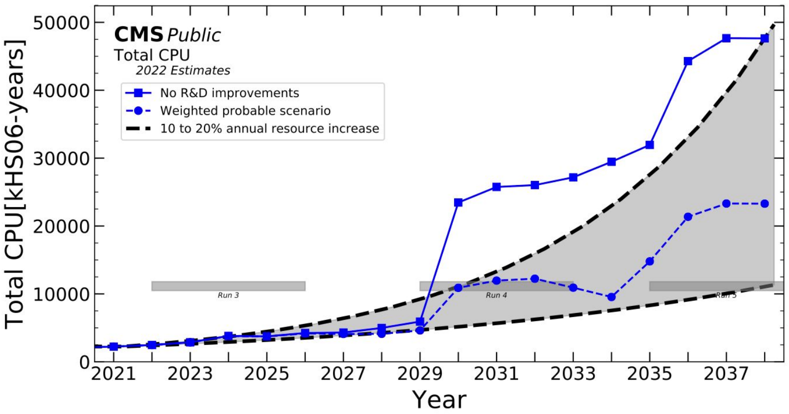
FlashSim: a ML simulation framework

Francesco Vaselli, Filippo Cattafesta, Gloria Cicconofri and Andrea Rizzi
On behalf of the **CMS Collaboration**

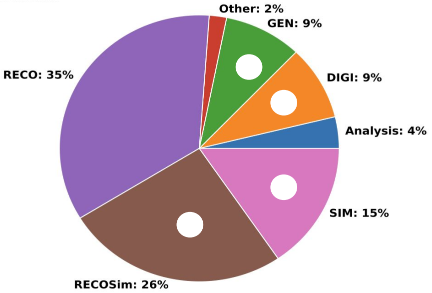
Faster simulation frameworks: Motivation



Event simulation is a non-negligible fraction of the total projected CPU need

Faster simulation frameworks are a part of the solution to the computing challenges posed by the HL-LHC era

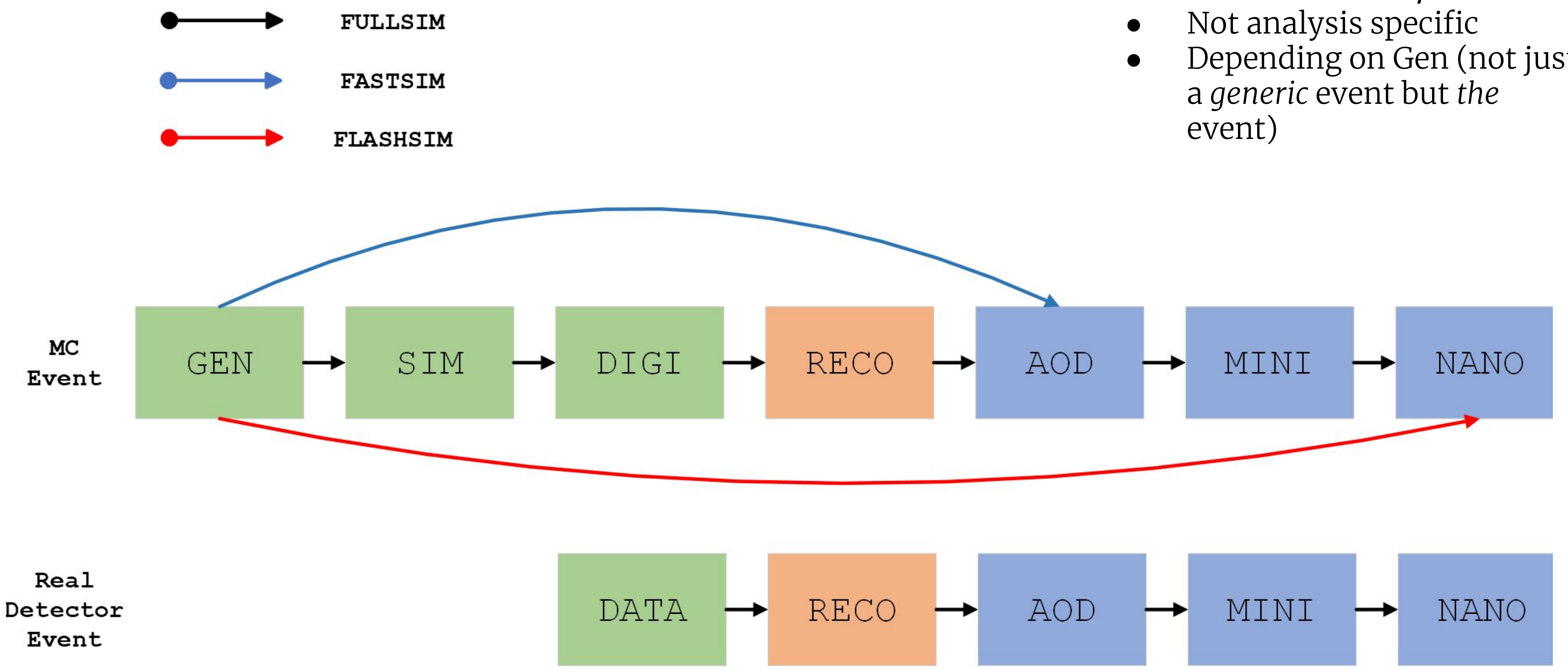
Machine learning is expected to provide both the speed and the accuracy we need



FlashSim means skipping all intermediate steps

We want something:

- fast(er):
  Reached 100 Hz/kHz!
- Not analysis specific
- Depending on Gen (not just a *generic* event but *the* event)

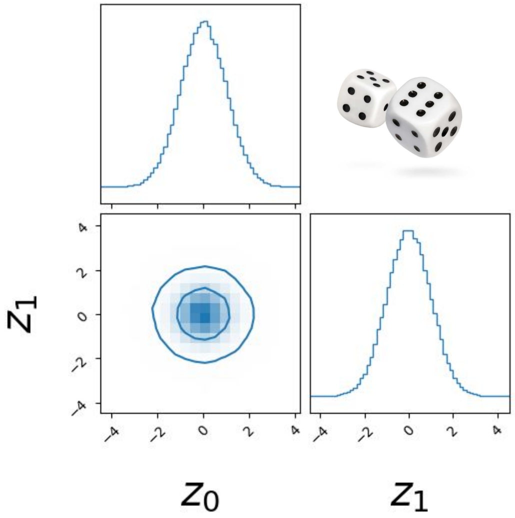


Outline

- Building blocks: *Normalizing Flows*
- Objects results:
  - AK4/AK8 Jets
  - Muons
  - Electrons
  - Fakes Jets
- Analysis test
- Fast Gen Variations
- Conclusions

Normalizing Flows: generative model for *pdfs*!

What we know

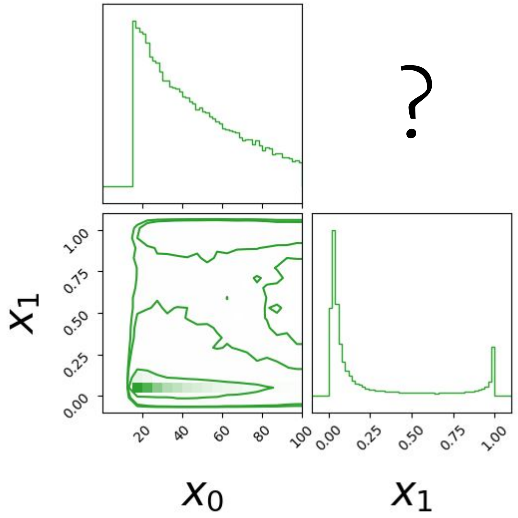


$\mathbf{z}$

multi-dimensional gaussian

$$f(\mathbf{z})$$

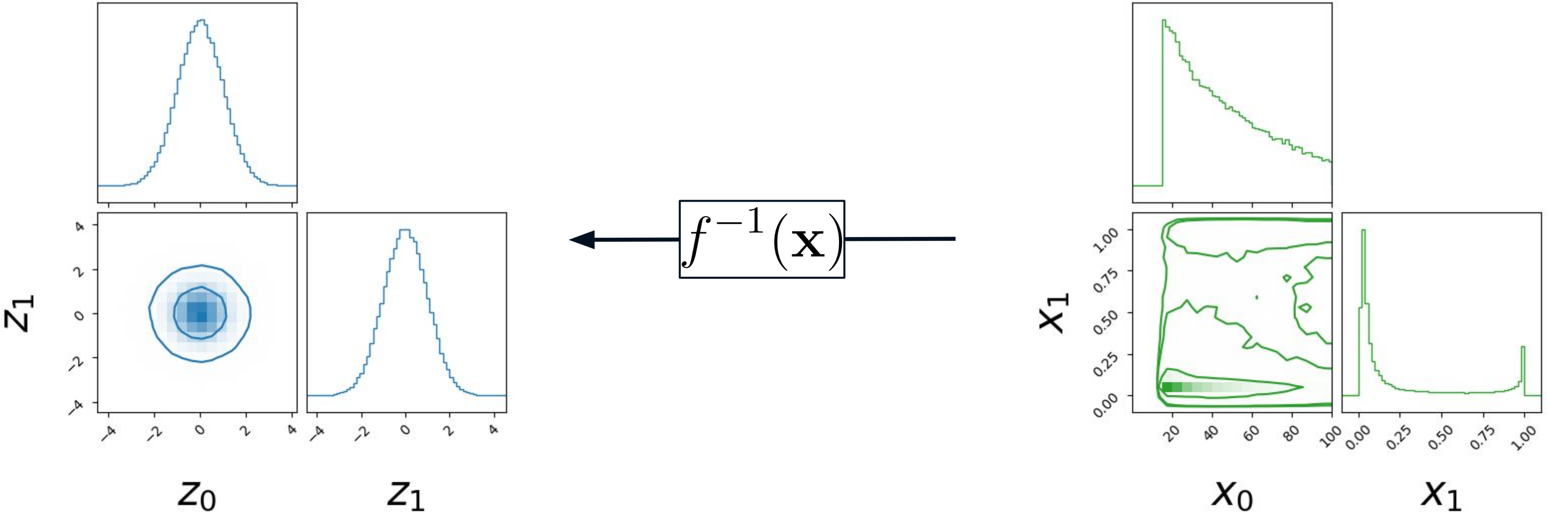
What we need



$\mathbf{x}$

FullSim data, pdf unknown!

Normalizing Flows: generative model for *pdfs*!



$$\mathbf{x} = f(\mathbf{z})$$

$$p_x(\mathbf{x}) = p_z(\mathbf{z}) \det \left| \frac{d\mathbf{z}}{d\mathbf{x}} \right|$$

$$\log(p_x(x)) = \boxed{\log(p_z(f^{-1}(\mathbf{x})))} + \boxed{\log\left(\det \mathbb{J}_{f^{-1}}(\mathbf{x})\right)}$$

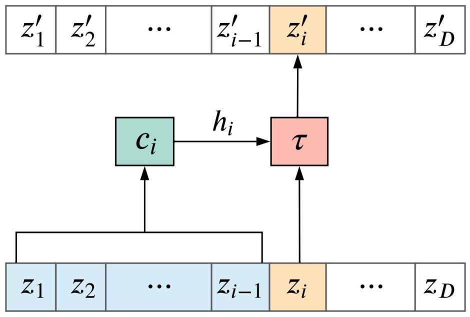
Flows building blocks: *conditioners*

How do we treat the input variables?
Want to correctly model correlations

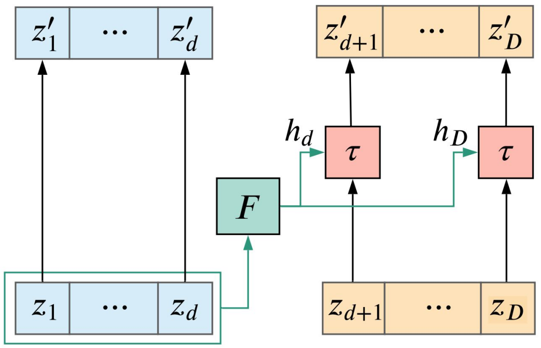
How do we use the other target variables
while transforming another?

Various way to do it!

Autoregressive:



Coupling:



Flows building blocks: *transforms*

How do we transform the variables?
Various ways to do it (as long as the transformation is invertible!)

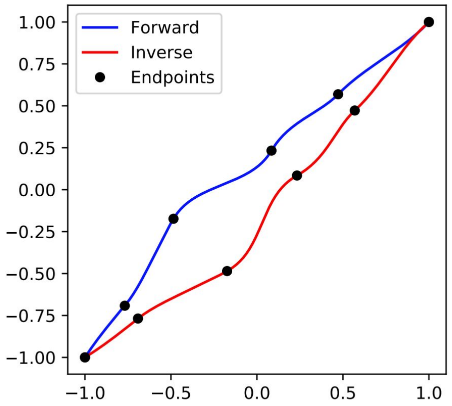
Each model is made up of multiple coupling+transformation blocks

This gives us an expressive final transformation with good correlations between variables

Affine:

$$\tau(\mathrm{z}_i; \boldsymbol{h}_i) = \alpha_i \mathrm{z}_i + \beta_i$$

Splines:



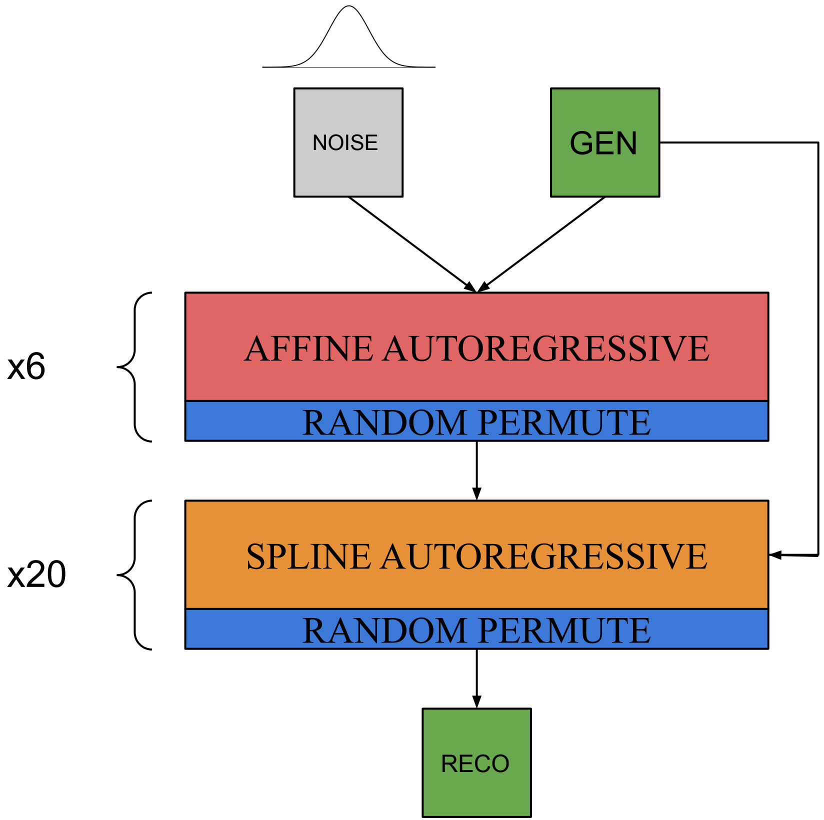
A *mixture* model as our baseline

Conditioning gen variables: 38
Target variables: 47

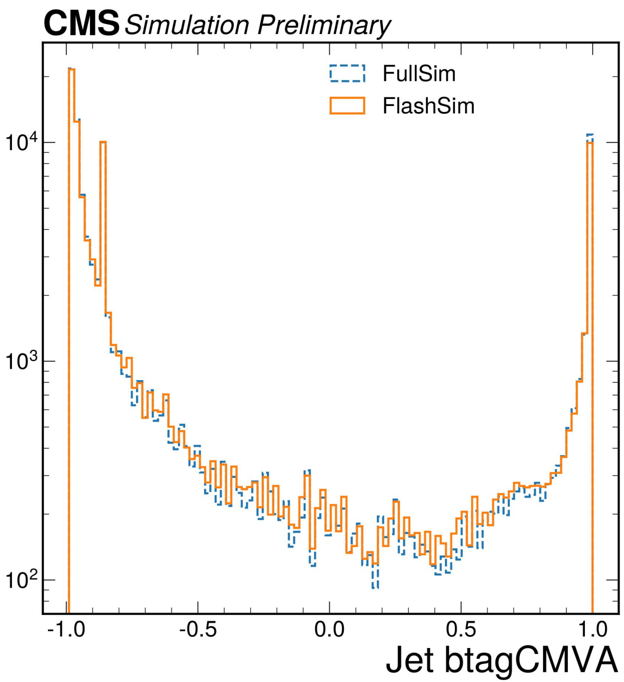
Total params: O(10e6)

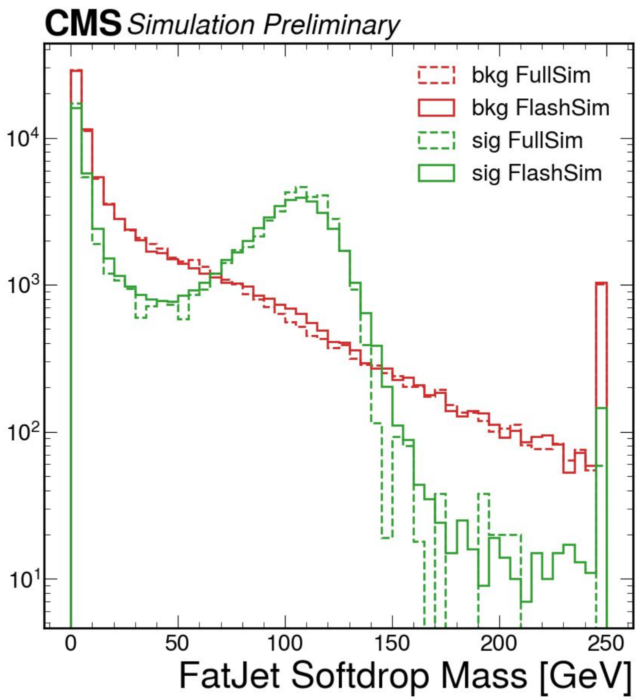
Trained on 5e6 objects from ttbar events

Newer model are based only on affine layers, good results with smaller dims



AK4 and AK8 Generator-matched Jets: good closure on 1d distributions



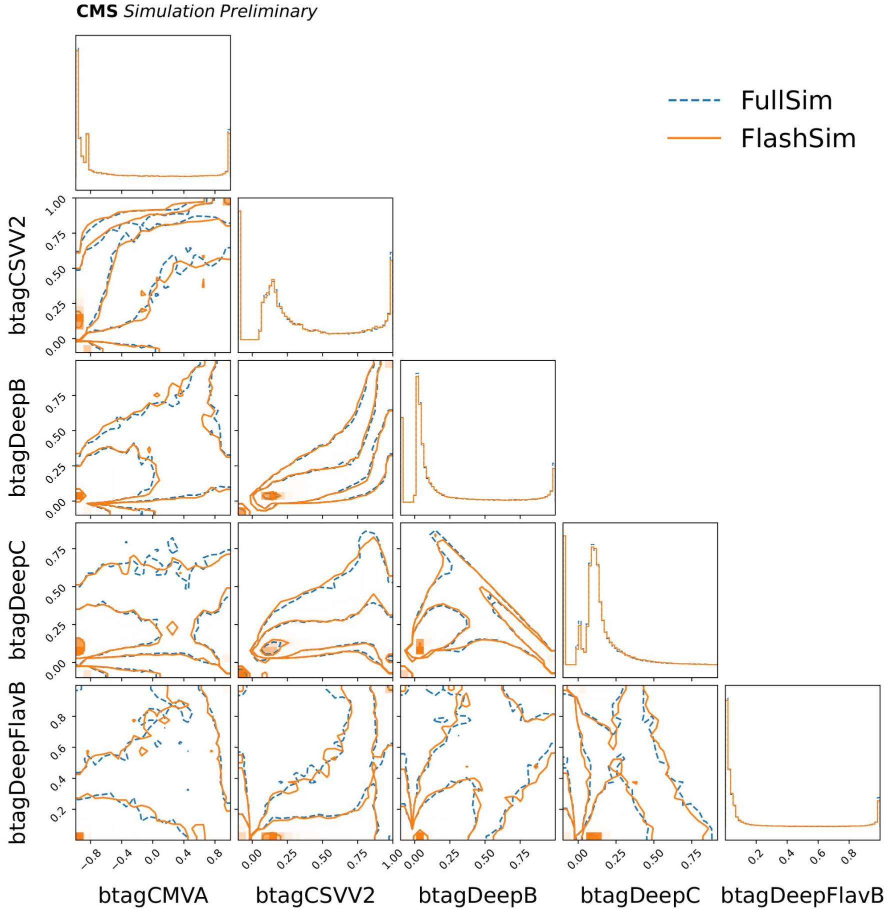


AK4 Generator-matched Jets: accurate correlations

The corner plot shows multivariate tagging algorithms

B-tag, C-tag algorithms relying on similar jet features

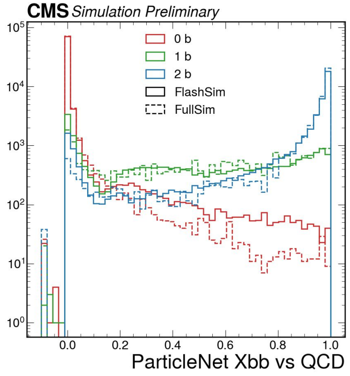
Even without specific information,
FlashSim learns them!

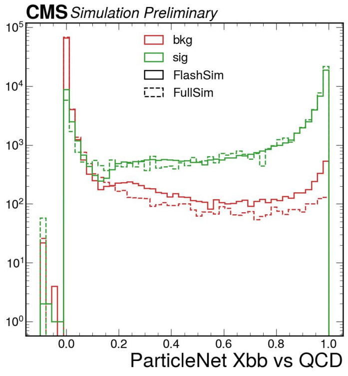


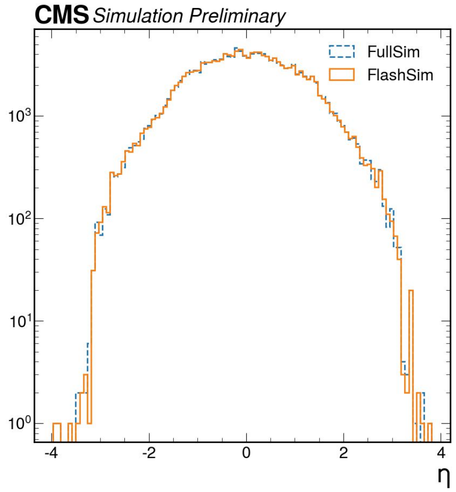
FatJets results

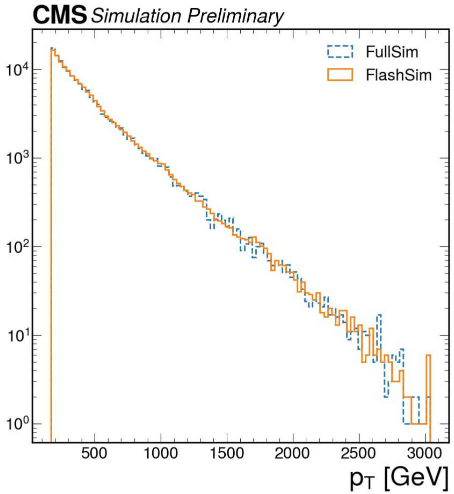
Trained of Phase 2 QCD multijet events and HH → 4b

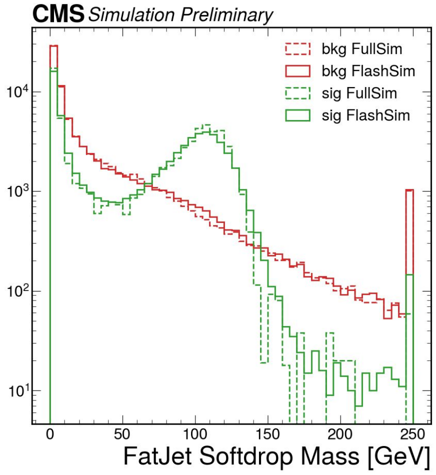
Smaller training sample compared to Run II samples



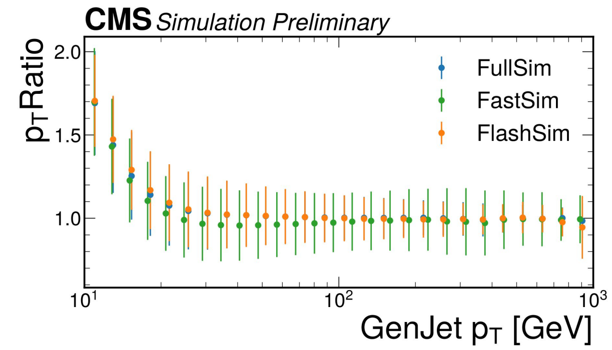


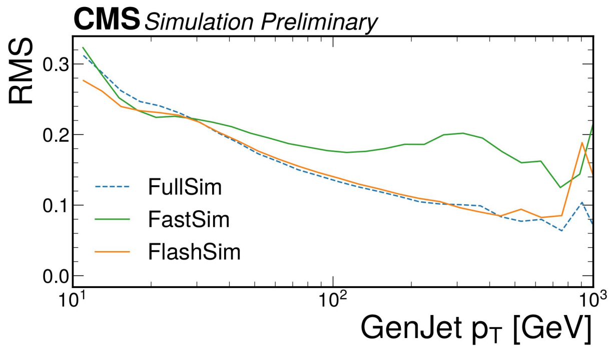


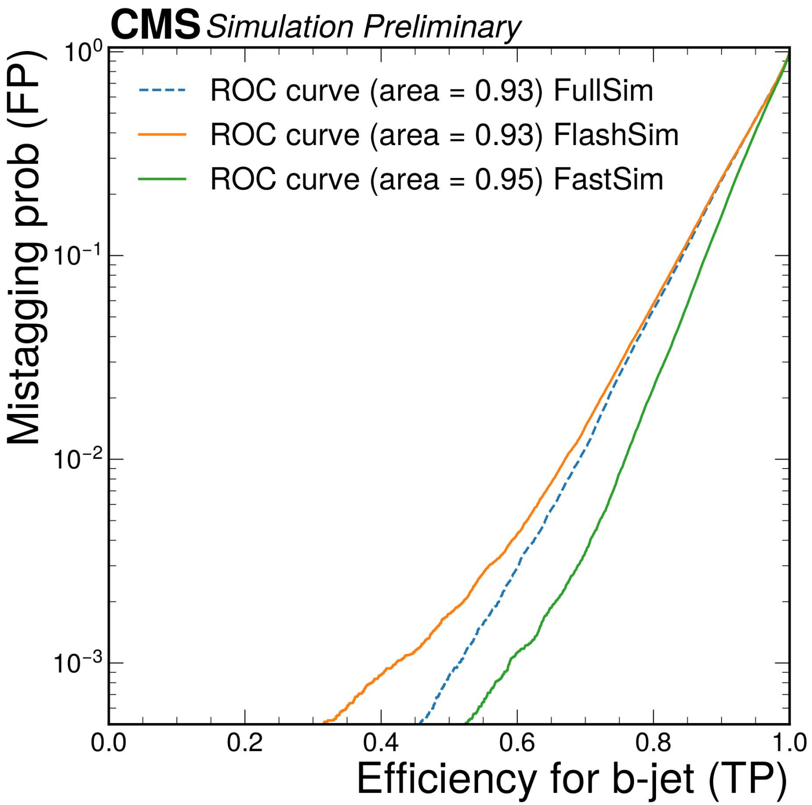




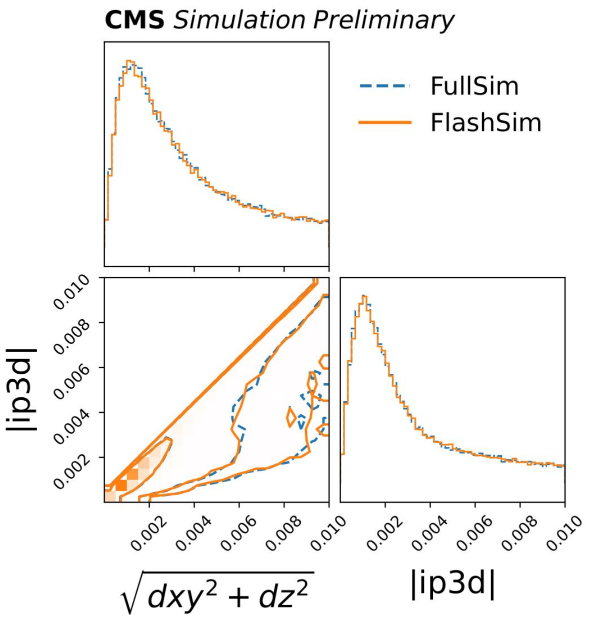
AK4 Generator-matched Jets *conditioning*

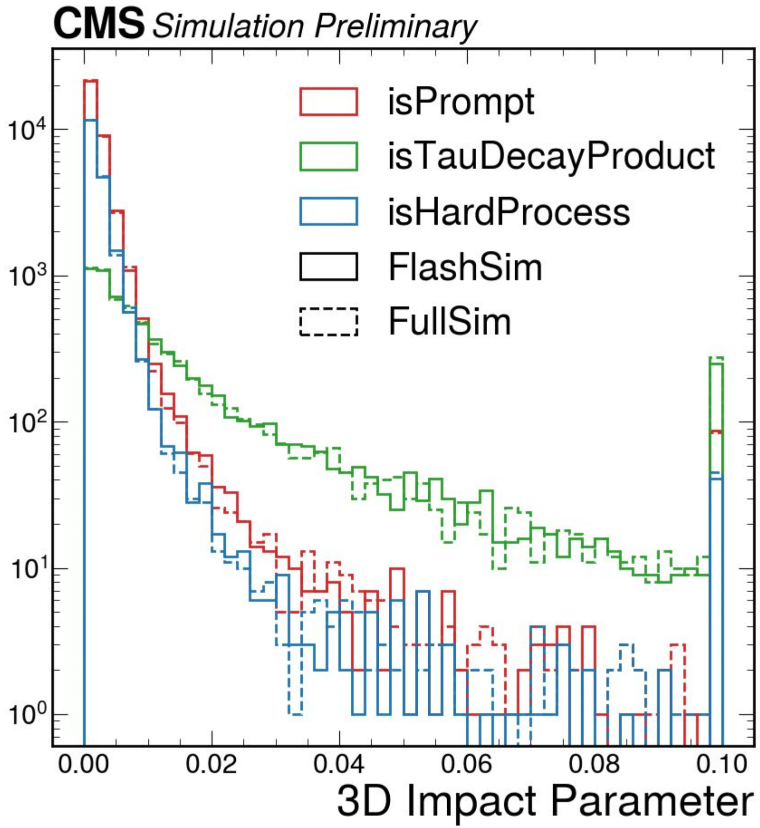




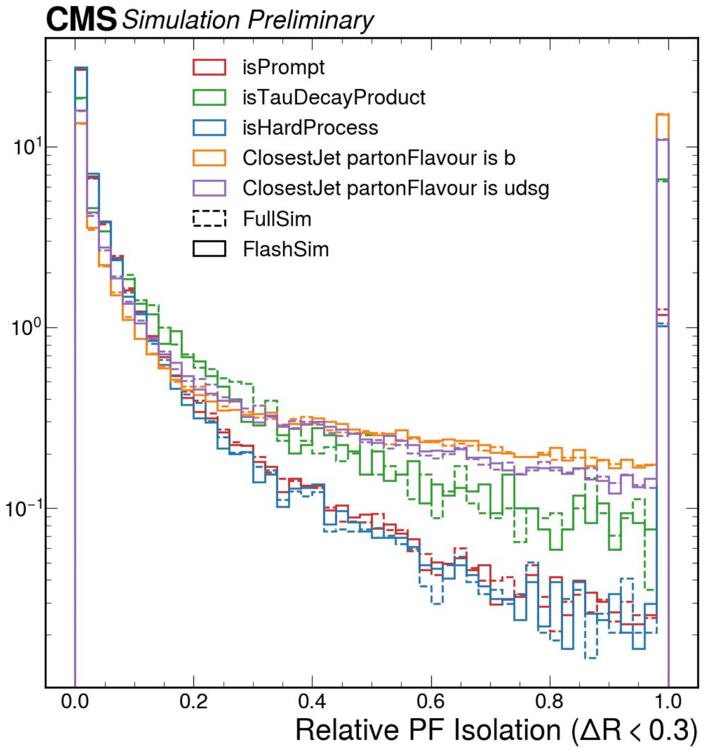


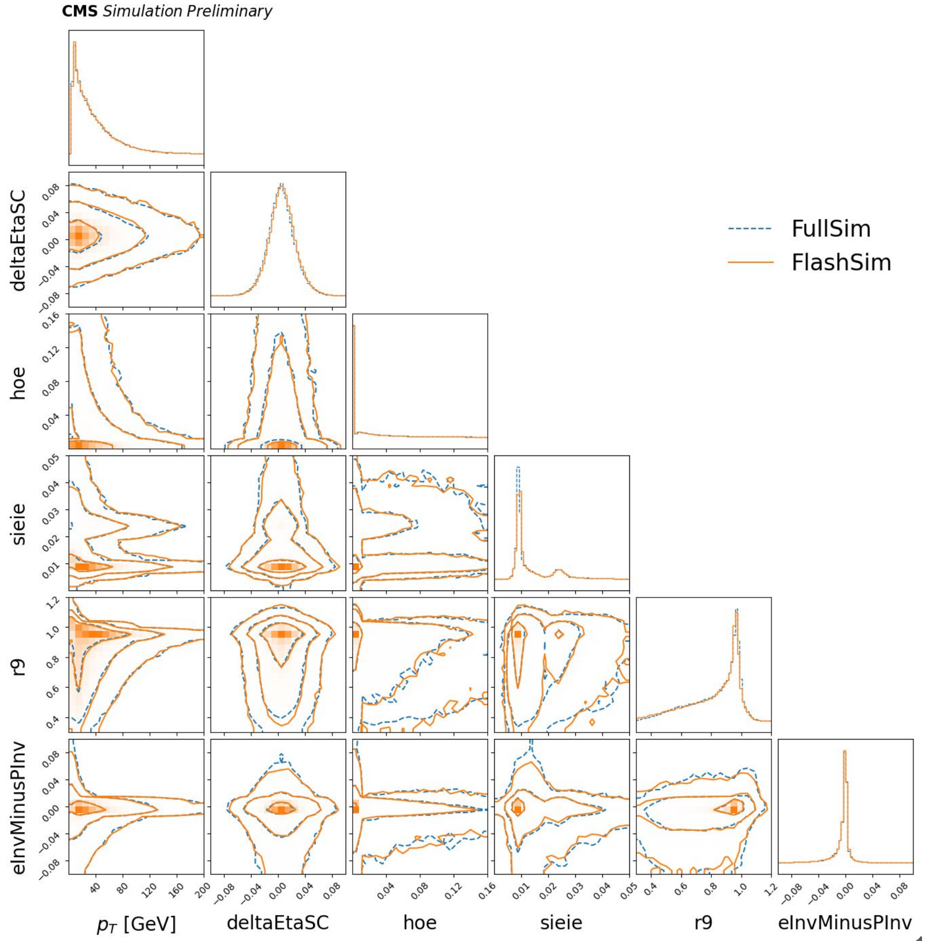
Promising results on Muons





Promising results on Electrons



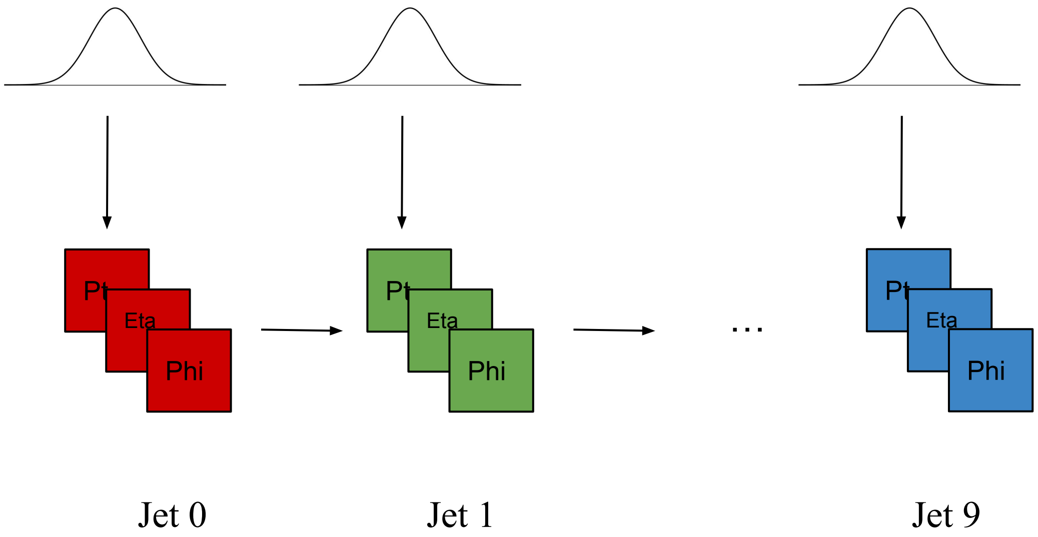


Fakes are a different type of problem: need to experiment

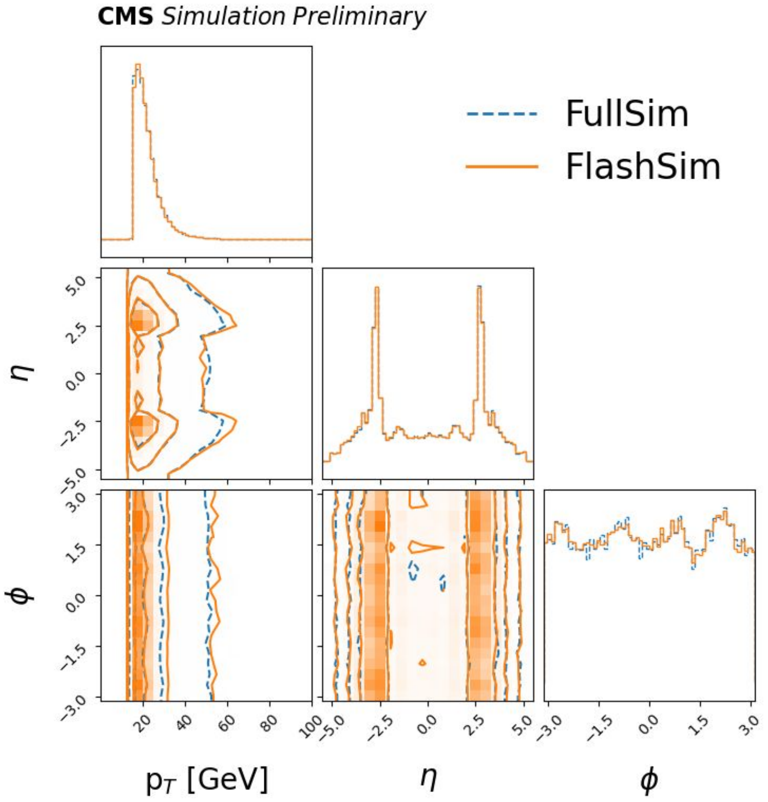
For fake jets there is no "generator" starting point to use!

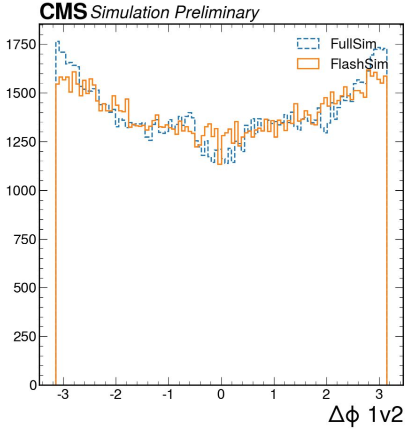
Autoregressive chain on a variable number of fake jets per event

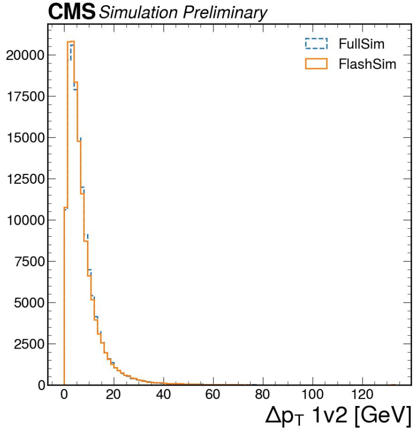
(pT, eta, phi) of jet N depends on those of jets N-1



Fake jets results







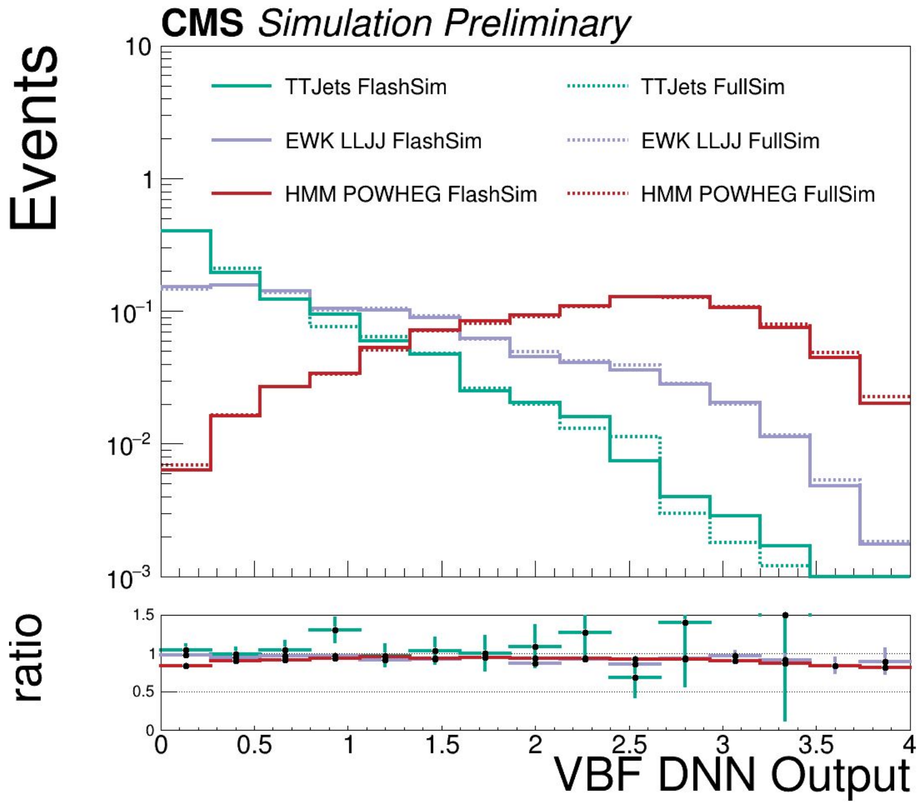
A real-world scenario: Analysis test

Simulate Jets, Muons on 4 different processes: TTJets, DY+Jets, EWK LLJJ, $H \to \mu^+\mu^-$

Perform VBF $H \to \mu^+\mu^-$ analysis selection

Compute derived quantities

Compute the DNN score used in the final fit!

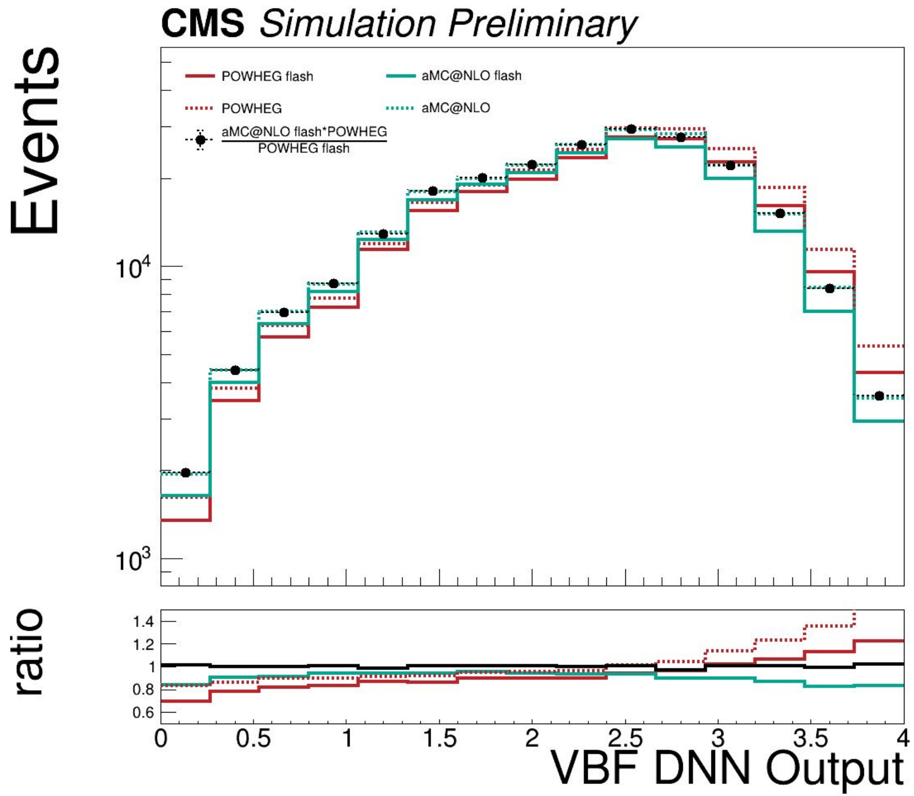


FlashSim for Generator-level variations

Given a FullSim sample with generator e.g. POWHEG
We can compute:

$$VAR_{Full} = REF_{Full}\frac{VAR_{Flash}}{REF_{Flash}}$$

To get quickly another precise sample for a different generator such as aMC@NLO



Conclusions

We compared results with Run II FullSim/ Phase-2 samples for:

- AK4/AK8 Jets
- Muons
- Electrons
- Fakes Jets
- Analysis use case

Still need to:

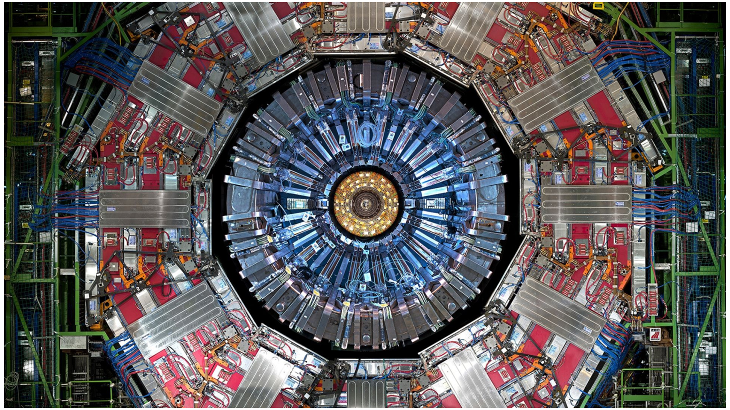
- R&D base models
- Cover missing Objects
- Crack Fakes

So far, **a fully-ML powered FlashSim simulation frameworks seems achievable!**
Speed of kHz/100Hz

A Public CMS note is available with all the results: https://cds.cern.ch/record/2858890

First concept repo: https://github.com/francesco-vaselli/FlashSim

Get in touch!
francesco.vaselli@cern.ch



Backup

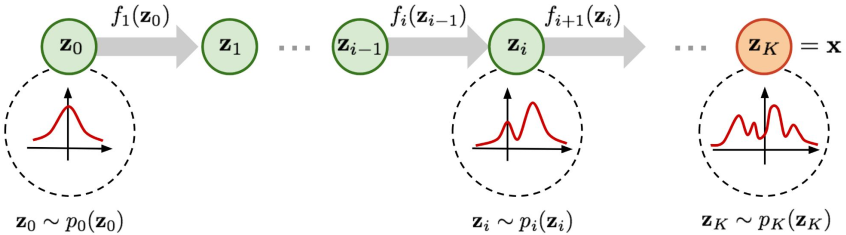
- Normalizing Flows in detail

Normalizing Flows in detail

Learn the *f(z)* to send $p_{\mathbf{z}}(\mathbf{z})$ into the (*unknown*) data distribution $p_{\mathbf{x}}(\mathbf{x})$

Needs:

- Basic distribution $p_{\mathbf{z}}(\mathbf{z})$, typically Gaussian
- Function called flow *f(z) invertible* and *differentiable*, with *tractable jacobian* (splines are a good choice)



$$p_x(\mathbf{x}) = p_z(f^{-1}(\mathbf{x}))\det\left|\frac{d\mathbf{z}}{d\mathbf{x}}\right|$$

$$\log(p_x(x)) = \log(p_z(f^{-1}(\mathbf{x}))) + \log\left(\det \mathbb{J}_{f^{-1}}(\mathbf{x})\right)$$