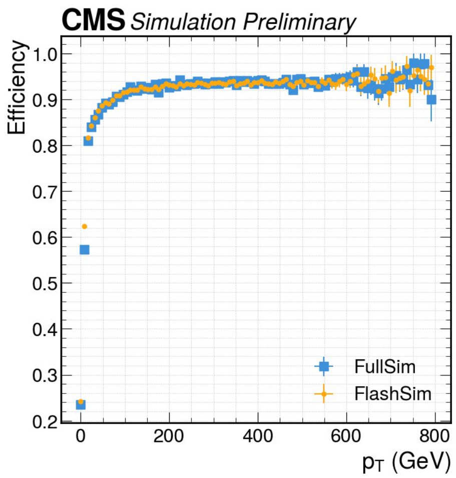


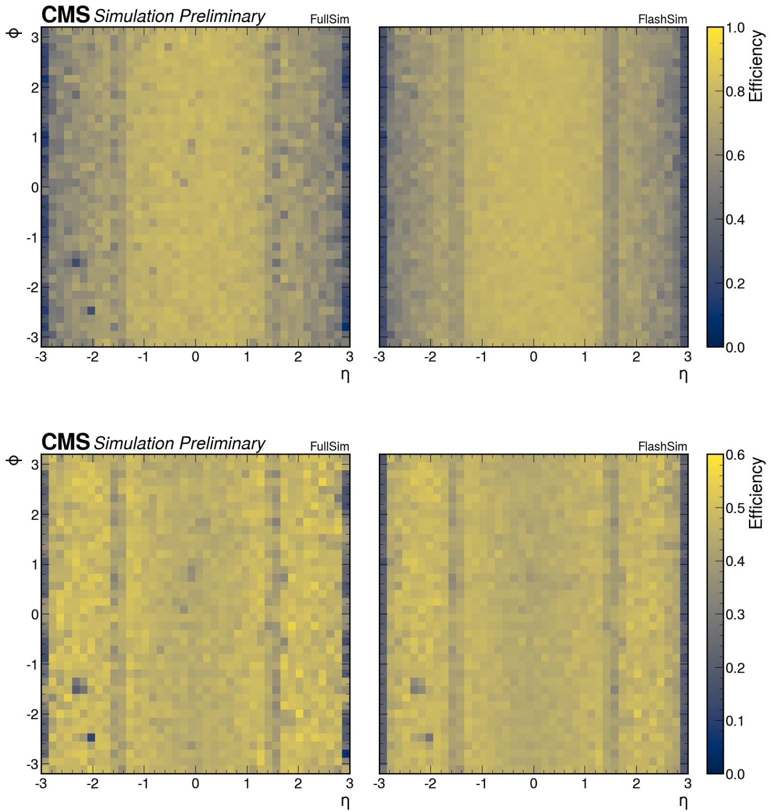
CMS Simulation Preliminary
FullSim
FlashSim
FullSim
FlashSim
H/E
R9
σiηiη
Δη_SC

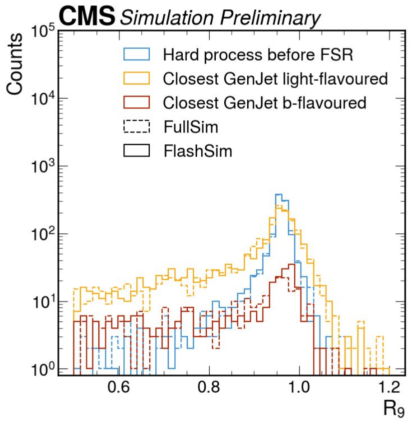


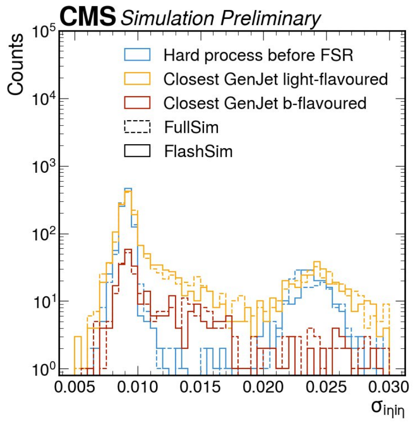
CMS Simulation Preliminary
FullSim
FlashSim
Efficiency
φ
η

Photon from generator level photons

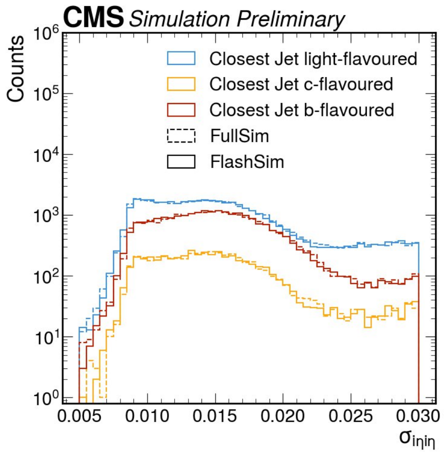


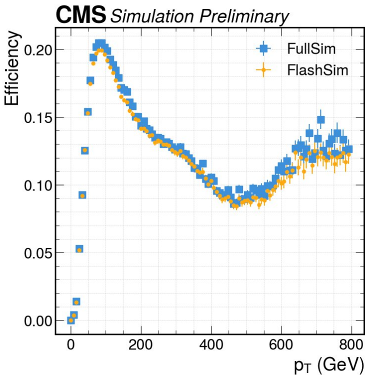


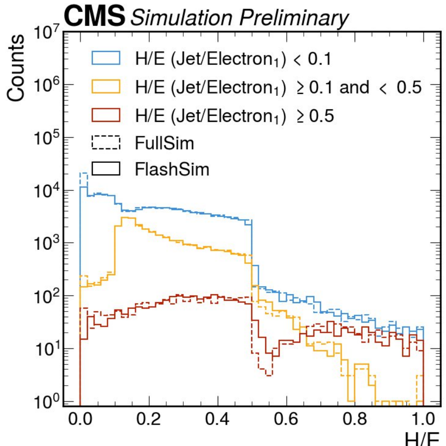


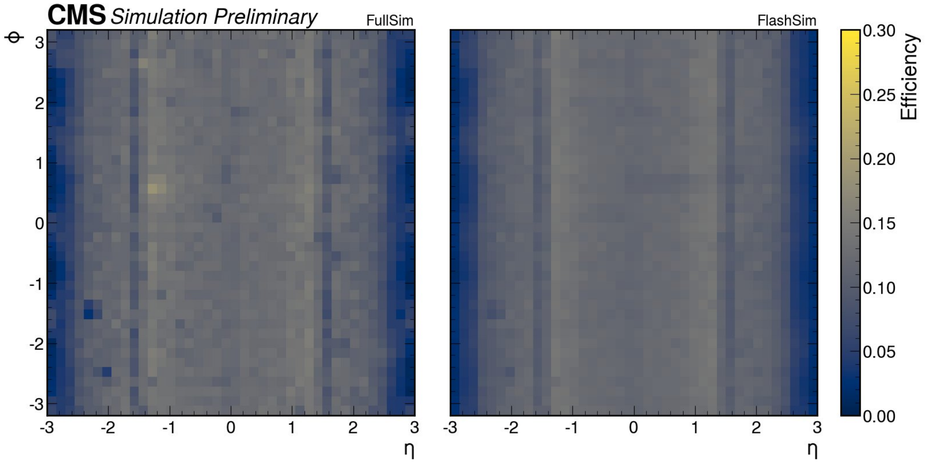


Photon from Jets

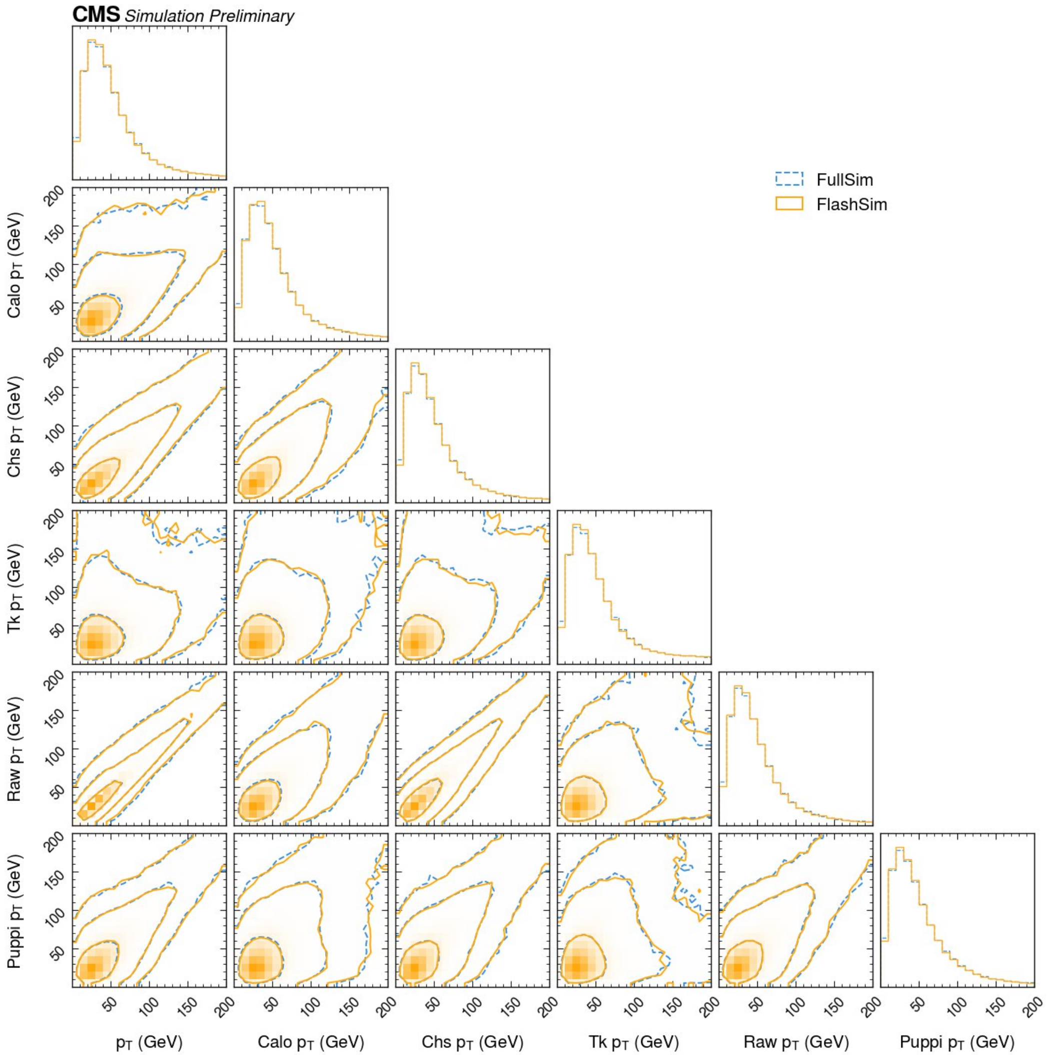




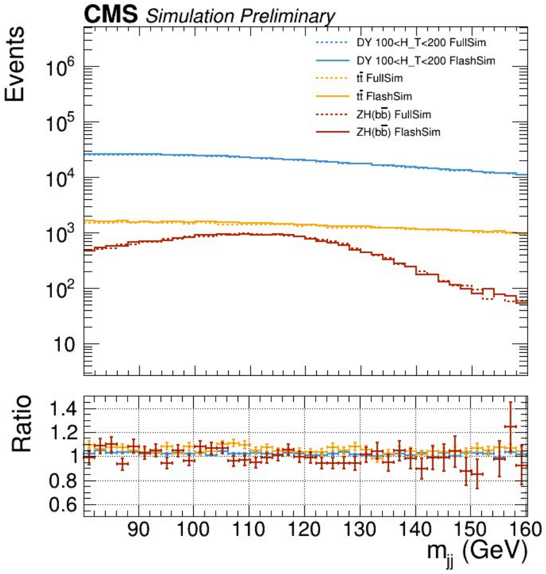


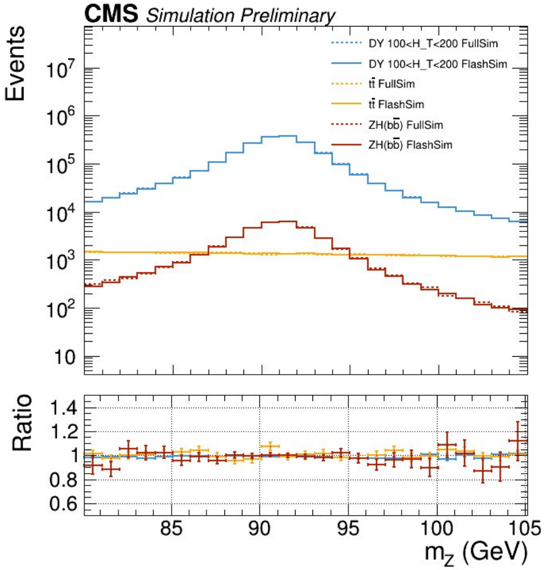


MET


CMS Simulation Preliminary
FullSim
FlashSim
Calo p_T (GeV)
Chs p_T (GeV)
Tk p_T (GeV)
Raw p_T (GeV)
Puppi p_T (GeV)
p_T (GeV)
50
100
150
200

Z(ll)H(bb)


CMS Simulation Preliminary
DY 100<H_T<200 FullSim
DY 100<H_T<200 FlashSim
$t\bar{t}$ FullSim
$t\bar{t}$ FlashSim
ZH($b\bar{b}$) FullSim
ZH($b\bar{b}$) FlashSim
Events
Ratio
m_{jj} (GeV)


CMS Simulation Preliminary
DY 100<H_T<200 FullSim
DY 100<H_T<200 FlashSim
$t\bar{t}$ FullSim
$t\bar{t}$ FlashSim
ZH($b\bar{b}$) FullSim
ZH($b\bar{b}$) FlashSim
Events
Ratio
m_Z (GeV)

VBF Higgs to mumu