PARTICLE CLOUD GENERATION WITH MESSAGE PASSING GANS

Raghav Kansal

ML at GGI Conference
09/09/2022

- LHC Simulations
- Deep generative models
    - Evaluation metrics
    - Data representations
- Current applications
    - MPGAN
- Discussion

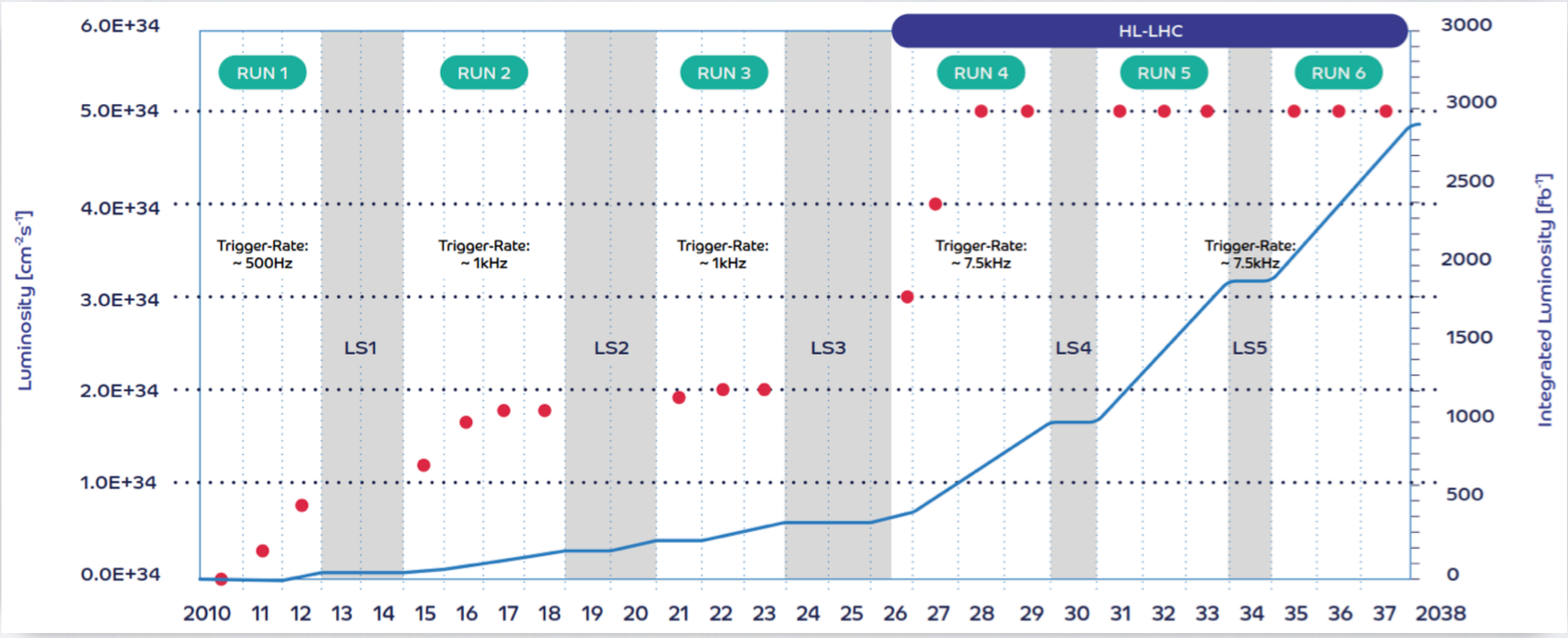
- LHC Simulations
- Deep generative models
  - Evaluation metrics
  - Data representations
- Current applications
  - MPGAN
- Discussion

LHC SIMULATIONS

Sources
K. Pedro, HSF 2020
J. Duarte, ANL 2021, Video

- Full detector simulation takes ~40% of grid CPU resources

*Delayed



- HL-LHC looming
  - Order-of-magnitude more simulations needed
  - Improved detectors ⇒ higher granularity, increased complexity
  - ML a possible solution?

LHC SIMULATIONS*

Sources
[S. Sekmen, LPC 2017]
[F. Krauss, Kyoto 2011]

Hard process
Showering
Hadronization
Underlying event

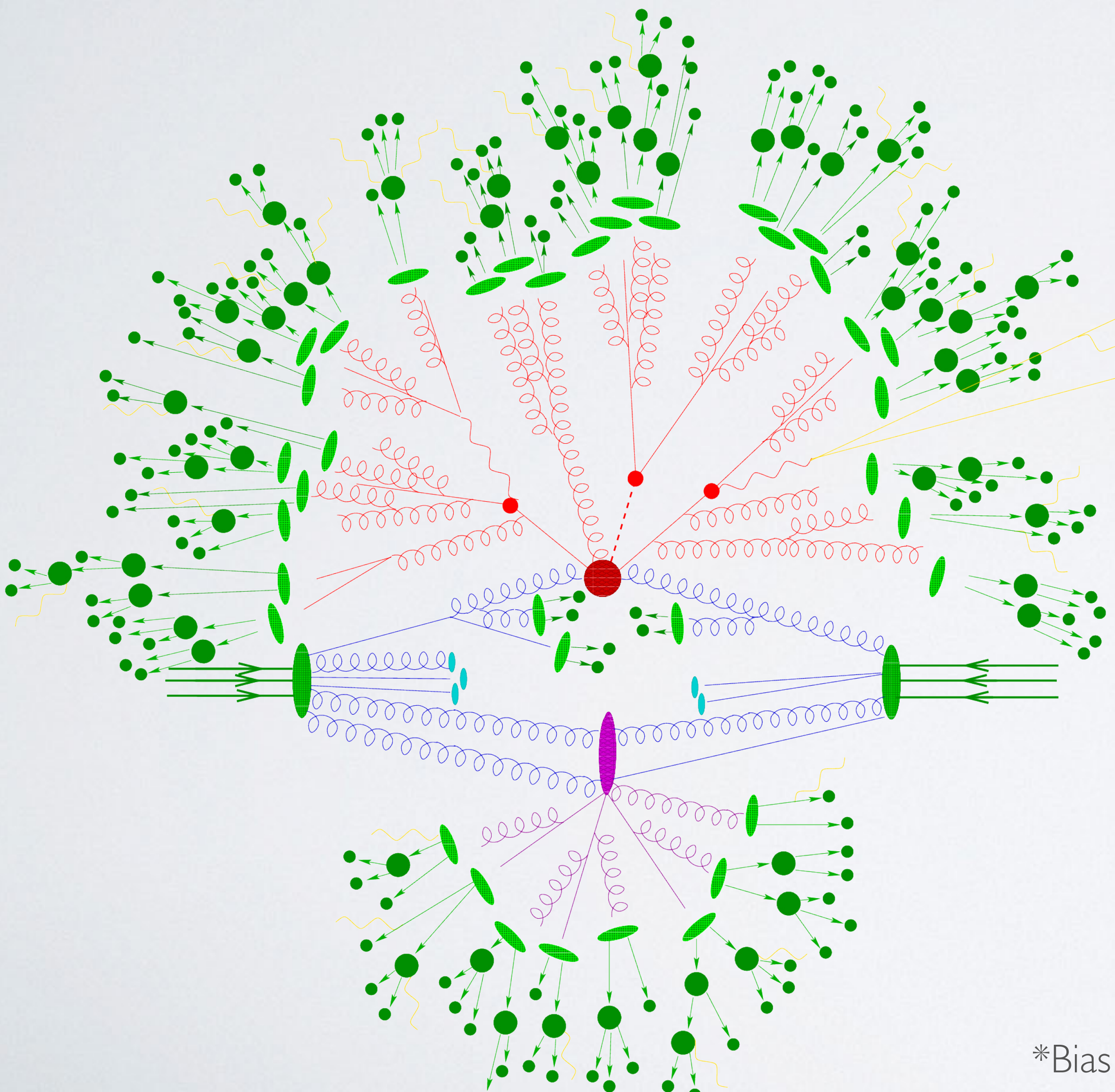
→ GenParticles → SimHits → RecoHits → Reco Particles → High level features

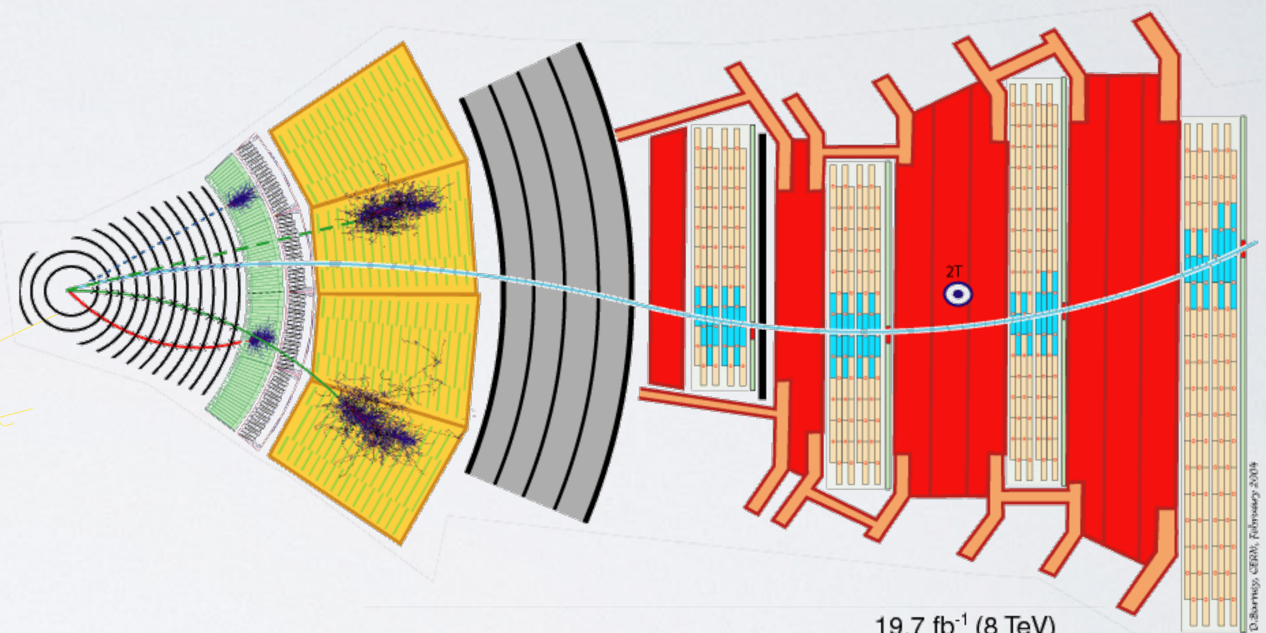
Energy depositions and interactions along detector

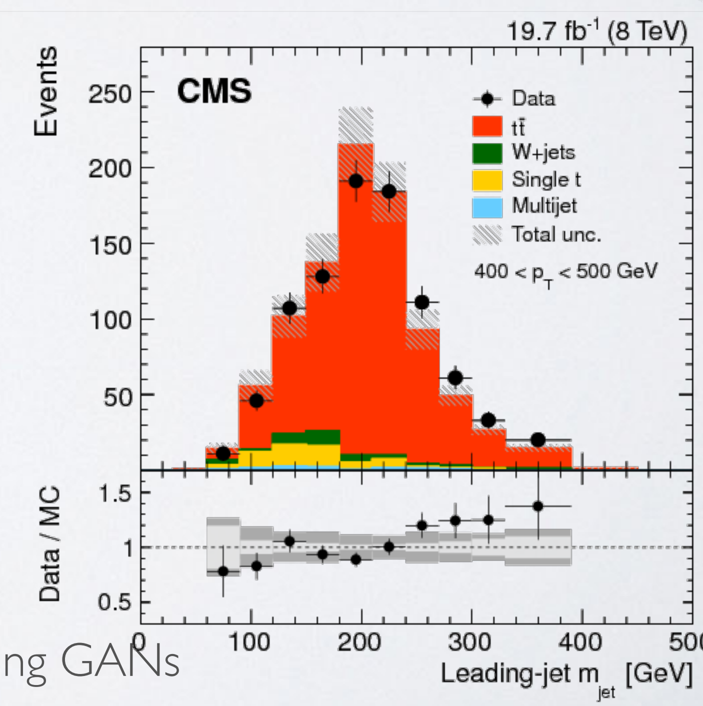
Digitization
Hit reconstruction

Particle reconstruction

Event-level analysis

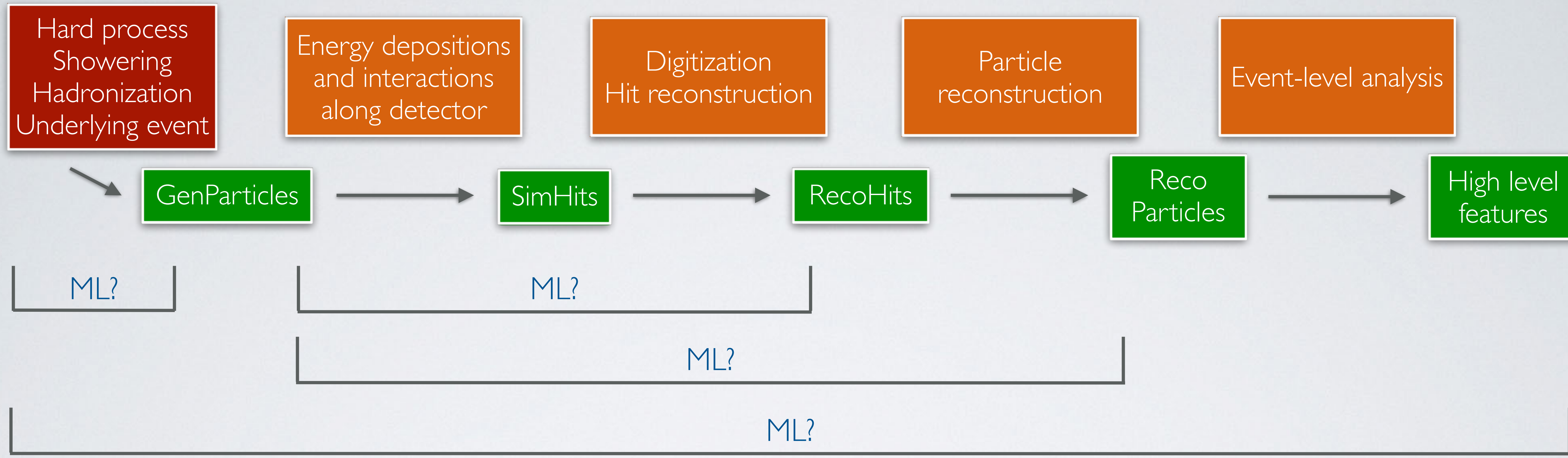




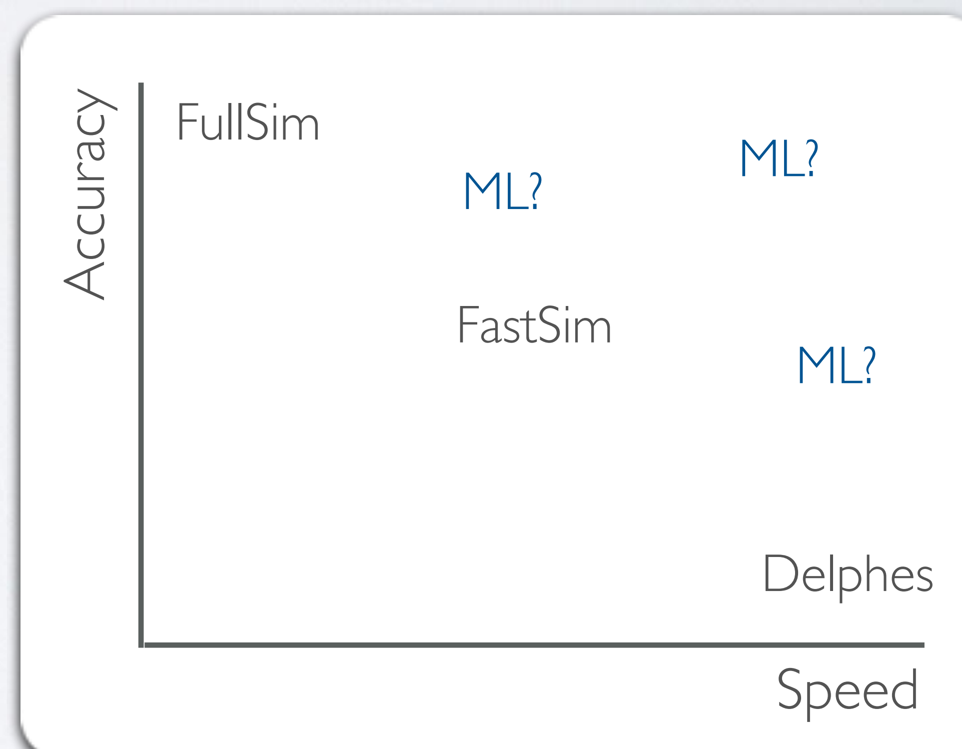


*Bias towards CMS

LHC SIMULATIONS

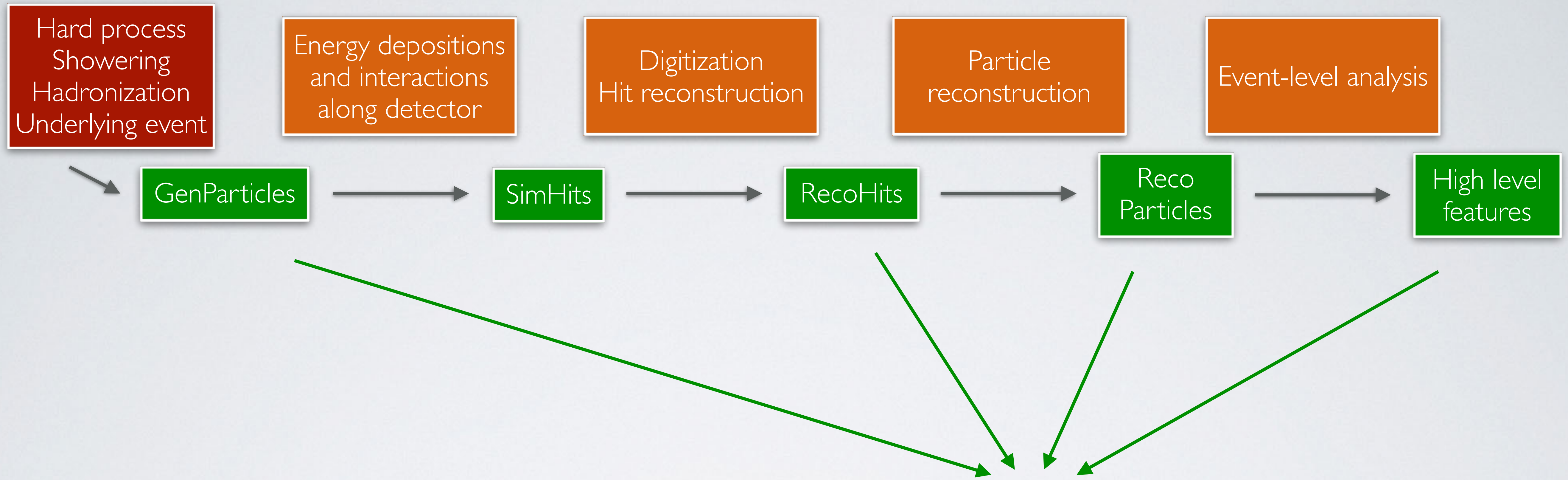


- Opportunity for ML alternatives in many steps
- Trading accuracy of "FullSim" (Geant) for speed
- Trading verifiability/trust for # of steps



K. Pedro, HSF 2020

LHC SIMULATIONS



- Want model $p_\theta(\mathbf{x})$ for underlying data distribution $p(\mathbf{x})$

- Rich area in machine learning: **Deep generative models**
  - deep neural networks are flexible and expressive
  - $p_\theta(\mathbf{x})$ typically modelled with high-capacity DNNs

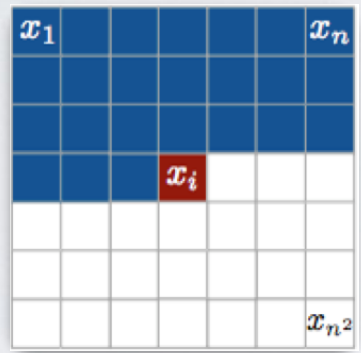
- LHC Simulations
- Deep generative models
  - Evaluation metrics
  - Data representations
- Current applications
  - MPGAN
- Discussion

DIRECT MODELLING

Explicit parametric specification of $p_\theta(x)$

Autoregressive

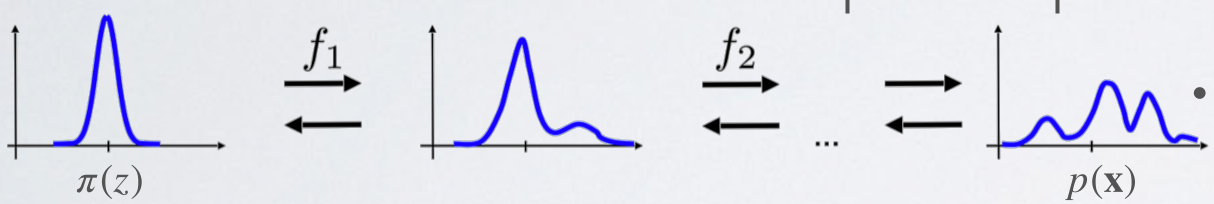
$$p(\mathbf{x}) = p(x_1)\prod_{i=2}^{N-1} p(x_i \,|\, x_{<i})$$



- Model assumption: x_i can be predicted entirely by $x_{<i}$
- Iteratively output x_i based on $x_{<i}$ using learnt $p(x_i \,|\, x_{<i})$

Flow-based

$$p(\mathbf{x}) = \pi\left(\mathbf{x} = f^{-1}(z)\right)\left|\frac{\partial f^{-1}(\mathbf{x})}{\partial \mathbf{x}}\right|$$

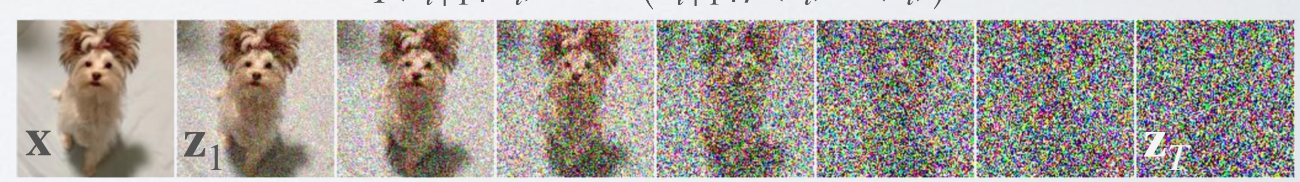


- Backward: learn transformation from $p(\mathbf{x})$ to simpler $\pi(z)$
- Forward: sample $\pi(z)$, **invert** transformation, get $p(\mathbf{x})$

Diffusion-based

- Backward: iteratively add gaussian noise
- Forward: $p(\mathbf{z}_i \,|\, \mathbf{z}_{i+1})$ network learns to denoise

$$q(\mathbf{z}_{i+1} \,|\, \mathbf{z}_i) = \mathcal{N}\left(\mathbf{z}_{i+1} \,|\, \mu(\mathbf{z}_i), \sigma(\mathbf{z}_i)\right)$$



$$p(\mathbf{x}) = p(\mathbf{x} \,|\, \mathbf{z}_1)\left(\prod_{i=1}^{T-1} p(\mathbf{z}_i \,|\, \mathbf{z}_{i+1})\right)p(\mathbf{z}_T)$$

1. Access to exact likelihood
2. Simple $-\ln p(\mathbf{x})$ loss
3. Stable training

- **But** in practice they are typically outperformed by GANs (next slide)

LATENT VARIABLE MODELS

- Assume high dimensional data $\mathbf{x}$ can be characterised by lower dimensional 'latent' (hidden) features $\mathbf{z}$
- Generative process: sample from simpler prior $\mathbf{z} \sim p(\mathbf{z})$ and learn $p(\mathbf{x}\,|\,\mathbf{z}) \Rightarrow p(\mathbf{x}) = \int p(\mathbf{x}\,|\,\mathbf{z})p(\mathbf{z})d\mathbf{z}$

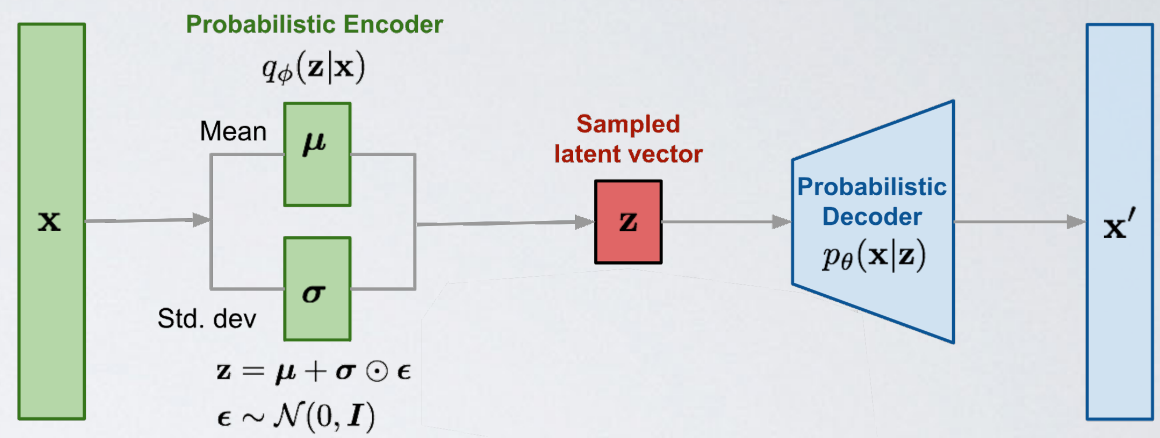
Variational Autoencoder

- 'Evidence-based Lower BOund' (ELBO) of $\ln p(\mathbf{x})$:

$$\mathrm{ELBO} = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}[\ln p(\mathbf{x}\,|\,\mathbf{z})] - \mathrm{KL}\left[q_\phi(\mathbf{z}\,|\,\mathbf{x})\,||\,p(\mathbf{z})\right]$$

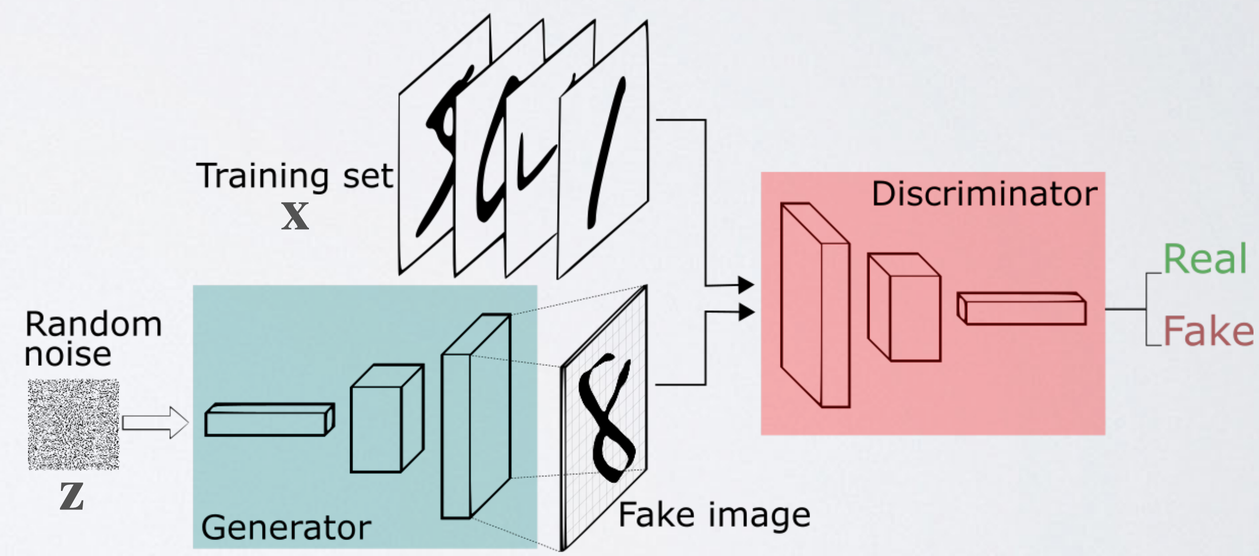
Autoencoder reconstruction loss

Divergence between variational posterior and assumed prior



Generative Adversarial Networks

- Abandon likelihood-based loss approach
- Iteratively train 'discriminator' network as an **adversarial loss** for the 'generator'
- 1) Hard to train; 2) lose likelihood; 3) adversarial; **but** when done right tends to be most performant*



- ***UNTIL** last year where **score-based** diffusion models started beating GANs for the first time!
- No time in this talk but very interesting direction - modelling $\nabla_{\mathbf{x}} \ln p(\mathbf{x})$ instead of $p(\mathbf{x})$

- How do we **trust** generated data?

- How do we compare generative models?

- How do we **trust** generated data? **Evaluation metrics**

- How do we compare generative models? **Evaluation metrics**

- How do we **trust** generated data?

  Evaluation metrics sensitive to:

  - Quality
  - Diversity
  - Physics performance (interpretable)

- How do we compare generative models?

  Evaluation metrics that are:

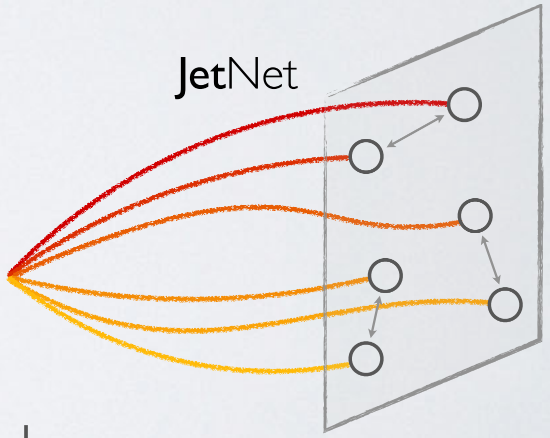
  - Standardised
  - Reproducible
  - Efficient

EVALUATION METRICS

- Most important aspect of generative modelling

- We propose two key metrics: Kansal et al., NeurIPS 2021
  - Physics-inspired: **1-Wasserstein** (W_1) distances between distributions
  - ML-inspired: adapt established **Fréchet Inception Distance** metric from CV

- Practically, found to together satisfy all criteria very effectively



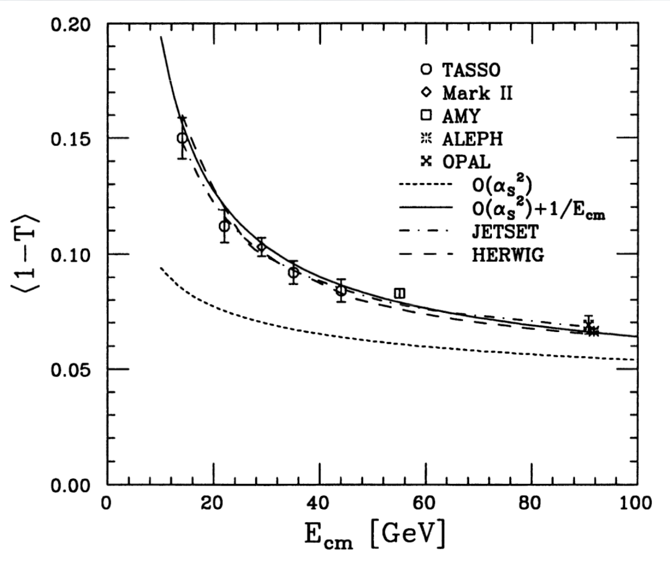
- In effort to **standardise**, we release these, and more, in JetNet package

W_1 DISTANCES

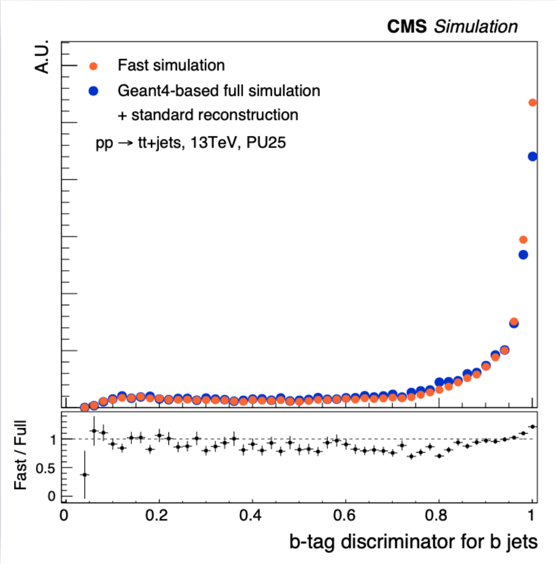
Kansal et al., NeurIPS 2021

- Traditional method for evaluating physics simulations is to compare physical distributions

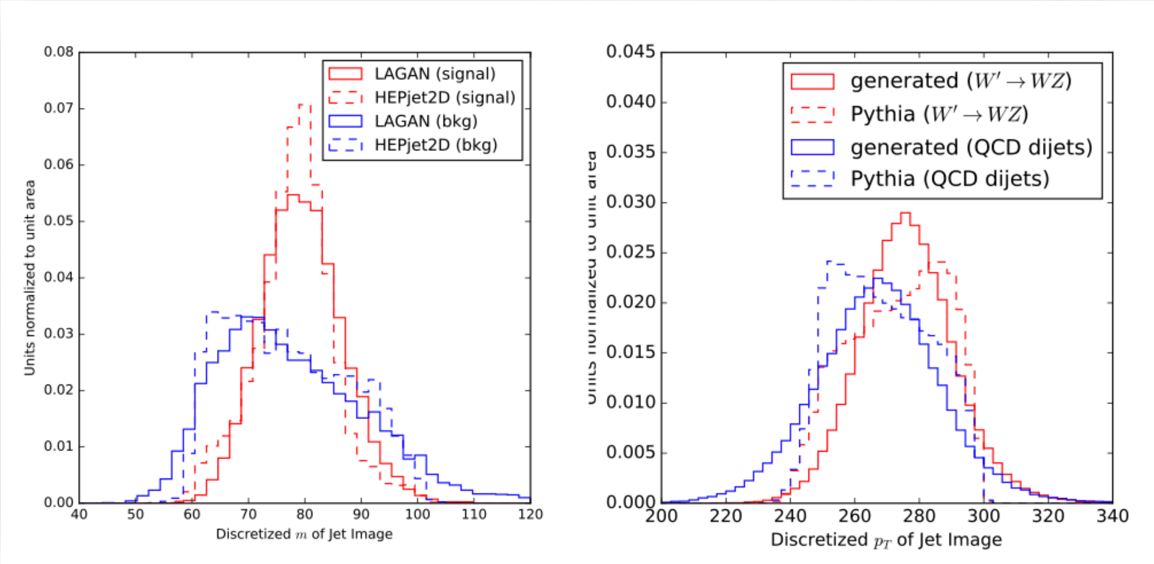
MC generator evaluation (Ellis et al '96)



FastSim (Sekmen '17)



LAGAN (de Oliveira et al '17)



- Proposal: quantify using 1-Wasserstein (earth mover's) distance (W_1)
  - Can evaluate multiple low- and high-level features: sensitive to **quality**
  - High scores for differing supports: sensitive to mode collapse (**diversity**)
  - **Efficient, Reproducible, Interpretable**
  - Can use boot-strapping with subsets of only real samples to derive **baseline**
- Cons: scaling to more dimensions (missing correlations), how to aggregate scores for different features?

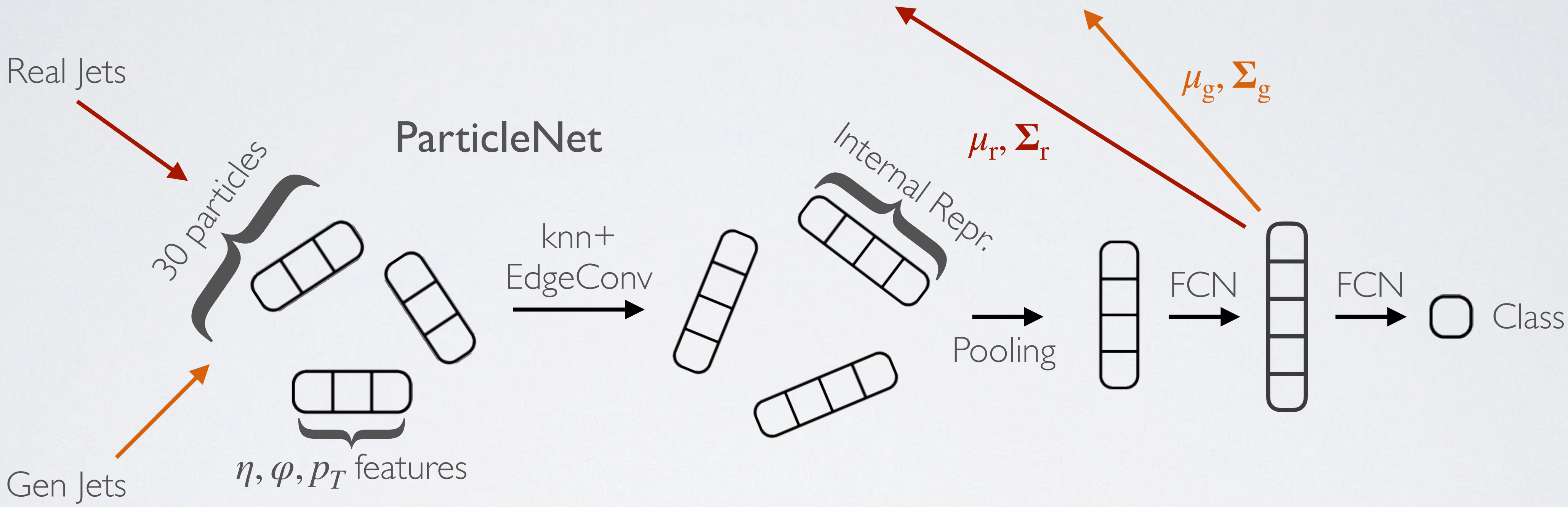
FRÉCHET <CLASSIFIER> DISTANCES

- Machine learning version of this: use classifier hidden features instead!

Kansal et al., NeurIPS 2021

- Example: apply to jet generation using pre-trained ParticleNet graph classifier:

$$\mathrm{FPND} = \mathrm{Frechet}(\mathcal{N}(\mu_\mathrm{r}, \boldsymbol{\Sigma}_\mathrm{r}), \mathcal{N}(\mu_\mathrm{g}, \boldsymbol{\Sigma}_\mathrm{g})) = ||\mu_\mathrm{r} - \mu_\mathrm{g}||^2 + \mathrm{Tr}\left[\boldsymbol{\Sigma}_\mathrm{r} + \boldsymbol{\Sigma}_\mathrm{g} - 2(\boldsymbol{\Sigma}_\mathrm{r}\boldsymbol{\Sigma}_\mathrm{g})^{1/2}\right]$$

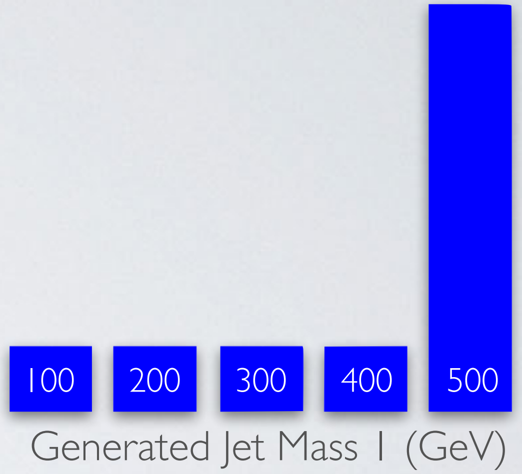


- High-performing classifier learns salient hidden features from data
- Retain sensitivity to **quality, diversity** from W_1, **reproducible** and **efficient** plus:
  - Single aggregate score, correlations (Σ) between features, easy to scale
  - But lose interpretability, hence used in conjunction with W_1 scores

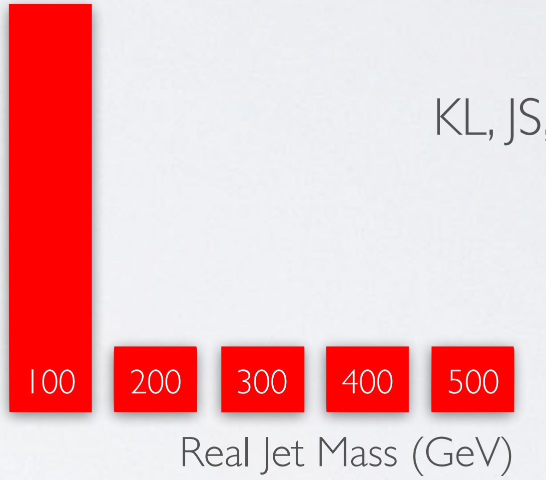
ALTERNATIVES

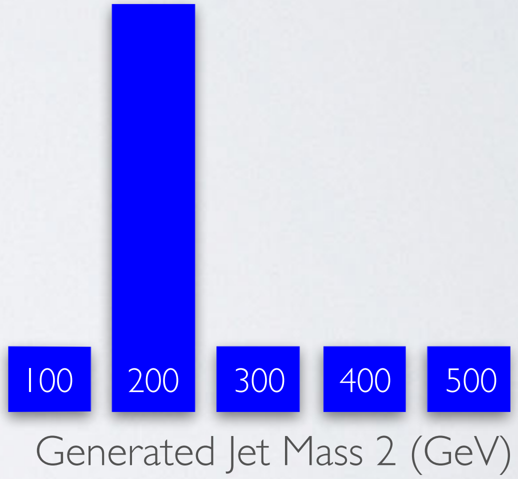
- **KL vs JS vs χ^2 vs W_1**
  - All reasonable, χ^2 particularly ubiquitous for GoF
  - Only W_1 takes account of metric space

Source



KL, JS, χ^2 for both is the same



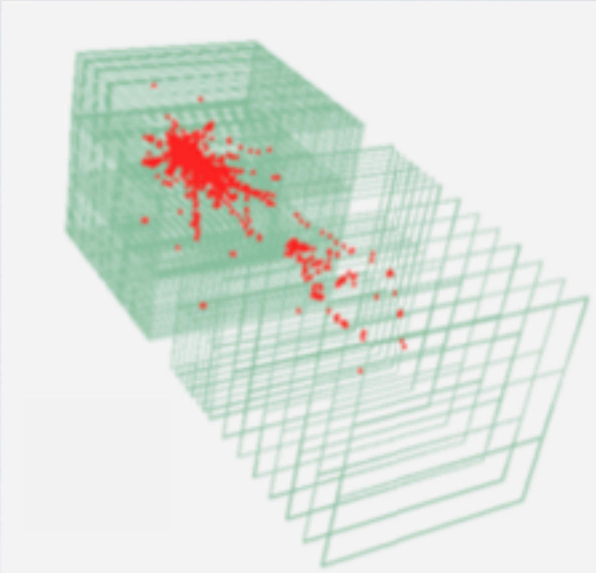


- **Classifier Metric**
  - Train a classifier between real and fake
  - Pros: **quality** and **diversity**
  - Cons: **Interpretability**, **reproducibility**, **efficiency**, hard to **standardise**

DATA REPRESENTATIONS

- Properties of LHC data:
  - Sparsity
  - High granularity
  - Irregular geometry
  - No fixed ordering
- Point clouds:
  - Store only the hits/particles
  - Retain feature precision
  - Are flexible, work for any geometry
  - Have no ordering

$\Rightarrow$ Natural representation for HEP data



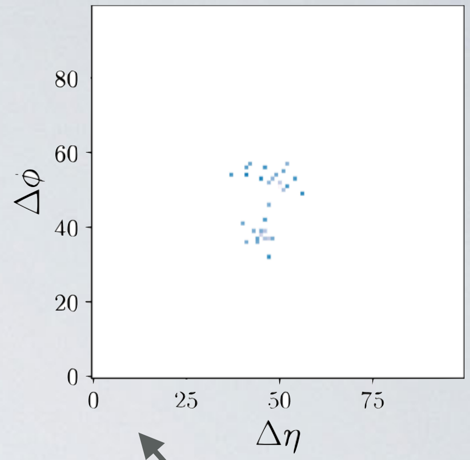
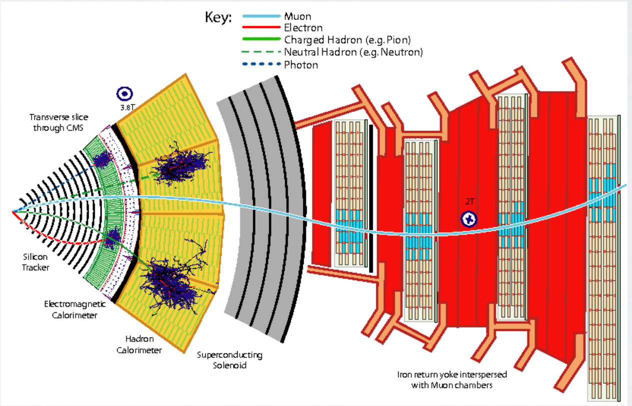


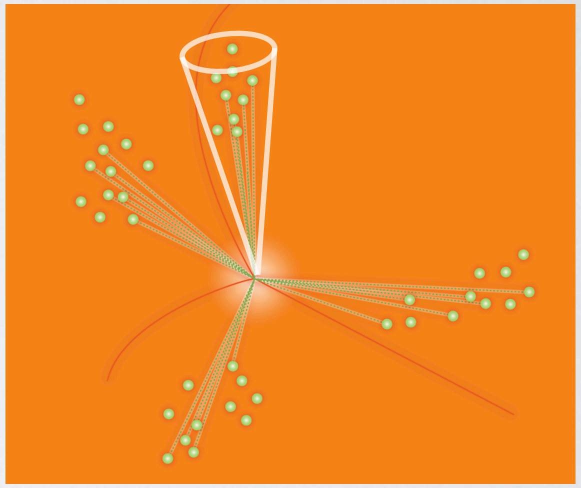
Image repr. for CNNs

Hits

Particles



Hit coordinates, energy



Particle coords, momentum

Row entry

Energy/p_T node features

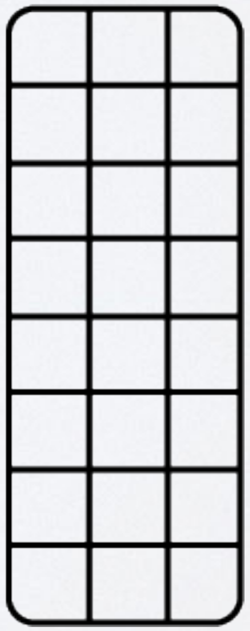
η φ p_T

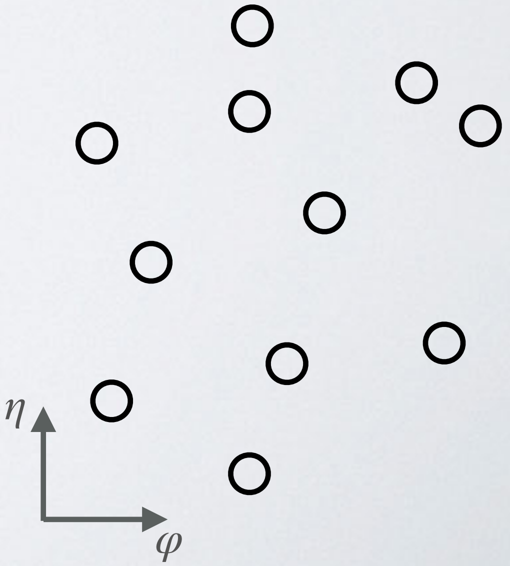
or p_x p_y p_z

Particle/Hit 1

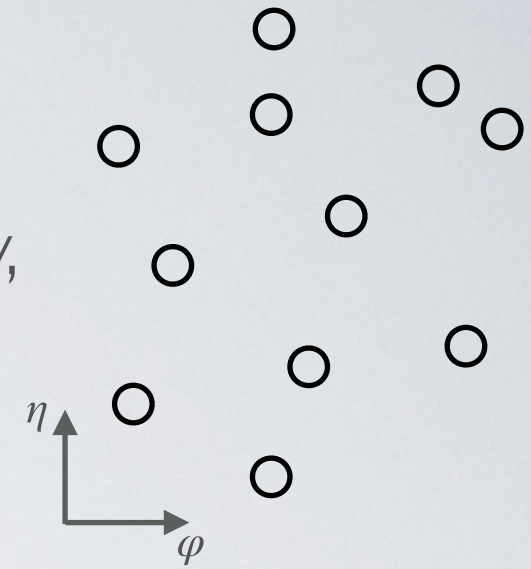
Particle/Hit 2

⋮

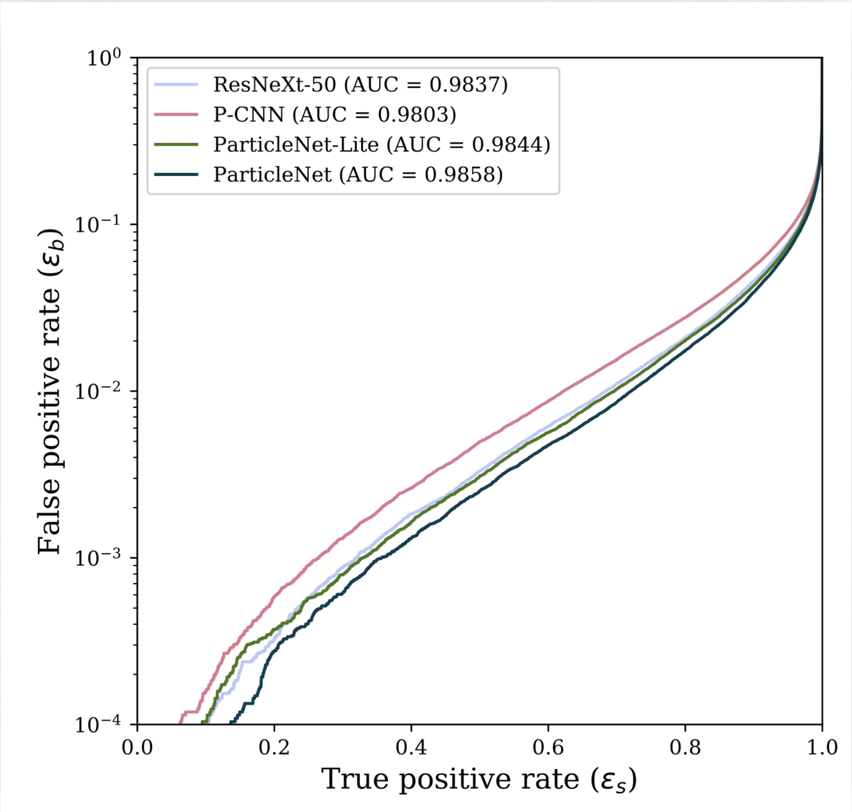




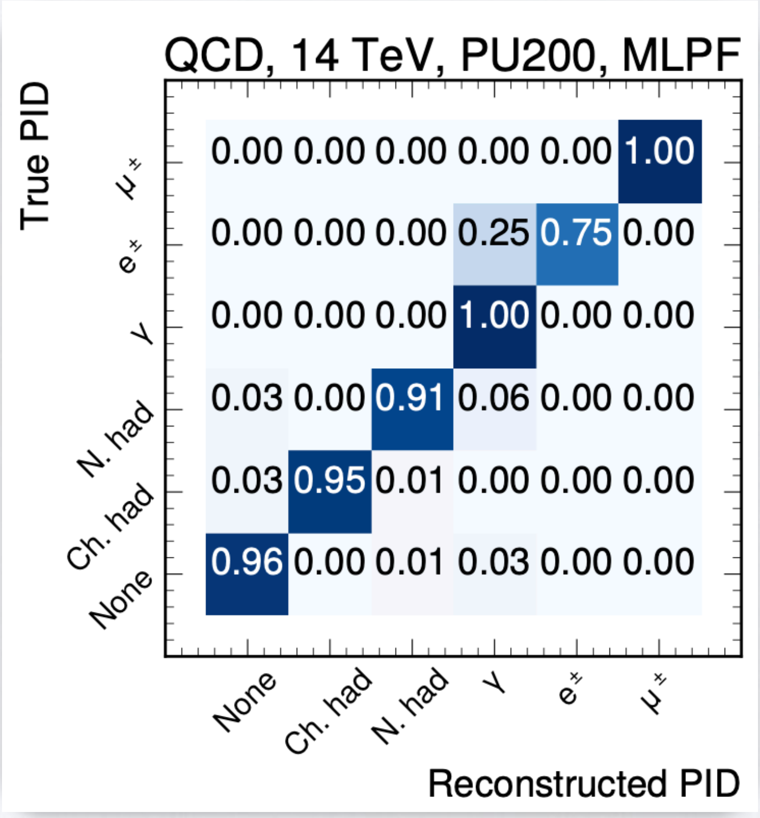
PARTICLE / HIT CLOUDS



- More physics-motivated representation, capturing geometry, respecting permutation symmetry
- This + **graph neural networks**, exploiting geometrical information, are SOTA in CMS



ParticleNet (Qu et al '20) for boosted jet tagging



MLPF (Pata et al '21) for Particle Flow reconstruction

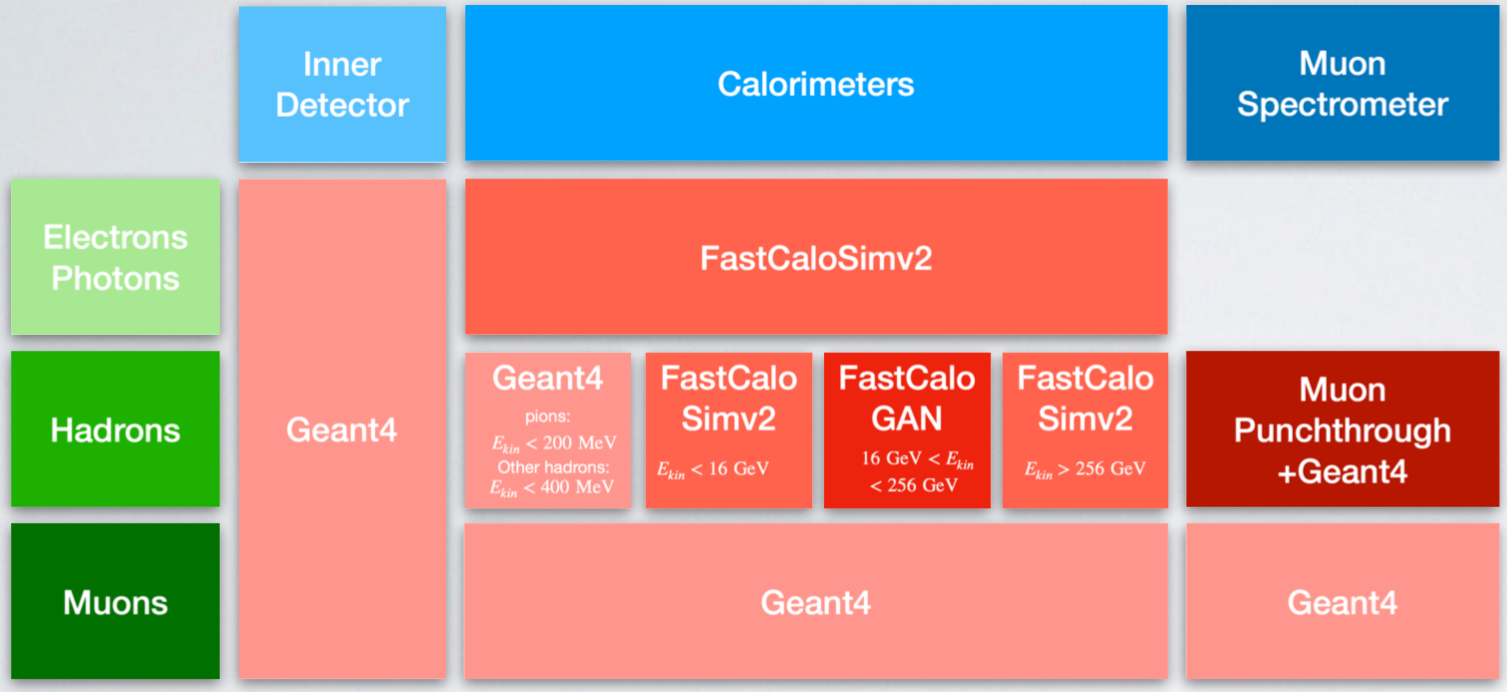
- **Can this be extended to generative models?**

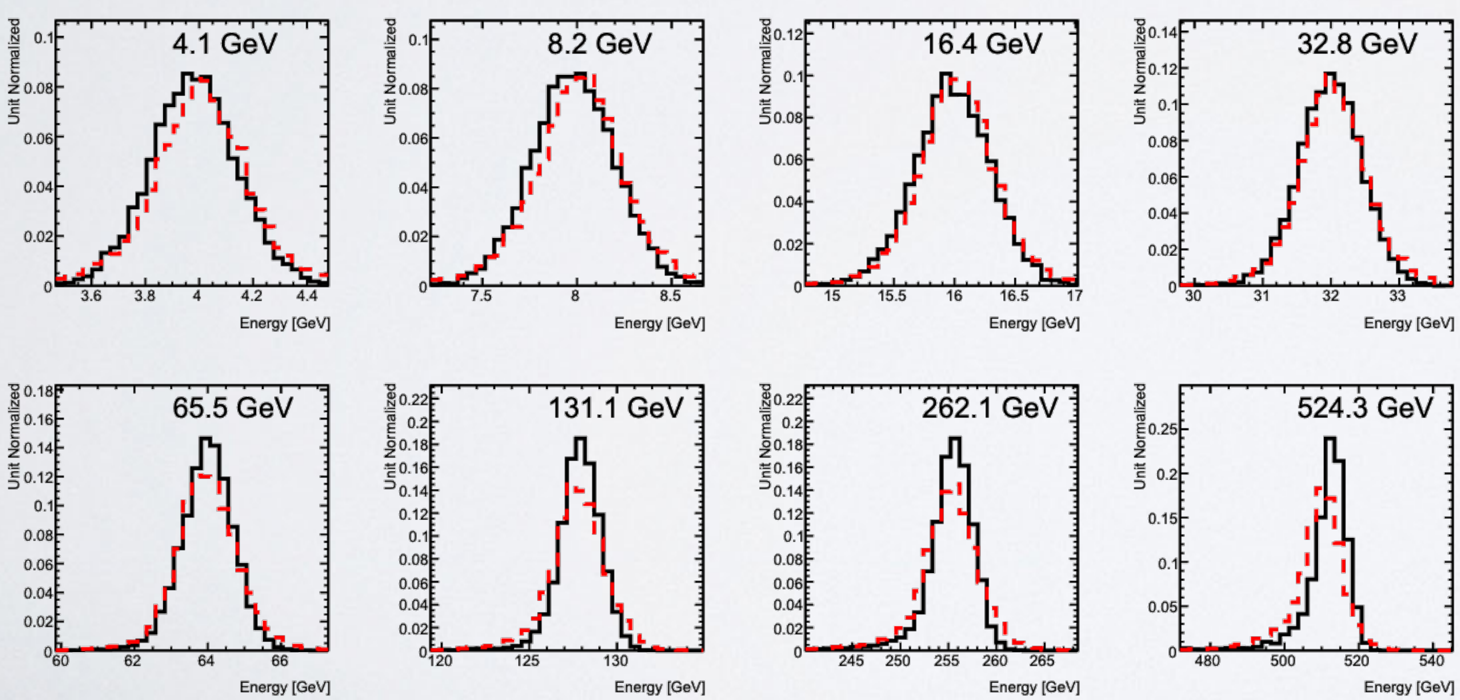
- LHC Simulations
- Deep generative models
  - Evaluation metrics
  - Data representations
- Current applications
  - MPGAN
- Discussion

- LHC Simulations
- Deep generative models
  - Evaluation metrics
  - Data representations
- **Some*** current applications
  - MPGAN
- Discussion

Apologies to those I didn't have time for!

ATLAS FASTCALOGAN



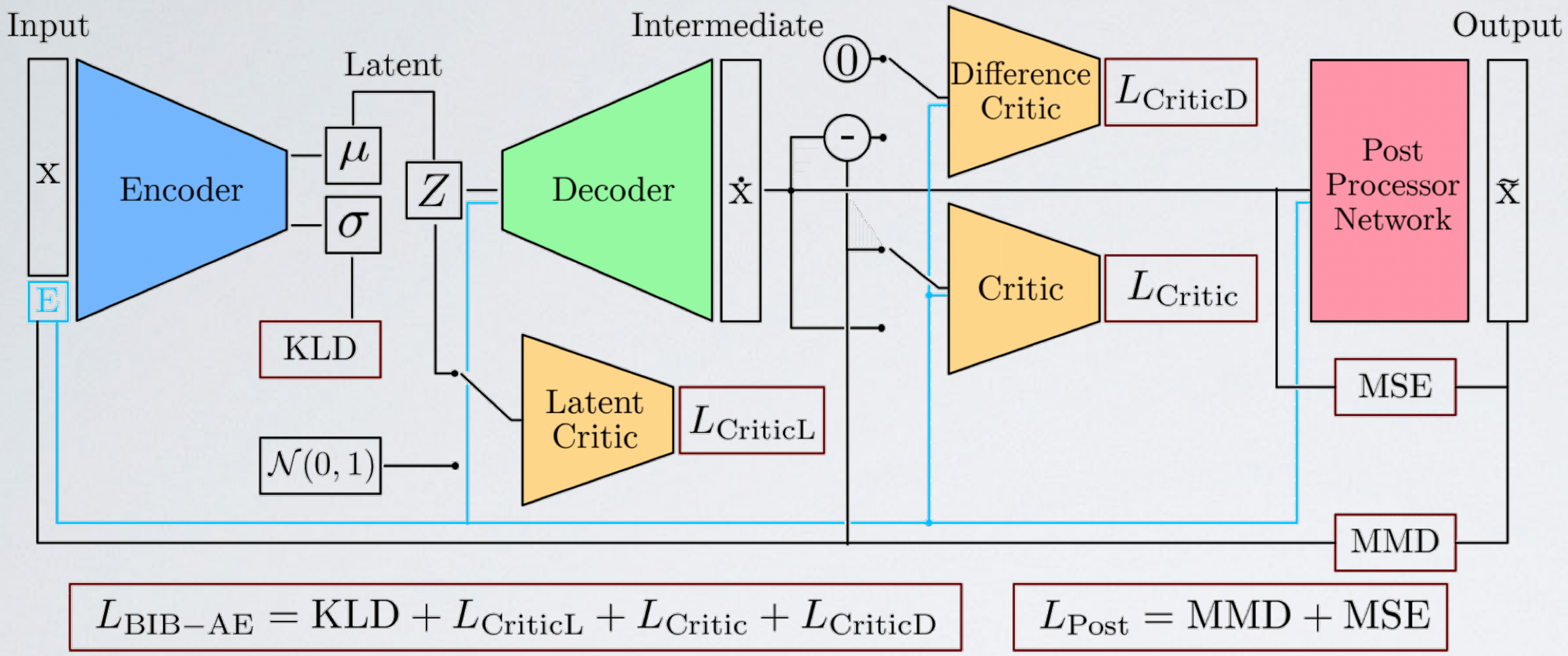


- Currently used for ATLAS fast simulations (AtlFast3) - 7B events for Run 2 analyses!
- Conditional 'Wasserstein GAN' using shower images
- Reasonable performance but:
  - Room for improvement
  - 'Voxelisation' to deal with sparsity and high granularity
  - 300 GANs trained for each η bin

AtlFast3, ATLAS Collaboration 2022

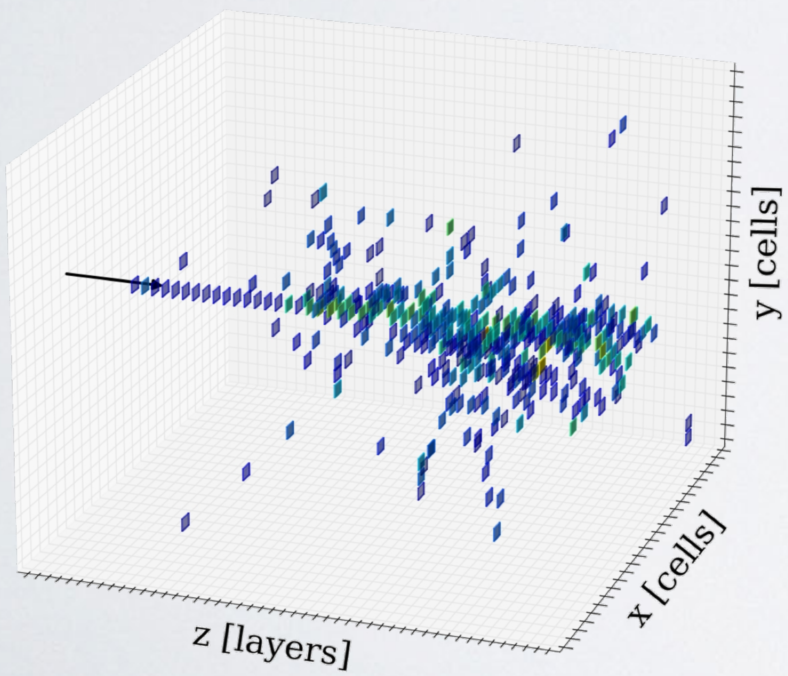
BIB-AE

Buhmann et. al., CSBS 2021
Buhmann et. al., MLST 2022

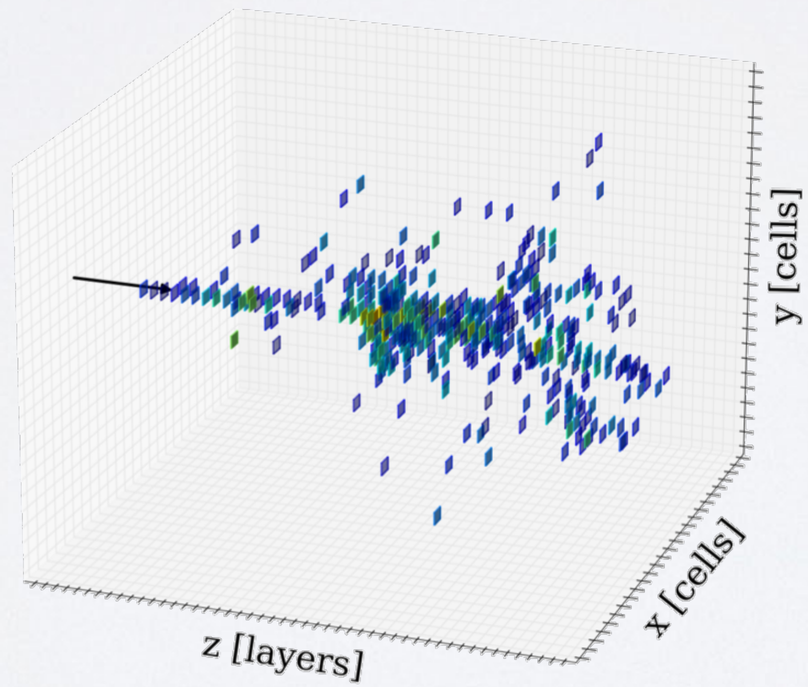


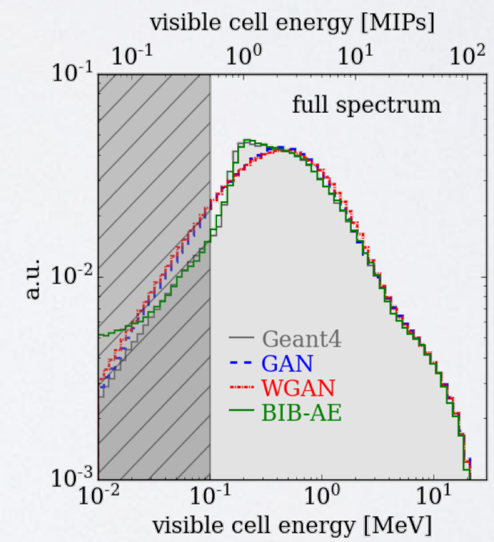
- Bounded-Information Bottleneck Autoencoder
- VAE + GAN + Post Processor
- Photon, pion shower images
- Good agreement with (simplified) CMS-like simulations

Sample Geant4 pion shower image



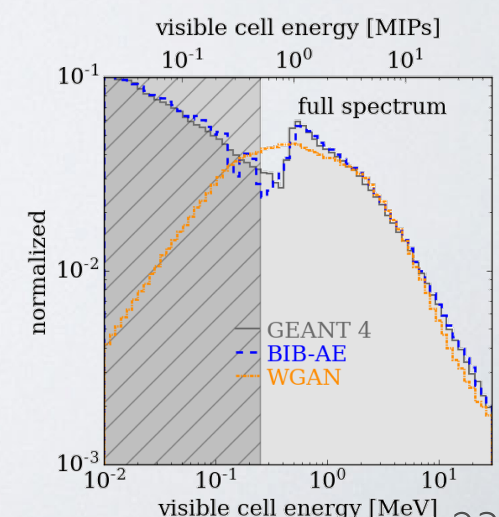
Sample BIB-AE image





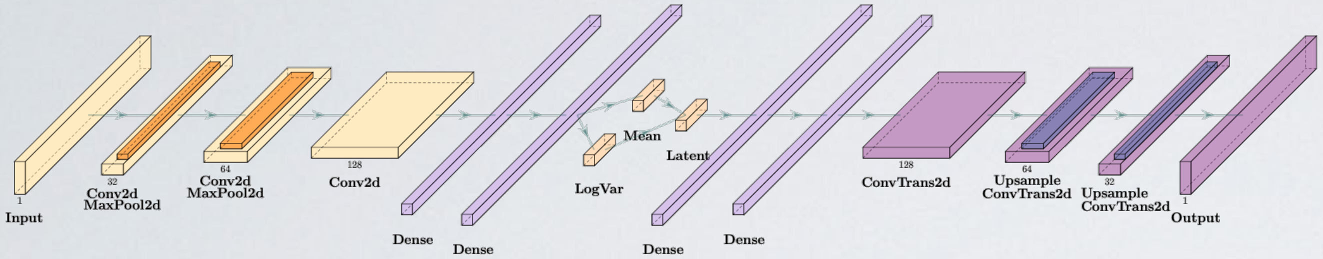
Photon showers

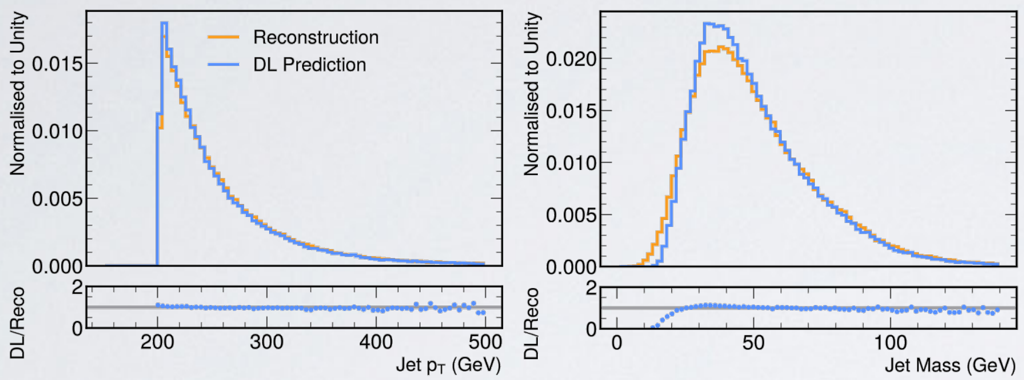
Pion showers



SPARSE APPROACHES

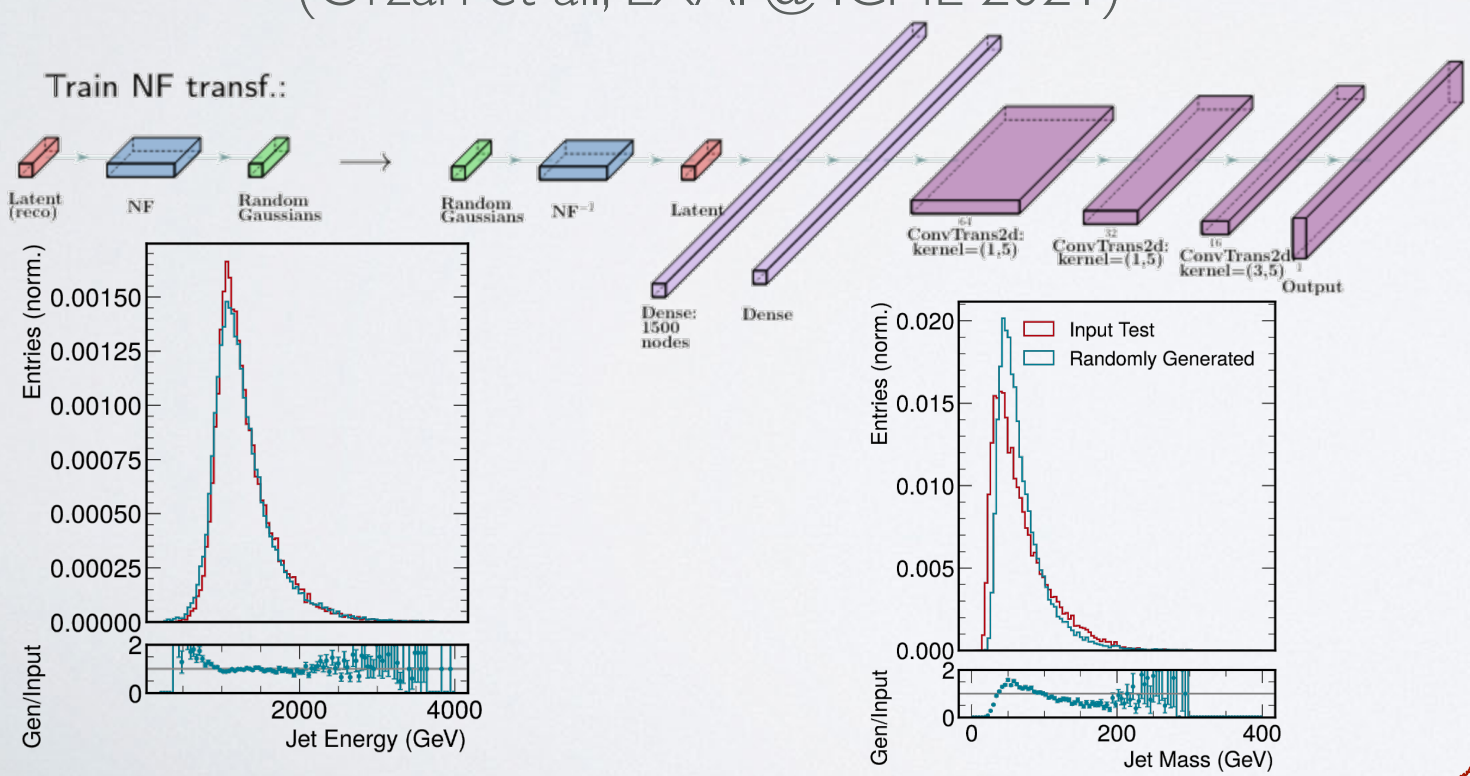
SparseVAE: Gen Jet → Reco Jet (Touranakou et al., MLST 2022)





VAE + Flow-based prior: Gen Particle → Reco Jet

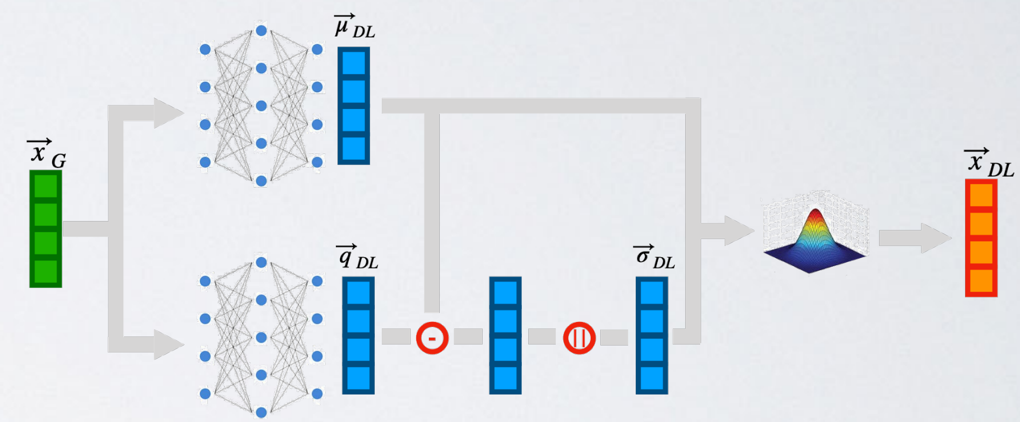
(Orzari et al., LXAI @ ICML 2021)

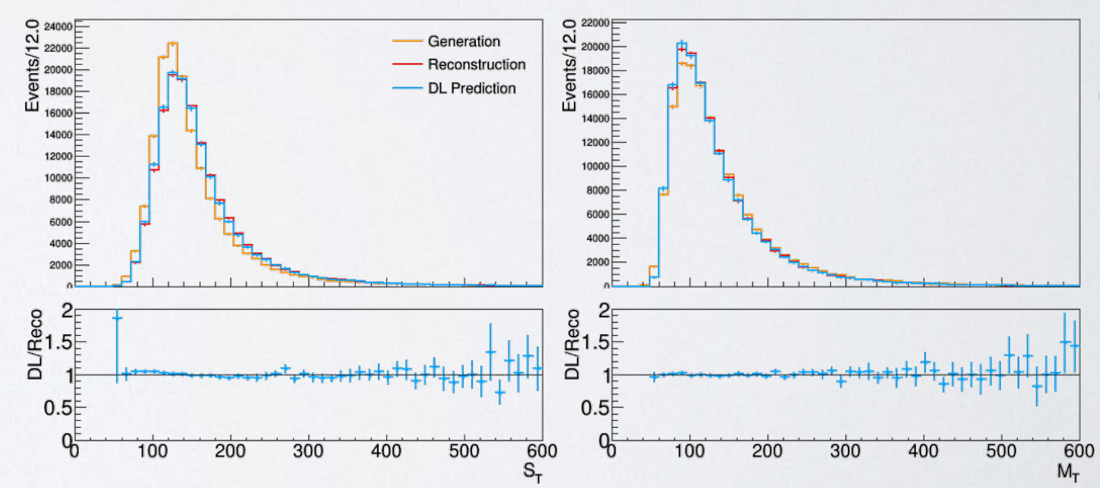


Analysis-Specific FastSim

(Chen et al., CSBS 2021)

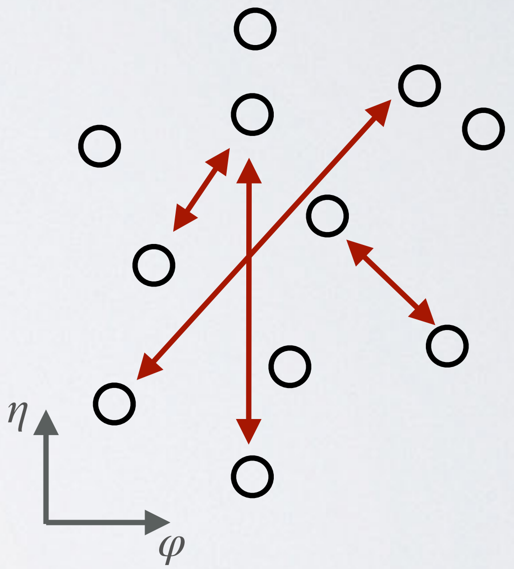
- Directly from gen-level high level features (jet $\overrightarrow{\mathbf{p}}$, MET etc.) to reco features





OUR APPROACH: MPGAN

- Majority of work, while successful, is image-based
- Difficult to scale to HL-LHC and apply to e.g CMS high-granularity calorimeter
- We develop a graph-based approach
- Key ideas:
  - Natural, sparse, and flexible representation for data
  - Learn global features *and* inter-particle correlations (i.e. jet, shower structure)



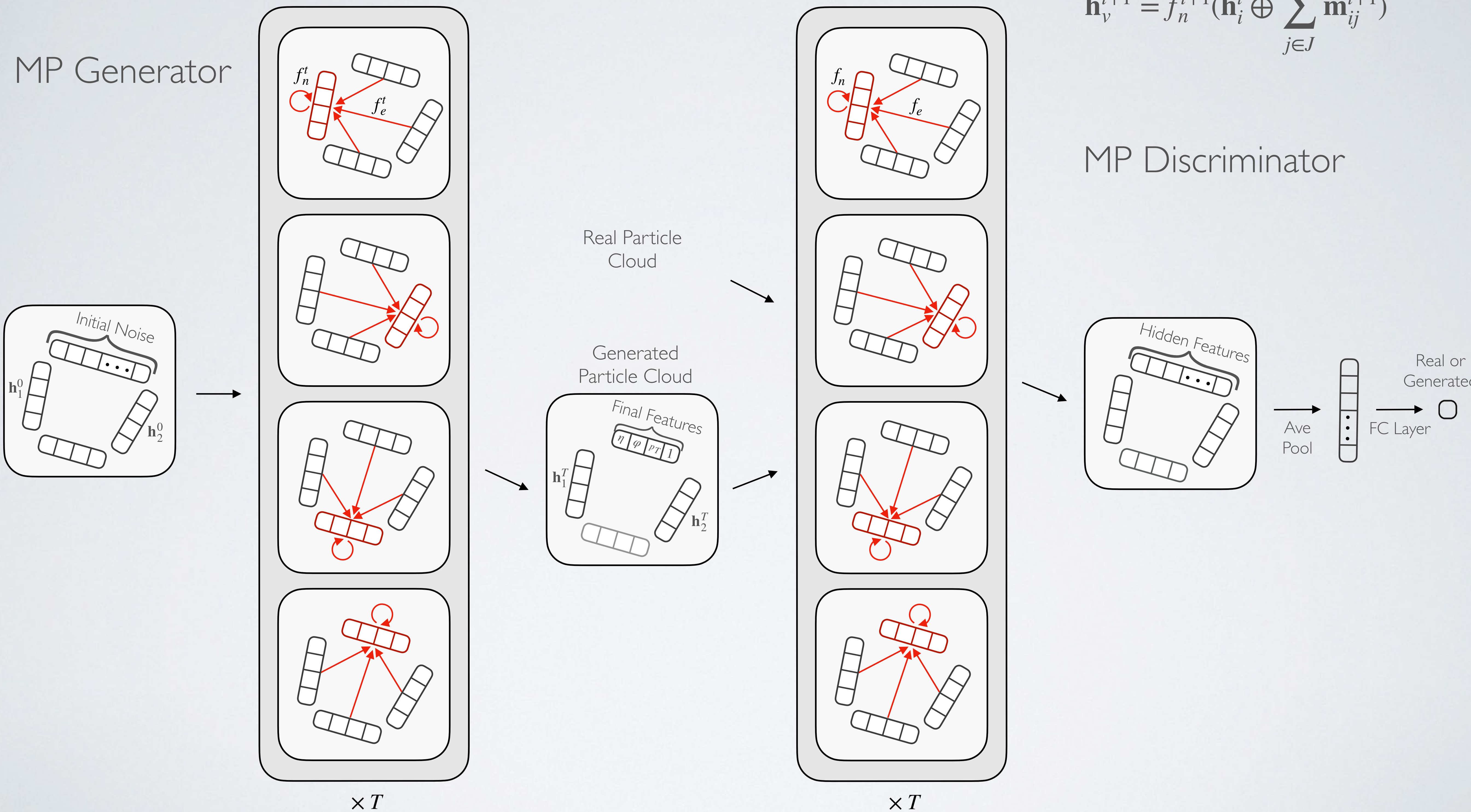
MPGAN

Kansal et al., ML4PS @ NeurIPS 2020
Kansal et al., NeurIPS 2021

- We develop a GAN with a fully-connected message-passing (MP) generator and discriminator

$$\mathbf{m}_{ij}^{t+1} = f_e^{t+1}(\mathbf{h}_i^t \oplus \mathbf{h}_j^t)$$

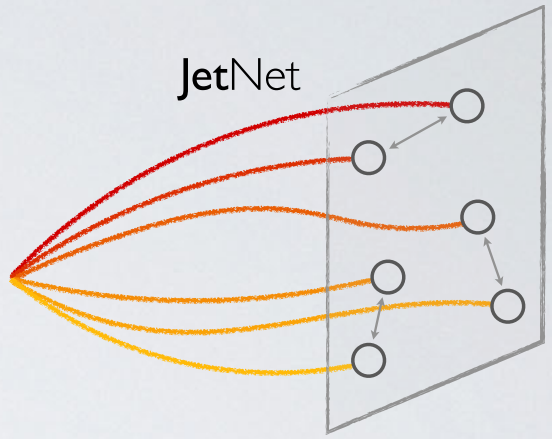
$$\mathbf{h}_v^{t+1} = f_n^{t+1}(\mathbf{h}_i^t \oplus \sum_{j \in J} \mathbf{m}_{ij}^{t+1})$$



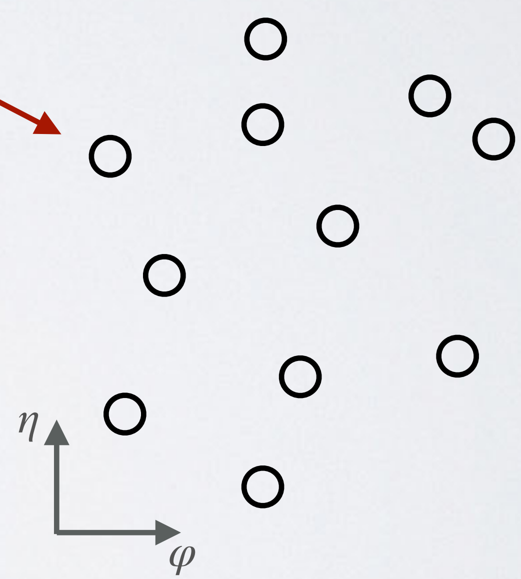
zenodo.org/record/5502543

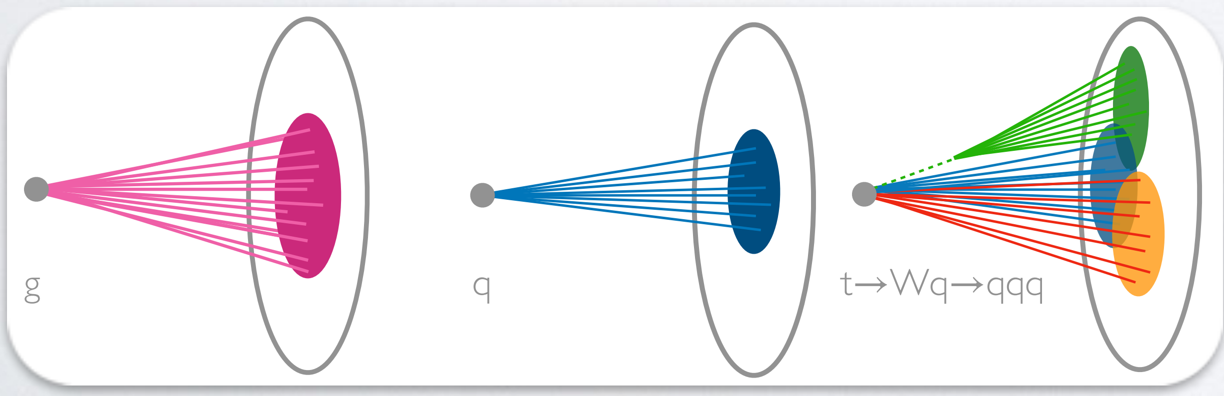
DATASET

Kansal et al., ML4PS @ NeurIPS 2020
Kansal et al., NeurIPS 2021
JetNet Python Package



- Test-bench: Pythia-simulated high p_T jets ("JetNet")
- 30 highest p_T particles, $(\eta^{rel}, \phi^{rel}, p_T^{rel})$ features
- **Gen particle → Reco jet**



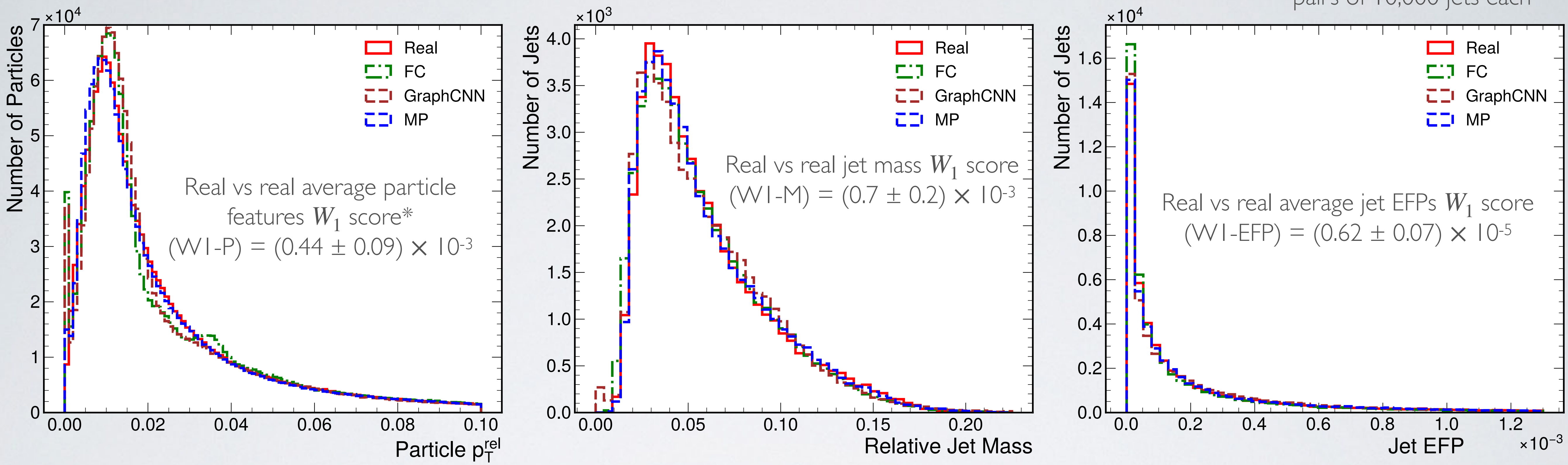


RESULTS: GLUON JETS

Kansal et al., ML4PS @ NeurIPS 2020
Kansal et al., NeurIPS 2021

Sample feature distributions, with MPGAN compared to baseline point cloud generators

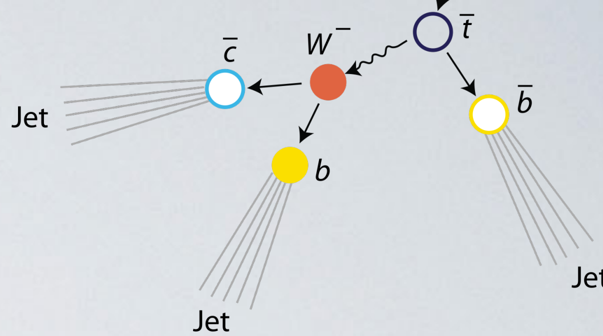
*Mean and error over 5 sets of pairs of 10,000 jets each

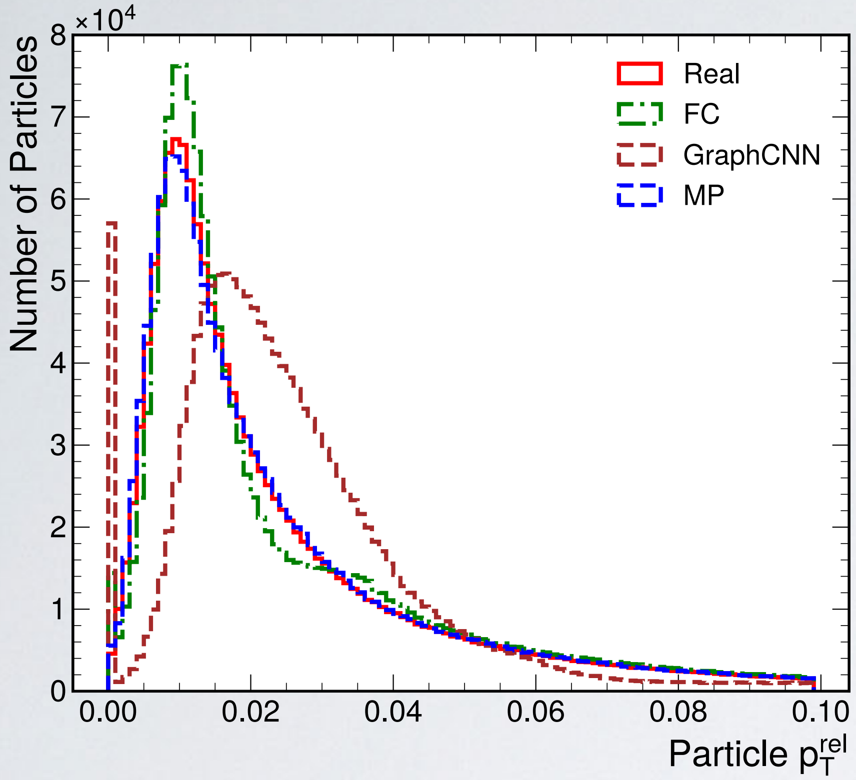


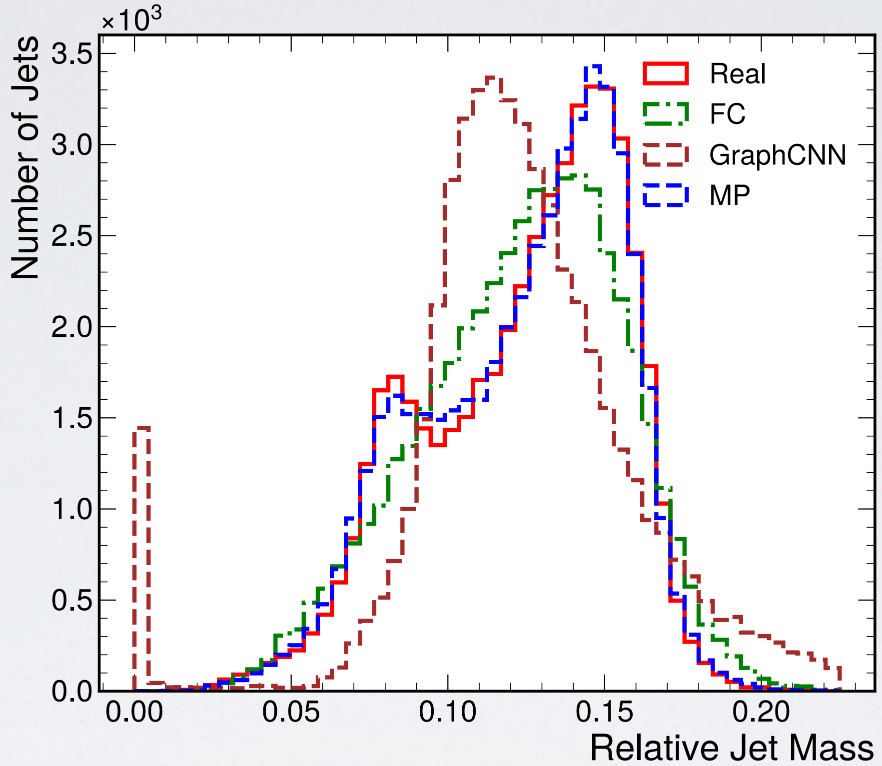
Generator	Discriminator	W1-P (10^{-3})	W1-M (10^{-3})	W1-EFP (10^{-5})	FPND
FC	PointNet	1.3 ± 0.2	1.3 ± 0.4	1.5 ± 0.9	5.0
GraphCNN	PointNet	16 ± 6	1.9 ± 0.2	200 ± 1000	7k
MP	MP	0.9 ± 0.3	0.7 ± 0.2	0.7 ± 0.2	0.12
MP	PointNet	1.2 ± 0.4	1.3 ± 0.4	4 ± 2	18

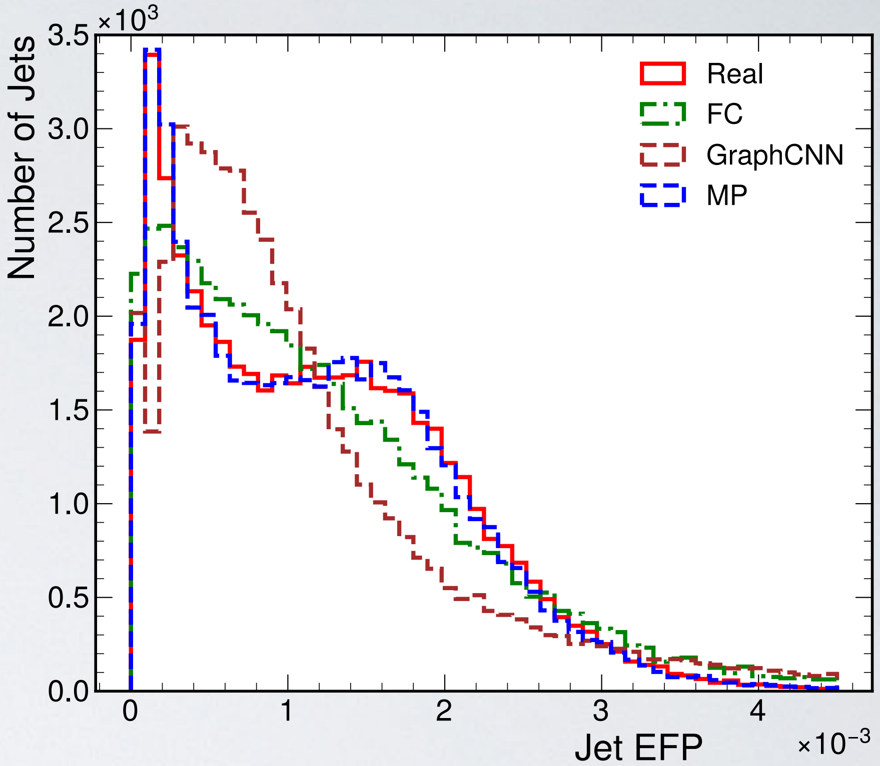
- MPGAN generator is the best performing on every metric
- Significantly outperforms alternatives on high level feature metrics (W1-M, W1-EFP, FPND)
- Mass and ave. EFP scores are within error of the real vs real baseline ⇒ learning jet substructure correctly

RESULTS: TOP QUARK JETS









Real vs real
W1-P = (0.55 ± 0.07) × 10^{-3}

Real vs real
W1-M = (0.51 ± 0.07) × 10^{-3}

Real vs real
W1-EFP = (1.1 ± 0.1) × 10^{-5}

Generator	Discriminator	W1-P (10^{-3})	W1-M (10^{-3})	W1-EFP (10^{-5})	FPND
FC	PointNet	1.6 ± 0.4	2.7 ± 0.1	7.7 ± 0.5	3.9
GraphCNN	PointNet	30 ± 20	11.3 ± 0.9	37 ± 2	30k
MP	MP	2.3 ± 0.3	0.6 ± 0.2	2 ± 1	0.37
MP	PointNet	1.6 ± 0.4	0.76 ± 0.08	4 ± 1	3.7

- MPGAN learns perfectly the complex bimodal jet feature distributions
- Mass and ave. EFP scores remain within error of real vs real baseline

MPGAN SUMMARY

- **Graph-based approach** is highly successful at learning complex physics substructure
- **Graph-based discriminator** loss is crucial: learns particle correlations ⇒ forces generator to learn as well
- Goal: extend this to CMS calorimeter showers for HL-LHC
    - **Gen particle → reconstructed hits**
- Next steps:
    - Conditional generation and metrics (learning and evaluating $p(\mathbf{x} \mid y)$)
    - Scaling to larger point clouds
    - Development / application to CMS datasets

- LHC Simulations
- Deep generative models
  - Evaluation metrics
  - Data representations
- Current applications
  - MPGAN
- Discussion

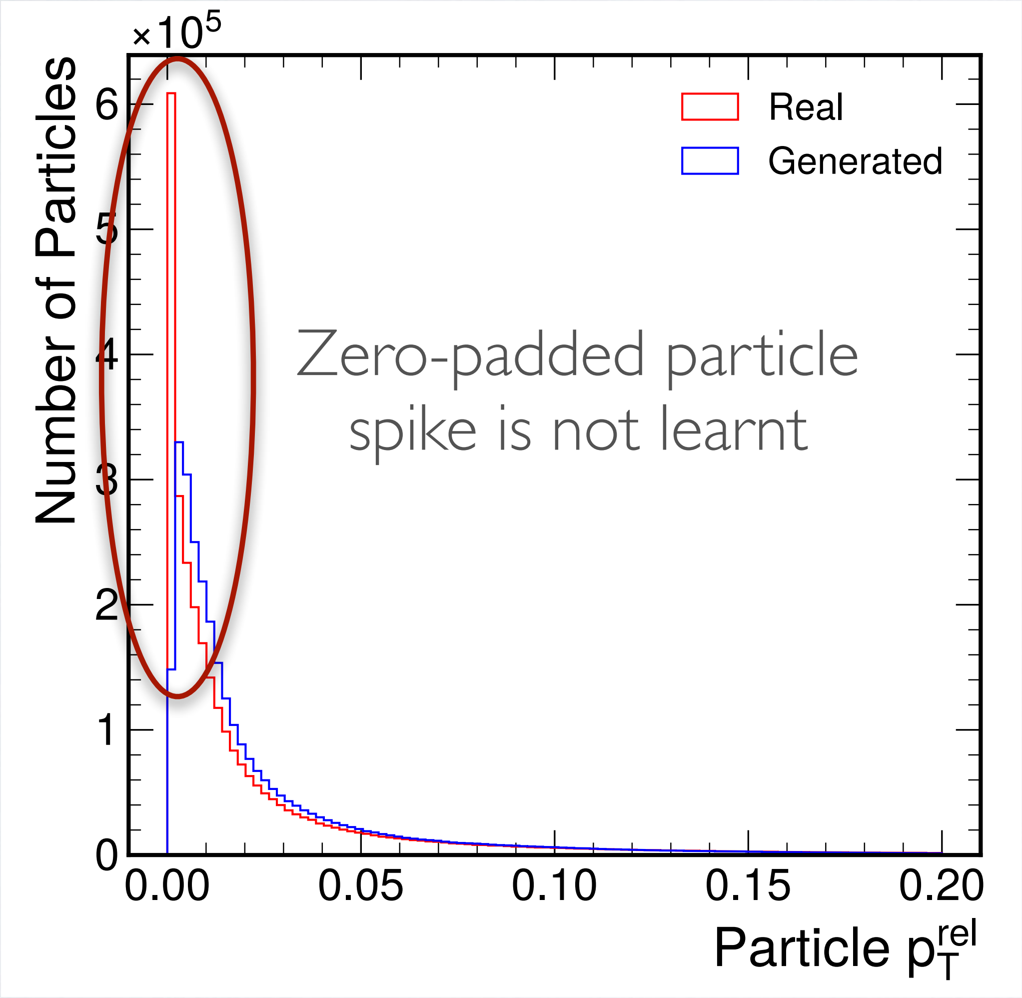
DISCUSSION

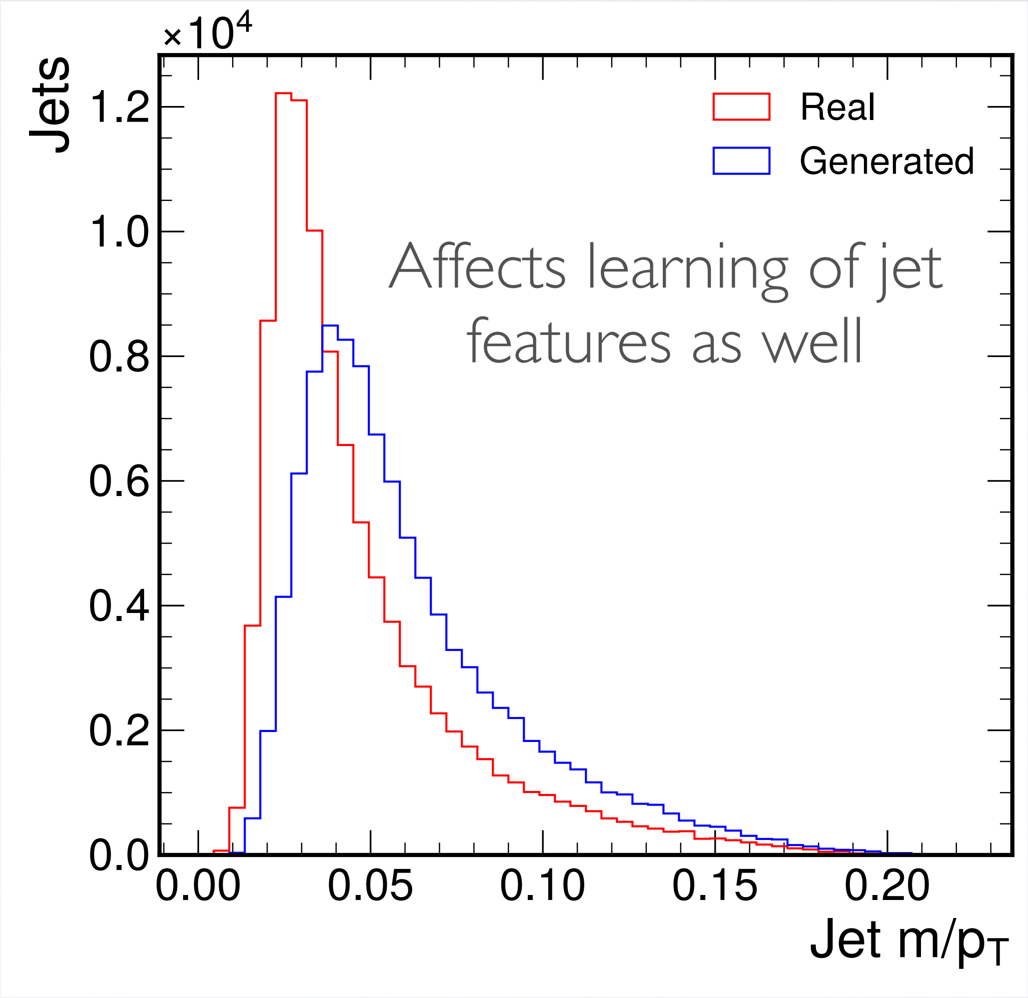
- Significant opportunity to accelerate simulations for HL-LHC using machine learning
- Rich active area of research in ML and HEP, with ATLAS already using GANs for fast simulation
- Lots of open questions:
  - Where in the simulation pipeline would be most effective?
    - **Gen particles → reco hits** seems to be a reasonable trade-off in speed vs. accuracy/trust
    - Plenty of phase space left to explore and test
  - Which model?
    - GANs (or variations thereof) and GNNs promising
    - Here also phase space left to explore
  - How to evaluate?
    - Fréchet distance and W_1 scores have been very effective
    - Community needs to converge on metrics (and datasets)

BACKUP

VARIABLE-SIZED CLOUDS

- Very few gluon jets have fewer than 30 particles, can get away with zero-padding
- More difficult with light quark jets:





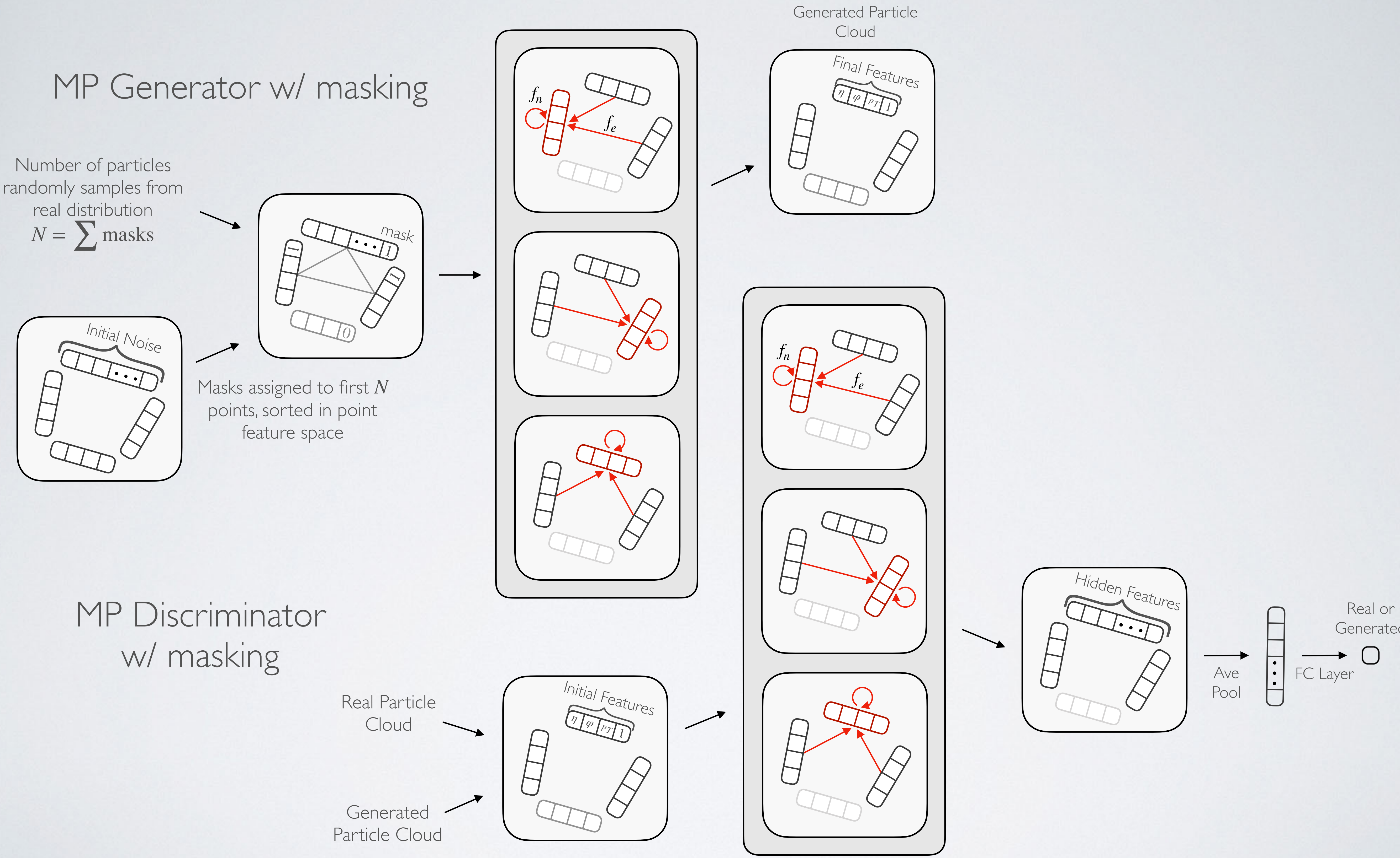
- We experiment with various masking architectures to handle this (adding a binary particle feature indicating if it's real or zero-padded)

Most successful masking architecture:

ARCHITECTURE

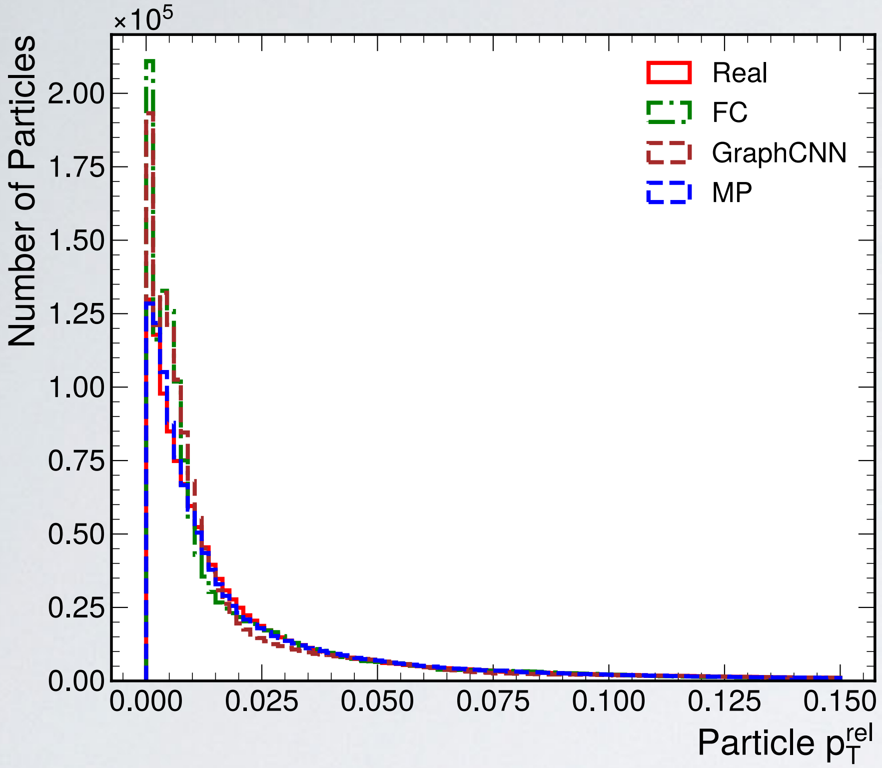
MP Generator w/ masking

MP Discriminator w/ masking

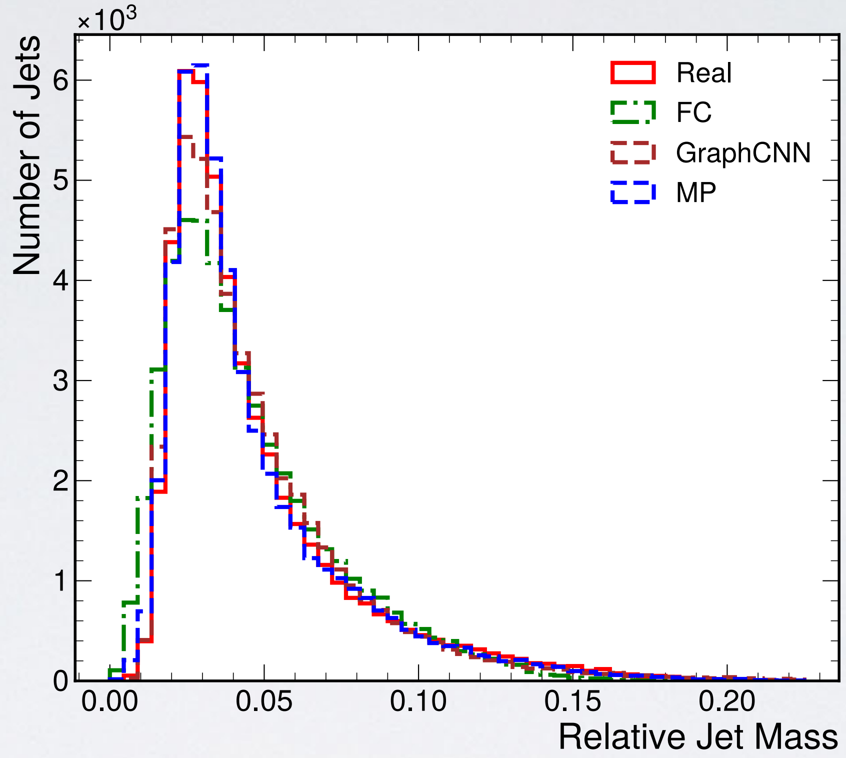


RESULTS: LIGHT QUARK JETS

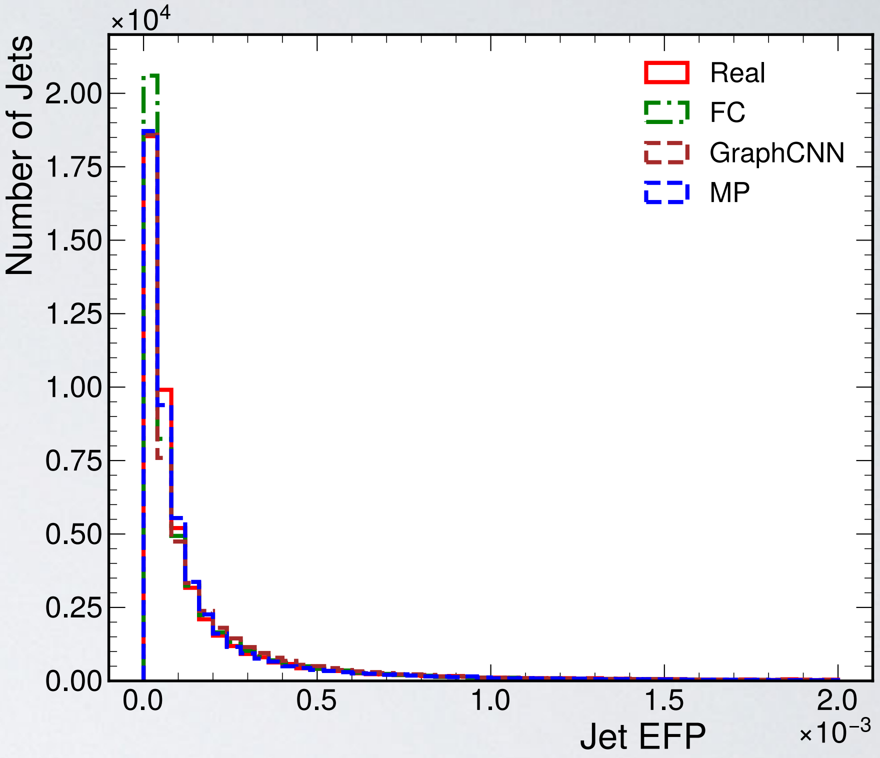
Sample feature distributions, with our MPGAN compared to FC and GraphCNN generators + PointNet discriminators



Real vs real
W1-P = (0.5 ± 0.1) × 10^{-3}



Real vs real
W1-M = (0.5 ± 0.1) × 10^{-3}



Real vs real
W1-EFP = (0.46 ± 0.04) × 10^{-5}

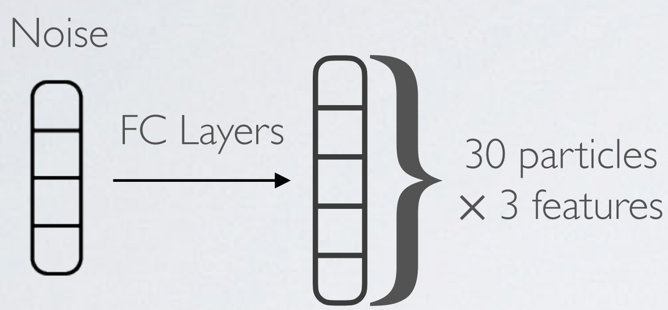
Generator	Discriminator	W1-P (10^{-3})	W1-M (10^{-3})	W1-EFP (10^{-5})	FPND	Coverage	MMD
FC	PointNet	1.5 ± 0.1	2.9 ± 0.2	2 ± 1	22k	0.36	0.026
GraphCNN	PointNet	3.9 ± 0.2	4.2 ± 1.6	20M ± 10M	19k	0.37	0.031
MP	MP	2.1 ± 0.1	0.6 ± 0.1	0.9 ± 0.4	2.4	0.54	0.026
MP	PointNet	22.0 ± 0.1	3.2 ± 0.2	5 ± 2	3.6k	0.22	0.035

- Masking strategy is successful
- MPGAN again best performing on every metric, apart from W1-P, significantly so on W1-M, W1-EFP, FPND
- Mass and ave. EFP scores all within error of the real vs real baseline

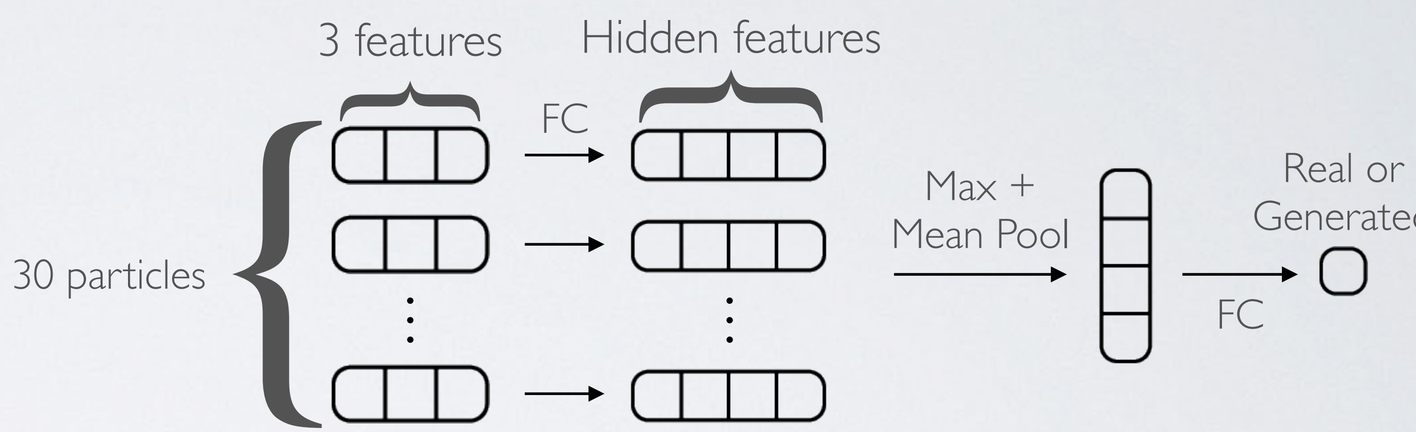
BASELINE POINT CLOUD GANS

We compare with existing point cloud GANs as baselines, two relevant architectures are:

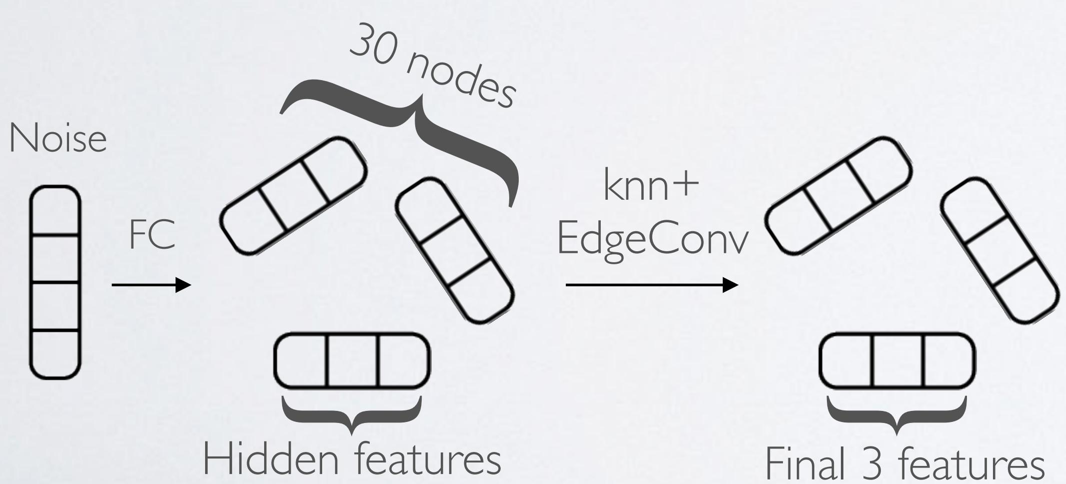
Fully Connected Generator (Achlioptas et al '18)



PointNet-Mix Discriminator is the most performant on ShapeNet, compared to FC and GraphCNN (Wang et al '21)



Graph Convolutional Generator (Valsesia et al '19)



These GANs work somewhat, but not well enough for our purpose (next slides)

Interestingly, Wang et al find the FC generator works better than the GraphCNN (with a PointNet-Mix discriminator)