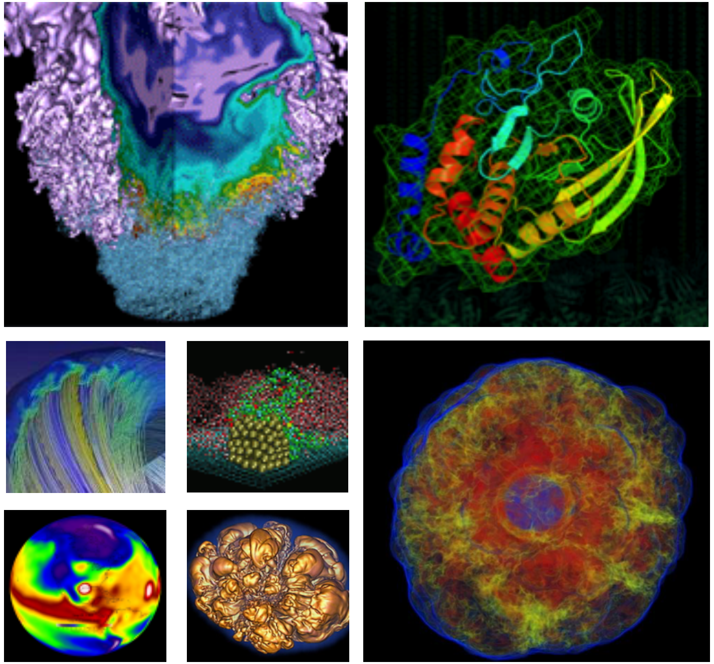
Next Generation Generative Neural Networks for HEP



Steve Farrell, Wahid Bhimji, Thorsten Kurth, Mustafa Mustafa, Debbie Bard, Zarija Lukic, Ben Nachman, Harley Patton

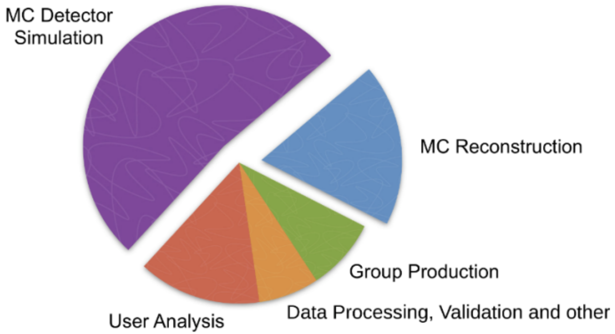
CHEP 2018, Sofia Bulgaria

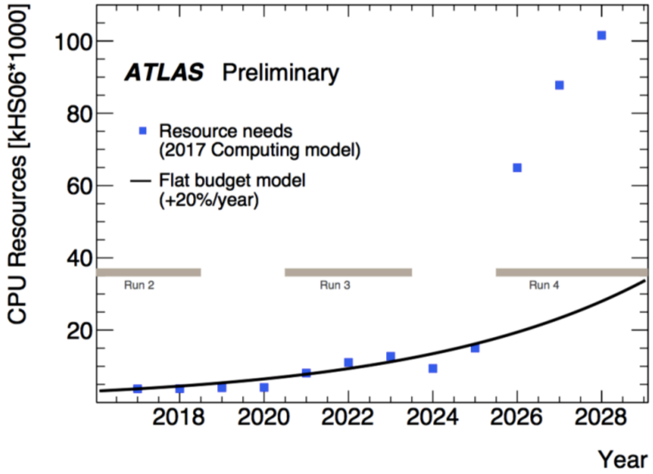
U.S. DEPARTMENT OF ENERGY | Office of Science

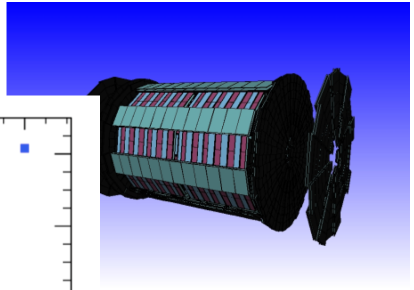
BERKELEY LAB

HEP simulation

- **Simulation is an essential application for HEP**
- **We have very powerful tools for simulation**
  - And active R&D programs
- **But gall darn is it *expensive*!**
  - Large cost in CPU resources
  - Large manpower cost in developing fast-simulation methods

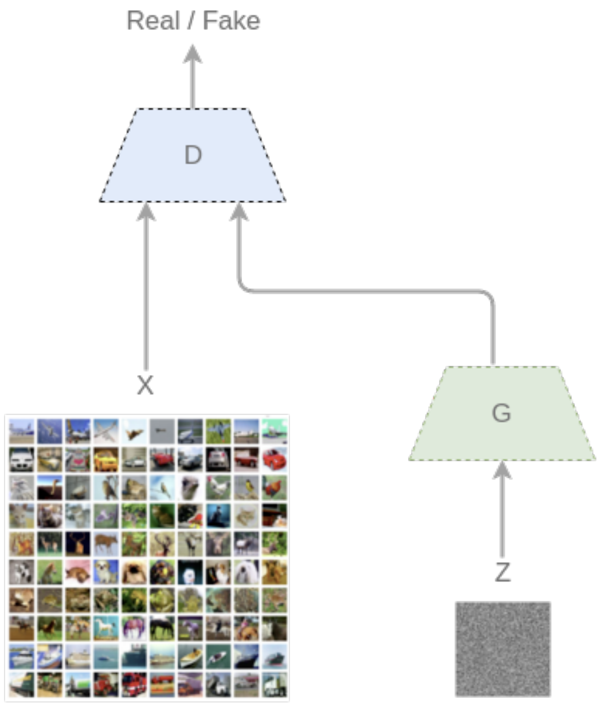






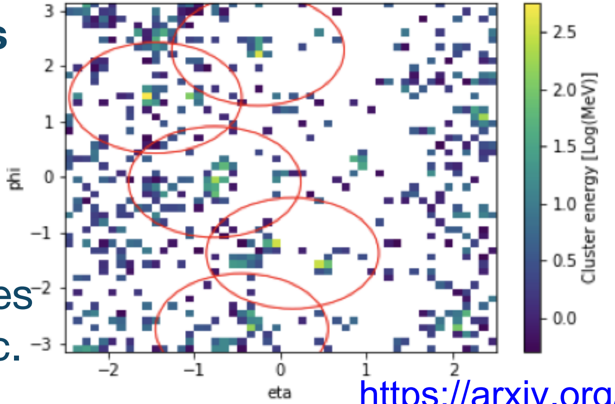
Deep learning generative models

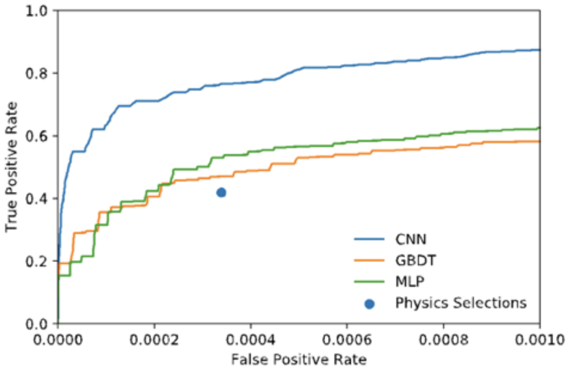
- **Deep neural networks that learn to sample from a data distribution**
  - Transform a simple "noise" distribution to the target distribution
- **Popular examples:**
  - Variational Autoencoder (VAE)
  - Generative Adversarial Network (GAN)
- **The GAN framework poses the problem as a trainable two-player game**
  - A *generator* tries to produce realistic samples
  - A *discriminator* tries to distinguish real from fake samples
- **GANs are notoriously unstable to train**
  - Difficult to define good metrics for learning



Related work

- **Deep learning on HEP images**
  - Jet images
  - Full detector images
- **Generative models for HEP**
  - Jet images, multi-layer images (CaloGAN), full 3D (CLIC), etc.

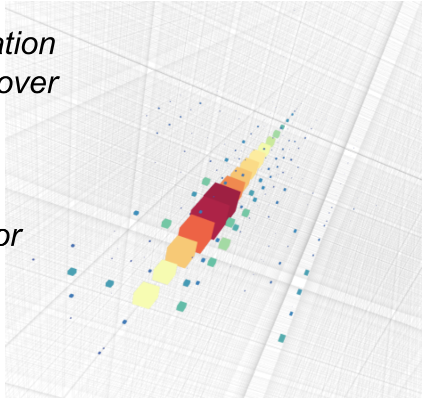


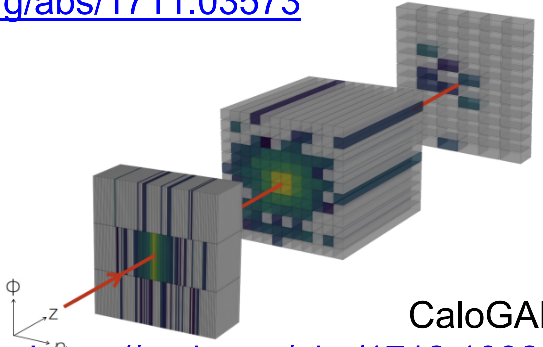


https://arxiv.org/abs/1711.03573

Relevant talks this week:

- S. Vallecorsa, *A Machine Learning Tool for fast simulation*
- J.R. Vlimant, *Training Generative Adversarial Models over Distributed Computing Systems*
- V. Chekalina, *Generative Models for Fast Calorimeter Simulation: LHCb Case*
- T. Trzcinski, *Using Generative Adversarial Networks for fast simulations in the ALICE Experiment*





CaloGAN

https://arxiv.org/abs/1712.10321

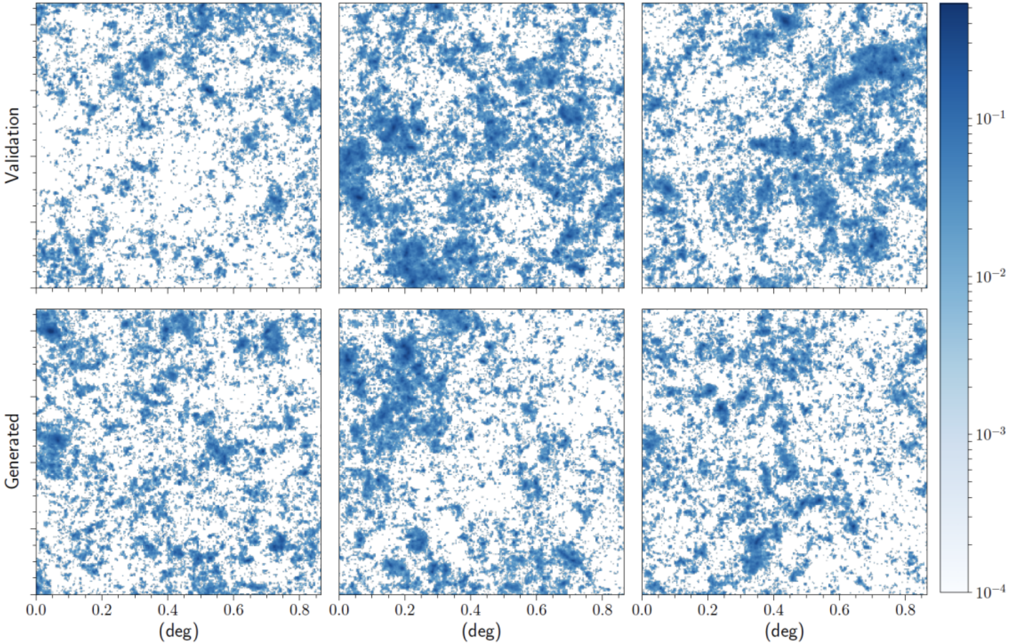
Next-generation models

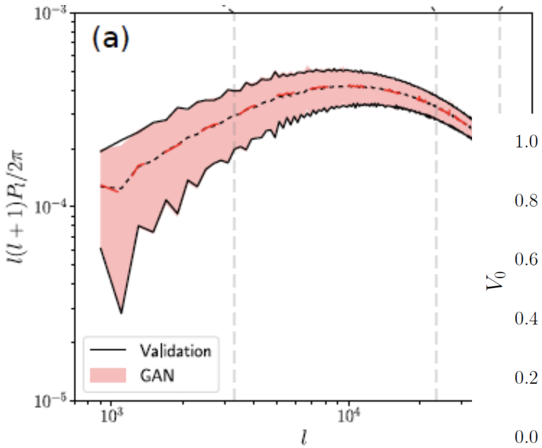
How do we take deep generative models for HEP to the next level?

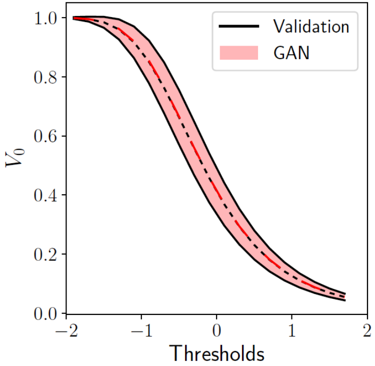
- **Develop bigger, smarter models**
  - Models that can learn more complex physics and structure
- **Improve training methods**
  - GAN stability innovations like Wasserstein GAN, Optimal-Transport GAN, Progressive GAN, Spectral-norm GAN
  - Training methods that incorporate physics knowledge
- **Improve representations and architectures**
  - Generalize beyond images
- **Improve research productivity**
  - Faster training with distributed methods
  - Improved, interactive development workflows

CosmoGAN

- Replace expensive cosmology simulations with a Deep-Convolutional GAN (DCGAN)
- Train the generator to produce weak lensing convergence maps







Resulting images have *very high fidelity* and reproduce the desired physical properties

- **2pt correlations (power spectrum)**
- **higher-order correlations (Minkowski functional)**

Mustafa Mustafa, Deborah Bard, Wahid Bhimji, Rami Al-Rfou, Zarija Lukić
https://arxiv.org/abs/1706.02390

Full HEP detector GAN

Can a GAN be trained to learn the distribution of full detector images?

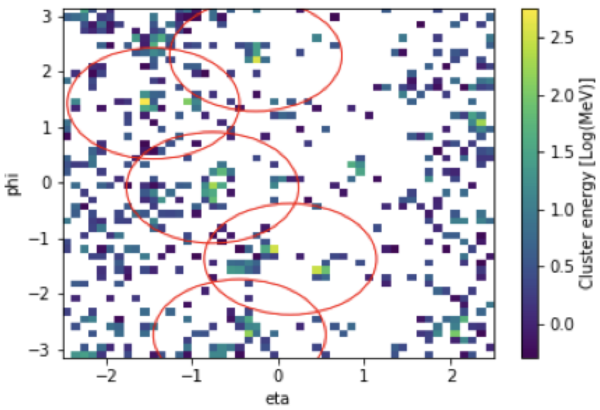
- In HEP, previously only applied to individual *particles*

Can the generator learn to produce realistic jets?

- Reconstruct-able, with the correct distributions?

How could we use such a model?

- Theory parameter interpolation
- Pileup simulation
- Other possibilities



RPV whole detector GAN

Data

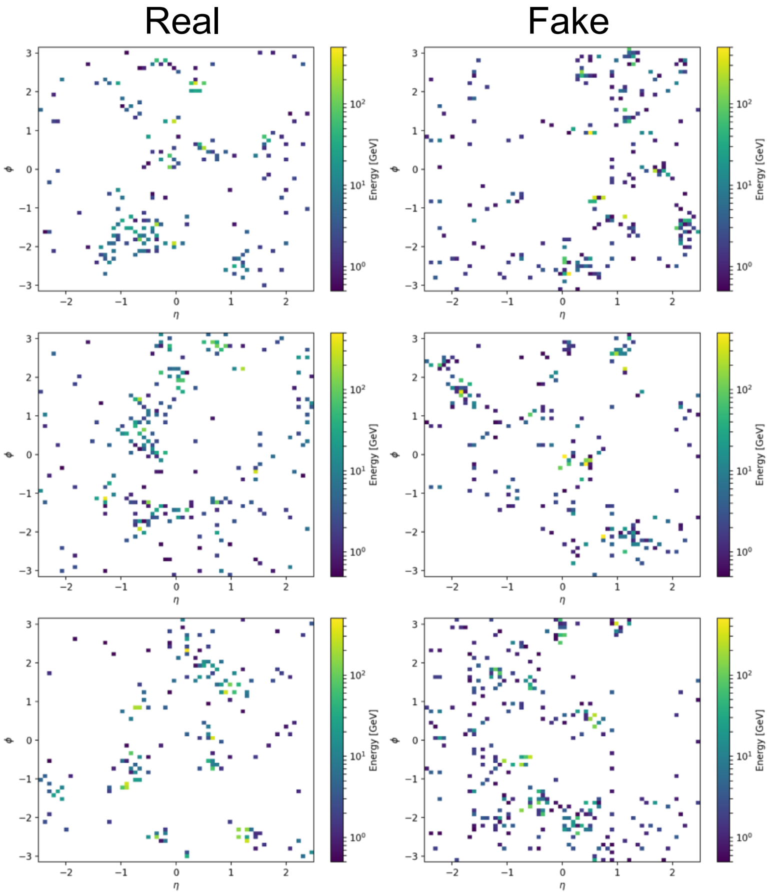
- 64 x 64 x 1 images representing calorimeter tower energy
- Delphes+Pythia simulation using ATLAS detector card

Architecture

- “Standard” DCGAN topology with 4 conv + 1 dense layer (with batch-normalization) in generator and discriminator
- Threshold on the generator output for sparsity

Analysis

- Images reconstructed with FastJet (R=1, pt>200GeV)
- *Kolmogorov-Smirnoff* test used to compare real and generated jet distributions
- KS metric used to select best model and epoch in *random hyper-parameter search*

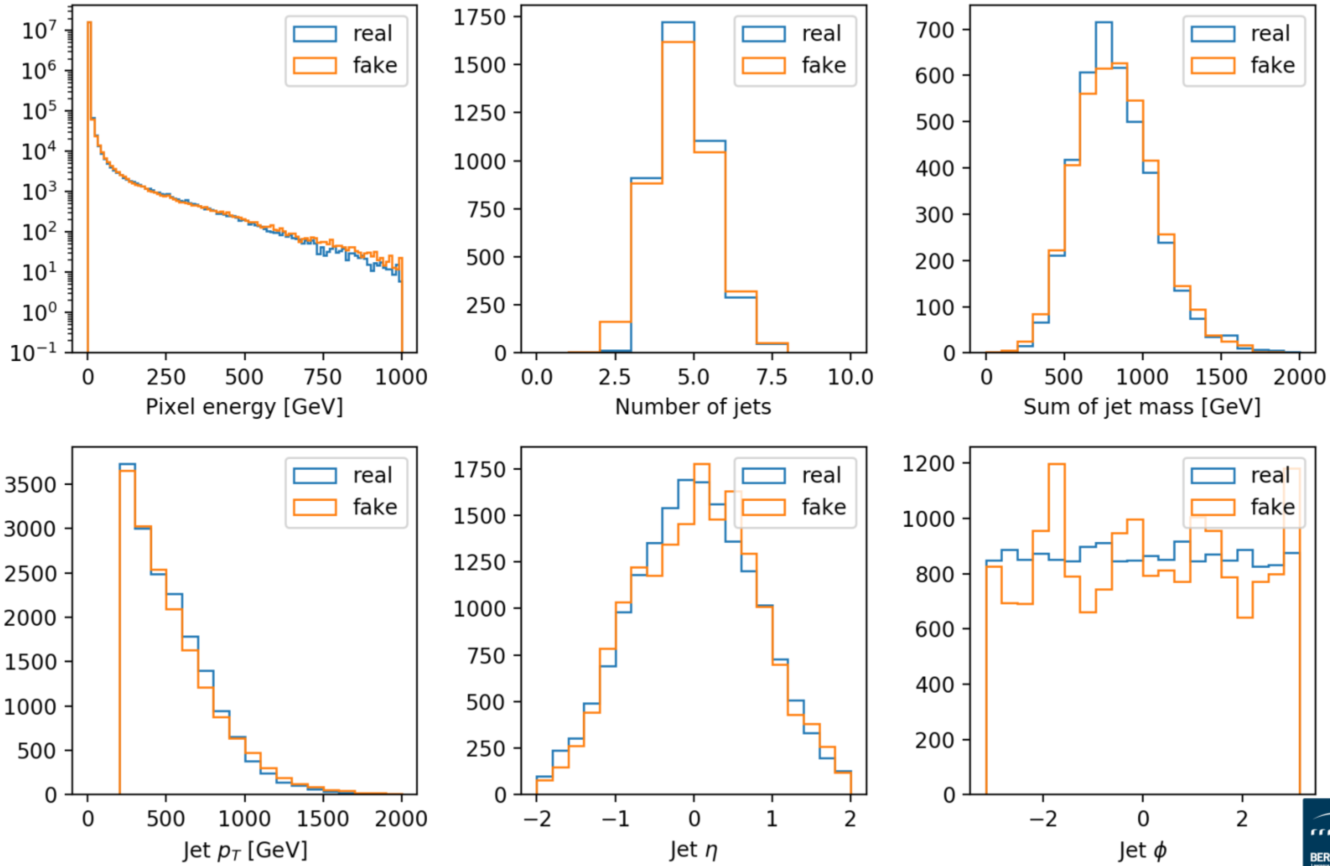


U.S. DEPARTMENT OF ENERGY | Office of Science

RPV whole detector GAN

The generated samples produce realistic jet multiplicities and kinematics

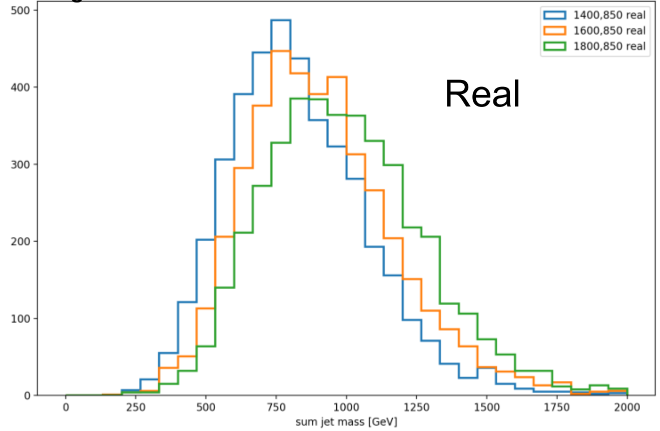
This is without imposing *any* physics knowledge!

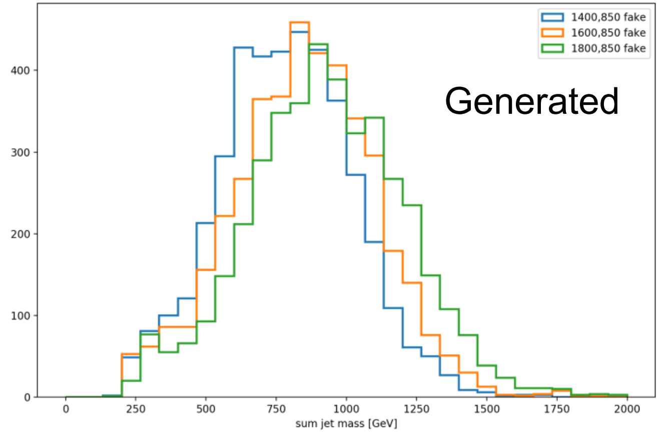


Conditional RPV GAN

- **Can the GAN learn to produce images conditional on the SUSY theory parameters?**
  - We augment the discriminator and generator to be conditioned on M^{glu},M^{neu}
- **The GAN is shown to learn the conditional distributions**
  - E.g., summed jet mass shifts as expected
- **Could use this to supplement full simulation in MC signal grids**
  - Coarse full-sim grid
  - Interpolate with GAN

M_{glu} = [1400, 1600, 1800] GeV





Work with Ben Nachman (LBNL), Harley Patton (Berkeley)

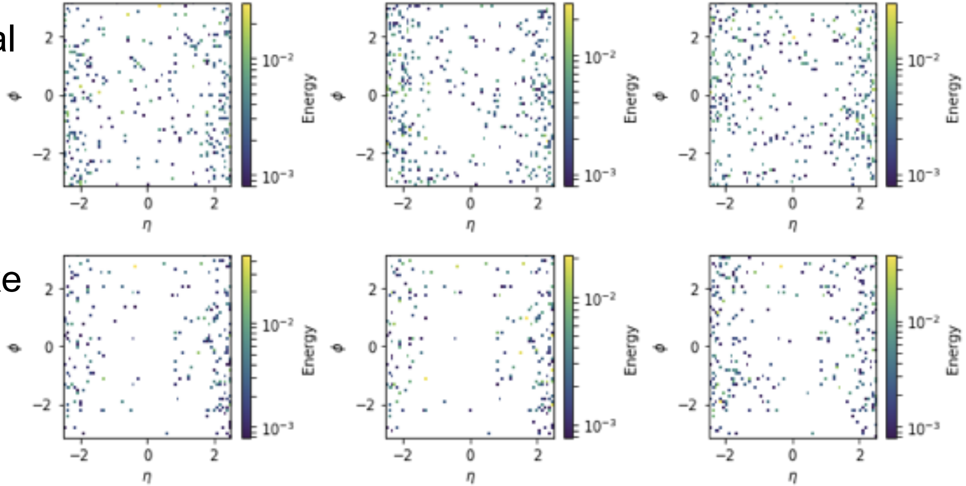
Pileup GAN

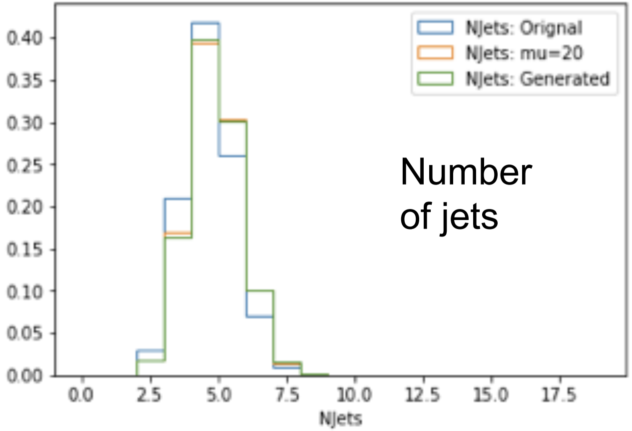
- **Pileup poses big challenges for HL-LHC computing workflows**
  - Need to simulate and store a very large volume of pileup events
  - Need to read this data from disk and overlay during digitization
- **The distribution can be modeled with a whole-detector GAN**
  - Simulate samples and train model *once*
  - Use the trained generator for fast, on-the-fly pileup sampling
- **To test fidelity, now we can evaluate the effects on reconstructed object kinematics**
  - Overlay real pileup or generated pileup
  - Compare the shifts in the distributions

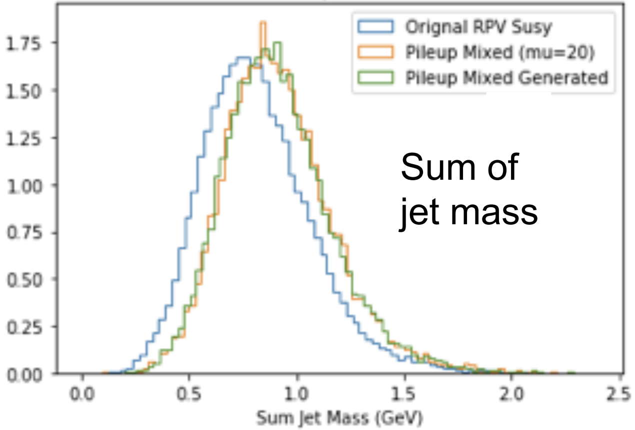
Pileup GAN - μ=20

Real

Fake



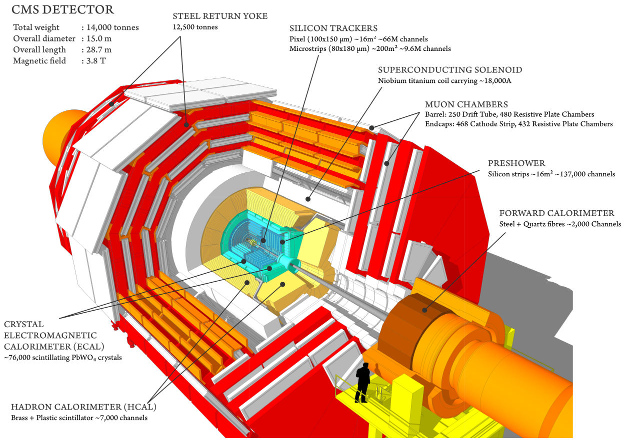


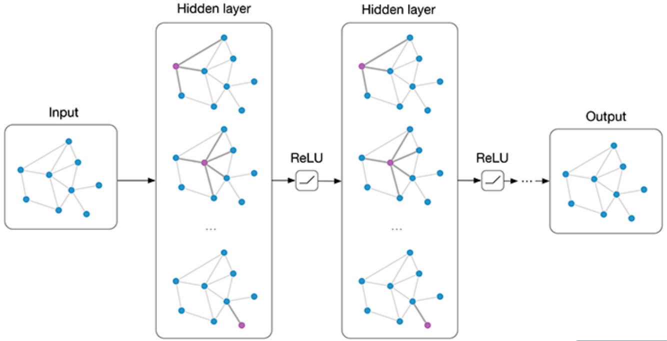


- The GAN gives realistic looking pileup images
- When overlayed onto RPV events, we see realistic shifts in the distributions

Generalizing geometry

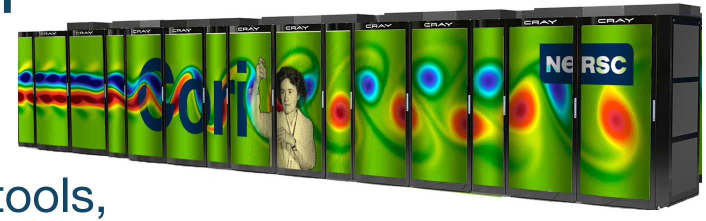
- **Not all HEP detector geometries map to an image**
  - We cannot always use standard convolutional architectures
- **How do you generalize to arbitrary geometries?**
  - *Geometric Deep Learning* methods
    - E.g., *Graph Neural Networks*
- **Already shown effective for some tasks in HEP**
  - Classification of jets
  - Pattern recognition for particle tracking
  - But not really explored yet for generative tasks
- **New sets of challenges, but new possibilities**
  - Work in progress





Increasing productivity with HPCs

- **At NERSC we have a lot of computing power**
  - The Cori supercomputer with 9668 KNL nodes and 2388 Haswell nodes



  - Cutting edge Deep Learning frameworks, tools, and methods, optimized for scale with industry collaborations
  - Next-generation supercomputer NERSC-9 (2020) will have accelerators
- **We're working on improving the Deep Learning experience on HPCs**
  - Scaling across nodes for training and hyper-parameter optimization
  - Jupyter-notebook-based distributed workflow solutions

Distributed training on HPCs

Distributed training is hard

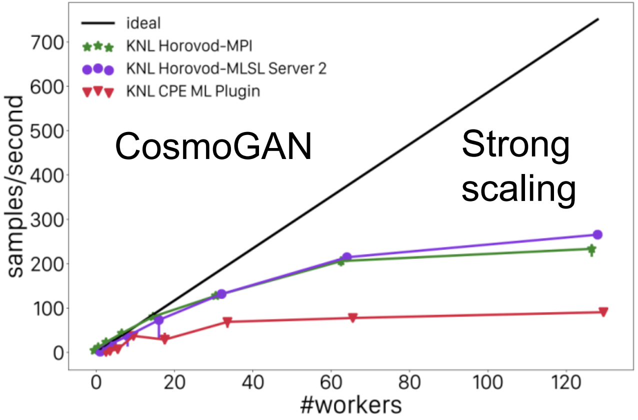
- Fixed batch size (strong-scaling) hits bottlenecks
- Growing batch size (weak-scaling) scales to *thousands of nodes*, but has convergence issues

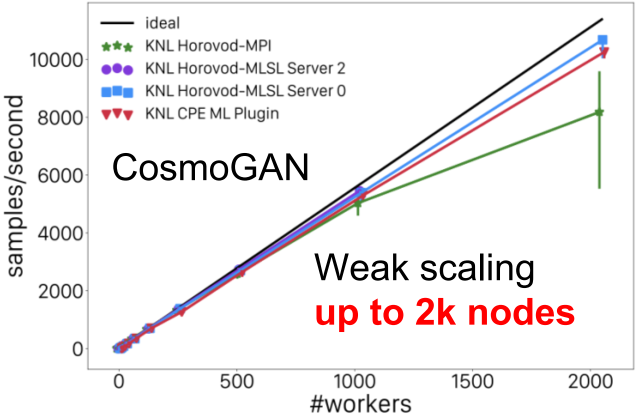
This is even harder with GANs

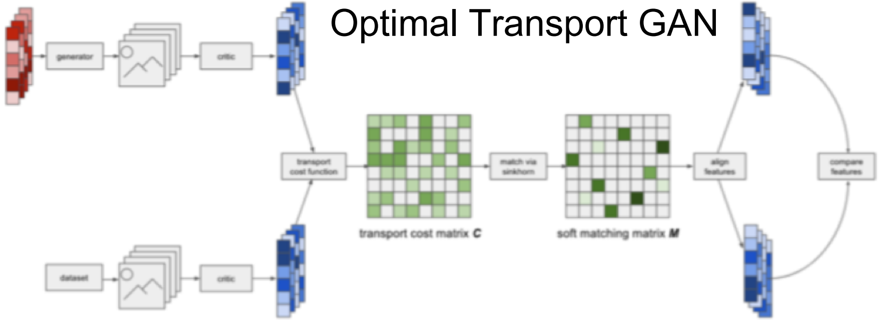
- They're already unstable!

Various methods promise to improve this

- E.g. Optimal Transport GAN
- But no magic bullet (yet)







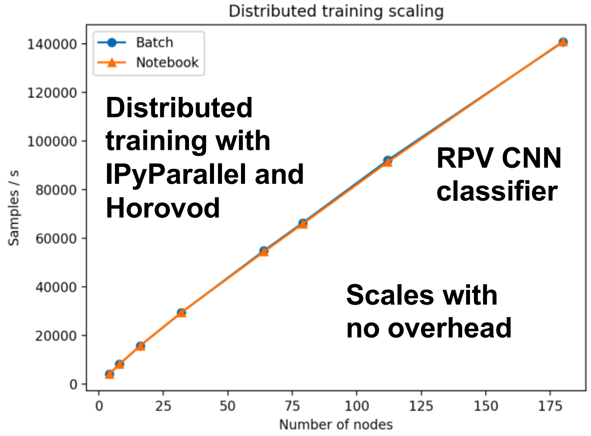
Presented at CUG 2018, PASC 2018
Thorsten Kurth

Distributed DL with Jupyter notebooks

Realizing the full power of supercomputers for Deep Learning with Jupyter notebooks

- Distributed training with IPyParallel+horovod
- Distributed HPO with IPyParallel

Example GAN HPO notebook



Presented at Interactive-HPC at ISC
S. Farrell, W. Bhimji, A. Vose, S. Cholia, O. Evans, M. Henderson, R. Thomas, S. Cannon, Prabhat

IHPC slides

```
# Load-balanced view
lv = c.load_balanced_view()

# Loop over hyper-parameter sets
results = []
for ihp in range(n_hpo_trials):
    checkpoint_file = os.path.join(checkpoint_dir, 'model_%i.h5' % ihp)
    result = lv.apply(build_and_train,
                      input_dir, n_train, n_valid,
                      conv_sizes=conv_sizes[ihp], fc_sizes=fc_sizes[ihp],
                      dropout=dropout[ihp], optimizer=optimizer[ihp], lr=lr[ihp],
                      batch_size=batch_size, n_epochs=n_epochs,
                      checkpoint_file=checkpoint_file)
    results.append(result)
```

Load-balanced task scheduler

Launch hyper-parameter training tasks

```
Hyperparameter trial 0 conv [ 64  16 128] fc [128] dropout 0.3234 opt Nadam, lr 0.0100
Hyperparameter trial 1 conv [ 4  8 64] fc [64] dropout 0.6747 opt Adadelta, lr 0.0010
```

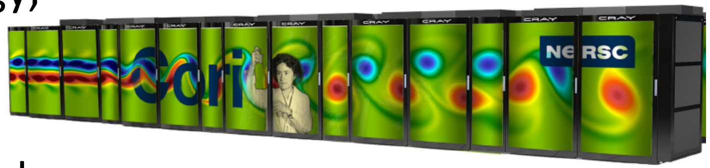
Conclusions

- **Deep generative models are showing a lot of promise for HEP simulation**
- **We've demonstrated that GANs can even learn full event physics**
  - RPV event images; conditioned on theory mass
  - Pileup event images
- **We're also been making progress on the computing challenges**
  - Extreme scale distributed training
  - HPC usability with Jupyter notebooks
- **A number of open challenges remain to bring high-fidelity generative models into practice**
  - Addressing stability, particularly at large scale
  - Generalizing geometry

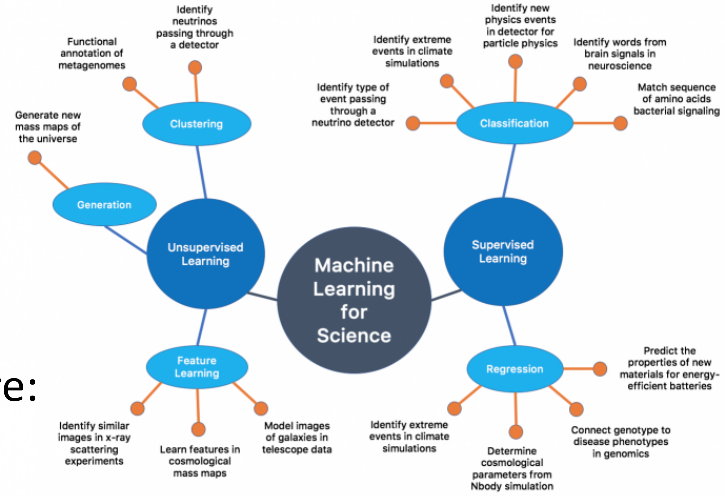
Thank You

NERSC

- **Mission HPC center for US Dept. of Energy**
  - 7000+ diverse users across science (e.g. cosmology, climate, biosciences, materials, particle physics)



- **Cori – Cray XC40 (31.4 PF Peak)**
  - 9668 Intel Knights Landing (KNL), 2388 Haswell nodes
- **Deep learning: Data and analytics (DAS) group:**
  - [Tools for machine learning](); optimized for scale
  - Cutting-edge methods/Collaborations/Training
- **Interactive Computing at NERSC:**
  - Modifications to SLURM including real-time and [interactive queues]() with dedicated resource
  - Also other interactive features not described here: (visualization; science gateways etc.)

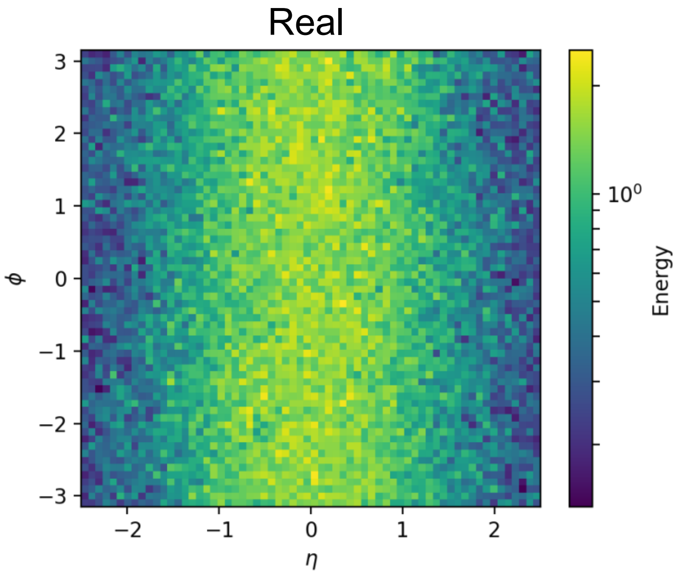


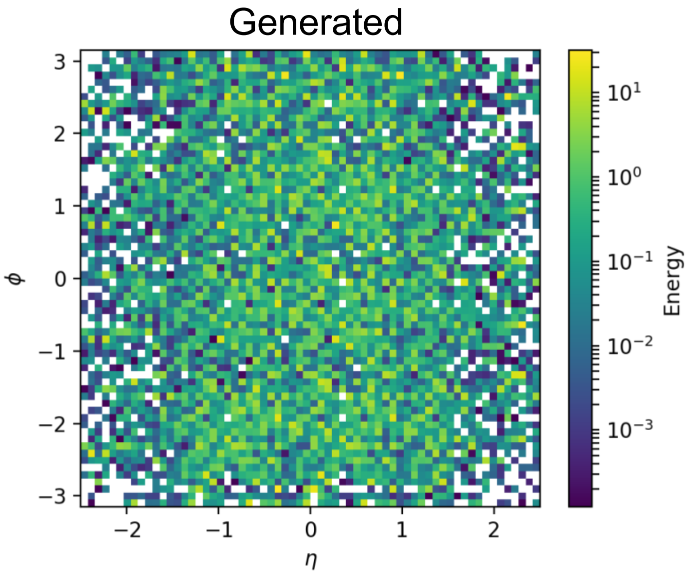
RPV-GAN hyper-parameter optimization

ks_nJet	ks_sumMass	ks_jetPt	ks_jetEta	ks_jetPhi	epoch	hp	flip_rate	noise_dim	image_norm	n_filters	lr	beta2	beta1	threshold	ks_comb
4.256377	2.841550	9.569770	25.151342	10.514511	25	8	0.135265	16	4000000.0	128	0.00010	0.999	0.5	0.000125	16.667698
5.357280	0.964445	10.920871	9.289027	28.375811	42	8	0.135265	16	4000000.0	128	0.00010	0.999	0.5	0.000125	17.242596
4.326334	10.338959	3.120948	21.986991	61.724127	42	21	0.038879	128	4000000.0	32	0.00010	0.999	0.5	0.000125	17.786242
6.152581	4.396782	7.689793	34.594309	60.834282	31	31	0.013889	64	4000000.0	32	0.00010	0.999	0.5	0.000125	18.239157
4.903664	13.920761	0.755197	26.283550	53.695622	53	21	0.038879	128	4000000.0	32	0.00010	0.999	0.5	0.000125	19.579622
7.798904	2.170124	11.107861	43.129680	6.218526	28	25	0.181805	128	4000000.0	128	0.00020	0.999	0.5	0.000125	21.076889
9.125388	9.723337	2.488834	39.824053	106.433264	38	10	0.115164	16	4000000.0	128	0.00001	0.999	0.5	0.000125	21.337559
5.137090	4.949499	12.484213	44.514930	49.054545	38	31	0.013889	64	4000000.0	32	0.00010	0.999	0.5	0.000125	22.570803
13.563612	4.396782	9.750795	29.163675	35.590484	55	21	0.038879	128	4000000.0	32	0.00010	0.999	0.5	0.000125	27.711188
5.357280	6.484448	17.111287	18.792253	15.805424	28	8	0.135265	16	4000000.0	128	0.00010	0.999	0.5	0.000125	28.953015

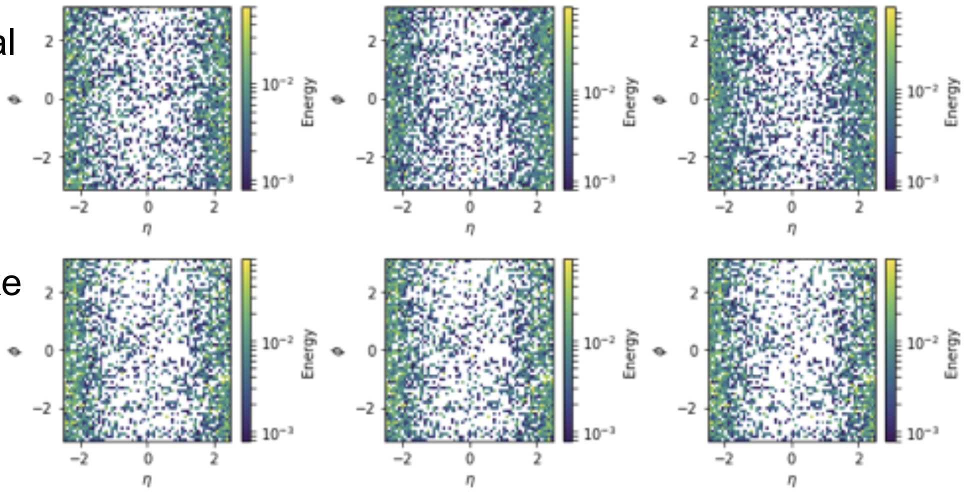
Ks_comb = ks_nJet + ks_sumMass + ks_jetPt
Where each KS metric is the negative log of the KS test p-value

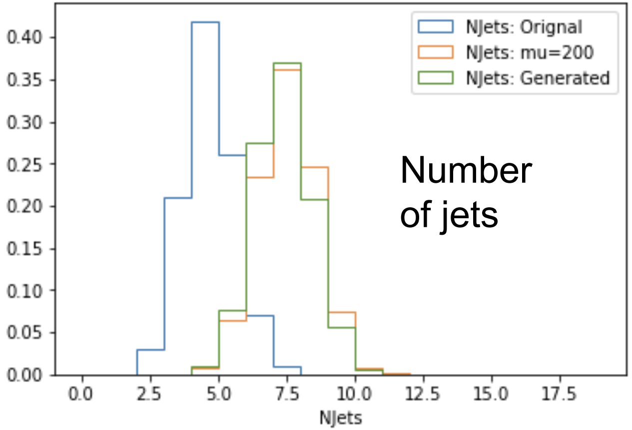
RPV-GAN average images

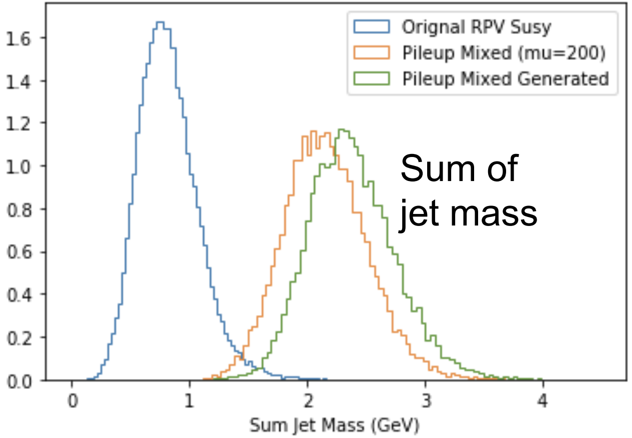




Pileup GAN - μ=200

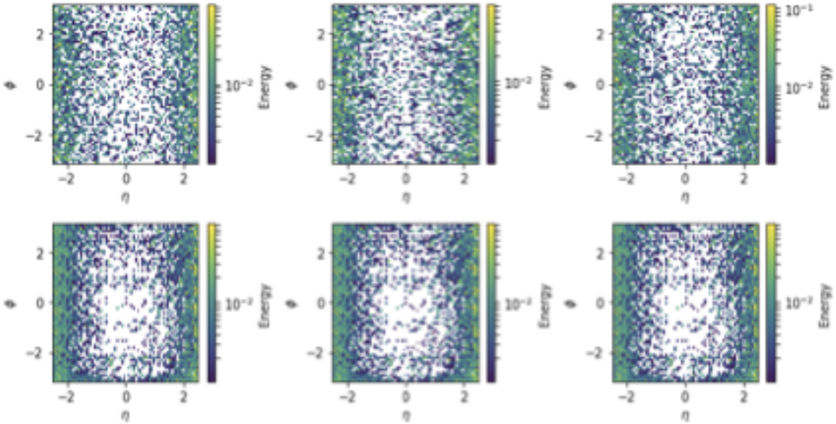




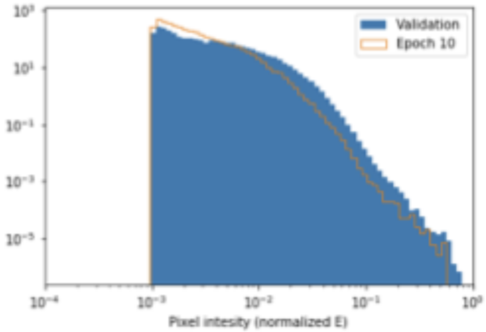


- **With μ=200 pileup, the mass shift isn't perfect**
  - But it's in the ballpark
- **Room for improvement**

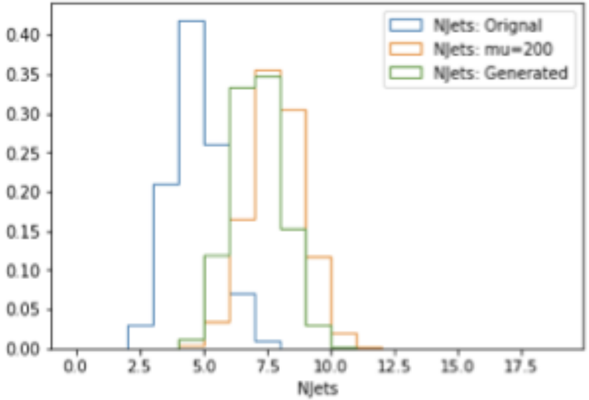
Wgan (PU 200)



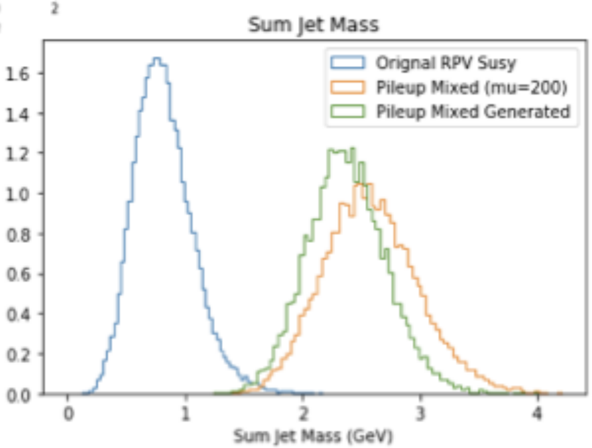
Energy



Validation
Epoch 10
Pixel intesity (normalized E)

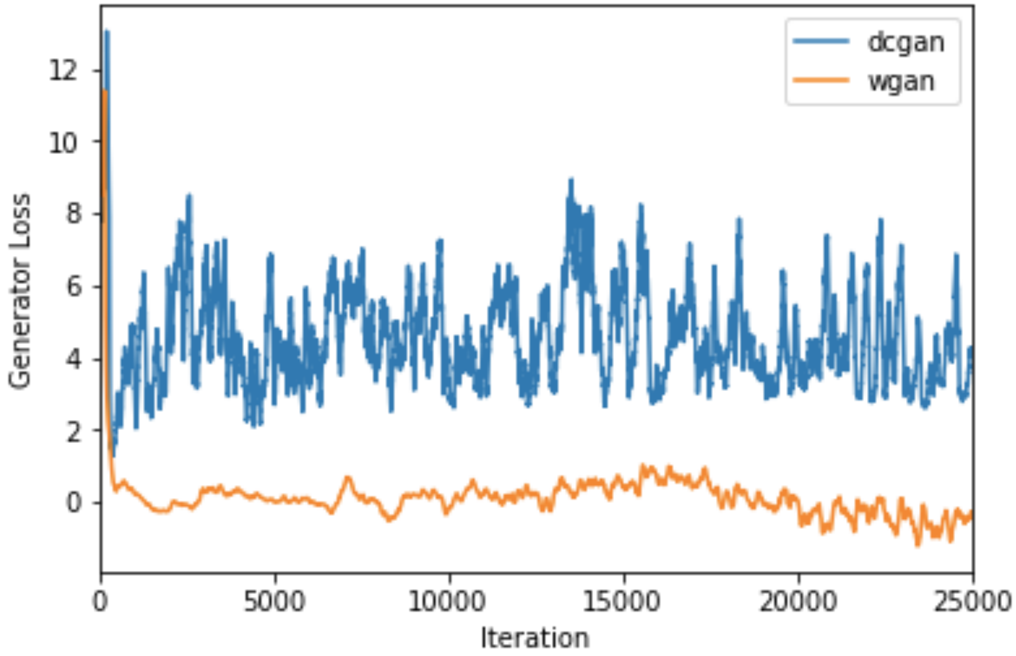


NJets: Orignal
NJets: mu=200
NJets: Generated
NJets



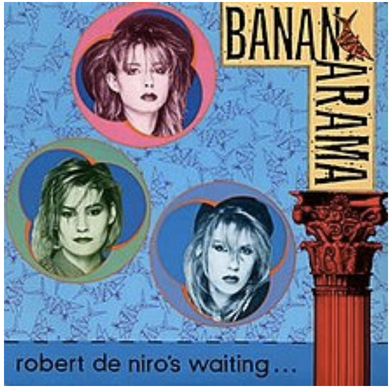
Sum Jet Mass
Orignal RPV Susy
Pileup Mixed (mu=200)
Pileup Mixed Generated
Sum Jet Mass (GeV)

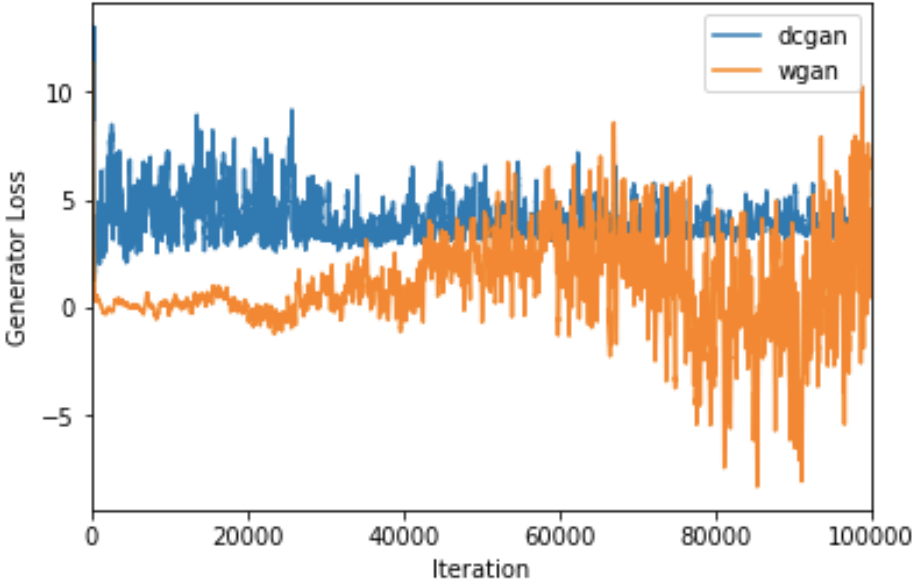
WGAN Generator Loss first 10 epochs (rolling average over 100 iter to smooth)



Looks like wgan is more stable in this respect but….

Then goes bananas





Jupyter architecture

- Allocate nodes on Cori interactive queue and start ipyparallel or Dask cluster
  - Developed %ipcluster magic to setup within notebook
- Compute nodes traditionally do not have external address
  - Required network configuration / policy decisions
- Distributed training communication is via MPI Horovod or Cray ML