A Containerised Solution for GAN-based Fast Calorimeter Simulation

Federico A.G. Corchia
Lorenzo Rinaldi
Matteo Franchini
Michele Faucci Giannelli

CaloChallenge Workshop
Villa Mondragone
Frascati (RM), Italy
30 May 2023

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

INFN
Istituto Nazionale di Fisica Nucleare

Summary

- We present a model of **calorimeter simulation** based on Generative Adversarial Networks (**GANs**) aiming at simulating calorimeter response as similarly as possible to Geant4, but much faster, using as training data the CaloChallenge public ones.

- We also present a general framework based on **containers to ensure full portability** of the code through different computing clusters: in this way, our framework may run not only on HEP-specific resources, but also on **supercomputing** HPC clusters providing further and more powerful resources.

The Context

- Simulation of electromagnetic and hadron interactions in calorimeters: **very demanding process**, both in terms of time and computing resources.

- Detector simulation: about half of total workload on CERN computing resources, of which calorimeter simulation takes about 80%.

- Only expected to increase with Run 3 and with High Luminosity-LHC!

- CERN resources (LXPLUS, Grid): good but not infinite!

Ways for Improvement

- Improvement through:
  - better and faster analysis techniques → BolognaGAN, based on Generative Adversarial Nets (GANs);
  - more efficient use of computing resources → containers and clusters other than LXPLUS.

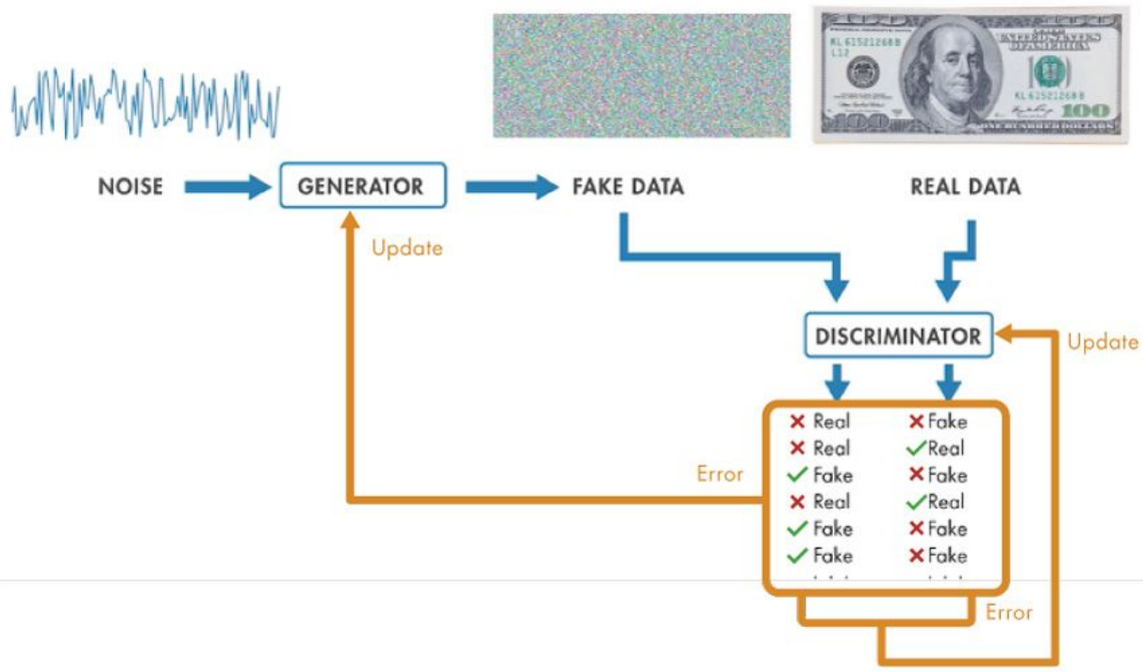
BolognaGAN

Basics on GANs

- Introduced in Goodfellow, I.J. *et al.*, *Generative Adversarial Nets*, *Advances in Neural Information Processing Systems* (2014).

- Training samples + generator (G) aiming at generating samples replicating the distribution of training data + discriminator (D) getting samples and determining where they come from (G or training data?).

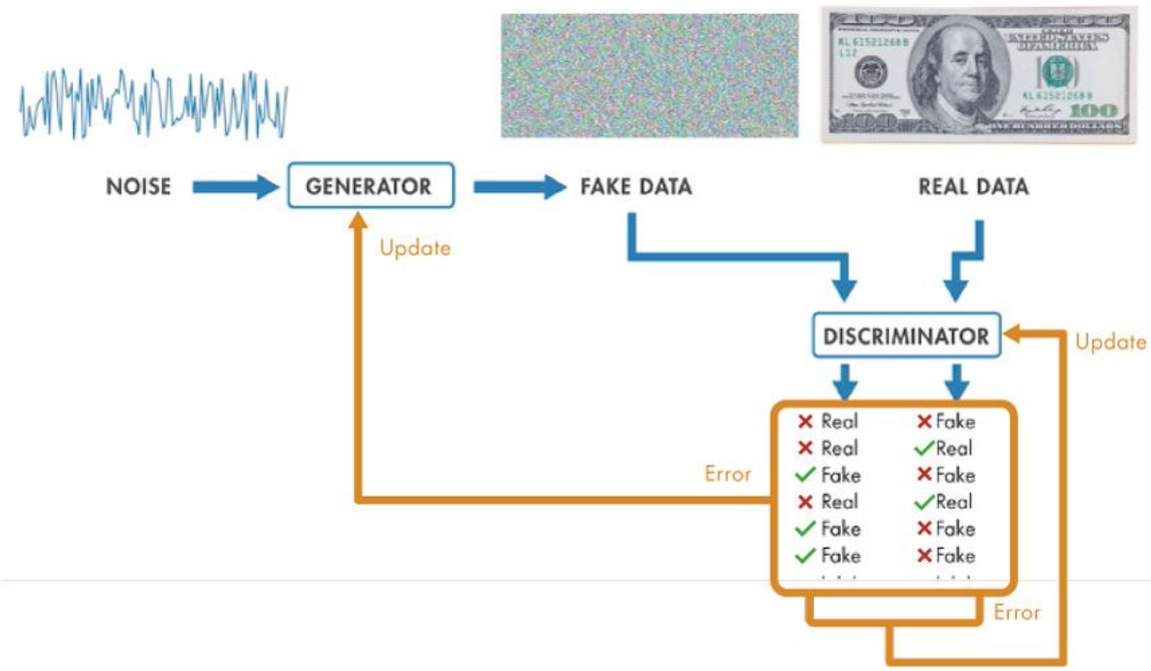
- Simultaneous training of G and D. G must maximise the probability for D to believe a sample from G is real data, D must make this hard for G.



BolognaGAN

Application of GANs to BolognaGAN

- In BolognaGAN, training data = Geant4 data. G must learn how to produce Geant4-like data. When finished training, BolognaGAN should generate calorimeter simulation data as Geant4 but much faster!



- For this contribution we have used as Geant4 data the ones of the challenge.

BolognaGAN

Further Technical Details

- Derived and evolved from *FastCaloGAN*, a fast simulation tool being evaluated in the ATLAS collaboration at CERN (M. Faucci Giannelli, ATL-SOFT-PUB-2020-006, cds.cern.ch/record/2746032/files/ATL-SOFT-PUB-2020-006.pdf);

- Mostly coded in Python + bash scripts interconnecting Python sources;

- Voxelisation phase (see next) coded in C++;

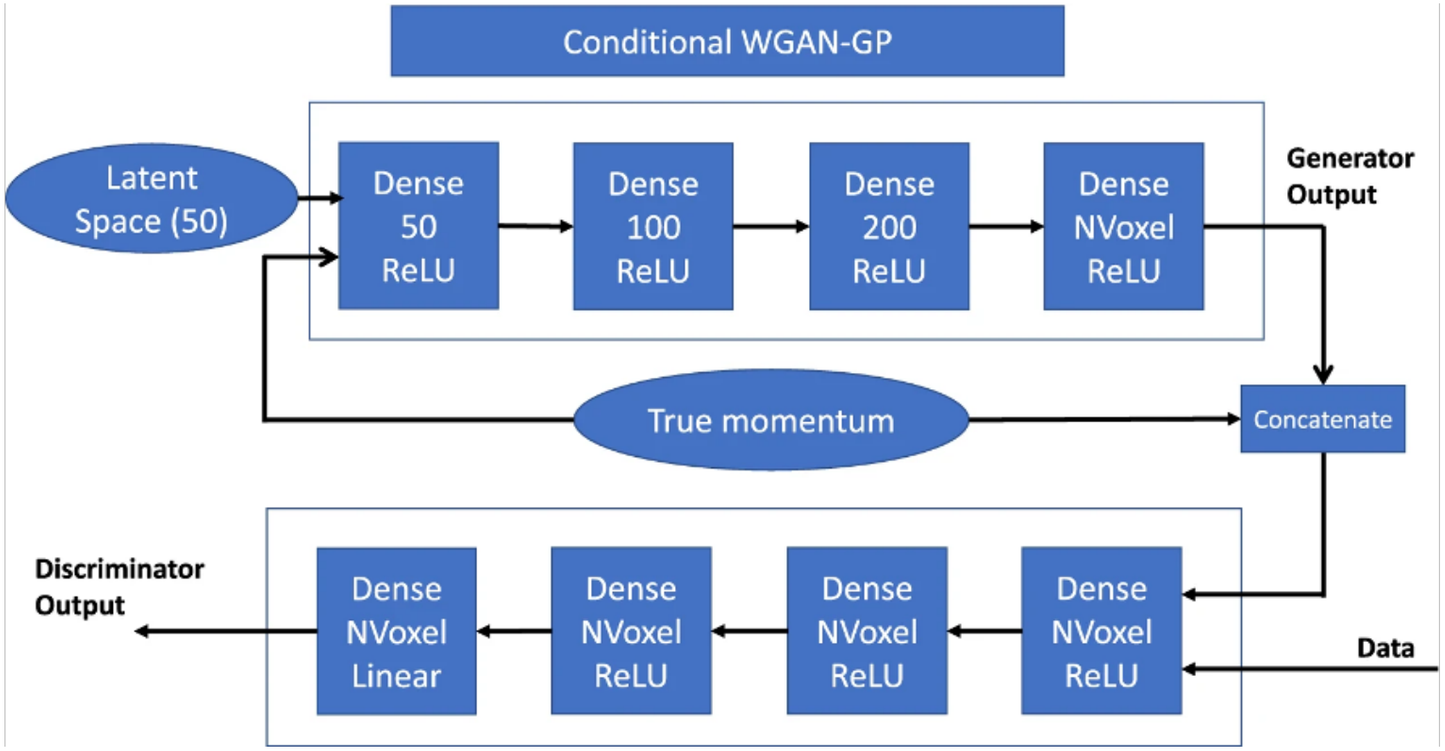
- Includes ROOT and Tensorflow; designed to run with HTCondor on CentOS 7 (x86_64 arch).

BolognaGAN

Further Technical Details (cont'd)

- Wasserstein GANs with gradient penalty (WGAN-GP) term in loss function of discriminator → good performance and training stability. Implemented in TensorFlow 2.0 so that training may be performed on either CPUs or GPUs.

- Both G and D using Adam optimiser.



BolognaGAN

Execution

- Execution in three stages, starting from event files: voxelisation → GAN training → GAN evaluation to choose the best iteration.
- Between them: scripts for checking and tidying.
- Challenge data already voxelised → start directly from GAN training.

BolognaGAN

Current Use Cases

- Currently able to simulate calorimeter showers for photons, electrons, pions and protons between 256 MeV and 4 TeV over full detector acceptance (protons only at $-0.25 \leq \eta \leq 0.25$).

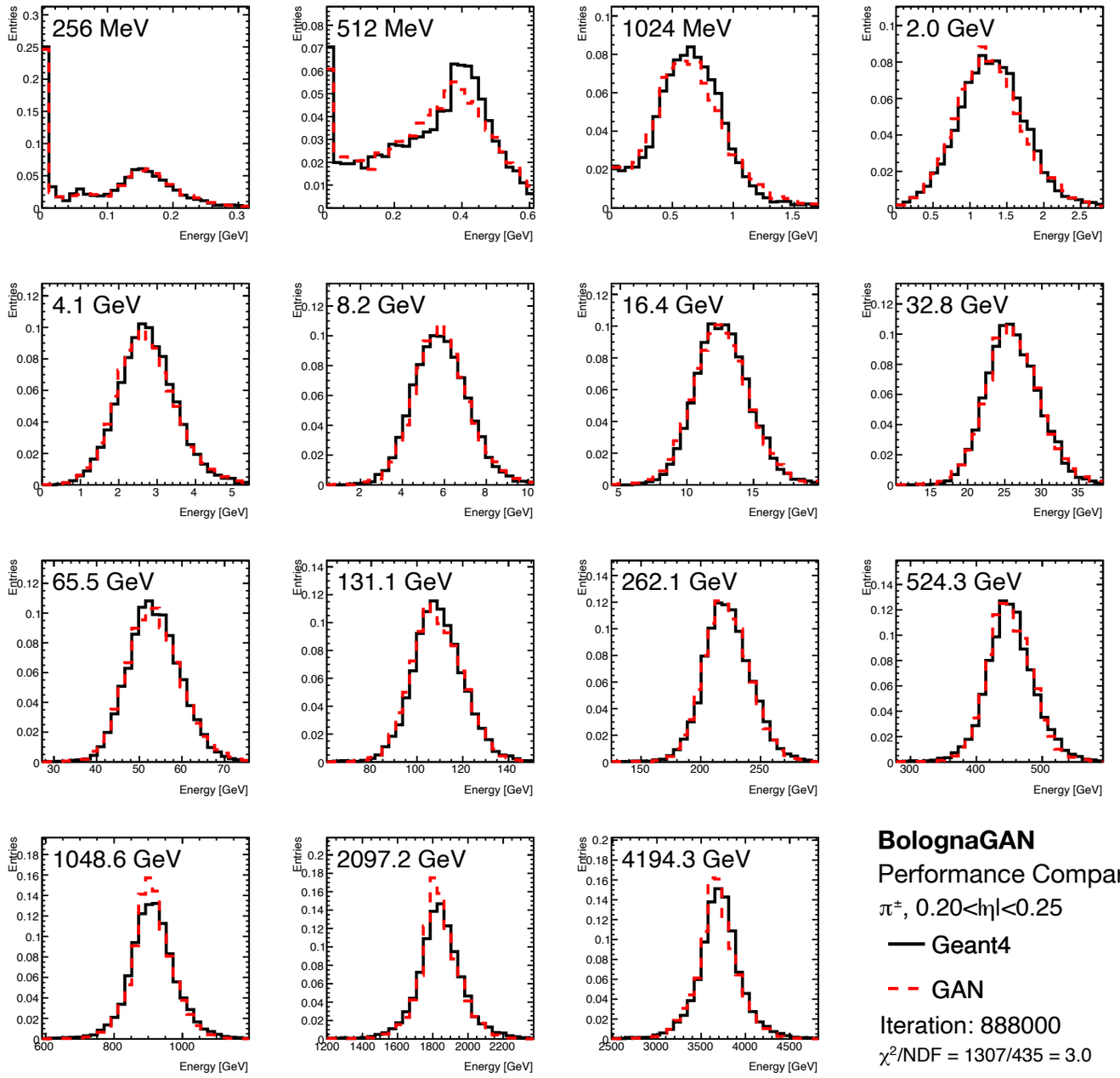
- Next slides: simulation examples, including the CaloChallenge analysis.

BolognaGAN

Performance Evaluation

- BolognaGAN performance evaluated through comparison of GAN-simulated data to the ones of Geant4.

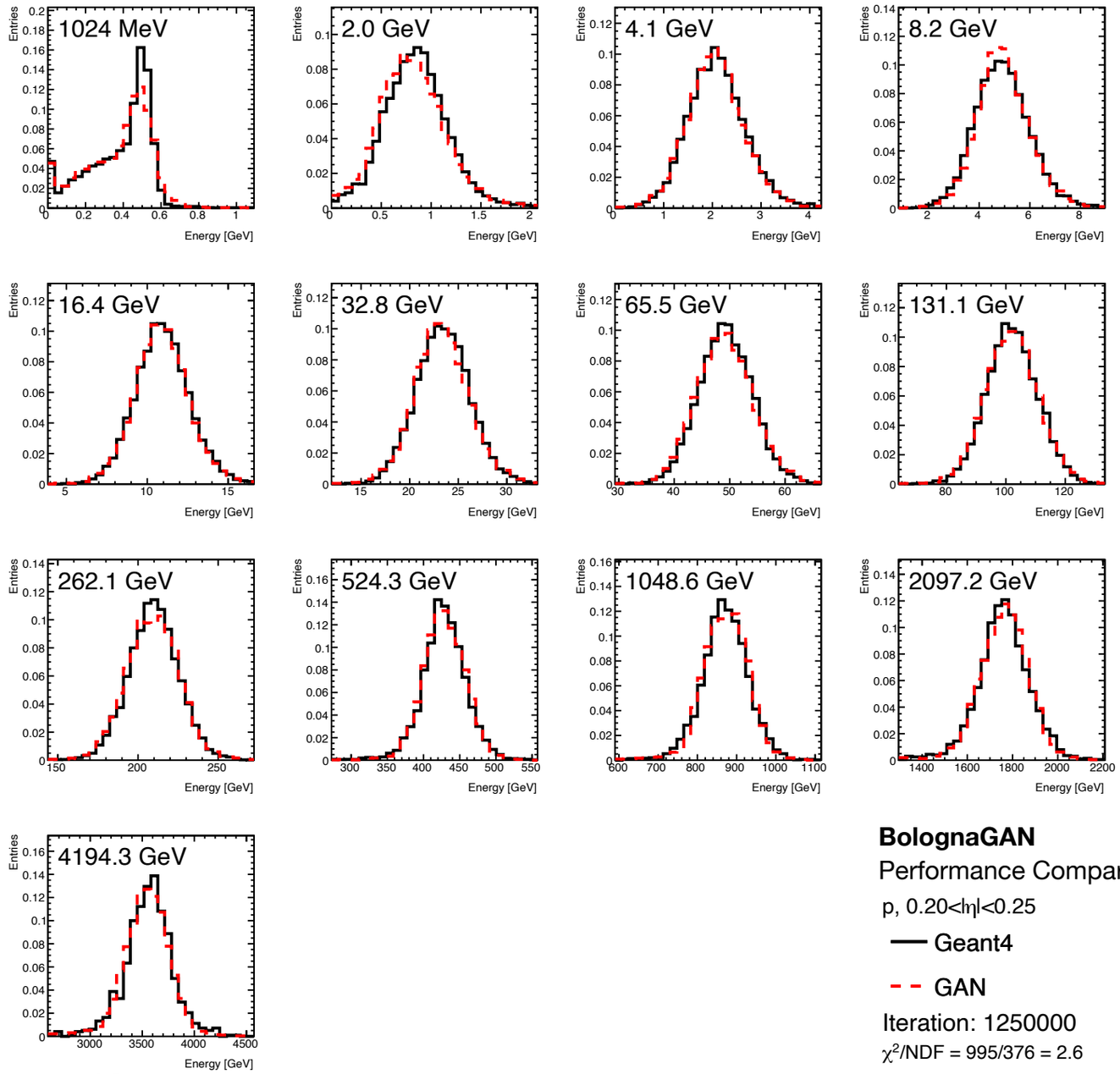
- Geant4-BolognaGAN comparison for pion simulation (own dataset). Good results, also observed for photons and electrons.



BolognaGAN

Performance Evaluation (cont'd)

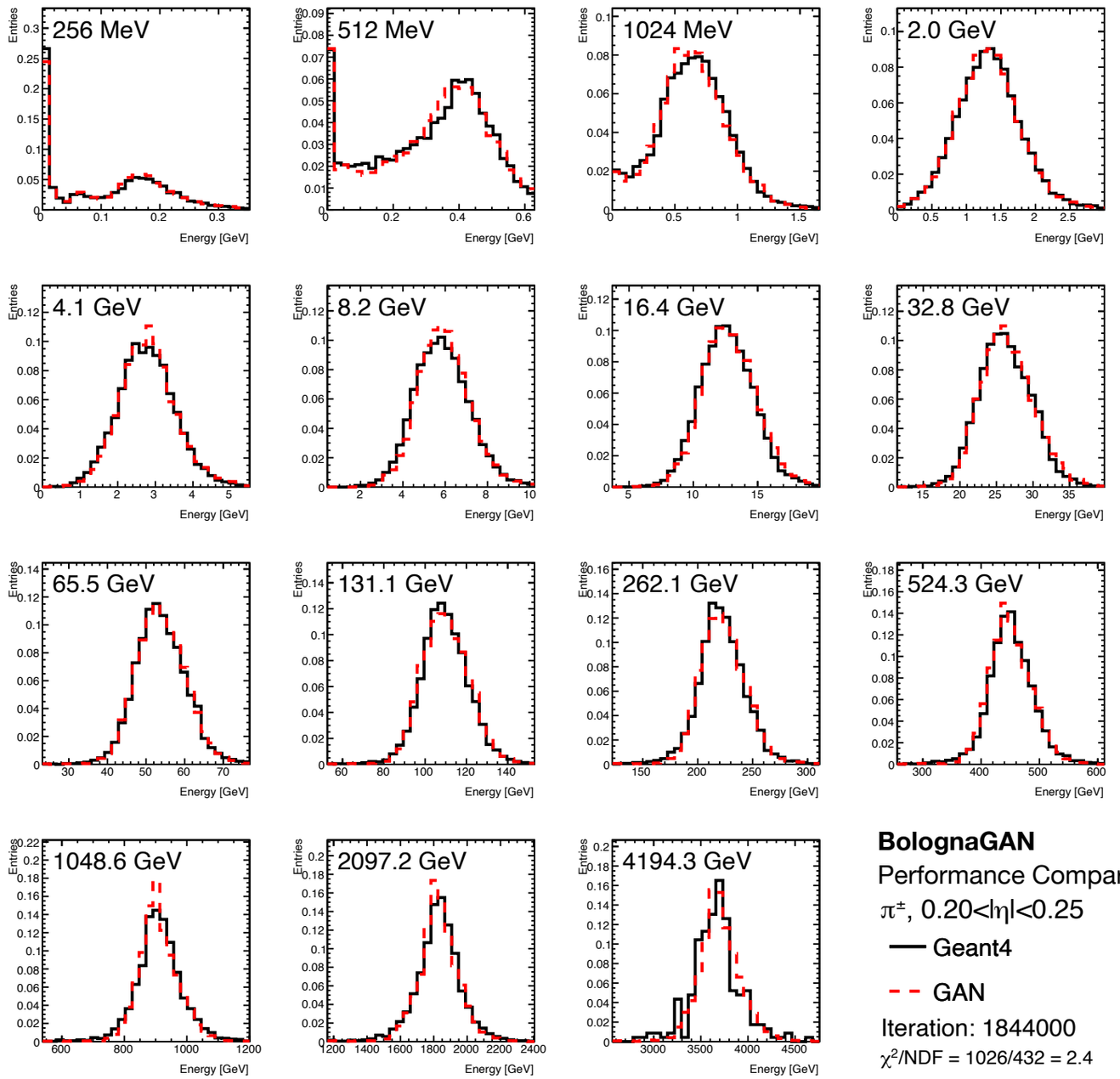
- Geant4-BolognaGAN comparison for proton simulation (own dataset). Remarkable results!



BolognaGAN

Performance with CaloChallenge Data

- Tested on Dataset 1 High Stats for pions.
- Good agreement with Geant4 data.
- To improve: modelling around peaks and of 4 TeV.



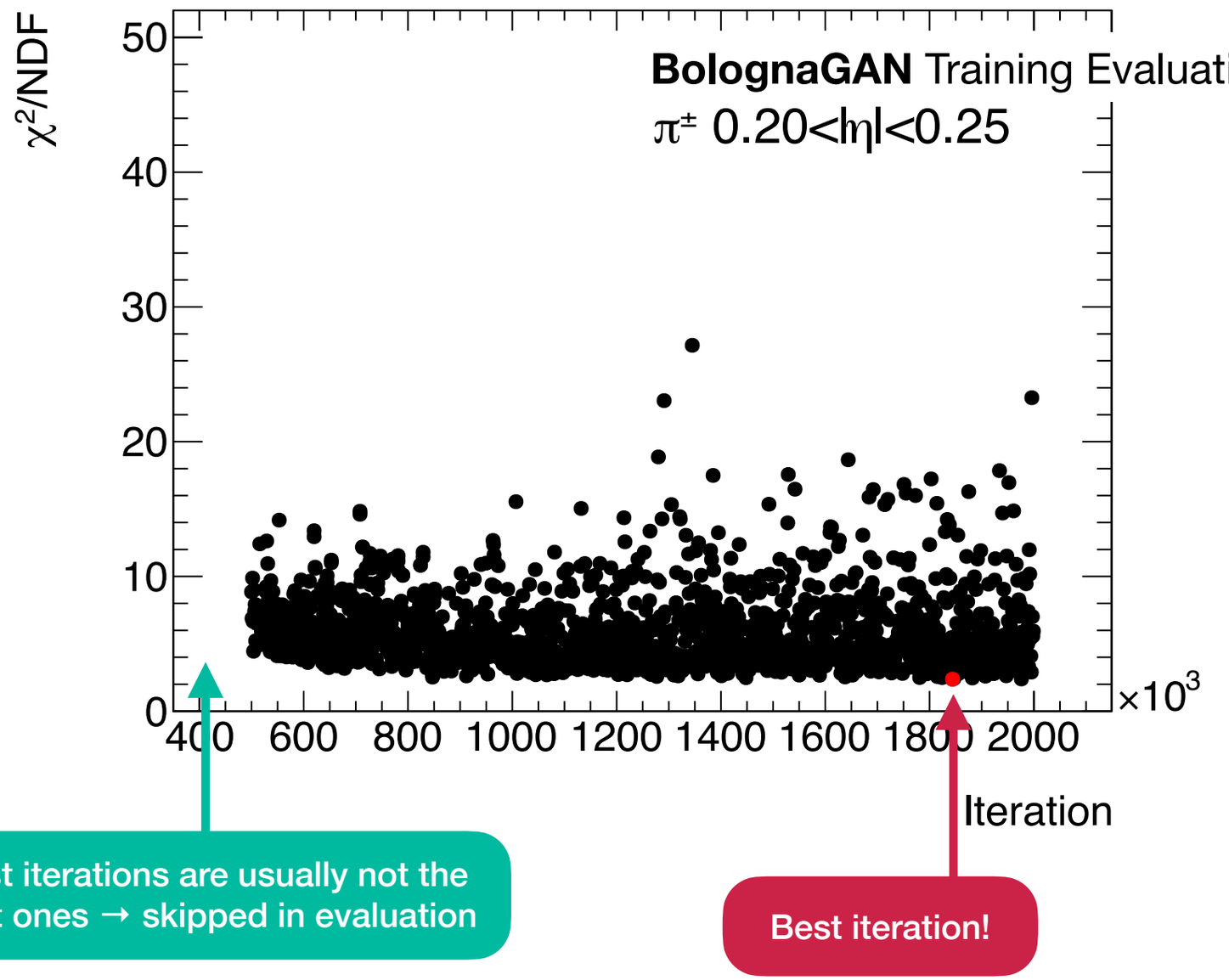
BolognaGAN

Performance with CaloChallenge Data

- After the training phase, the programme chooses the iteration returning the best χ^2/ndf;

- Right: the best iteration for the CaloChallenge pions-trained GANs is no. 1844000 with χ^2/ndf = 2.4;

- The best iteration of a GAN might not be one of the last ones but, contrarily to other Machine Learning systems, it's not just because of overfitting!



BolognaGAN

Further Development Ideas

- Improvement of results for currently studied particles (new GAN settings);

- BolognaGAN can handle all the workflow from event files to model, including voxelisation → improve voxelisation;

- Extension of use cases (further particles, further energies).

Containers

Usefulness

- BolognaGAN was designed to run on CERN resources
  - CentOS7 (intel x86_64 cores) with V100 GPUs
  - managed by HTCondor scheduler
- In order to have it run on other more powerful clusters, including general usage (i.e. non-HEP-specific) supercomputers, it must be ported.
  - Containers technology

Containers

Basic Ideas

- Technology packing all dependencies needed by a software to work into a ready-to-use environment (the container itself), inside which that software can run.

- Most common container engines (handling software): *Apptainer* (formerly *Singularity*) and *Docker*.

- The user starts the container and the program runs inside of it as if it were on another Operating System with another environment/installed software but without the burden of a Virtual Machine!

BoGANtainer

Framework Prototype

- We are making a container **prototype** so that BolognaGAN may run on other clusters and so does its analysis.

- Using Apptainer engine (no need to be a privileged user)

- **BoGANtainer**: Apptainer container replicating LXPLUS environment (CentOS7+ CVMFS + CUDA driver/libs)

BoGANtainer

Container Build recipe

```
BootStrap: docker
From: centos:centos7

%setup
 mkdir ${APPTAINER_ROOTFS}/data
bolognagan /data/bolognagan %post

%post
  yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm
 yum-config-manager -y --add-repo http://
developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/
cuda-rhel7.repo
 yum install -y nvidia-driver-latest-dkms cuda cuda-drivers
 yum install -y glibc libXpm libXft libSM libXext gcc-c++
 yum install -y git
 mkdir -p /data/cvmfs
 git clone https://github.com/cvmfs/cvmfsexec

%runscript
 if [ ! -d /cvmfs ]; then
   echo "Installing cvmfs with cvmfs-exec"
  /tmp/$USER/cvmfs/cvmfsexec/makedist default
   /tmp/$USER/cvmfs/cvmfsexec/cvmfsexec grid.cern.ch atlas.cern.ch
atlas-nightlies.cern.ch atlas-condb.cern.ch sft.cern.ch sft-
nightlies.cern.ch --
 fi
```

Build from centos7

Local FS within the container

Installation of

- **CUDA**
- **cvmfs-exec (unprivileged)**
- **Other needed sw utils**

Runscript for cvmfs-exec start (if cvmfs not present on the host node)
BolognaGAN code directory band

BoGANtainer

Container Build recipe

```
BootStrap: docker
From: centos:centos7

%setup
 mkdir ${APPTAINER_ROOTFS}/data
bolognagan /data/bolognagan %post

%post
  yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm
 yum-config-manager -y --add-repo http://
developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/
cuda-rhel7.repo
 yum install -y nvidia-driver-latest-dkms cuda cuda-drivers
 yum install -y glibc libXpm libXft libSM libXext gcc-c++
 yum install -y git
 mkdir -p /data/cvmfs
 git clone https://github.com/cvmfs/cvmfsexec

%runscript
 if [ ! -d /cvmfs ]; then
   echo "Installing cvmfs with cvmfs-exec"
  /tmp/$USER/cvmfs/cvmfsexec/makedist default
   /tmp/$USER/cvmfs/cvmfsexec/cvmfsexec grid.cern.ch atlas.cern.ch
atlas-nightlies.cern.ch atlas-condb.cern.ch sft.cern.ch sft-
nightlies.cern.ch --
 fi
```

The recipe produces a container image (SIF)

The container image can be run interactively or within a batch scheduler (SLURM or HTCondor)

BolognaGAN code directory is installed inside the container

Data directory is bind-mounted at runtime

BoGANtainer

Where it has been tested

- INFN-Bologna cluster
    - (old) nodes with CentOS7, cvmfs, HTCondor, no GPUs
- Department of Physics and Astronomy computing cluster (UniBo *Open Physics Hub* project);
    - Several nodes with Rocky Linux 8, SLURM, no-cvmfs, no GPUS
- INFN-CNAF HPC cluster (close to WLCG INFN-T1)
    - Centos7 nodes, no-cvmfs, SLURM, V100 GPUs

BoGANtainer

Where we would like to test it

- Deployment ideas:
  - LEONARDO (the 4th most powerful supercomputer in the world, at CINECA in Bologna). Agreement being established between INFN and CINECA: pre-production access to resources expected shortly;
  - cloud resources (e.g. AWS, Google), only recommended when resource request peaks.

BoGANtainer

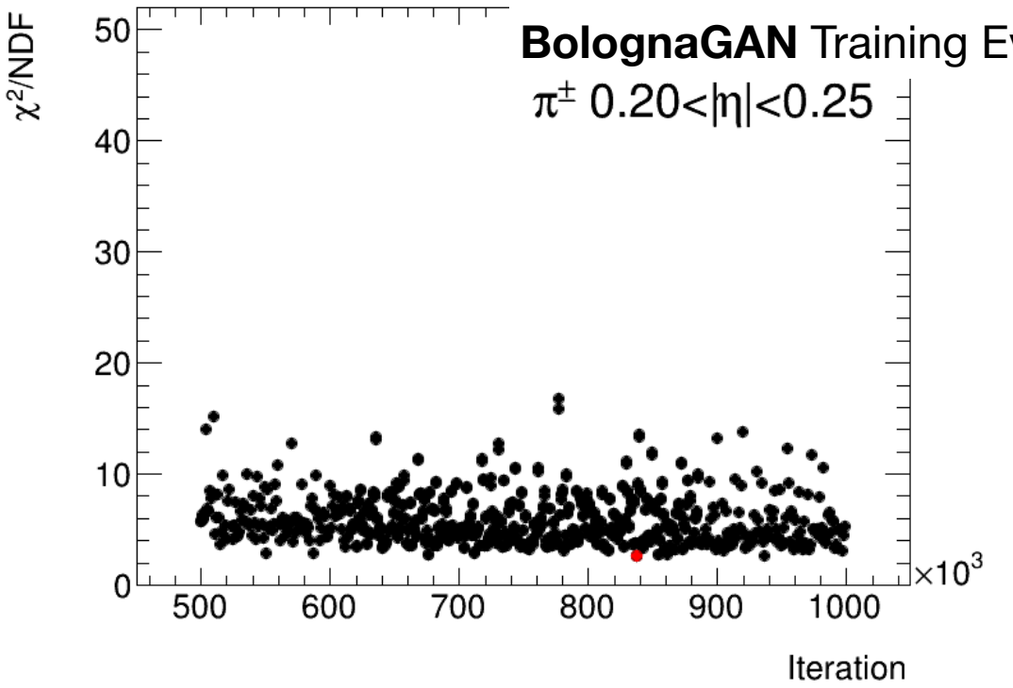
Results

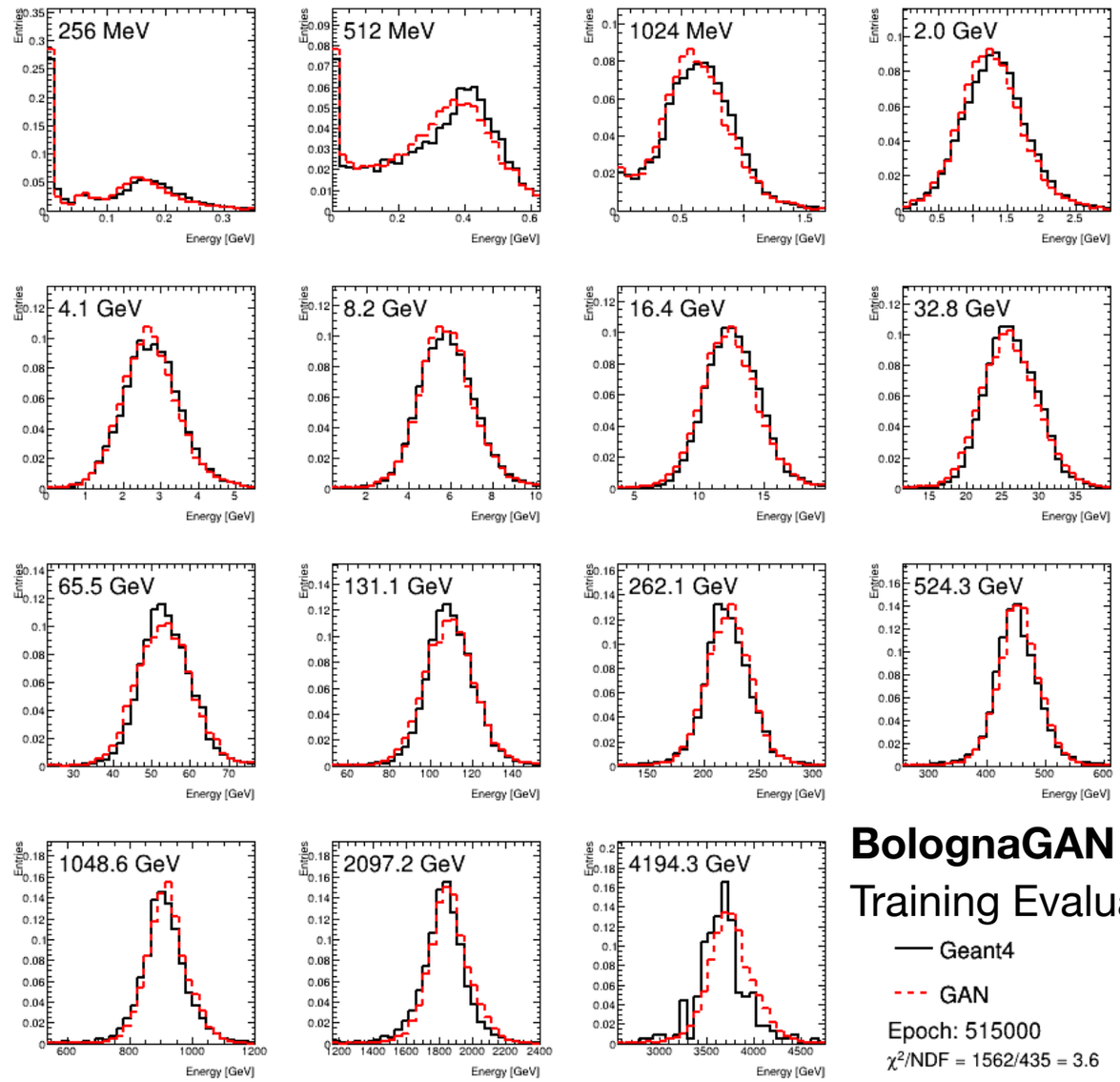
- BoGANtainer prototype successfully run on the three testbeds
  - Test performed with GAN training+validation on CaloChallenge pion dataset
- Computing performance on GPUs (INFN-CNAF) comparable with CERN machines (~6 hours)
- Much slower on single CPUs machines: ~12 hours (unibo OPH)

BoGANtainer

Results

- BoGANtainer prototype model performance are OK (here CNAF HPC)





BoGANtainer

Further Development Ideas

- BoGANtainer prototype is working
    - Improve **integration** with BolognaGAN (i.e. tree structure of the code and parameter handling)
    - Improve **parallelism** (multi-cpu, multi-GPU)
    - Test on more **resources** (possibly with GPUs)
    - Test on more **architectures** (i.e. ARM, very challenging…)
    - Make BoGANtainer **available** for you (Docker-hub), with proper **documentation**

Conclusion

- Improvement of calorimeter simulation through the usage of GANs has been proven effective and viable.
  Enhancement of this activity through containers has also been shown as full of potential;

- Further development, combined with extra ideas (the ones exchanged here!), may provide further benefit.

Thank you!