Next Generation Generative Neural Networks for HEP

Steven Farrell[1,*], *Wahid* Bhimji[1], *Thorsten* Kurth[1], *Mustafa* Mustafa[1], *Deborah* Bard[1], *Zarija* Lukic[1], *Benjamin* Nachman[1] and *Harley* Patton[2]

[1]Lawrence Berkeley National Laboratory
[2]University of California, Berkeley

Abstract. Initial studies have suggested generative adversarial networks (GANs) have promise as fast simulations within HEP. These studies, while promising, have been insufficiently precise and also, like GANs in general, suffer from stability issues. We apply GANs to to generate full particle physics events (not individual physics objects), explore conditioning of generated events based on physics theory parameters and evaluate the precision and generalization of the produced datasets. We apply this to SUSY mass parameter interpolation and pileup generation. We also discuss recent developments in convergence and representations that match the structure of the detector better than images. In addition we describe on-going work making use of large-scale distributed resources on the Cori supercomputer at NERSC, and developments to control distributed training via interactive jupyter notebook sessions. This will allow tackling high-resolution detector data; model selection and hyper-parameter tuning in a productive yet scalable deep learning environment.

1 Introduction

Simulation of physics processes, particle propagation, and detector response are essential components of physics analyses in High Energy Physics (HEP) experiments such as those at the Large Hadron Collider (LHC) [1]. High fidelity simulation software toolkits such as Geant4 [2] are heavily used by experiments to model their building-sized detectors with extremely fine detail. However, producing accurate simulations of such complex physics and detectors requires considerable computing resources. As a result, considerable person-power effort is spent in the HEP experiments to develop fast simulation solutions that can supplement or replace the full-fidelity simulation.

One promising direction in the area of fast simulation development lies in the growing field of Deep Learning [3]. Deep generative models such as Generative Adversarial Networks (GANs) [4] have shown significant promise in recent studies in Cosmology [5] and Particle Physics [6–8] to supplement or replace existing simulation codes. These models can be trained to learn a data distribution to quickly sample from. In the GAN framework, the learning problem is posed as a two-player game between a generator network (tasked with producing realistic looking samples) and a discriminator network (tasked with distinguishing real from generated samples).

In this paper we extend the work of using GANs to emulate physics data to full HEP detector images of collision events. We demonstrate, in section 4, the capability of GANs

[*]e-mail: SFarrell@lbl.gov

to learn to produce physics-realistic samples from a new-physics theory. We then show, in section 5, how the model can be extended to produce samples conditional on the new physics theory parameters. In section 6 we turn to a different application and explore the capability of GANs to learn to produce the underlying collision event background known as "pileup". Finally in section 7 we describe ongoing computing work at NERSC to run these models at scale on the Cori supercomputer and to provide a productive interface via Jupyter as well as discussing possible future directions for GAN modelling.

2 Related work

This work is built on a number of related contributions in HEP and Cosmology. First, in [9], the full-detector image representation for HEP collisions was developed and used with Convolutional Neural Networks (CNNs) trained to distinguish new physics samples from background. This work showed the potential in low-level image representations of entire collision events for physics analysis. CosmoGAN [5] is an application of Deep Convolutional GANs (DCGANs) to the problem of simulating weak-lensing convergence maps for Cosmology. In that work, GANs were shown to be able to produce images of very high fidelity with very little physics knowledge imposed on the model. In HEP, there have been studies [6–8] that demonstrate the ability of customized GAN models to produce 2D or 3D particle shower signatures in calorimeter detectors. These models worked on the level of individual particles and could be conditioned on the particle kinematics.

3 Datasets

For this study, we take as a use-case, new massive supersymmetric ('RPV-SUSY') particles in multi-jet final states at the LHC. We focus on 'gluino-cascade decays' with varied gluino masses. We use the Pythia event generator [10] interfaced to the Delphes fast detector simulation [11] and using the default Delphes ATLAS detector configuration. We generate events for the RPV-SUSY signal and also a minimum-bias soft-QCD sample for pileup generation studies.

To produce the jet variables we use a standard jet algorithm used in the physics analyses of these signals (with Radius R=1, and transverse momentum $p_T > 200$ GeV) and applied to the final calorimeter images (described below) using 'FastJet' [12] via pyjet [13].

Data from the surface of the cylindrical detector is represented as a 2D image with coordinates corresponding to azimuthal angle ϕ and pseudorapidity η. For the pixel intensity in this image, we use the overall energy deposited in the combined calorimeter. We choose to bin the energy into uniform 64x64 bins,which correspond to the approximately 0.1×0.1 ($\eta \times \phi$) resolution of the ATLAS hadronic calorimeter.

4 RPV whole-detector GAN

The architecture employed for this study is based on the original DCGAN topology [14]. The generator network consists of five transposed convolutional layers with batch-normalization, rectified linear unit (ReLU) hidden activations, and a final thresholded output sigmoid activation which ensures sparsity in the generated samples. The discriminator network consists of five convolutional layers with batch normalization, leaky ReLU hidden activations, and a final sigmoid activation on the output. As in the original DCGAN approach, the generator takes as input a vector of random noise (with values sampled from the standard normal distribution),

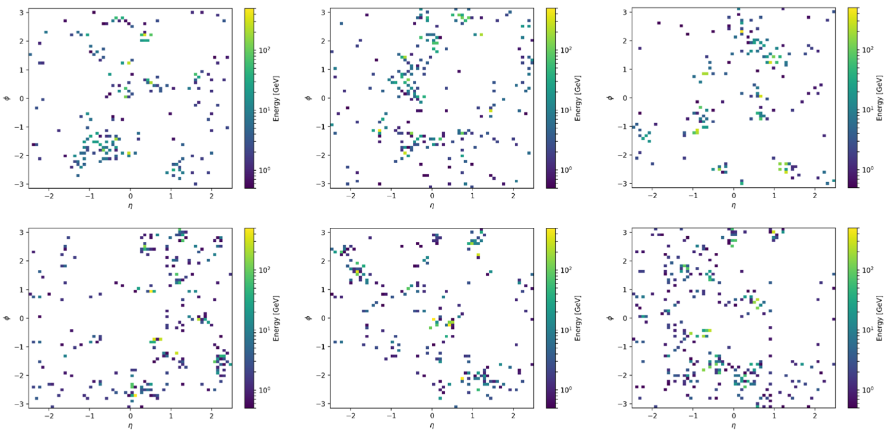

Figure 1. Comparison of original (top) with GAN generated images (bottom)

produces fake detector images, and is trained to try and fool the discriminator. The discriminator takes images as input and is trained to classify them as real or fake. We use random label flipping to regularize and stabilize the training. The model is trained using the Adam optimization algorithm [15] with learning rate, random vector size, number of convolutional filters, and label flip rate treated as hyper-parameters.

To identify the best values of the hyper-parameters we use a random search over the parameter space. To evaluate and select the best configuration, a physics quality metric is computed for a validation set of generated and real samples at every epoch of training. First, jets are reconstructed from both sets of samples. Then, we compare the jet kinematic distributions with Kolmogorov-Smirnoff (KS) tests [16]. The final evaluation metric is the sum of the negative log of the KS test p-values for three jet quantities: the jet multiplicity, the transverse momentum, and the summed jet mass. The model and epoch which minimize this quantity are selected as the best.

Example real and generated samples are shown in figure 1. The generated samples can be seen to exhibit qualitatively similar structure and sparsity to the real ones. A comparison of the reconstructed jet variables is shown in figure 2. The generated samples produce realistic jet multiplicities and kinematics without having those distributions explicitly used in training.

5 Conditional RPV GAN

We extend the GAN architecture used in the last section to incorporate the gluino and neutralino masses in training. This allows the GAN to learn the conditional data distributions. The generator is augmented to take the two mass parameters as additional input, and for the discriminator the mass values are included in the input image by filling them in new image channels.

For this study, we generate SUSY-RPV samples for a range of gluino and neutralino masses and train on a mixture of them. Once the model is trained, we can sample from the generator for specific mass values and see that the reconstructed physics distributions change accordingly. In figure 3 we show that the conditional dependence is also learned and that the shift in sum of jet masses is reproduced in GAN generated data. Such an approach could be used to supplement full simulation samples in physics analysis. One can use the high fidelity

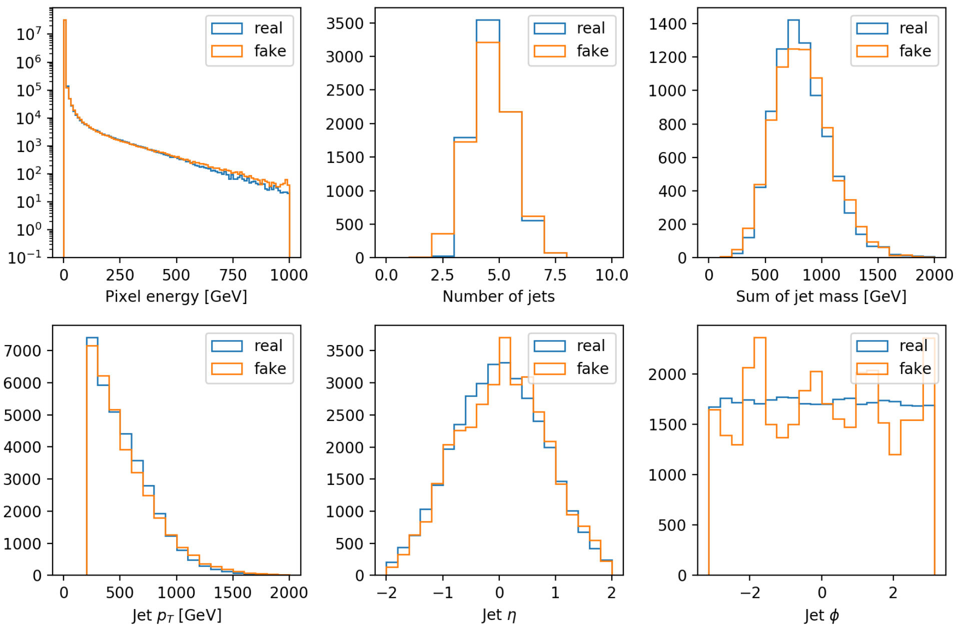

Figure 2. Reconstructed jet kinematics from the Delphes simulated samples ("real") and the GAN-generated samples ("fake").

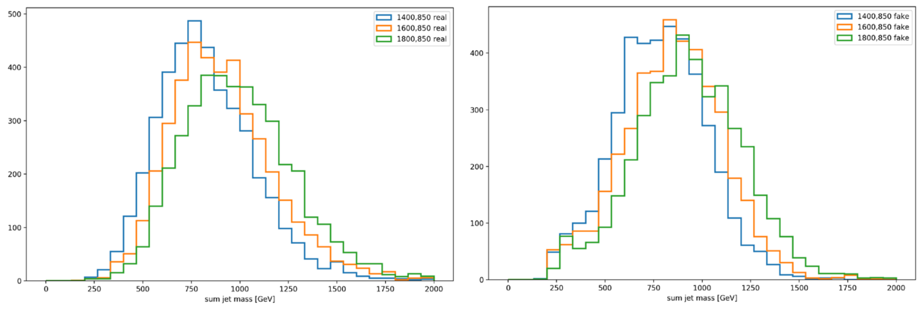

Figure 3. Summed jet mass dependence on RPV theory masses in real samples (left) and GAN samples (right).

full simulation on a coarse grid of theory parameters and use the GAN to interpolate in theory parameter search and quickly generate samples at new points.

6 Pileup GAN

Pileup poses big challenges for HL-LHC computing workflows. Currently a very large volume of min-bias events are simulated and stored requiring both CPU resources to generate and disk space, I/O and memory resources when overlayed on other samples during digitization. Instead this distribution could be modeled with a whole-detector GAN, where a single

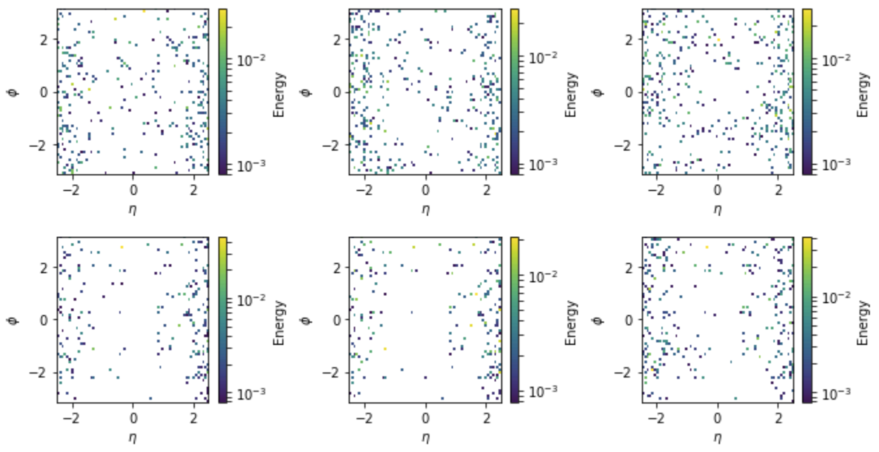

Figure 4. Comparison of min-bias (mu=20) images (top) with GAN generated examples (bottom).

generated sample was used to train a model which is then employed for fast, on-the-fly pileup sampling.

To test the feasibility of this approach we train a GAN on min-bias events. The training,validation and test samples are generated using the Pythia event generator for soft-QCD events, passed through the same Delphes detector simulation and image creation described above in section 3. N image histograms are then summed to produce images representing a pileup of $\mu = N$ on which a GAN is trained with the same DCGAN architecture as described in section 4 though no hyper-parameter optimization is performed. A WGAN implementation [17, 18] was also used which achieved similar results to those shown here. We then evaluate the effects on reconstructed object kinematics by overlaying the original and generated pileup on the RPV-SUSY images used above, reconstructing jets on these on these overlaid images and comparing the resultant shifts in the variables used by the RPV-SUSY analyses.

Figure 4 shows the a comparison of the images generated by the GAN and those in a validation set for $\mu = 20$ and figure 5 shows distributions for two key jet variables. It can be seen that the shift in jet variables due to pileup is modelled by the GAN generated events. However we have observed that at higher pileup the distributions are not modelled with the required level precision so further work will be required to tune models for this purpose.

7 Discussion

The results of the previous sections demonstrate some new capabilities of GANs to supplement HEP simulation, but challenges remain in developing sufficiently sophisticated solutions for use by experiments. In this section we describe some of the remaining challenges and potential solutions.

GANs infamously suffer from instabilities and other difficulties in training. Many augmentations have been proposed to stabilize GAN training [17, 19, 20] and prevent issues like mode-collapse or otherwise poor performance. Some of these solutions, such as Wasseserstein-GAN, have been explored in this work and in the related HEP studies from section 2. Other studies (e.g. CaloGAN) have leveraged additional physics knowledge or constraints to improve GAN results. While it’s clear that such approaches can help, there is still more to be done to evaluate and compare different techniques.

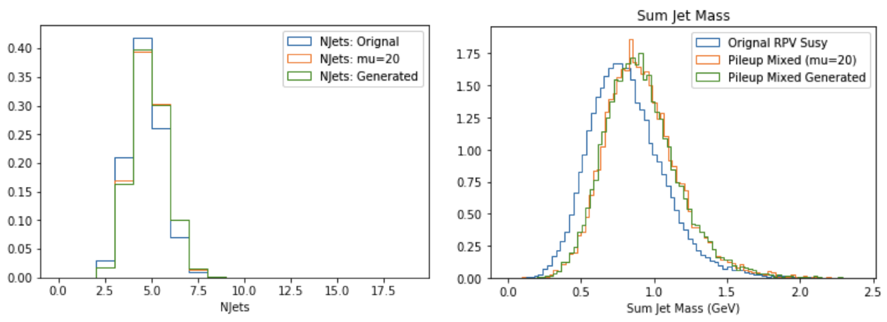

Figure 5. Shift in the number of fat jets (left) and sum of jet mass (right) for RPV-Susy samples when overlaid with events generated from a GAN trained on min-bias events (mu=20) compared with that from when a validation set of original events are overlaid.

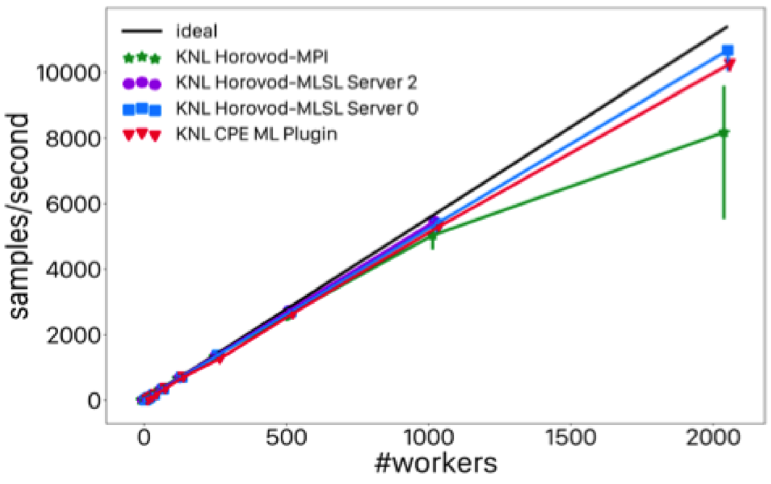

Figure 6. Weak scaling of Cosmology DCGAN network using Horovod [22] and CrayPE [23] MPI libraries with Tensorflow at NERSC

Most modern Deep Learning applications require large compute resources because of the large datasets and complex models needed to solve tasks. HPC facilities are particularly well suited to address this demand and work has already been done at NERSC to study GANs on large-scale HPC systems [21]. In figure 7 we show that we are able to scale GAN architectures up to 1000s of compute nodes with reasonable efficiency using modern MPI libraries. However, training GANs at scale generally exacerbates the instability issues and how to resolve this is still very much an open question.

To effectively utilize HPC resources for GAN applications, it is helpful to have infrastructure which enables high productivity and interactive, iterative development. At NERSC we are developing Jupyter notebook solutions for deploying distributed Deep Learning applications on HPC [24] which we anticipate will be useful for large-scale GAN training as well.

Finally, HEP experiments use complex detector geometry. The layout of detector sensors gives data which cannot always be mapped into 2D or 3D image formats without lossy transformations. There is, however, a growing area of research into deep learning methods for graph- and manifold-structured data known as Geometric Deep Learning [25]. Deep generative models for graph-structured HEP data may be effective but haven't yet been studied to our knowledge.

8 Conclusion

In this paper we demonstrate the applicability of using Generative Adversarial Networks to generate low-level LHC collision events. We extend previous work to consider whole detector events; define a systematic KS-based procedure for more stable performance and hyper-parameter optimization; and add the capability to condition on physics parameters of interest. We apply this to key problems that face major computational challenges in future LHC running, theory parameter interpolation and pileup generation. We demonstrate that our DCGAN architectures are able to learn min-bias and RPV SUSY events and reproduce the reconstructed jet features used in these analyses, and the effect of pileup, without having been explicitly trained on those features.

We also discuss how the computational challenges related to training these large and unstable models are beginning to be addressed on HPC facilities at NERSC, scaling to thousands of nodes and harnessing the power of those resources through productive Jupyter interfaces. This work presents an important next step in pushing the computational and methodological frontier of generative networks for HEP.

References

[1] L. Evans, P. Bryant, JINST **3**, S08001 (2008)

[2] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand et al., Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **506**, 250 (2003)

[3] Y. LeCun, Y. Bengio, G. Hinton, Nature **521**, 436 (2015)

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, in *Advances in Neural Information Processing Systems 27* (2014), pp. 2672–2680, `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`

[5] M. Mustafa, D. Bard, W. Bhimji, R. Al-Rfou, Z. Lukić, arXiv:1706.02390 (2017)

[6] L. de Oliveira, M. Paganini, B. Nachman, Computing and Software for Big Science **1**, 4 (2017)

[7] M. Paganini, L. de Oliveira, B. Nachman, Physical Review D **97**, 014021 (2018)

[8] G. Rukhkhattak, S. Vallecorsa, F. Carminati, *Three Dimensional Energy Parametrized Generative Adversarial Networks for Electromagnetic Shower Simulation*, in *2018 25th IEEE International Conference on Image Processing (ICIP)* (IEEE, 2018), pp. 3913–3917

[9] W. Bhimji, S.A. Farrell, T. Kurth, M. Paganini, E. Racah, Prabhat, arXiv preprint arXiv:1711.03573 (2017)

[10] T. Sjöstrand, S. Mrenna, P. Skands, Computer Physics Communications **178**, 852 (2008)

[11] J. de Favereau, C. Delaere et al., JHEP **2014**, 57 (2014)

[12] M. Cacciari, G.P. Salam, G. Soyez, The European Physical Journal C **72**, 1896 (2012)

[13] N. Dawe, E. Rodrigues, *pyjet: the interface between fastjet and numpy* (2010), `https://github.com/scikit-hep/pyjet`

[14] A. Radford, L. Metz, S. Chintala, CoRR **abs/1511.06434** (2015), `1511.06434`

[15] D.P. Kingma, J. Ba, CoRR **abs/1412.6980** (2014)

[16] R. Simard, P. L'Ecuyer, Journal of Statistical Software, Articles **39**, 1 (2011)

[17] M. Arjovsky, S. Chintala, L. Bottou, arXiv preprint arXiv:1701.07875 (2017)

[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A.C. Courville, *Improved training of wasserstein gans*, in *Advances in Neural Information Processing Systems* (2017), pp. 5767–5777

[19] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, CoRR **abs/1802.05957** (2018), `1802.05957`

[20] T. Salimans, H. Zhang, A. Radford, D.N. Metaxas, CoRR **abs/1803.05573** (2018), `1803.05573`

[21] T. Kurth, M. Smorkalov, P. Mendygral, S. Sridharan, A. Mathuriya, Concurrency and Computation: Practice and Experience p. e4989 (2018), `https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4989`

[22] A. Sergeev, M.D. Balso, arXiv preprint arXiv:1802.05799 (2018)

[23] P. Mendygral, N. Hill, K. Kandalla, D. Moise, J. Balma, M. Schongens, *High performance scalable deep learning with the cray programming environments deep learning plugin, cray users group* (2018), `https://www.cray.com/sites/default/files/resources/CUG-DeepLearning.pdf`

[24] S. Farrell, W. Bhimji, A. Vose, O. Evans, M. Henderson, S. Cholia, R. Thomas, S. Canon, Prabhat, *Interactive distributed deep learning with jupyter notebooks*, `https://drive.google.com/open?id=1lAK16LthN0NpCp97sX-tY7--WyUYojOZ`

[25] M.M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, CoRR **abs/1611.08097** (2016), `1611.08097`