DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC

Riccardo Di Sipio, Michele Faucci Giannelli, Sana Ketabchi Haghighat, Serena Palazzo

3rd Machine Learning Workshop

April 15-18, 2019

THE UNIVERSITY *of* EDINBURGH

Introduction

Motivations:

- In the next years experiments at the LHC need to cope to a substantial increase of data.
- At the same time, the strategy to produce accurate and statistically large samples of simulated *pp* collisions will be facing both technological limitations and new opportunities.

New approach:

- In this context **Machine Learning** techniques are considered.
- A Generative Adversial Network (GAN) is used to simulate the production of particle pairs in *pp* collisions at LHC.
- This approach is applied to dijet events production:
  - In the SM precision measurements and searches for physics beyond the SM is one of most common background source.

Paper draft on arXiv and submitted to JHEP: ▸ arXiv 1903.02433

Monte Carlo Simulation @ the LHC

- Both the ATLAS and CMS experiments at the LHC rely on:
  - Event generators such as MadGraph5, Powheg-Box and aMC@NLO to simulate the hard scattering process.
  - Pythia8 and Herwig7 to simulate the parton shower process.
  - Geant4 to simulate the detector response.

Limitations:

- Several minutes to simulate a single event:
  - Huge computational footprint with O(10^9) events, both in terms of CPU usage and disk space.

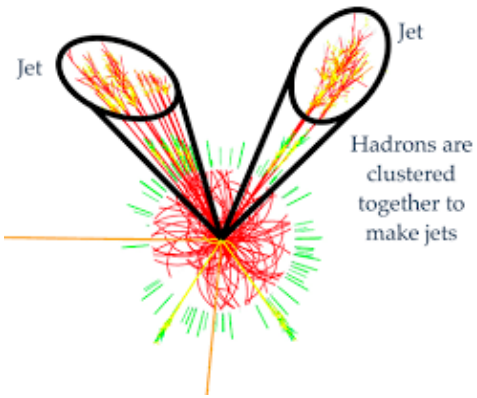
Current solution:

- Experiments rely on simpler but less accurate generators.

Providing a solution to extend the simulated events to the requirements of the LHC experiments will significantly enhance a wide range of measurements.

Physics of QCD dijet events

- At the LHC, *pp* collisions result in interaction of quarks and gluons referred to as *partons*.
- Parton scattering processes via strong interaction result in most of the cases in two outgoing partons producing parton showers that hadronize into cluster called **jets**.
- Parton scattering processes with a pair of jets in the final state are called **dijet** events.



Monte Carlo training sample

Events generation:

- A sample of 2 milion dijet events has been used.
  - Generated using MadGraph5 and Pythia8 samples.

Detector response:

- Delphes3 Fast Simulation with setting that reproduce the ATLAS detector geometry.
- An average of 25 additional soft QCD *pp* collisions were overlaid to reproduce more realistic data-taking conditions.

Object definition and selection

Object Reconstruction:

- Objects (e^-, μ, E_T^{miss}, jets) are reconstructed using DELPHES3 algorithm before and after the detector simulations:
  - Particle and reconstruction levels, respectively.
- Jets were reconstructed using the anti-k_T algorithm as implemented n FastJet with a distance parameter R=1.0.

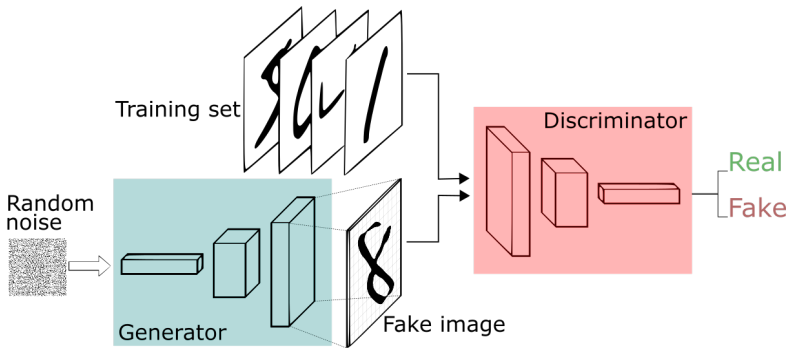
Selection:

- Applied a cut on the scalar sum of p_T of the outgoing partons ($H_T >$ 500 GeV) to increase the number of events with $p_T >$ 250 GeV.
- $\sim$ 1.5 million of events passed this selection at particle level and about 800,000 at reco level.
- These events were used to train the network.

Generative Adversial Networks (GANs)

GANs are generative models that try to learn the model to generate the input distribution as realistically as possible.

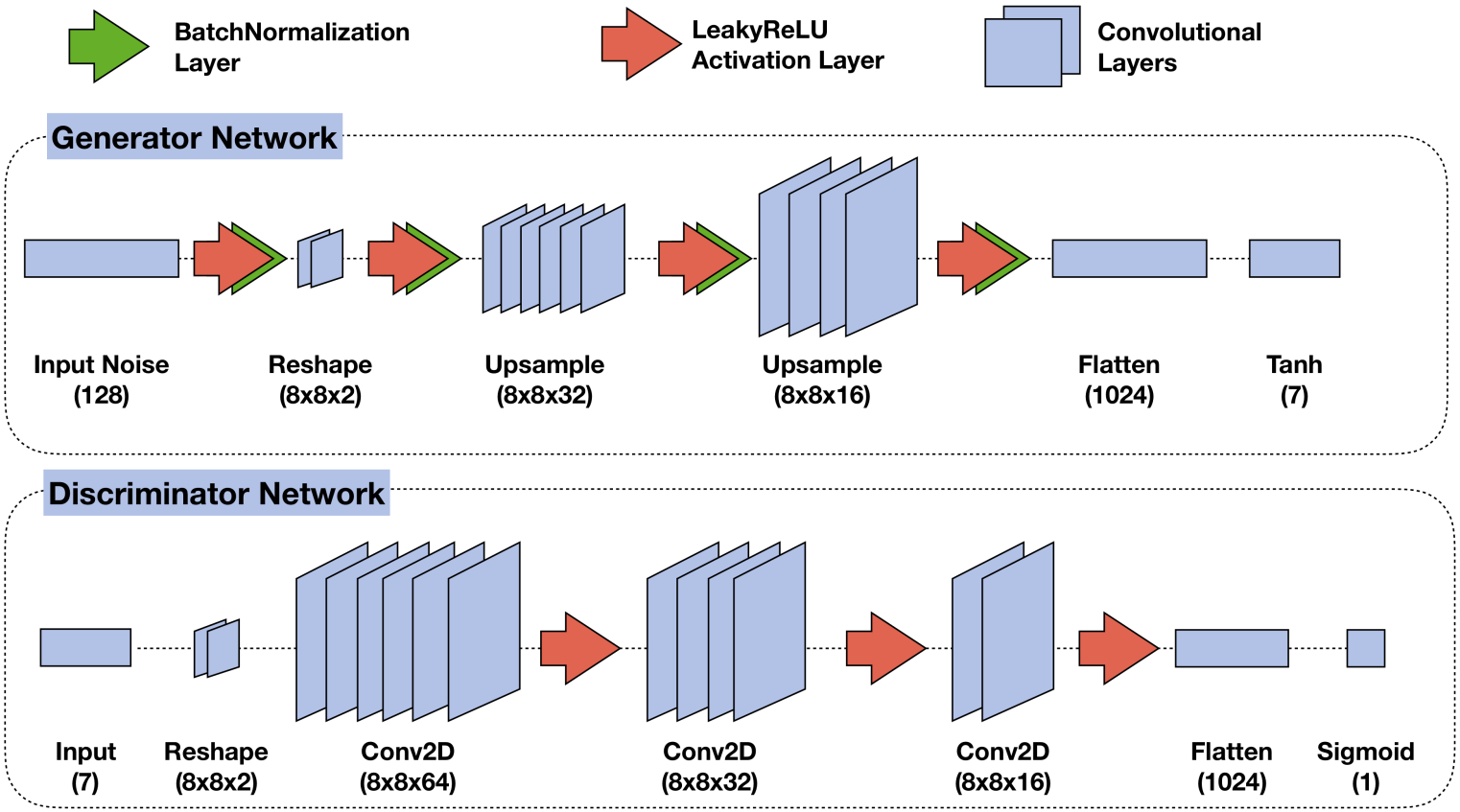
- A GAN is an unsupervised learning technique and consists of two neural networks:
  - A generative model **Generator** (G) generates new data points from some random uniform distribution.
  - A discriminative model **Discriminator** (D) identifies fake data produced by Generator from the real data.



Network architecture (1/2)

- The overall architecture of the network is composed of two main blocks: a Generator (G) and a Discriminator (D), based on convolutional layers.
- All layers have LeakyReLU activation functions except the last layer of both G and D.
  - The last layer of G has a *tanh* activation function.
  - The last layer of D has a *sigmoid* activation function.
- The network is implemented and trained using KERAS v2.2.4 with TENSORFLOW v1.12 back-end.
  - Input features are scaled in the range [-1,1] and pre-processed using SCIKIT-LEARN and PANDAS libraries.

Network architecture (2/2)



Training process

- The **Generator** takes as input random noise:
  - It transforms a vector of 128 random numbers from a uniform distribution in the [0,1] range, into a vector of 7 elements that represent:
    - p_T, η and m of the leading jet
    - p_T, η ϕ and m of the 2nd-leading jet.
- The **Discriminator** is trained with MC events.
- Achieved significant improvement by using the intrinsic ϕ symmetry of dijet events $\rightarrow$ ϕ of the leading jet set to 0.
- All events were used twice by switching η sign (η-flip).
  - For the event generation η randomly flipped to remove any not physical effects.
- 100K iterations with mini-batches of 32 events each.
- 1 hour to complete the training on a GPU NVIDIA Quadro P6000.

Network parameters optimization

- Several parameters of the model have been optimised.
- Once satisfactory performances have been reached, no further parameter optimisation was carried on.

Loss Function settings:

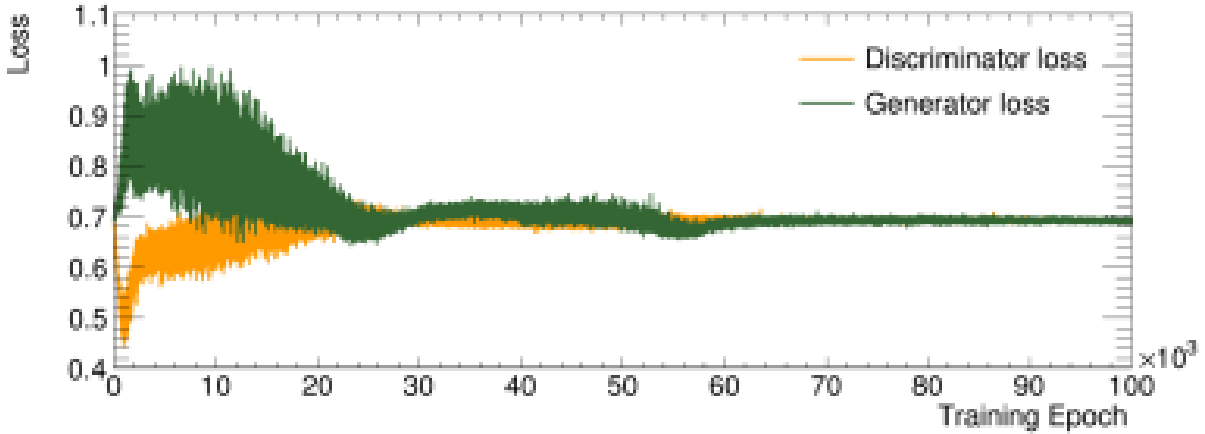
- Generator $\rightarrow$ Mean squared error.
- Discriminator $\rightarrow$ Binary cross-entropy.

Optimizer setting:

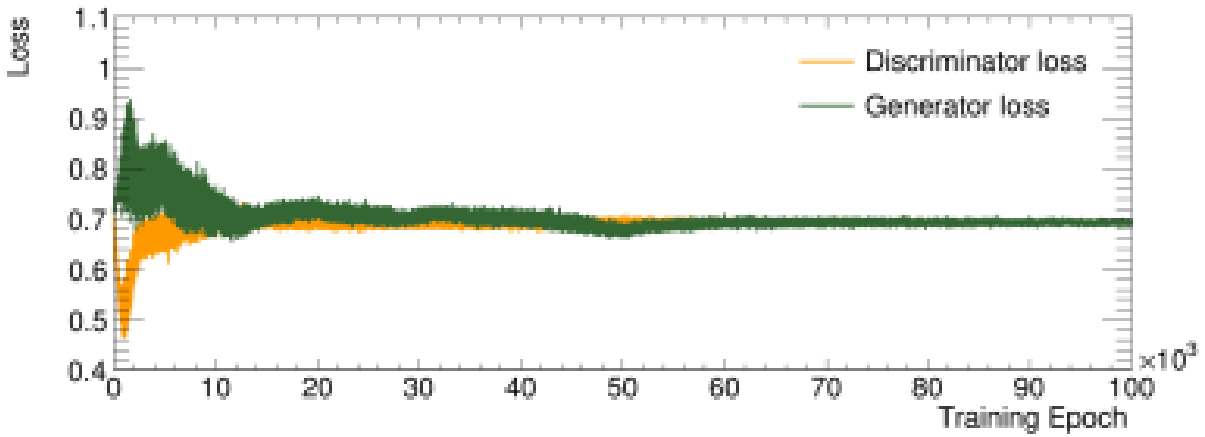
- for both Discriminator and Generator the optimizer is *Adam* with learning rate lr $= 10^{-5}$, $\beta_1 = 0.5$ and $\beta_2 = 0.9$

These parameters gave the best results among many values and configuration tested.

Loss functions



↬ Reco Level

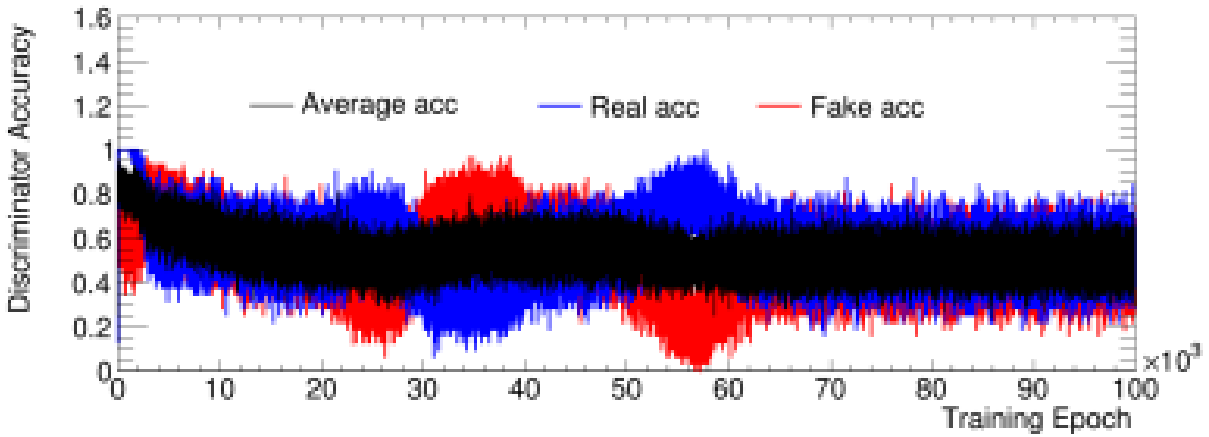


↬ Particle Level

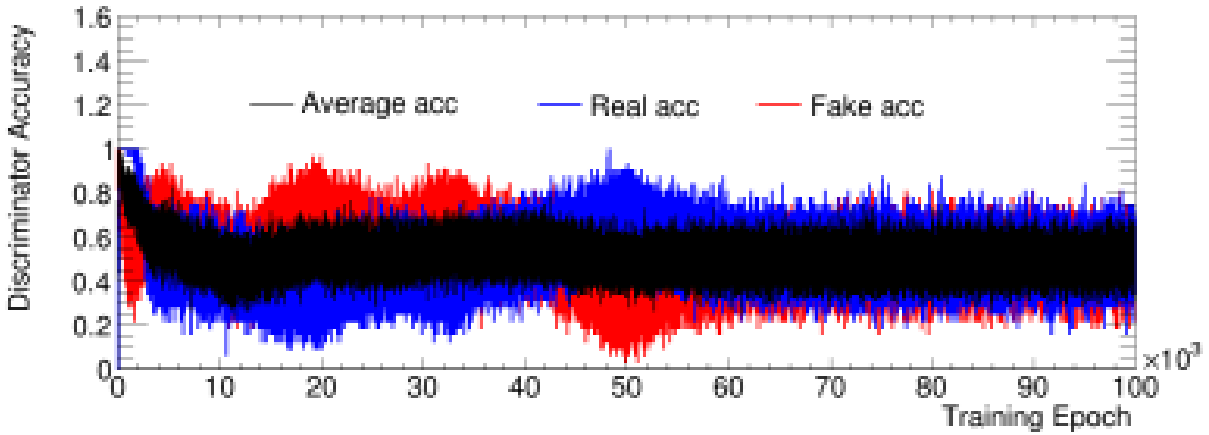
↬ The stationary state between generator and discriminator (Nash equilibrium) is reached after few thousands epochs.

↬ This is true for both particle and reco levels.

Discriminator accuracy



↬ Reco Level



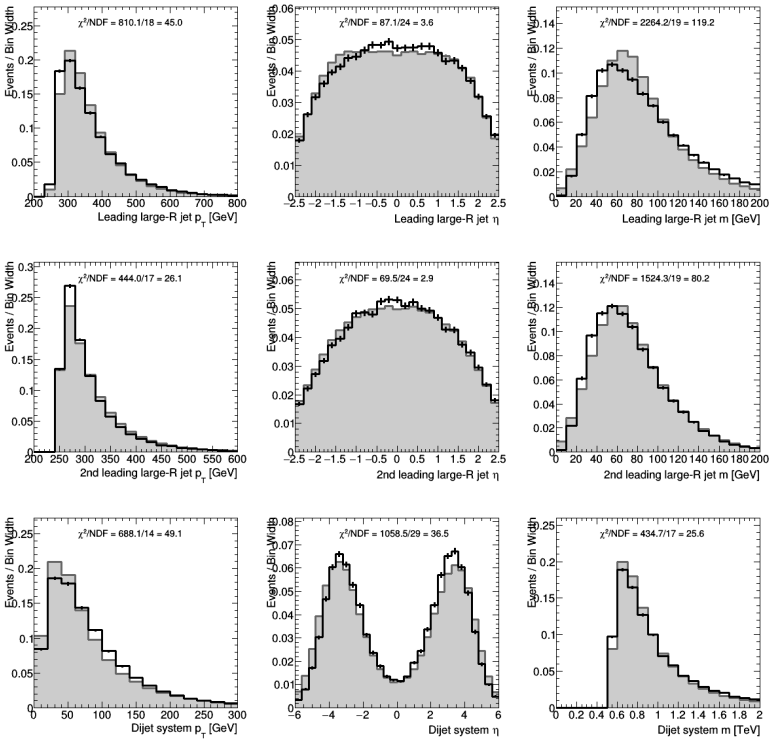
↬ Particle Level

Event Generation

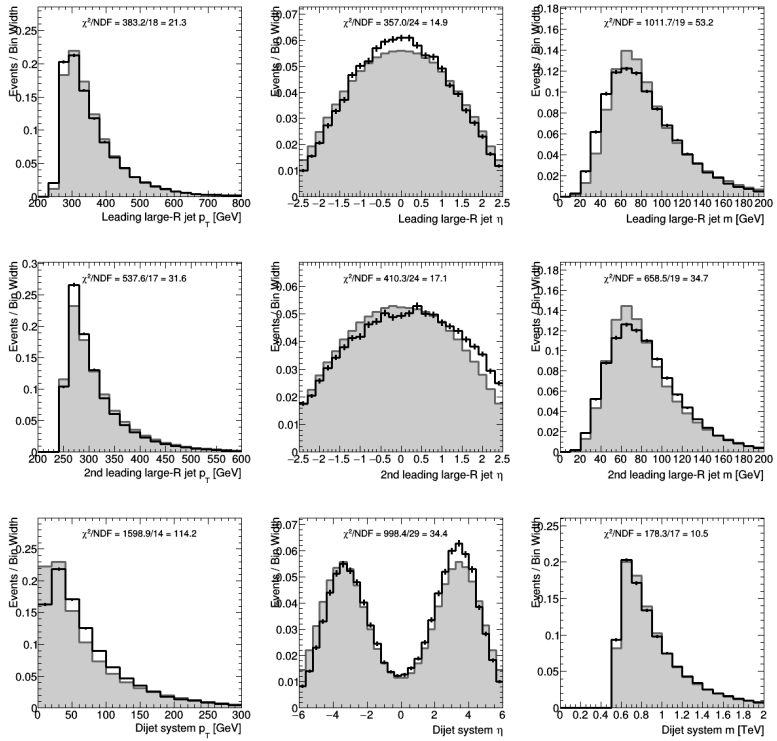
- Weights of generator model are saved during the training every 5000 epochs and used to generate events.
  - $\sim$ 80 s to generate 1 million events.
  - Events are filtered by applying the same cut that are applied to the real dataset:
    - both jets with $p_T > 250$ GeV.
- These events are used to fill the histograms.
- The comparison between the GAN output and the MC production is done for the following kinematic variables:
  - Leading large R-jet p_T, η, m
  - Sub Leading large R-jet p_T, η, m
  - Dijet system p_T, η, m

Results

Reco Level



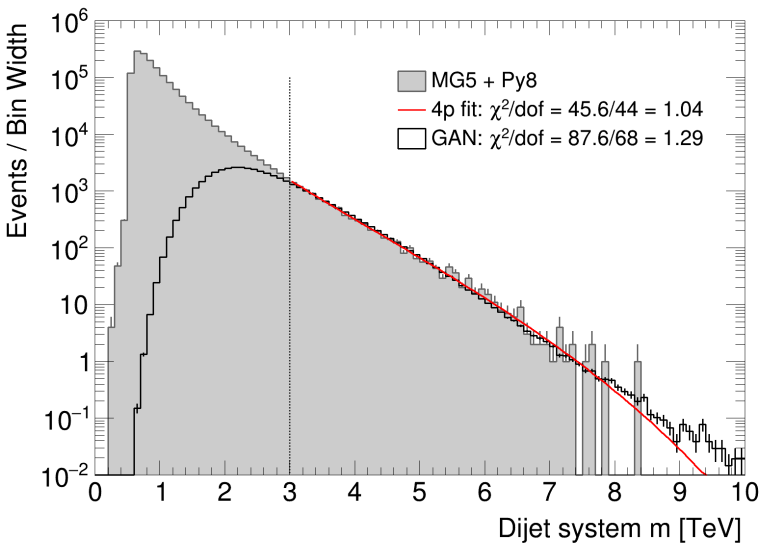
Particle Level



- Comparison of leading jets and dijet system kinematis as they appear at the iteration that yields the best agreement in terms of overall χ^2.
- Satisfactory level of agreement over a large range of kinematic regime.

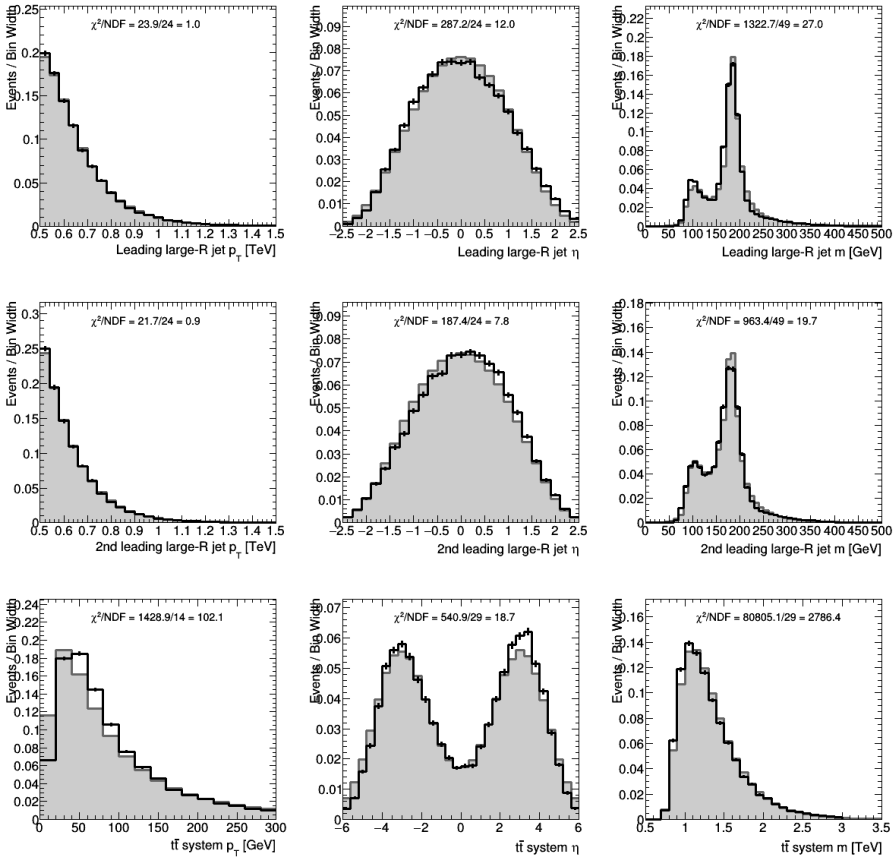
Further investigations (1/2)

- Performed further investigations in regions of the phase spaces with low cross-section $\rightarrow$ kinematic region of particularly interest for searches of physics BSM.
- Fit the MC with this 4p analytic function: $f(x) = \frac{p_0(1-x)^{p_1}}{x^{(p2+p3 log x)}}$, $x = m_{jj}/\sqrt{s}$
- Trained GAN with $\sim$ 40K events with $m_{jj} > 1.5$ TeV. Then used the trained model to generate 10 million events (sample » of MC).



- The fit in the region between 3 and 10 TeV can predict the shape of the MC distribution with a $\chi^2/\text{NDF} = 1.04$.
- The sample generated with the GAN shows an agreement of $\chi^2/\text{NDF} = 1.29$.

Further investigations (2/2)



- Trained GAN on a sample of top quark pairs decaying in the all hadronic channel ($t\bar{t} \to WbW\bar{b} \to bq\bar{q}'\bar{b}q\bar{q}'$)
- Both jets at particle level are required to have $p_T > 350$ GeV and $m < 500$ GeV.
- In this phase space region the jet mass is expected to have a peak around the top mass (172.5 GeV) in the MC simulation.
- When b-quarks are produced at an angle where only W bosons are found, a second peak is expected around W boson mass (~ 80 GeV).

- The figure shows that the GAN can learn these physics processes.

Conclusions

- A Generative Adversial Network approach for the simulation of QCD dijet events at the LHC has been presented.
- This novel approach is an attractive solution to reduce CPU usage and disk space to generate and simulate events.
- Very promising approach that will allow to improve the quality of MC production at the LHC.
- Results show that the GAN can reproduced simulated events with very high accuracy.

BACKUP

GANs functions (1/2)

Generator

z $\rightarrow$ Noise vector

G(z) $\rightarrow$ Generator's output for x_{fake}

Discriminator

x $\rightarrow$ training sample

D(x) $\rightarrow$ Generator's output for x_{real}

D(G(z)) $\rightarrow$ Discriminator's output for x_{fake}

Generator G	Discriminator D
D(G(z)) $\rightarrow$ should be maximized	D(x) $\rightarrow$ should be maximized D(G(z)) $\rightarrow$ should be minimized

Loss functions

G_{loss} = log(1-D(G(z)) or -log(D(G(z))

$D_{loss_{real}}$ = log(D(x))

$D_{loss_{fake}}$ = log(1-D(G(z))

$D_{loss} = D_{loss_{real}} + D_{loss_{fake}} =$ log(D(x)) + log(1-D(G(z))

GANs functions (2/2)

- Discriminator and Generator play a two player min-max game.

$$\begin{aligned}\min_G \max_D L(G,D) &= \mathbb{E}_{x\sim p_{data}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1-D(G(z))\right)\right] \\ &= \mathbb{E}_{x\sim p_{data}(x)}\left[\log D(x)\right] + \mathbb{E}_{x\sim p_{fake}}(x)\left[\log\left(1-D(x)\right)\right]\end{aligned}$$

- $x \sim p_{data}$ probability density function of trainig sample generated using MC method.
- $z \sim p_{fake}$ probability density function of input noise.
- The Nash equilibrium (min-max game) is reached when D is unable to distinguish fake examples from real data.
  - Hence the generator has been trained to be a good approximator of the data pdf, i.e. $p_{fake} \sim p_{data}$.

Training GANs process

Training steps:

- The main idea is to train 2 different networks to compete with each other with two different objectives:
  1. The Generator G tries to fool the discriminator D into believing that the input sent by G is real.
  2. The Discriminator D identifies that the input is fake
  3. Then, the Generator G learns to produce similar type of training data inputs.
  4. This process, called **Adversial Training**, is repeated for a while or until Nash equilibrium is found.