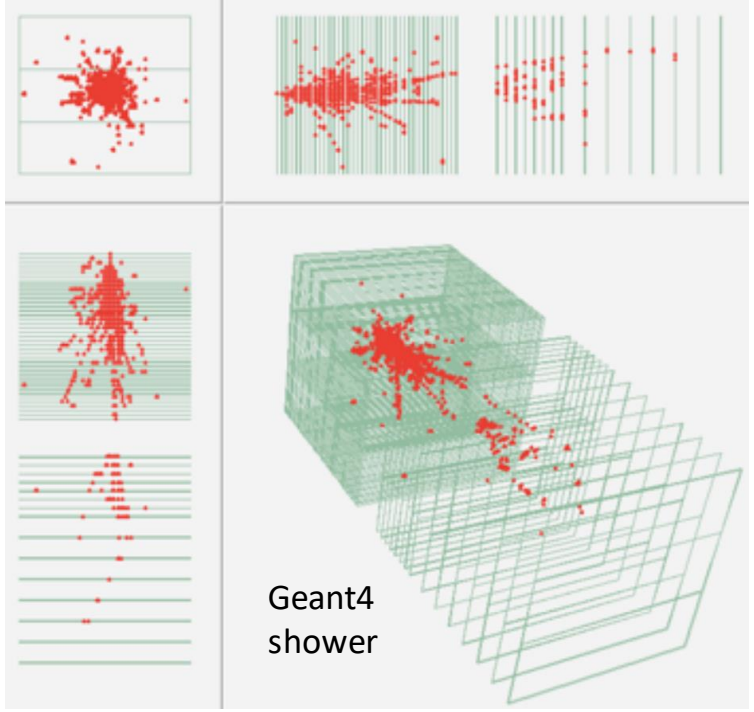
3D convolutional GAN for fast simulation

CHEP 2019

R. Brito Da Rocha, F. Carminati, G. Khattak, S. Vallecorsa

Calorimeter simulation

Data is essentially a 3D image



Electromagnetic calorimeter detector design(*)

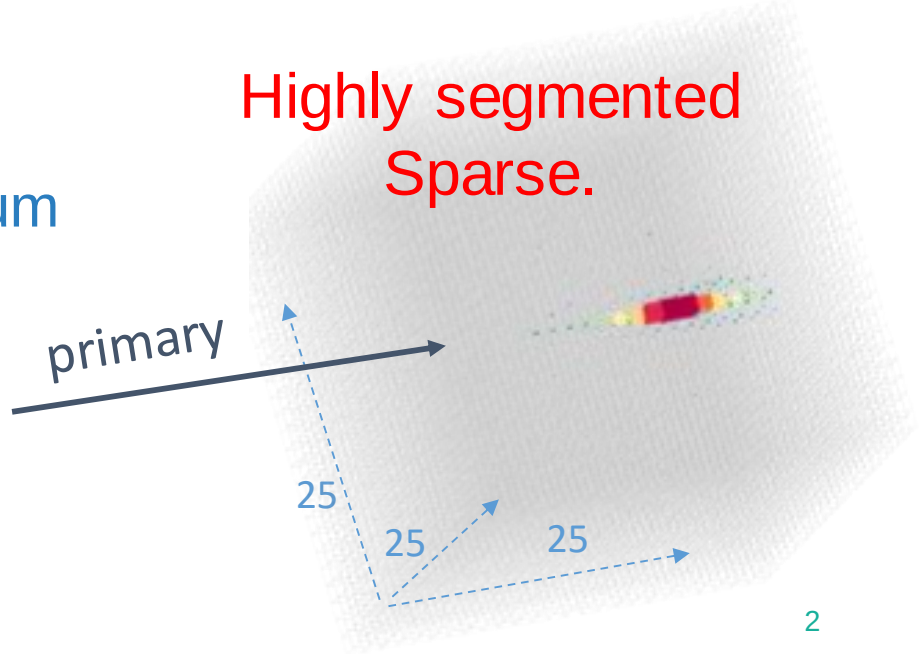
(Linear Collider Detector studies)

- 1.5 m inner radius, 5 mm × 5 mm segmentation: 25 tungsten absorber layers + silicon sensors

1M single particle samples (e,γ,π) with flat energy spectrum (10-500) GeV

- +/- 30° random incident angle (51x51x25 pixels)

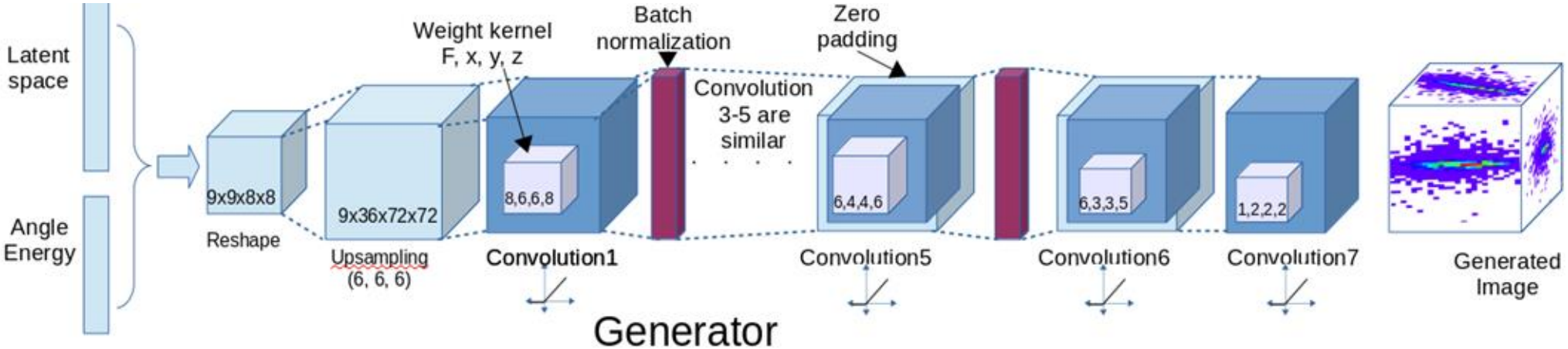
Highly segmented
Sparse.



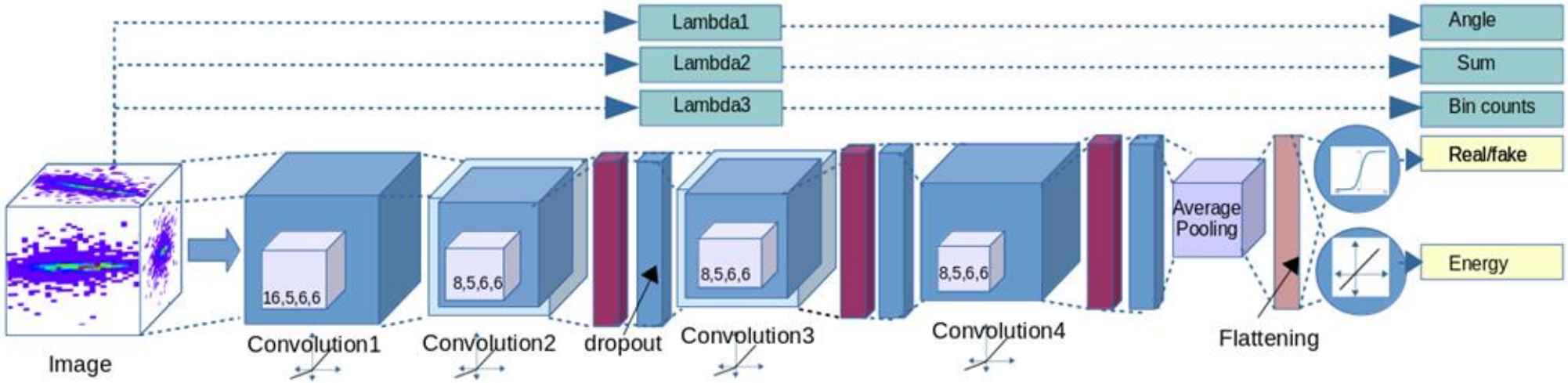
(*) http://cds.cern.ch/record/2254048#

3D convolutional GAN

Similar discriminator and generator models (1.2 M parameters total)



Generator

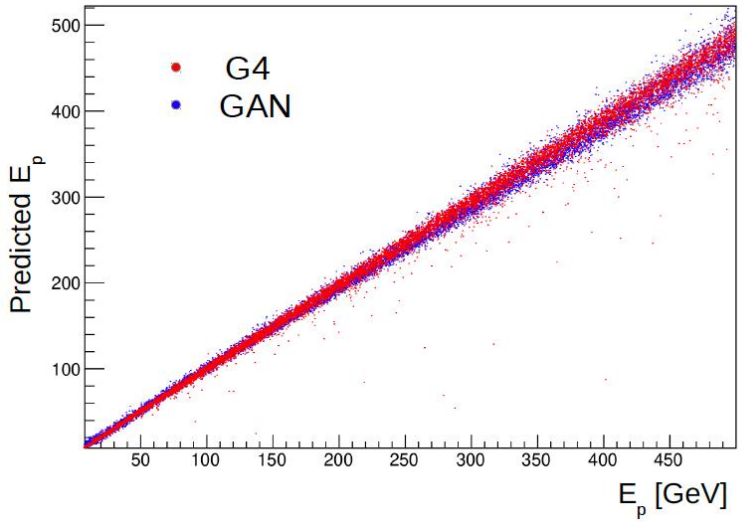


Discriminator

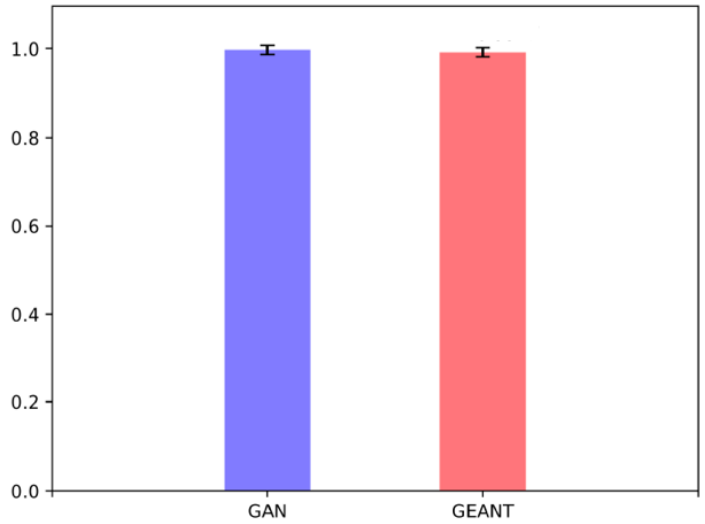
Validation

- Convergence and discriminator performance
- Comparison to GEANT4
  - Shower Shapes, Sampling Fraction
  - Correlations
  - Sparsity, etc..
- Compare TriForce[1] classification and regression results on GAN/GEANT4 events
  - Trained on Geant4 data

TriForce energy regression



TriForce electron identification



[1]Matt Zhang, https://github.com/BucketOfFish/Triforce_CaloML

Convergence

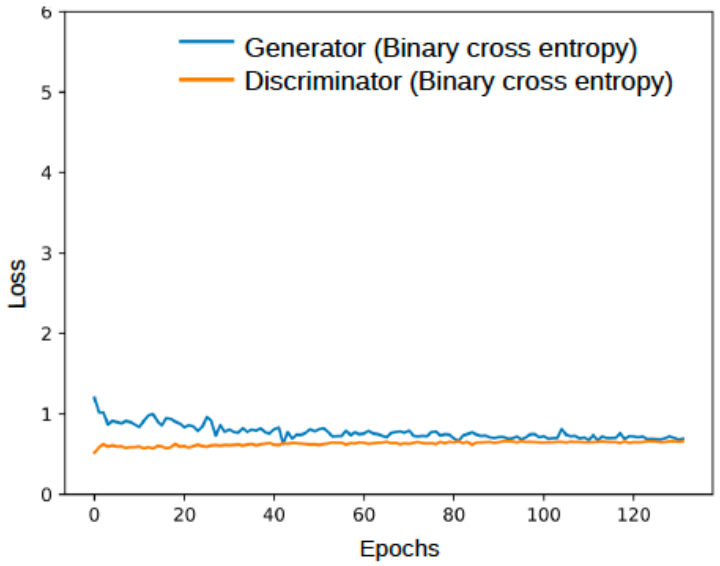
and discriminator performance

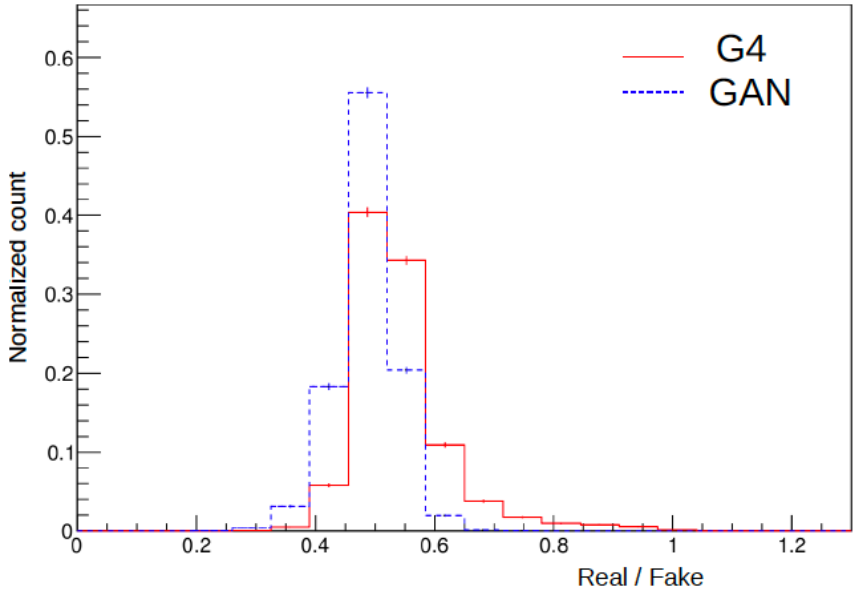
3DGAN is trained following a two-steps procedure.

- Initially restrict training set to 100-200 GeV energy range
- Transfer learning to full energy range (10-500 GeV)

Stable test losses

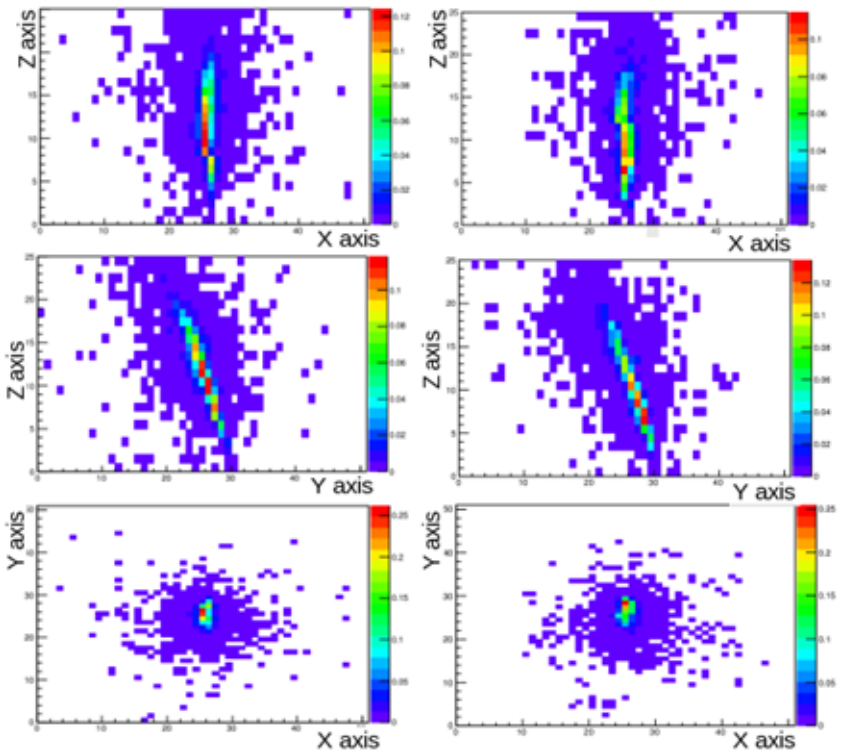
Discriminator Real/Fake probability peaks at ~50%





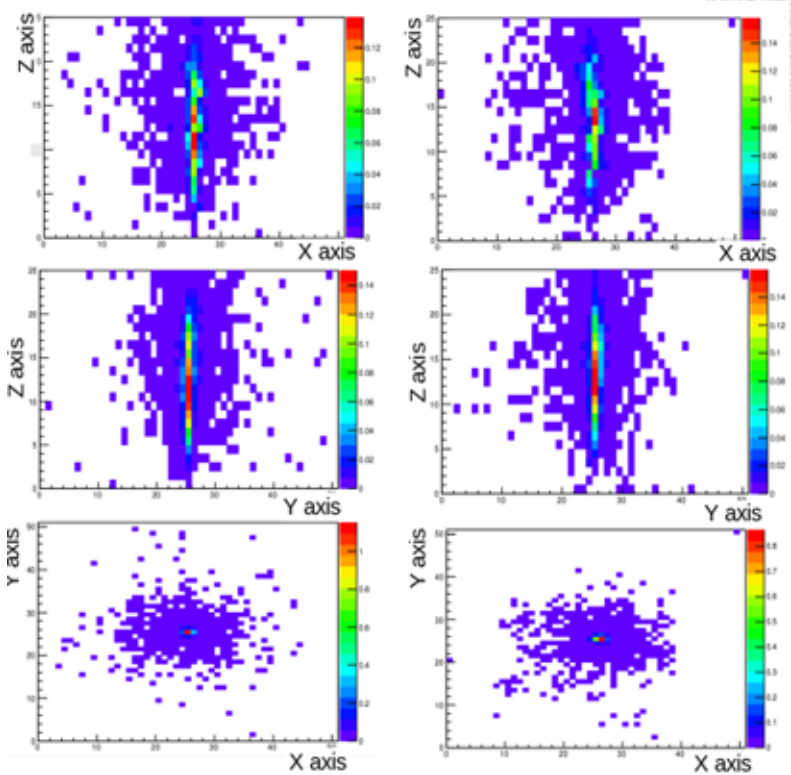
Example events

E_p=111 GeV, α=115°

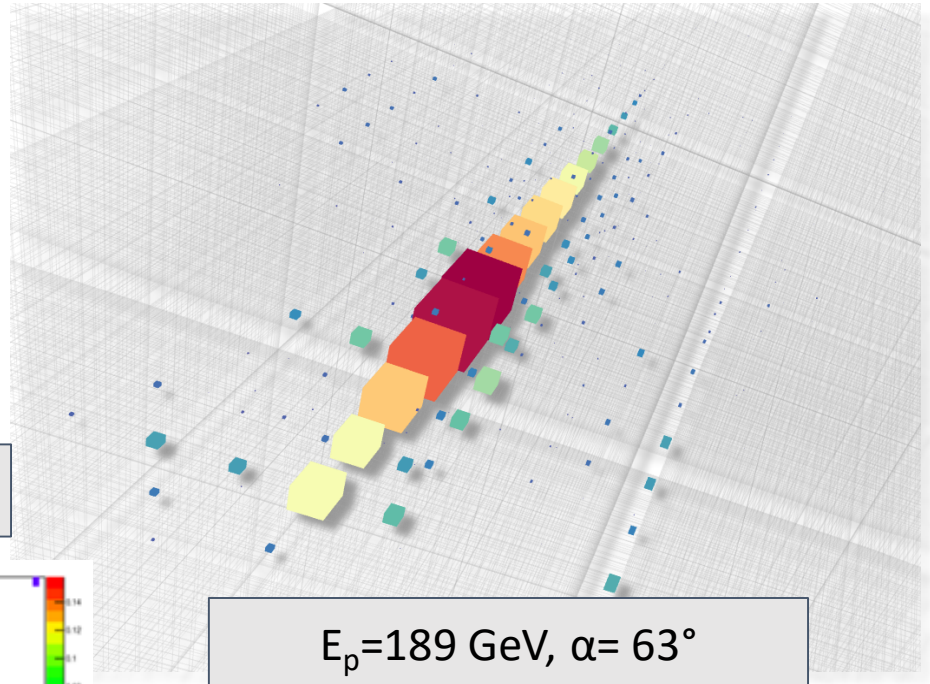


G4 GAN

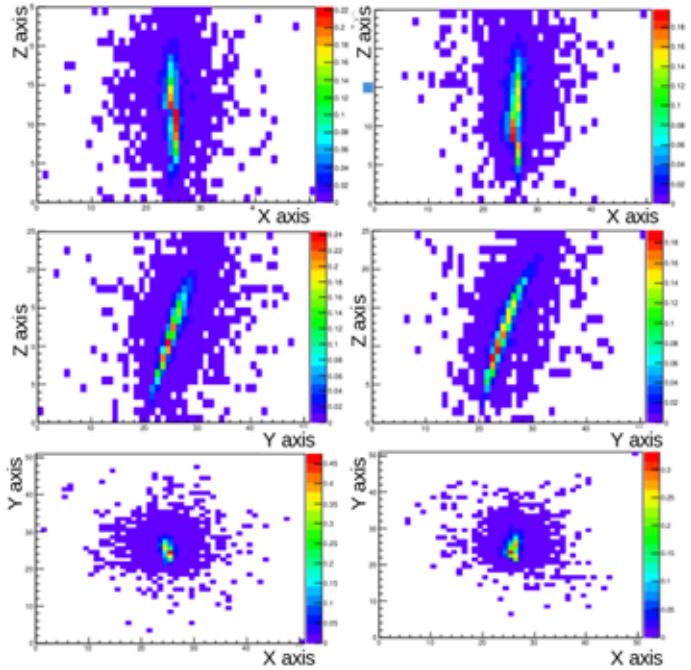
E_p=147 GeV, α= 88°



G4 GAN



E_p=189 GeV, α= 63°

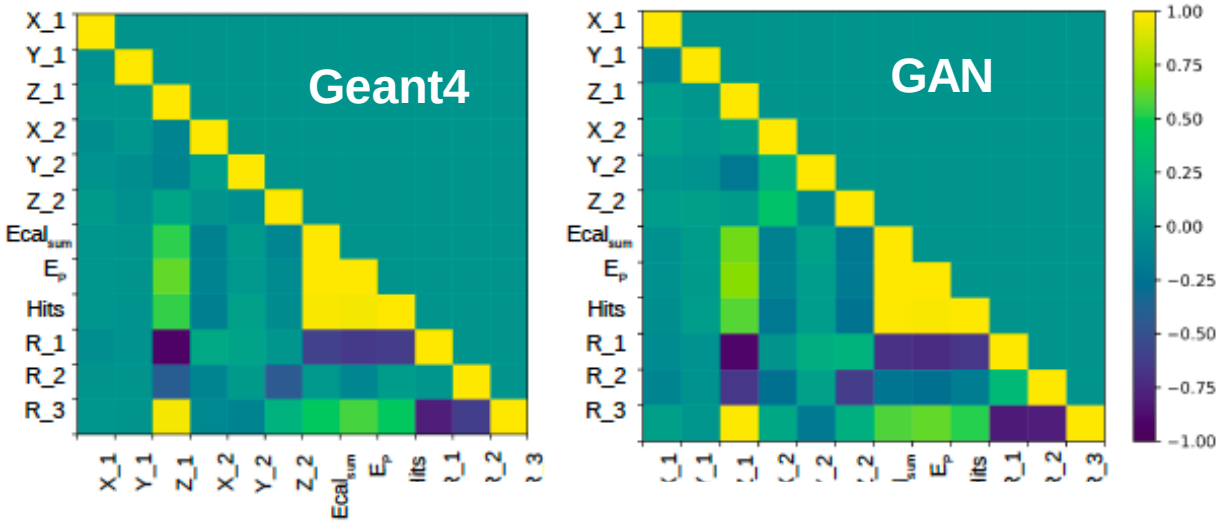


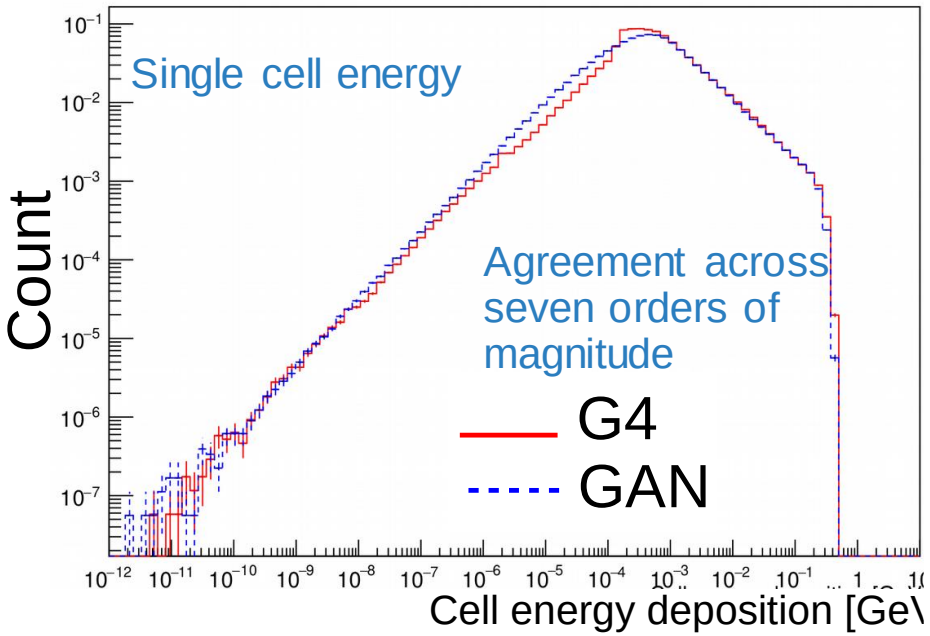
G4 GAN

CERN openlab

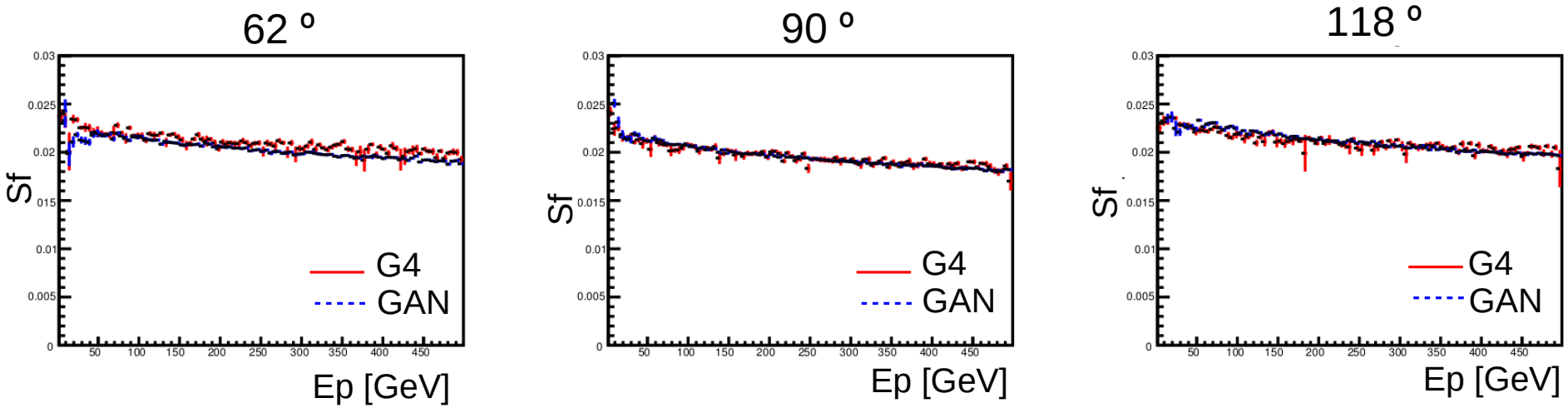
Physics validation

GAN can reproduce Geant4 internal correlations almost perfectly!



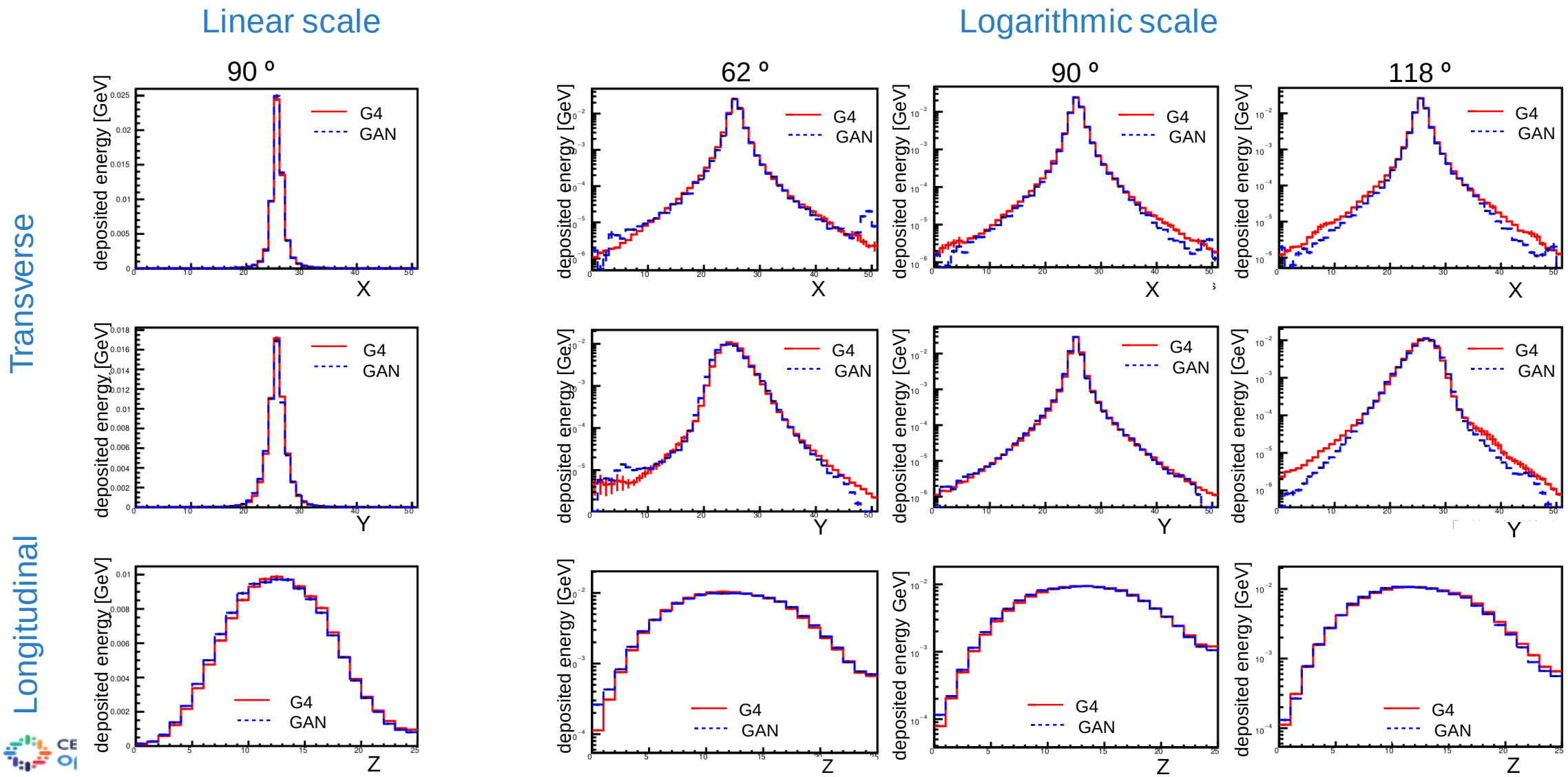


Sampling Fraction:



CERN openlab

Energy shower shapes



Distributed training on cloud

```
$ # Master + workers
$ docker run --network=host -P \
  -e
PATH="/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:
/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin" \
  -e
LD_LIBRARY_PATH="/usr/local/nvidia/lib:/usr/local/nvidia/lib64"
\
  -e LIBRARY_PATH="/usr/local/cuda/lib64/stubs" \
  -e CUDA_VERSION="9.0.176" -e CUDA_PKG_VERSION="9-0=9.0.176-1"
\
  -e NVIDIA_VISIBLE_DEVICES="all" -e
NVIDIA_DRIVER_CAPABILITIES="compute,utility" \
  -e NVIDIA_REQUIRE_CUDA="cuda>=9.0" \
  -v "/usr/local/nvidia:/usr/local/nvidia:ro" \
  --device "/dev/nvidia0" --device "/dev/nvidiactl" \
  --device "/dev/nvidia-uvm" \
  gitlab-registry.cern.ch/kosamara/ecal/svalleco \
  "./master.sh" "2"

$ docker run ... \
  gitlab-registry.cern.ch/kosamara/ecal/svalleco \
  "./worker.sh"
```

Docker

For every VM, Rebuild on Code Change

master

```
apiVersion: batch/v1
kind: Job
...
        image: gitlab-registry.cern.ch/kosamara/ecal/svalleco
        command: ["./master.sh", "${num_replicas}"]
        resources:
          limits:
            nvidia.com/gpu: 1
...
```

worker

```
apiVersion: apps/v1
kind: Deployment
...
  replicas: ${num_replicas}
...
        image: gitlab-registry.cern.ch/kosamara/ecal/svalleco
        command: ["./worker.sh"]
        resources:
          limits:
            nvidia.com/gpu: 1
...
```

Global, but still needs SSH keys

Kubernetes

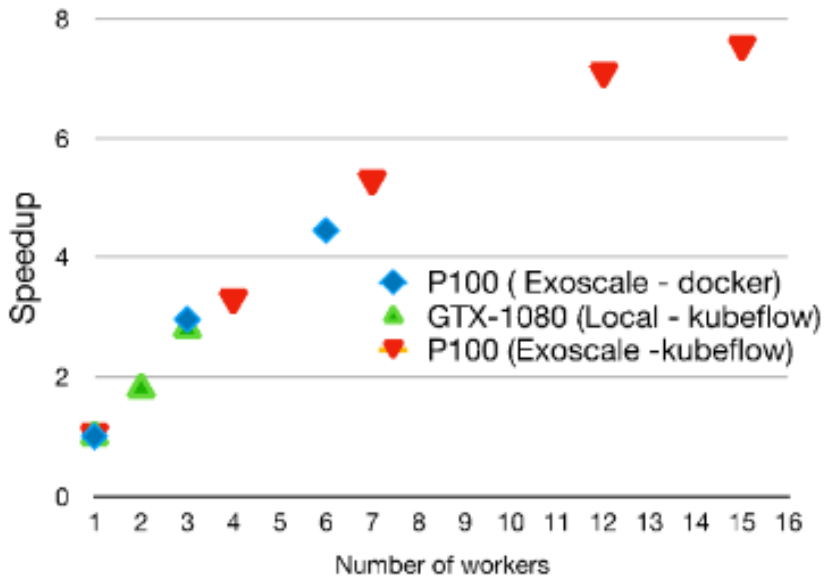
Distributed training on cloud

```
$ cat train-mpi_3dGAN.yaml
apiVersion: kubeflow.org/v1alpha1
kind: MPIJob
metadata:
  name: train-mpijob
spec:
  backoffLimit: 6
  replicas: ${num_replicas}
  template:
    spec:
      hostNetwork: true
      containers:
      - image:
gitlab-registry.cern.ch/kosamara/ecal/mpijob
        name: train-mpijob
        resources:
          limits:
              nvidia.com/gpu: 1
```

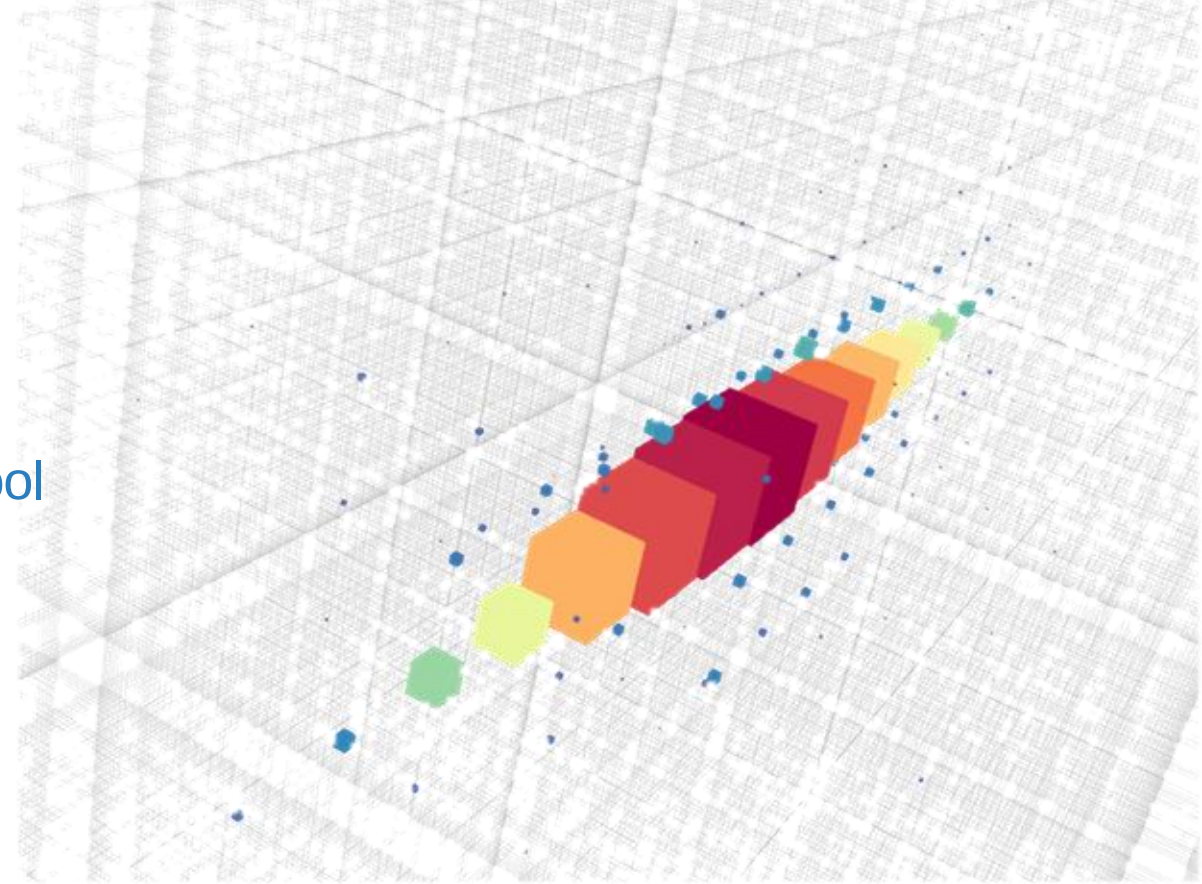
No Master / Worker Differentiation

MPI, TensorFlow and PyTorch Possible

Easy Scaling Out



Summary & Plans



3DGAN: first step towards customizable simulation tool

- Agreement to Monte Carlo within few percent
- Work in CERN simulation group to test integration

Meta-optimization and hyper-parameters scans

- Test of several distributed training approaches
- Understand / optimize performance at scale

Simple, efficient and fast deployment of the training process is key

- First integration with kubernetes yields promising results
- Starting to experiment with TPUs