Sparse autoregressive models for scalable generation of sparse images in particle physics

Yadong Lu,[1] Julian Collado,[2] Daniel Whiteson,[3] and Pierre Baldi[2,*]

[1]*Department of Statistics, University of California, Irvine, California 92627, USA*
[2]*Department of Computer Science, University of California, Irvine, California 92627, USA*
[3]*Department of Physics and Astronomy, University of California, Irvine, California 92627, USA*

(Received 12 October 2020; accepted 11 January 2021; published 16 February 2021)

Generation of simulated data is essential for data analysis in particle physics, but current Monte Carlo methods are very computationally expensive. Deep-learning-based generative models have successfully generated simulated data at lower cost, but struggle when the data are very sparse. We introduce a novel deep sparse autoregressive model (SARM) that explicitly learns the sparseness of the data with a tractable likelihood, making it more stable and interpretable when compared to generative adversarial networks (GANs) and other methods. In two case studies, we compare SARM to a GAN model and a nonsparse autoregressive model. As a quantitative measure of performance, we compute the Wasserstein distance (W_p) between the distributions of physical quantities calculated on the generated images and on the training images. In the first study, featuring images of jets in which 90% of the pixels are zero valued, SARM produces images with W_p scores that are 24%–52% better than the scores obtained with other state-of-the-art generative models. In the second study, on calorimeter images in the vicinity of muons where 98% of the pixels are zero valued, SARM produces images with W_p scores that are 66%–68% better. Similar observations made with other metrics confirm the usefulness of SARM for sparse data in particle physics.

DOI: 10.1103/PhysRevD.103.036012

I. INTRODUCTION

Experiments in particle physics seek to uncover the building blocks of matter and their interactions, which determine the structure of the Universe from subatomic to cosmic distances. Analyses of the data produced by these experiments make extensive use of simulations to predict the experimental signature of particle interactions under various theoretical hypothesis. These simulations are used in likelihood-free inference as well as in the development of data selection and analysis strategies which optimize the statistical power of the data. Current state-of-the-art simulators apply Monte Carlo techniques to the microphysical processes governing individual particles' propagation and interaction [1], making them computationally expensive [2,3].

Detectors in particle physics experiments have a multilayer architecture which produces highly structured data. One essential layer, the calorimeter, measures the energy of passing particles, and is subdivided into small cells to ensure spatial resolution. In collider experiments, the calorimeter is typically cylindrical [4], while in fixed-target experiments it may be a surface [5]. In both cases, the data can be represented as an image, allowing for the application of image-processing methods initially developed for natural images. However, in contrast to natural images, pixels in calorimeter images (Fig. 1) are very sparse, where usually 90% or more of the pixel values are zero. In addition, these images are not as uniform as natural images, featuring clusters in the center and noise in the periphery.

Recently, deep generative models [8–10] have produced high-quality artificial natural images [11–13] at a relatively low computational cost. The successful application of machine learning in high-energy physics [14–21] and generative models in natural images have inspired the use of these models for generating imagelike data in physical sciences applications [6,22–32], often employing generative adversarial networks (GANs) [8] or, less frequently, variational autoencoders (VAEs) [9]. However, the extreme sparsity of the images in particle physics and other areas of the physical sciences [33] presents unique challenges for generative models.

The leading applications of GAN-based generative models for sparse image synthesis in high-energy physics, LAGAN [6] and CaloGAN [34], make use of the ReLU activation function in the final layer to induce sparsity in the output image. The flat portion of the ReLU activation function can lead to many error gradients being zero at the output layer, creating challenges [35] for stochastic gradient descent [36,37] methods. In addition, GANs are notoriously unstable during training [38] and can suffer

*Corresponding author.
pfbaldi@ics.uci.edu

from mode collapse, which restricts the diversity of events in the generated data [39,40]. Despite these difficulties, GANs have been one of the most popular deep generative models in particle physics.

However, other generative models may be better suited for sparse data. For example, deep autoregressive models (ARMs) have also demonstrated impressive performance for generating natural images among likelihood-based generative models [10,41]. In this paper, we develop *sparse* autoregressive models (SARMs), a class of ARMs specifically tuned to produce sparse images. We present a systematic approach for designing SARMs and demonstrate their effectiveness through multiple experiments. SARMs are stable during training with respect to hyperparameter variations and weight initializations. SARMs are also interpretable in the sense that it is possible for these models to produce an analytic likelihood for any given sample. We then evaluate SARMs on two benchmark datasets. Given their flexibility, SARMs may be applicable to areas beyond particle physics where sparse images must be generated.

II. DATASETS

An important statistical task in the analysis of particle physics data is identifying the particle source of a particular detector signature. Below, we describe two datasets, one which distinguishes between the detector signatures of single quarks and collimated pairs of quarks, and a second which distinguishes between muons produced in isolation and those produced as part of a shower of particles.

A. Jet substructure study

Quarks or gluons produced in collisions leave a particular detector signature: a *jet*, or shower of collimated particles, which deposit most of their energy in a tight core. In many applications, it is important to distinguish the signatures of a single quark or gluon from that of a collimated pair of quarks, which may leave two potentially overlapping cores. This task is a natural setting for image-recognition algorithms, and has been the focus of many deep learning studies [33,42–45] which rely on simplified calorimeter simulations due to the cost of generating realistic samples. Thus, an inexpensive generation of realistic datasets would be very valuable as a classification training sample.

We use a set of benchmark jet images from Ref. [6], where a full description of this dataset can be found as well as the code to generate it. In this dataset, quark pairs from W-boson decay are labeled as signal and single quark or gluon jets are labeled as background images. The intensity of each pixel value represents the sum of the momenta transverse to the beam (P_{T}) over the particles which strike a particular cell. The images are generated using PYTHIA8.219 [46] simulations of proton collisions at a center-of-mass energy $\sqrt{s} = 14$ TeV, selecting jets with $250 < P_{\mathrm{T}} < 300$ GeV. Instead of a realistic detector simulation, the calorimeter response is mimicked via a regular 0.1×0.1 grid in the η and ϕ coordinates. The jet images are constructed and preprocessed as described in [43], including the centering and rotations of the images. The resulting images are 25×25 pixels, with intensity values in the [0,276] range. We divide them into a training set containing 400,000 images for the signal and 400,000 images for the background, and a testing set containing 36,000 images for the signal and 36,000 images for the background. A typical image from this dataset is shown in Fig. 1. This dataset has a high degree of sparseness: more than 90% of its pixels are zero valued.

B. Muon isolation study

Muons leave a very clear detector signature which is difficult to mimic. However, physicists must distinguish between two modes of muon production: a rare mode in which muons are produced from the decay of a heavy boson and are isolated in the detector, and a second prolific mode in which muons are produced inside a jet, surrounded by other particles. Fluctuations in the jet can occasionally produce apparently isolated muons.

We use a set of benchmark calorimeter images from [7], where muons from heavy bosons are labeled as signal and muons produced within jets are labeled as background. The signal muons are generated with the process $pp \to Z' \to \mu^+\mu^-$ with a Z' mass of 20 GeV/c^2. Background muons are generated with the process $pp \to b\bar{b}$. Both signal and background datasets are generated at a center-of-mass energy $\sqrt{s} = 13$ TeV. The collisions and immediate decays are simulated with MADGRAPH5 2.3.3 [47], showering and hadronization with PYTHIA6.428 [46], and detector response with DELPHES3.4.0 [48] using the DELPHES ATLAS detector model. Additional proton interactions are overlaid on top of the primary process, at a rate of 50 additional interactions per event. This dataset only considers muons with P_{T} in the range: $P_{\mathrm{T}} \in [10, 15]$ GeV/c. The signal events are weighted to match the transverse muon momentum distribution of the background events. The calorimeter images in the vicinity of the muon are created from the calorimeter deposits within the $\eta - \phi$ radius of $R < 0.4$, where each pixel represents the momentum transverse to the beam axis. The deposits are preprocessed by centering the image on the coordinates of the identified muon propagated to the calorimeter. The images are pixelated using a 32×32 grid to roughly match the granularity of the calorimeters of ATLAS and CMS, and the pixels have values in the range [0, 172]. The training set contains 41,250 signal images and 41,246 background images, and the testing set contains 41,344 signal images and 41,151 background images. A typical image from this dataset is shown in Fig. 1. This dataset has an even greater level of sparsity: more than 98% of its pixels have zero value.

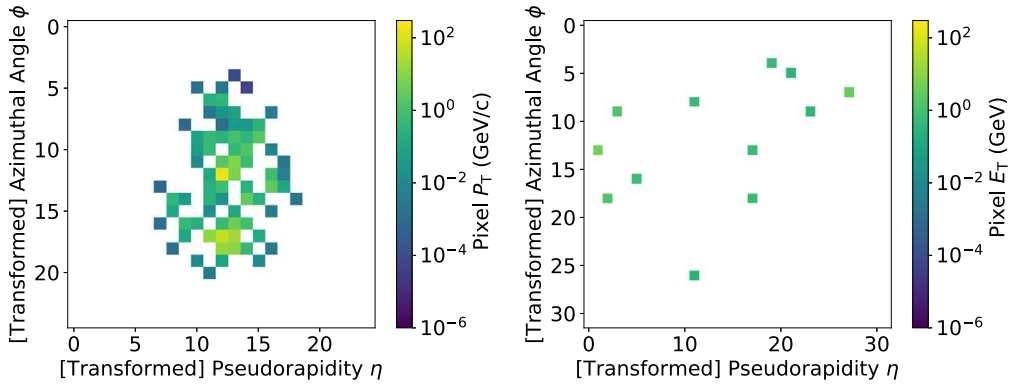

FIG. 1. Calorimeter images in particle physics are often very sparse, where most of their pixels have very small values. Left panel: typical signal image of a hadronic jet from [6]. Right panel: typical signal image of the vicinity of a muon from [7].

III. AUTOREGRESSIVE MODELS

ARMs approximate a high-dimensional data distribution $P_{\text{data}}(\mathbf{x})$ with $P(\mathbf{x})$, the distribution induced by the model where $\mathbf{x} \in \mathbb{R}^D$. For example, when working with images, $P_{\text{data}}(\mathbf{x})$ represents the distribution of the values of D pixels in the image. ARMs are generative models that create outputs sequentially, where each new output is conditioned on the previous output [49]. Formally, ARMs transform the problem of learning the joint distribution $P_{\text{data}}(\mathbf{x})$ into learning a sequence of tractable conditional distributions $P(x_i|x_{j<i})$. The ordering of the pixels can influence the model's performance and will be discussed later in the paper. ARMs rely on the basic factorization

$$\begin{aligned} P(\mathbf{x}) &= P(x_0, x_1, \ldots, x_D) \\ &= P(x_0)P(x_1|x_0)P(x_2|x_0, x_1)\ldots P(x_{D-1}|x_0\ldots x_{D-2}). \end{aligned} \tag{1}$$

The conditional densities $P(x_i|x_{j<i})$ can be parametrized by deep neural networks [10,41,50,51] so that (1) $P(x_i|x_{j<i}) = P(x_i|\theta_i)$, where θ_i represents the parameters of a distribution (e.g., mean and standard deviation); (2) $\theta_i = f_i(x_0, \ldots, x_{i-1})$, such that θ_i depends on previous output; and (3) the function f_i is implemented by a neural network. At generation time, the pixel values x_i are generated sequentially by sampling in order from the distributions $P(x_i|\theta_i)$. A simplified implementation of this process using a single neural network is depicted in Fig. 2. The weights of the neural networks that compute the θ_i's are shared across different values of i, for regularization [51] purposes and to reduce computational costs, hence the zero padding of the input vector.

A common concern with ARMs is that by generating pixels in sequence, conditioning only on previously visited pixels, the model may not be able to take into account the dependence of a current pixel on subsequent pixels. However, this is not the case because the weights are trained using all the data (i.e., "past" and "future" pixels) and the model always learns to generate the joint marginal distribution of previous and current pixels. This idea is further illustrated with a toy example in Appendix A.

Learning in ARMs is different from learning in other generative models such as GANs and VAEs. ARMs directly minimize the discrepancy, in terms of Kullback-Leibler (KL) divergence, between the data distribution $P_{\text{data}}(\mathbf{x})$ and the model distribution $P(\mathbf{x})$ which is produced explicitly. In contrast, neither GANs nor VAEs produce a tractable marginal likelihood model $P(\mathbf{x})$ and, as a result, they have to resort to approximations for minimizing the KL divergence between the data and model distributions. ARMs avoid this issue by sequentially modeling each conditional probability distribution, allowing them to minimize the KL divergence directly with a tractable likelihood $P(\mathbf{x})$. Leveraging the flexibility of deep neural networks to learn each conditional probability, ARMs are able to approximate a large family of continuous distributions in $\mathbb{R}^D$ [52].

The implementation of ARMs for images can follow several approaches [10,41,50,53]. For scalability during training and generation, we use a single neural network to model the parameters of the conditional probabilities at each step, where some connections are intentionally disabled to preserve the autoregressive structure (see Appendix B), similar to the structure used in [50]. Given

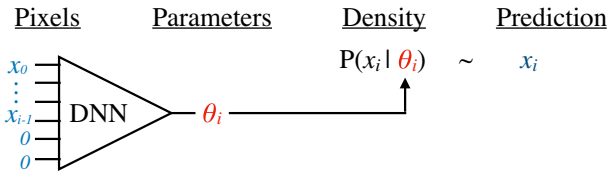

FIG. 2. Pixel generation process by a deep ARM to create an image with D pixels. For the pixel x_i, a deep neural network (DNN) is evaluated on a vector with values $x_0, \ldots, x_{i-1}$, zero padded to length D. The output of the network is the parameters θ_i of a parametric probability density $P(x_i|\theta_i)$, from which x_i is sampled.

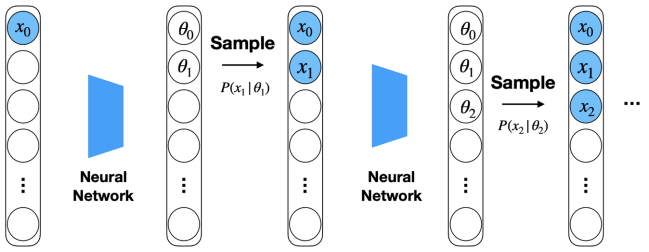

FIG. 3. Generation process of a deep autoregressive model. During generation, the first pixel x_0 is sampled from $x_0 \sim P(x_0|\theta_0)$. Next, the pixel x_0 is zero padded to a D-dimensional vector and passed to the neural network ARM model, which evaluates the parameters $\boldsymbol{\theta} = \{\theta_0, \ldots, \theta_{D-1}\}$, though only θ_1 is needed to sample the next pixel $x_1 \sim P(x_1|\theta_1)$. The pixels x_0 and x_1 are again zero padded to create a D-dimensional vector which is passed into the neural network to generate the next pixel. This process is repeated until all pixels are generated. Note that the same neural network is used at each generation step, and part of its weight connections are disabled to preserve the autoregressive structure.

a training image, this makes it possible to calculate *all* the parameters $\theta_0, \ldots, \theta_{D-1}$ in parallel, instead of calculating each θ_i sequentially. During generation, the model generates the output elements one by one as illustrated in Fig. 3.

IV. SPARSE AUTOREGRESSIVE MODELS

To deal with sparsity in images, we introduce SARMs in which each conditional distribution is a mixture comprising a Dirac delta distribution at the zero-pixel value, as one of its components. The probability associated with the zero-pixel value is learnable by gradient descent, providing a flexible and efficient way of modeling and fitting highly sparse datasets. The other components of the mixture can be modeled in different ways, as described below.

A. Sparse image likelihood models

In SARMs, the likelihood function for the ith pixel x_i is formulated as

$$p(x_i|\theta_i) = \gamma_i \cdot \delta_{x_i=0} + (1-\gamma_i) \cdot \delta_{x_i\neq 0} \cdot p(x_i|\phi_i), \quad (2)$$

where the parameters $\theta_i = \{\gamma_i, \phi_i\}$ are predicted by the underlying neural network taking $x_0, \ldots, x_{i-1}$ as its inputs. Since the pixel values in the calorimeter images represent the physical deposition of energy, they must be nonnegative, i.e., $p(x_i|\phi_i) > 0$ only when $x_i > 0$. To satisfy this constraint, we explore two options. First, we use a mixture of a Dirac delta distribution at zero with a discrete distribution for the nonzero pixels (D + D). Second, we use a mixture of Dirac delta distribution at zero with a continuous distribution for the nonzero pixels (D + C).

Discrete mixture model (D + D).—We discretize each pixel value x_i by rounding it to the nearest value in a predetermined grid with points $\{0, g_1, \ldots, g_N\}$, where $g_j > 0$ for j from 1 to N, and g_N corresponds to the largest pixel value after rounding. The model learns the probability of each discrete value as a categorical distribution:

$$p(x_i|\theta_i) = \gamma_{i,0} \cdot \delta_{x_i=0} + \sum_{j=1}^{N} \gamma_{i,j} \cdot \delta_{x_i=g_j}, \quad (3)$$

where each $\gamma_{i,j}$ is predicted by the parameter $\theta_i = (\theta_{i0}, \ldots, \theta_{iN})$ using a softmax function. When the grid is uniform, this likelihood is the same as the discretized softmax likelihood used by Pixel RNN [10], which has achieved state-of-the-art results on benchmark datasets of natural images [54]. However, in particle physics the distribution of pixel values is typically far from uniform. In many typical cases, there is a large number of pixels with small values, and a few pixels with large values, as seen in Fig. 5. To better represent the pixel distribution and minimize the error due to quantization, we assign more grid points to the region of low pixel values. We achieve this by using a power transformation $\hat{x} = x^{1/p}$ on the pixel values, where p is a hyperparameter such that $p \geq 1$.

Discrete and continuous mixture model (D + C).—The pixel values of natural images are usually represented by unsigned integer values between 0 and 255. However, in particle physics images, the pixel values are typically real valued. To avoid explicit rounding, SARM (D + C) is built with a truncated logistic distribution that models the nonzero distribution component of each pixel. To generate the D + C mixture, we reparametrize each pixel as $x_i = \tilde{x}_i \cdot z_i$, where $\tilde{x}_i$ follows a truncated logistic distribution $TL(\mu_i, s_i)$ with mean μ_i and scale parameter s_i. Here $z_i \sim \text{Bern}(\gamma_i)$ is a Bernoulli random variable with probability $p(z_i = 1) = \gamma_i$, which controls the sparsity level. By assuming independence of $\tilde{x}_i$ and z_i, the likelihood function of x_i becomes

$$p(x_i|\theta_i) = \gamma_i \cdot \delta_{z_i=0} + (1-\gamma_i) \cdot \delta_{z_i\neq 0} \cdot p(\tilde{x}_i|\mu_i, s_i), \quad (4)$$

where $\theta_i = \{\mu_i, s_i, \gamma_i\}$ are functions of the previous pixel values $x_{0:i-1}$, to ensure the autoregressive structure. In order to allow for unconstrained optimization, we treat $\log(s_i)$ as the learning parameter and take its exponential in the likelihood equation (4). Since the pixel distribution could be multimodal, we use a mixture of truncated logistic distributions for $\tilde{x}_i$ which is more flexible. An example of the generation process is depicted in Fig. 4

The mixture of truncated logistic likelihood differs from the discretized logistic mixture used in Pixel CNN++ [41] in the way it handles continuous pixel values. Pixel CNN++ requires discretizing x_i and then maximizing the probability on the discretized grid. In contrast, SARMs can directly maximize the probability density function of x_i,

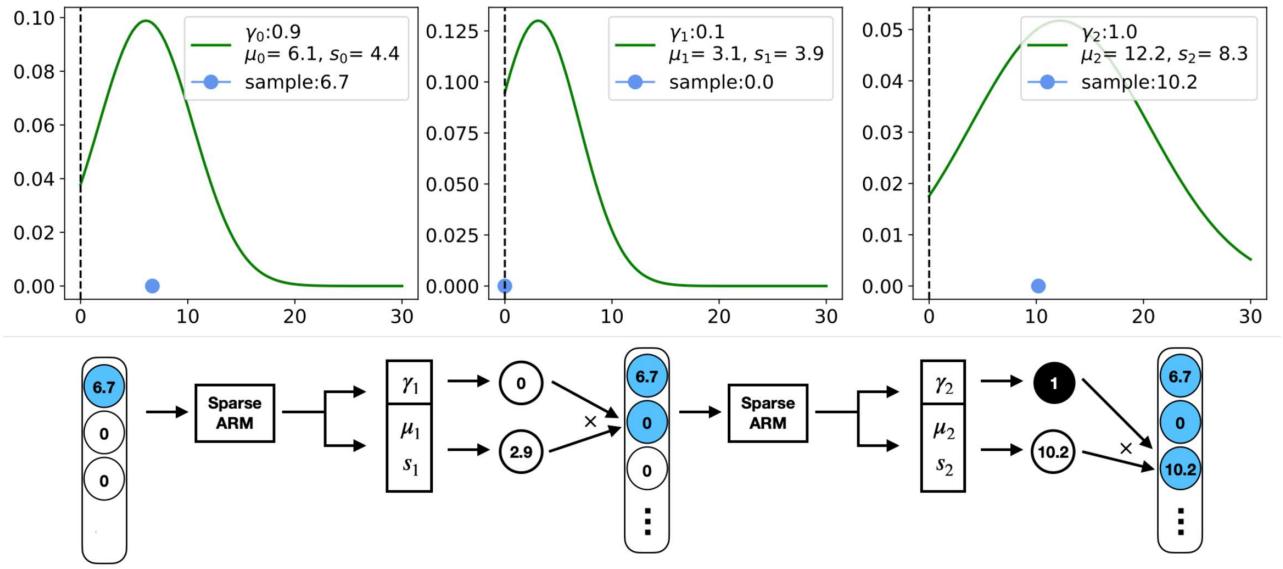

FIG. 4. Generation process for the D + C model. The blue circle dots represent the value sampled for each pixel. For example, given the first pixel value of 6.7, sampled from the empirical distribution of the dataset, the neural network outputs the distribution parameters $\gamma_1 = 0.1, \mu_1 = 3.1, s_1 = 3.9$ to generate the second pixel. Then a Bernoulli random variable is sampled from $z_1 \sim \text{Bern}(\gamma_i)$ and a logistic random variable is sampled from $\tilde{x}_i \sim \text{Logistic}(\mu_i, s_i)$. The value of the second pixel x_i is produced by the product of these two variables as $x_i = z_i \cdot \tilde{x}_i = 0 \cdot 2.9 = 0$. This sequential process is repeated until every pixel is generated.

allowing it to handle continuous pixel values without incurring quantization errors.

There are several differences between the D + D and the D + C models. The D + D model allows enough flexibility to represent multimodal distributions, as each grid point has its own learnable probability. However, there is a price for this flexibility. It is significantly more time consuming to generate an $(N + 1)$-way softmax vector and sample from a discrete mixture (D + D) than it is to generate the parameters of γ, μ, s and then sample from a discrete and continuous mixture (D + C). Other constrained domain distributions such as the exponential and the gamma distributions were also considered but led to inferior results. The exponential distribution suffers from a lack of flexibility due to having only one learnable parameter.

B. Multistage generation for heterogeneous areas

In many ARM applications, a single network is used to predict the parameters θ_i of the conditional probability distribution $P(x_i|\theta_i)$. This approach works well if the distribution of pixel values is similar across pixels, as is often the case in natural images. However, as shown in Fig. 5 (left panel), the pixel value distribution in the central square of a calorimeter image containing a jet is very different from the distribution in the rest of the image (see also [43]). In order to handle these heterogeneous regions,

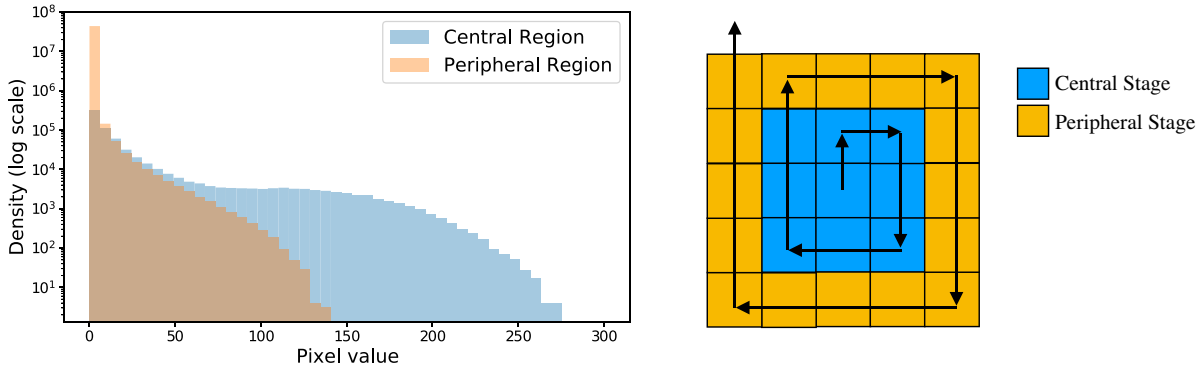

FIG. 5. Left panel: Distribution of pixel values in the jet substructure dataset for the nine pixels in the center of the images (central region) and the rest of the pixels (peripheral region). Note that the majority of the pixels in the peripheral region are zero valued and in general have lower variance than pixels in the central region. Right panel: Two-stage generation for the central and peripheral regions using a spiral path and two different SARM modules. Using different networks for each region improves performance.

we use a two-stage approach by stacking two distinct deep SARM modules, one for the center and one for the periphery. When the model generates the image from the inside out, the outer module generates pixels conditioned on the outputs of the center module, as illustrated in Fig. 5 (right panel). We refer to this model as SARM-2 while the single-stage model is SARM-1. Since the center may not have a clear border, we treat the size of the center relative to the periphery as a hyperparameter during training. Note that in general the number of stages depends on the structure of the data and is not limited to two. Furthermore, it is possible to learn the SARMs associated with each region in any order.

Thus, in summary, through the experiments to be presented, we show that a good heuristic approach for SARM design is to (1) decompose the images into relevant regions (e.g., center vs background); (2) use a different SARM for each region type; and (3) within each region type, preferably choose a systematic and congruent order for generating the pixels, as these compare favorably to random generation orders. By systematic and congruent orders we mean orders that have some kind of continuity for the location of the pixels being generated—subsequent generated pixels should be close in the image—while respecting the geometry of the highly activated region (e.g., a spiral order for a globular region, a linear order for a linear region).

V. EVALUATION METHODS

The goal is to train generative models which create images indistinguishable from images created by the slower Monte Carlo methods. We compare the performance of our models, both in terms of image quality and generation time, against two other generative models: LAGAN [6], the current state-of-the-art generative model for sparse images in particle physics; and Pixel CNN++ [41], a widely used autoregressive model for natural images not tuned for sparse images. We evaluate all models on both datasets described above; note that LAGAN was designed to handle images typically found in the jet substructure dataset, while the muon dataset features extreme sparsity in comparison. We measure the quality of the generated images both qualitatively and quantitatively.

Qualitative evaluation.—We examine typical images generated by each model, as well as the pixelwise average intensity of the generated images, using the images produced by the Monte Carlo methods, which in the jet substructure study are referred to as the PYTHIA images. Additional qualitative comparisons are described in Appendixes C and D.

Quantitative evaluation.—Comparisons of distributions in high-dimensional datasets should focus on the scientific context and potential applications. In particle physics, the calorimeter information is typically used to calculate physical quantities, such as invariant mass or transverse momentum (P_T), which are especially revealing as metrics because they have not been explicitly optimized by the models. In addition, calorimeter images are used to train classifiers which can identify particles from their patterns of depositions.

One-dimensional distributions of mass and P_T can be evaluated in comparison to the distributions from Monte Carlo generators using the Wasserstein distance, the minimum cost to transform one distribution into the other one, expressed by

$$W_p(P,Q) = \left(\inf_{J\in\mathcal{J}(P,Q)} \int \|x-y\|^p dJ(x,y)\right)^{1/p}, \quad (5)$$

where $\mathcal{J}(x,y)$ is the family of joint probability distribution of x and y, P and Q are marginal distributions, and $p \geq 1$. When $p = 1$, this metric is also known as the earth mover's distance [55]. To match the results in [6], we computed $W_1(P,Q)$, where P represents one of the jet observable distributions from the PYTHIA images, and Q represents the corresponding jet observable distribution from the generated images.

An important motivation for developing generative models for fast simulations is to provide a computationally inexpensive method to augment existing datasets in classification tasks [43,56]. The jet substructure dataset was generated to train classifiers to distinguish between jets from W-boson decays (signal) and those from single quarks and gluons, a well-known classification task [43,56]. The muon isolation dataset was generated to train classifiers to distinguish isolated muons from those due to heavy-flavor jet production. Therefore, an essential test for the quality of the generated images is whether they can be used in these classification tasks. To quantify this, the generated images were used as training sets to develop a classifier whose performance was assessed using the Monte Carlo images. The same convolutional neural network architecture was trained with the same hyperparameters on five different datasets: Monte Carlo images, images generated by SARM-2 (D + C) images generated by SARM-2 (D + D), images generated by LAGAN, and images generated by Pixel CNN++. Because higher quality images should lead to improved classification of the Monte Carlo images, we used the classification performance as the evaluation metric.

Speed.—Each generative model was used to generate batches of images multiple times to measure the average speed of image generation.

VI. RESULTS

A. Jet substructure study

1. Qualitative analysis

An example image from each generative model and from the PYTHIA Monte Carlo generator is shown in Fig. 6. It is clear that SARM-2 (D + C) excels at generating pixels with

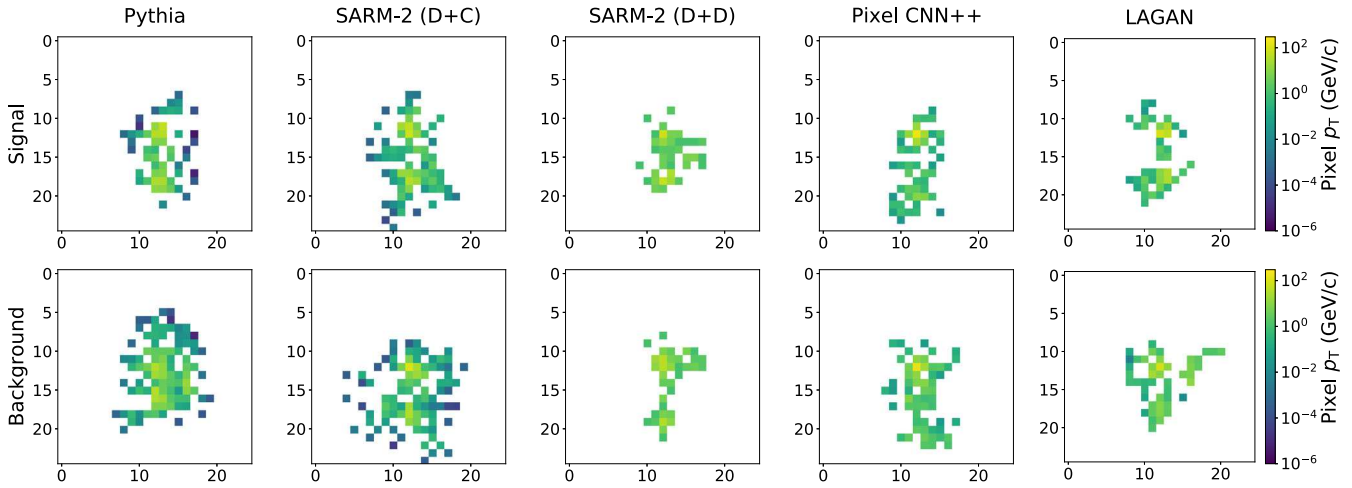

FIG. 6. Example of jet images generated from each model. Notice that SARM-2 (D + C) is able to produce small value pixels in the periphery of the images. The intensity of each pixel is shown on a log scale, where the white space represents pixels with value zero.

small values around the periphery in comparison to the other models. Additional samples for each model can be seen in Appendix H. To assess the overall quality of the generated images, Fig. 7 shows the pixelwise average of each dataset. The autoregressive models, SARMs and Pixel CNN + +, are able to model the peripheral radial region around the center more accurately. This region has higher degree of sparseness than the center region, making it more challenging for the generative models to accurately capture. We note that the images from the SARM-2 (D + C) model appear to be the most similar to the PYTHIA images, while the other models are less able to generate the peripheral region faithfully. In addition, Pixel CNN + + struggles to achieve the radial structure present in the PYTHIA images and creates a squarelike structure instead. In general, the images from Fig. 7 generated by the autoregressive models show a smooth transition from the highly activated center to the sparse border, similar to that seen in the PYTHIA dataset. In contrast, the border of the LAGAN images is irregular, which could be due to its reliance on the ReLU activation function to induce the sparsity, making the model unable to estimate the sparseness level directly.

2. *Quantitative analysis: Jet observables as metrics for quality*

To quantify the fidelity of the images generated by each model as compared with the original samples, we insert them into typical applications in particle physics. In the context of collisions that produce jets, it is common to calculate the invariant mass of the jet, and the transverse momentum. Distributions of jet mass and P_T are shown in

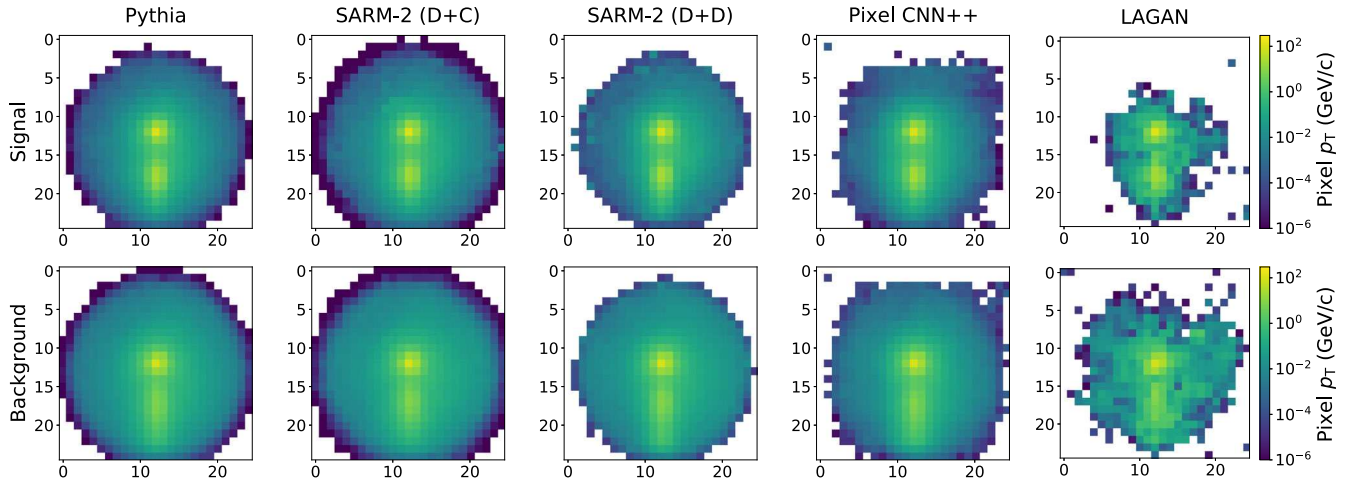

FIG. 7. Pixelwise average of the images generated by each model. Notice that LAGAN struggles to capture the distribution of low value pixels in the periphery of the images and has a nonsmooth radial transition compared to the autoregressive models. The intensity of each pixel is shown on a log scale, where the white space represents pixels with value zero.

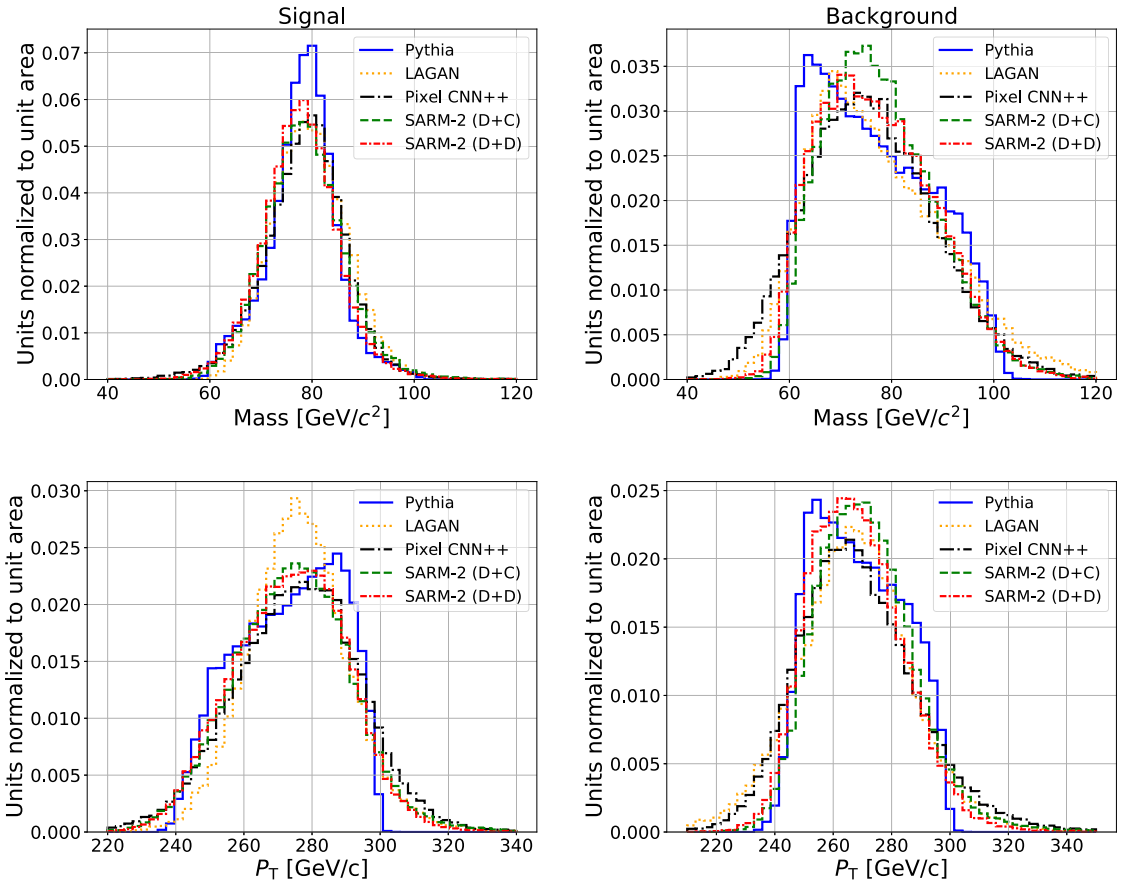

FIG. 8. Distributions of jet observables (top panels: mass, bottom panels: P_T) calculated from images generated by several generative models and from the original images generated by PYTHIA. Signal images, with two collimated quarks, are on the left; background images, with a single quark or gluon, are on the right.

Fig. 8 for all models, which all succeed in matching the general shape, though discrepancies are visible, and Wasserstein distances are shown in Table I.

TABLE I. Comparison of images created by various generative models with original images from PYTHIA, evaluated using the Wasserstein distance (with $p = 1$) between one-dimensional distributions of physical quantities calculated from the images: jet P_T and invariant mass, also shown in Fig. 8. Smaller values indicate a closer match to the PYTHIA images. Four SARMs are evaluated, those with either one-stage (SARM-1) or two-stage (SARM-2) models, and those with either discrete and continuous distributions (D + C) or a mixture of discrete distributions (D + D). The boldface is used to highlight the best performances and thereby also the best models.

	P_T		Mass	
Model	Signal	Background	Signal	Background
LAGAN	3.15	3.29	1.45	1.39
Pixel CNN + +	3.46	3.59	1.09	1.56
SARM-1 (D + C)	2.33	2.46	1.07	1.54
SARM-2 (D + C)	2.32	2.71	1.06	1.39
SARM-1 (D + D)	1.95	2.52	1.34	2.45
SARM-2 (D + D)	**1.44**	**1.66**	**0.94**	**0.92**

All SARM variants achieve lower distances in the P_T distributions than LAGAN and Pixel CNN+, and comparable or better distances in jet mass. The best results in all categories are obtained by the SARM-2 (D + D). Compared to the best of Pixel CNN + + and LAGAN, SARM-2 (D + D) provides a 51.92% improvement for P_T, and a 23.79% improvement for mass, averaged over the signal and background sets. These results demonstrate the effectiveness of taking sparseness into account during learning and generation. Secondly, the SARM-2 models clearly outperform the SARM-1 models for both the (D + D) and (D + C) likelihoods, which shows the effectiveness of the multistage approach in modeling heterogeneous areas in the images.

3. Classification of generated images

An important application of generated calorimeter images is to augment training sets for networks learning to perform vital signal-background classification tasks. As a high-level test of the image quality, we train networks using images generated by each model (2×10^5 signal, 2×10^5 background), and evaluate the performance on the original images from PYTHIA (2×10^4 signal, 2×10^4

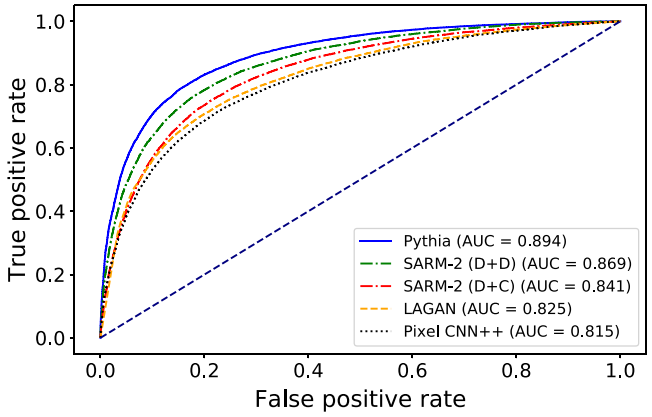

FIG. 9. Evaluation of the fidelity of images generated by several models in the context of a classification task. Images generated by the model are used to train a network to discriminate between signal and background, but performance is measured using the original PYTHIA images.

TABLE II. Quality of jet substructure signal images generated by SARM-1 (D + D) with various pixel-generation orderings. The quality is measured by the Wasserstein distance for the physical observables (P_T and mass) between the generated images and the original PYTHIA images. Spiral-in clockwise/counterclockwise (CW/CCW), spiral-out CW/CCW, column-wise, row-wise, and two random approaches are compared. The outward spiral orders show good performance due to the radial structure of the images.

	P_T (std.)	Mass (std.)
Spiral-out CCW	1.94 (0.09)	1.38 (0.10)
Spiral-out CW	2.47 (0.23)	1.53 (0.22)
Spiral-in CCW	3.64 (0.32)	1.62 (0.14)
Spiral-in CW	3.20 (0.22)	1.45 (0.16)
Row-wise	3.06 (0.30)	2.01 (0.11)
Columnwise	3.38 (0.39)	1.90 (0.08)
Random I	4.05 (0.51)	1.74 (0.53)
Random II	3.41 (0.33)	1.25 (0.26)

background). Training sets which best mimic the original PYTHIA images should lead to networks which most closely match the performance of a network trained on PYTHIA images. Detailed information about the classifier and training procedure is given in Appendix F. The receiver operating characteristic (ROC) curves for networks trained on images from PYTHIA, SARM-2 (D + C), SARM-2 (D + D), Pixel CNN++, and LAGAN are shown in Fig. 9. Classifiers trained on both SARM generated datasets have higher area under the ROC curve (AUC) scores than the classifiers trained on the LAGAN images and Pixel CNN + + images.

4. Generation order

SARMs generate images pixel by pixel, conditioning each step on the previously generated pixels. The order of the pixel generation corresponds to a dependency decomposition in Eq. (1), which may impact training performance. The traversal path is especially important for images containing heterogeneous areas. For natural images, Ref. [50] uses an ensemble of models with random paths, while Pixel CNN++ and other models [10,41] use the row-by-row pixel ordering.

The average performance of various pixel orderings for SARM-1 (D + D) over ten repeated runs is shown in Table II. Each order is evaluated by using the Wasserstein distance between the distributions of the generated signal images and the PYTHIA signal images for the jet P_T and invariant mass.

The spiral paths, clockwise (CW) and counterclockwise (CCW), achieve the stronger results. This could be understood in terms of mutual information between neighboring pixels. Unlike the other orderings, the spiral ordering is continuous, i.e., it always generates a pixel adjacent to the previously generated pixel. Furthermore, the spiral order is congruent with the globular shape of the highly activated region in the jet images, e.g., Fig. 7. Starting the spiral from the center outperforms inward spirals, indicating that it may be easier to learn the correlations between the pixels starting with pixels that are more active (more nonzero pixel values). The difference between CW and CCW is likely due to asymmetries generated by the rotation and centering steps in the preprocessing of the data. We use this asymmetric version of the data in order to enable direct comparison to the LAGAN model. These results confirm that nonrandom, systematic generation orders that have good continuity and congruence properties perform well (and outperform random orders). A full exploration of the ordering dependency is beyond the scope of this work and computationally challenging due to the factorial number of possible orderings.

5. Computational costs

Table III shows the speed of the generative models in comparison to the Monte Carlo method (PYTHIA). The SARM-2 models are 5 times slower than LAGAN, which is mainly due to the extra computational cost of the autoregressive structure. On the other hand, the SARM-2 models

TABLE III. Comparison of image generation speed between the Monte Carlo approach (PYTHIA) and various generative models. The SARM-2 models are slower than LAGAN, but still considerably faster than PYTHIA and Pixel CNN + +.

Model	Speed (images/sec)
PYTHIA [6]	34
Pixel CNN + +	50
SARM-2 (D + D)	1612
SARM-2 (D + C)	2480
LAGAN	10,176

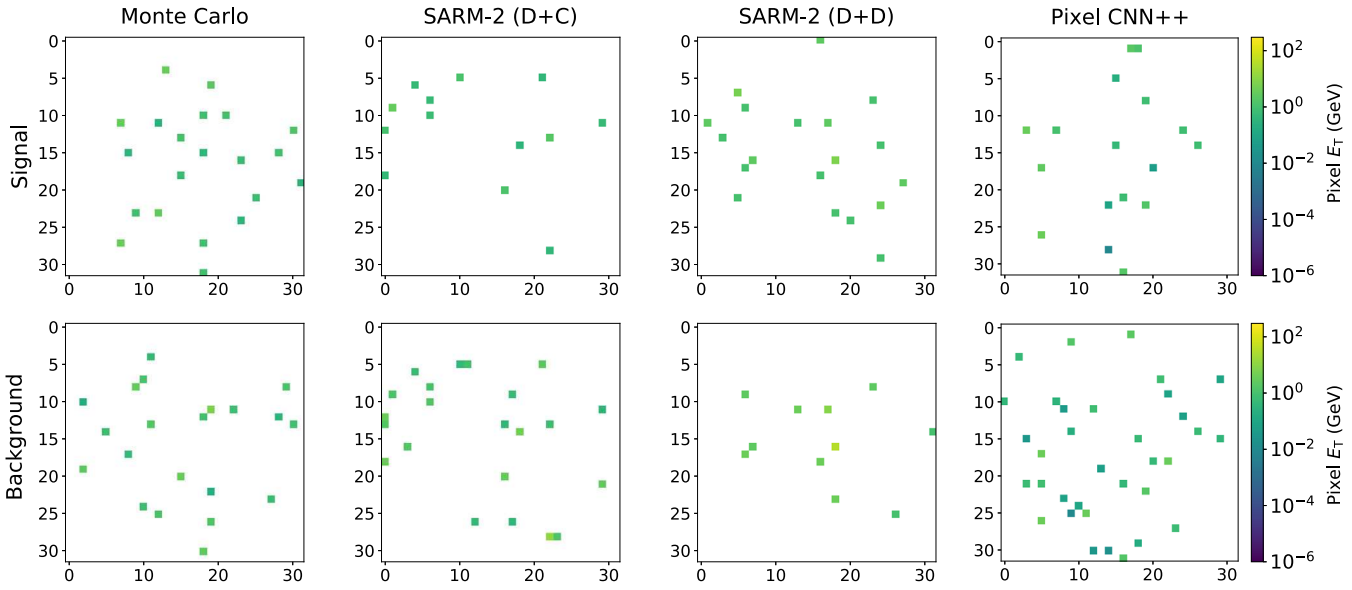

FIG. 10. Example calorimeter images in the vicinity of a muon from the generative models as well as the original Monte Carlo generator. The top row shows isolated muons (signal), while the bottom shows muons produced in association with a jet (background). The intensity of each pixel is shown on a log scale, where the white space represents pixels with value zero.

are 2 orders of magnitude faster than PYTHIA and Pixel CNN++. The forward pass of the Pixel CNN++ model is computationally expensive due to the ResNet blocks with convolutional layers and skip connections [41,57]. In contrast, SARMs use a simple feed forward network with disabled connections to preserve autoregressive structure. The speed of the generative models is measured on a machine with 4 TITANX graphics processing unit (GPU) cards each with 12 gigabytes of memory. The speed of PYTHIA was assessed in [6] using Amazon Web Services and an IntelR XeonR E5-2686 v4 at 2.30 GHz CPU.

There is room to further optimize the speed of the SARM models. For instance, we find that reducing the size of the intermediate upsampling layer of the SARM (D + D) drastically reduces the memory requirements and improves the generation speed. Another direction is to explore model pruning and compression.

B. Muon isolation study

1. Qualitative analysis: Average generated images

Typical calorimeter images in the vicinity of a muon generated by the standard Monte Carlo method, Pixel CNN++ as well as two SARMs are shown in Fig. 10. In this context, LAGAN suffered from mode collapse and failed to generate reasonable quality images (see Fig. 18 in the Appendix). This is a well-known problem when training GANs [6,38,39], especially with sparse data.

Figure 11 shows the pixelwise average images. The SARM-2 models and the Pixel CNN++ reproduce the radial symmetry seen in the original images. However, the average images produced by Pixel CNN++ contain noticeable artifacts, potentially due to the convolutional layers in the model [58].

2. Quantitative analysis: Calorimeter observables as metrics for quality

To assess the fidelity of the images quantitatively, we calculate physical quantities which summarize the content of the images and allow for comparison of one-dimensional distributions. While calorimeter images in the vicinity of a muon do not necessarily contain a clustered jet, the total P_T and invariant mass of the entire image do have physical meaning. Figure 12 shows the distributions of these quantities for the original Monte Carlo images, as well as for the generated images, and Table IV provides the corresponding Wasserstein distances.

The datasets generated by both SARM-2 models have considerably smaller Wasserstein distances than the datasets generated by the Pixel CNN++ model for both signal and background. The distributions of all the generated datasets approximate the shape of the Monte Carlo distributions quite well for P_T and mass, but the distributions of the Pixel CNN++ dataset have a small shift toward higher values, for both the signal and the background. In addition, for the background they are more concentrated around the mean. This is potentially due to the fact that Pixel CNN++ fails to model the right tail of the pixel distribution, where the pixels have higher values but appear much less frequently in the data (Fig. 21 in the Appendix). The SARM-2 (D + D) has the best overall performance, with improvements of 68.08% for P_T and 66.44% for mass, averaged over the signal and background datasets.

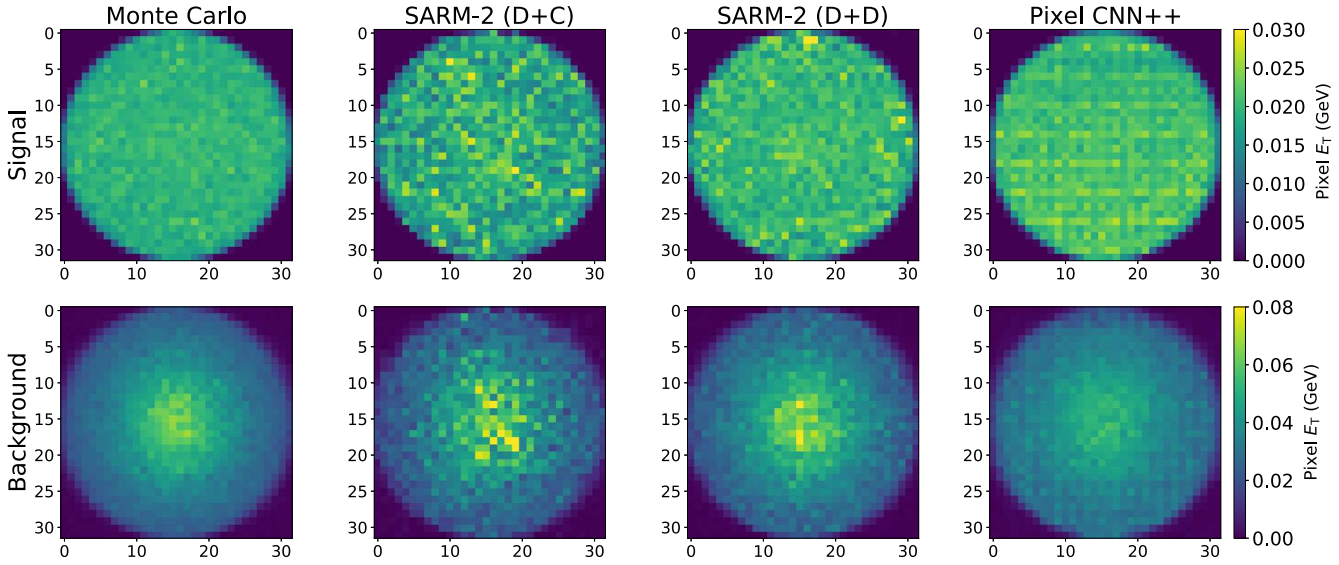

FIG. 11. Pixelwise averages of calorimeter images in the vicinity of a muon from the generative models as well as the original Monte Carlo generator. The top row shows isolated muons (signal), where little calorimeter activity is expected. The bottom row shows muons produced in association with a jet (background), which deposits significant energy near the muon. A linear scale is used to reveal the differences between signal and background images.

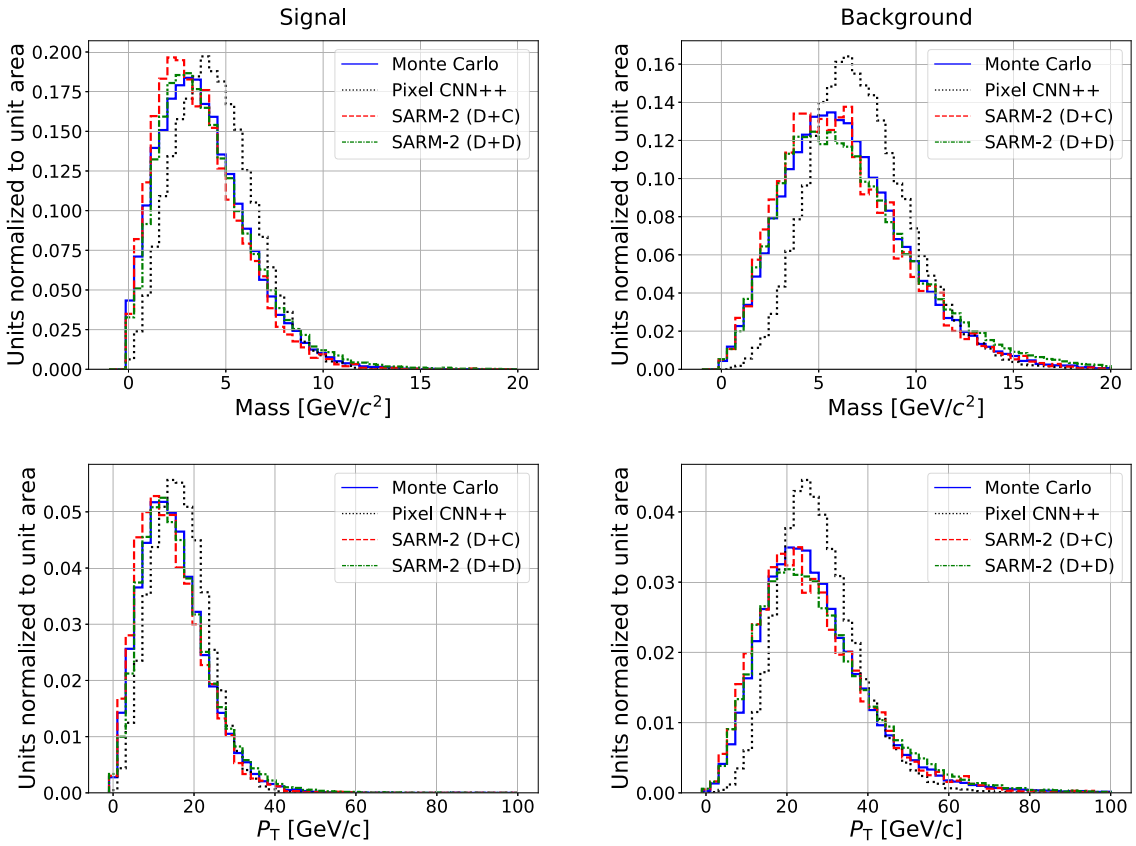

FIG. 12. Distributions of calorimeter observables (top panels: invariant mass, bottom panels: total P_T) calculated from images generated by several generative models and the originals generated by a Monte Carlo generator. Signal images, in the vicinity of an isolated muon, are on the left. Background images, in the vicinity of a muon produced with an associated jet, are on the right.

TABLE IV. Comparison of images created by various generative models to the original Monte Carlo images using the Wasserstein distance (with $p = 1$) between one-dimensional distributions of physical quantities calculated from the images: P_T and invariant mass, also shown in Fig. 12. Smaller values indicate a closer match to the Monte Carlo images. Two SARMs are evaluated, with either discrete and continuous distributions (D + C) or a mixture of discrete distributions (D + D). The boldface is used to highlight the best performances and thereby also the best models.

	P_T		Mass	
Model	Signal	Background	Signal	Background
PixelCNN + +	1.75	2.92	0.58	0.82
SARM-2 (D + C)	0.79	0.97	0.25	**0.21**
SARM-2 (D + D)	**0.56**	**0.93**	**0.17**	0.31

3. Classification of generated images

The fidelity of the images can be evaluated in the context of the data analysis task for which they were created, training a network to distinguish between signal (calorimeter images near isolated muons) and background (calorimeter images near nonisolated muons).

A convolutional neural network classifier was trained using images generated exclusively by each of the models [SARM-2 (D + C), SARM-2 (D + D), or Pixel CNN + +]; one additional network was trained using images from the Monte Carlo generator. The quality of the images is measured by comparing the classification performance of these networks on images from the Monte Carlo generator, see Fig. 13. The classifiers trained on each SARM dataset have higher AUC scores than the classifier trained on the Pixel CNN + + dataset, providing additional evidence that the SARM datasets are more similar to the Monte Carlo images and thus better suited for downstream tasks such as data augmentation.

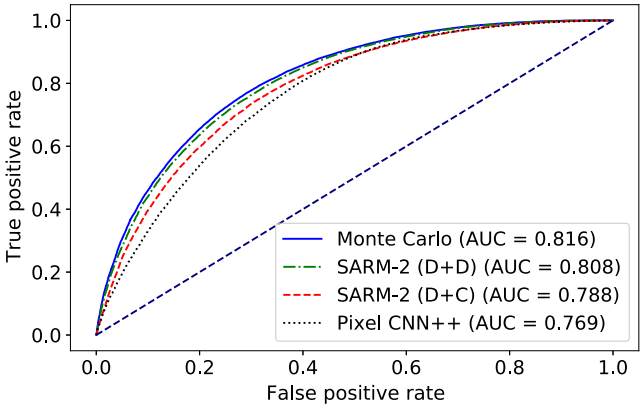

FIG. 13. Evaluation of the fidelity of images generated by several models in the context of a classification task, distinguishing muons produced in isolation from those produced in association with a jet. Images generated by the model are used to train a network to discriminate between signal and background, but performance is measured using the original Monte Carlo images.

TABLE V. Quality of images generated by SARM-1 models with various pixel-generation orderings for the muon isolation signal dataset. The quality is measured by the Wasserstein distance for the physical observables (P_T and mass) between the generated images and the original Monte Carlo images.

	P_T (std.)	Mass (std.)
Spiral-out CCW	0.99 (0.37)	0.27 (0.10)
Spiral-out CW	0.92 (0.33)	0.26 (0.09)
Spiral-in CCW	0.81 (0.23)	0.20 (0.05)
Spiral-in CW	0.95 (0.24)	0.24 (0.07)
Row-wise	0.99 (0.28)	0.20 (0.05)
Columnwise	0.90 (0.26)	0.22 (0.05)
Random I	1.17 (0.30)	0.32 (0.08)
Random II	1.34 (0.41)	0.37 (0.11)

4. Generation order

In this section, we discuss the impact of the pixel order for SARMs associated with the signal dataset of the muon isolation study. Similarly to Sec. VI A 4, we conducted ten repeated experiments for each of the orders and summarized the results in Table V.

In contrast to the jet substructure study, the muon isolation data is not rotated and the pixel value distribution is quite uniform. Therefore we see that different generation orders have a similar performance in terms of mass and P_T distances. In addition, all the models trained using systematic orders that have some continuity in the sequence of pixels slightly outperform the models trained using random orders. In combination, these results confirm the validity of the heuristic strategy outlined at the end of Sec. IV, providing general guidelines for SARM design and pixel generation when applying these models to other datasets.

5. Computational costs

Calorimeter image generation speeds in the context of the muon isolation study are shown in Table VI for

TABLE VI. Comparison of image generation speed between the Monte Carlo approach and various generative models. The SARM-2 models are considerably faster than Pixel CNN + + and the Monte Carlo generator.

Model	Speed (images/ sec)
Monte Carlo	5
Pixel CNN + +	10
SARM-2 (D + D)	625
SARM-2 (D + C)	1136

the SARM models, Pixel CNN + +, and the Monte Carlo generator. The SARM models are 1 to 2 orders of magnitude faster than Pixel CNN + +, similar to the observation of the jet substructure study. The generation speed of each generative model is measured with the same hardware as described in Sec. VI A 5. The speed for the Monte Carlo generator is measured on an Intel(R) Xeon(R) E5-2680 at 2.70 GHz CPU.

VII. CONCLUSION

Sparse images, prevalent in particle physics datasets, present unique challenges for generative models. We have developed and applied a new class of models, deep SARMs, specifically designed to handle extreme sparseness. These compositional models are also able to take advantage of the structure present in particle physics images by using a multistage generation approach. Using several different metrics, we compared SARMs to other generative models, in particular to Pixel CNN + +, a popular autoregressive model not adapted for sparsity, and to LAGAN, a state-of-the-art GAN for sparse images. The comparisons were carried using two benchmark datasets.

In the first case study on jet substructure, the adaptation to sparseness enables SARMs to produce qualitatively and quantitatively higher quality images than Pixel CNN + + and LAGAN. SARM are also orders of magnitude faster than traditional Monte Carlo methods and Pixel CNN + +, but slower than the nonautoregressive model LAGAN, showing a trade-off between speed and quality. The second case study features extremely sparse images corresponding to calorimeter images in the vicinity of muons. While competing models produce artifacts or suffer from mode collapse, SARMs are able to handle and model extreme degrees of sparseness.

In sum, given the prevalence of sparse images in particle physics and beyond, SARMs can be expected to provide an important option for rapid, high-quality, image generation from training data. Because of their quality, the generated images in turn will be able to benefit a variety of downstream data analyses.

Original data and software will be made available from the UCI Machine Learning in Physics Web portal [59].

ACKNOWLEDGMENTS

We wish to acknowledge a hardware grant from NVIDIA. The work of Y. L., J. C., and P. B. is in part supported by Grants NSF NRT 1633631 and ARO 76649-CS to P.B. D.W. is supported by the Department of Energy Office of Science. The authors would like to thank Benjamin Nachman for helpful feedback on an early draft.

APPENDIX A: 2D TOY EXAMPLE

We simulate a dataset containing pairs of two variables x_0 and x_1, such that $x_0 \sim p(x_0|x_1)$ and $x_1 \sim p(x_1)$. In this toy example we show that the autoregressive model is still able to learn to generate the joint distribution of x_0 and x_1, even though during training it is forced to learn $x_0 \sim p(x_0)$ first, and then to learn the dependency $p(x_1|x_0)$. The simulated training data contains 1000 pairs of $\{x_0, x_1\}$ according to $x_1 \sim N(0,1)$ and $x_0 = x_1 + \epsilon$, where $\epsilon \sim N(0,1)$, a standard normal distribution independent of x_1. The joint distribution of x_0, x_1 is shown in Fig. 14. The toy autoregressive model learns to generate x_0 using two learnable parameters, μ_0 and $\log(\sigma_0)$, corresponding to the mean and log standard deviation of x_0. It has a single linear layer for predicting μ_0 and $\log(\sigma_0)$, which corresponds to the mean and log standard deviation of x_1. The model is trained for 5000 iterations, by maximizing the likelihood $p(x_0, x_1)$. During the generation stage, the model generates x_0 without knowing x_1. Since the goal of the model is to generate the joint distribution of $(x_0, x_1) \sim P(x_0, x_1)$, to do this it only needs to learn the marginal distribution, which is $x_0 \sim N(0,2)$ and the relationship $x_1 = x_0 - \epsilon$. Figure 14 shows the result of training

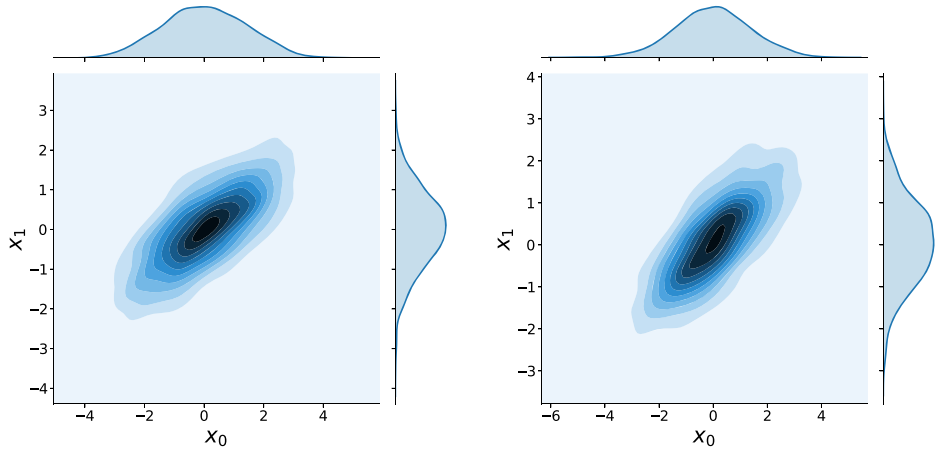

FIG. 14. Left panel: density plot of training data. Right panel: density plot of generated data. The two distributions are very close, showing that the ARM is able to learn the joint distribution of x_0 and x_1 well.

this model and we can see it correctly learns the means and variances of $\{x_0, x_1\}$ along with the data distribution despite the fact that it has to generate x_0 before generating x_1.

APPENDIX B: MADE STRUCTURE

The masked autoencoder for distribution estimation (MADE) structure enforces the autoregressive property on fully connected layers by using a carefully selected binary mask on the weights of the layer. The joint likelihood of the MADE structure can be evaluated in one forward pass of the network during training, which is not possible in other models like Pixel-RNN [10] and Pixel CNN++ [41]. This allows MADE to take advantage of the GPU acceleration. In our SARM implementation, we consider a simple MADE structure with input x and a stack of multiple hidden layers $\mathbf{h}(\mathbf{x})$, where each $\mathbf{h}(\mathbf{x})$ follows

$$\mathbf{h}(\mathbf{x}) = \mathbf{f}(\mathbf{b} + (\mathbf{W} \odot \mathbf{M}^{\mathbf{W}})\mathbf{x}),$$
$$\boldsymbol{\theta} = \mathbf{f}(\mathbf{c} + (\mathbf{V} \odot \mathbf{M}^{\mathbf{V}})\mathbf{h}(\mathbf{x})). \tag{B1}$$

Here $\boldsymbol{\theta}$ is the output, and $\mathbf{f}$ is the activation function of the hidden layer. In practice, we found Gaussian Error Linear Units [60] work better in our experiments than other activations such as sigmoid and tanh. Both $\mathbf{W}$ and $\mathbf{V}$ are weight matrices, with corresponding masks: the hidden mask $\mathbf{M}^{\mathbf{W}}$, and the output mask $\mathbf{M}^{\mathbf{V}}$. Each matrix is multiplied elementwise with each mask.

Suppose $\mathbf{x} \in R^D$, it can be shown that for the input mask

$$\mathbf{M}^{\mathbf{W}}_{k,d} = 1_{k \bmod D \le d} = \begin{cases} 1 & \text{if } k \bmod D \le d, \\ 0 & \text{otherwise.} \end{cases} \tag{B2}$$

Likewise, suppose $\mathbf{h}(\mathbf{x}) \in R^H$, then for the output mask

$$\mathbf{M}^{\mathbf{V}}_{k,d} = 1_{k \bmod D < d} = \begin{cases} 1 & \text{if } k \bmod D < d, \\ 0 & \text{otherwise.} \end{cases} \tag{B3}$$

Then the output $\boldsymbol{\theta}$ satisfies autoregressive structure: for any i, θ_i only depends on $x_{j<i}$. As shown in Fig. 3, the parameter θ_i is used to generate the ith pixel during generation. For example, if the likelihood is a logistic distribution, then $\theta_i = [\mu_i, s_i]$, where μ_i, s_i corresponds to the mean and scale of a logistic distribution.

During generation, at step i we take the previously generated $x_0, x_1, \ldots, x_{i-1}$ and pad the remaining $x_i, \ldots, x_{D-1}$ with zeros. Then we input this vector in the MADE structure so that the output θ_i depends only on $x_0, \ldots, x_{i-1}$. Finally, we sample the pixel x_i conditioned on θ_i and repeat this process until every pixel is generated.

APPENDIX C: FURTHER ANALYSIS OF THE JET STRUCTURE STUDY

Figure 15 shows the subtraction between the pixelwise average of the images from each generative model and the pixelwise average from PYTHIA. Notice the differences are concentrated in the middle of the images where there are higher value pixels. The images generated by both SARM models have small differences compared to the ones

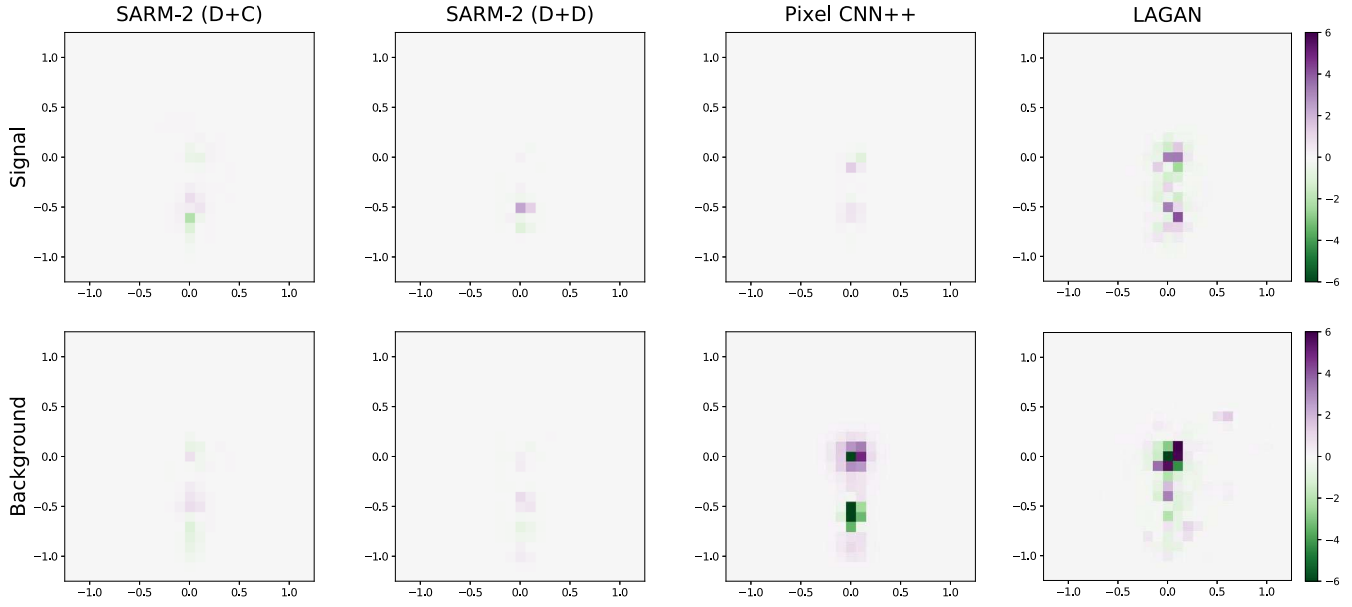

FIG. 15. Error measured by subtracting of the pixelwise average of the images created by each generative model and the pixelwise average of the images generated with PYTHIA. The SARM models have lower error than both Pixel CNN++ and LAGAN with most of the errors are concentrated in the center of the image.

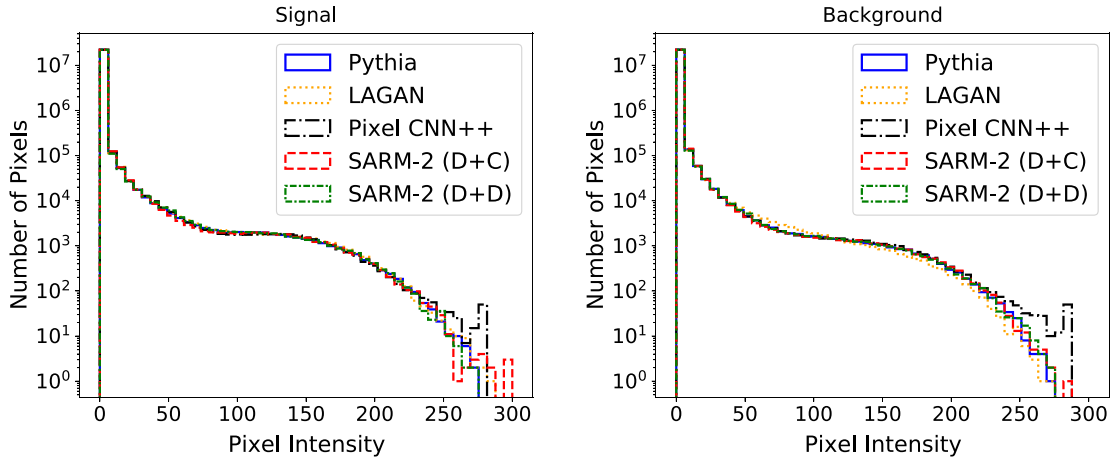

FIG. 16. Distribution of aggregated pixel intensity in the generated images for jet substructure study. Notice most of the differences happen at high pixel values where there are fewer events. LAGAN also has a harder time replicating the distribution of background images across all pixel values compared to the other models.

generated by LAGAN for both signal and background and by Pixel CNN++ for background. Also, Pixel CNN++ has higher errors in background images compared to signal images.

Figure 16 shows the distribution of pixel values across all the generated images. For the signal images, all the models match the PYTHIA distribution for pixel values below 200 but the models have difficulties at higher values. SARM-2 (D + D) and LAGAN have the closest match at high pixel values while SARM-2 (D + C) and Pixel CNN++ overestimate them. For the background images, most of the models accurately predict low value pixels, but LAGAN

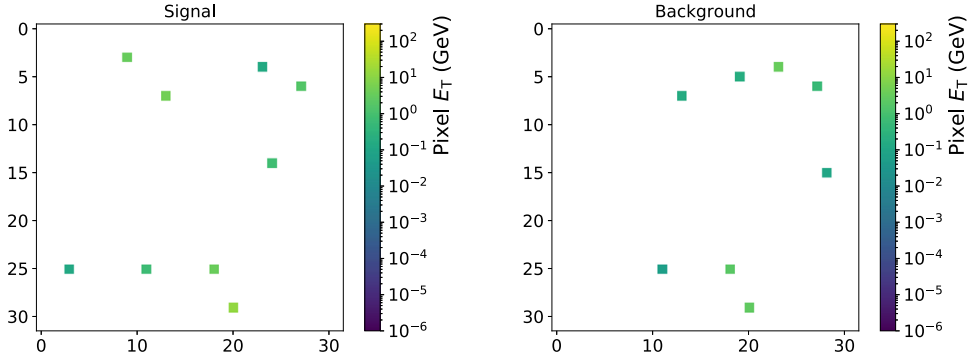

FIG. 17. Typical muon images generated using LAGAN. The figures are plotted in log scale, where the white space represents pixels with value zero.

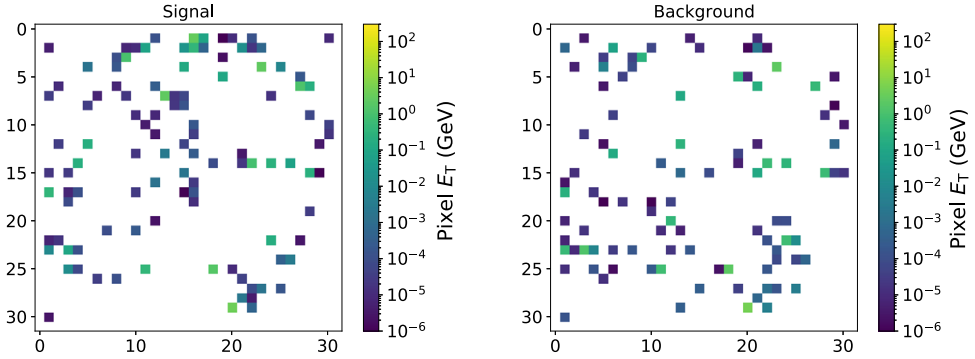

FIG. 18. Pixelwise average of muon images from LAGAN for signal and background. The average images generated by LAGAN fail to reproduce the radial structure present in the average Monte Carlo images (Fig. 11).

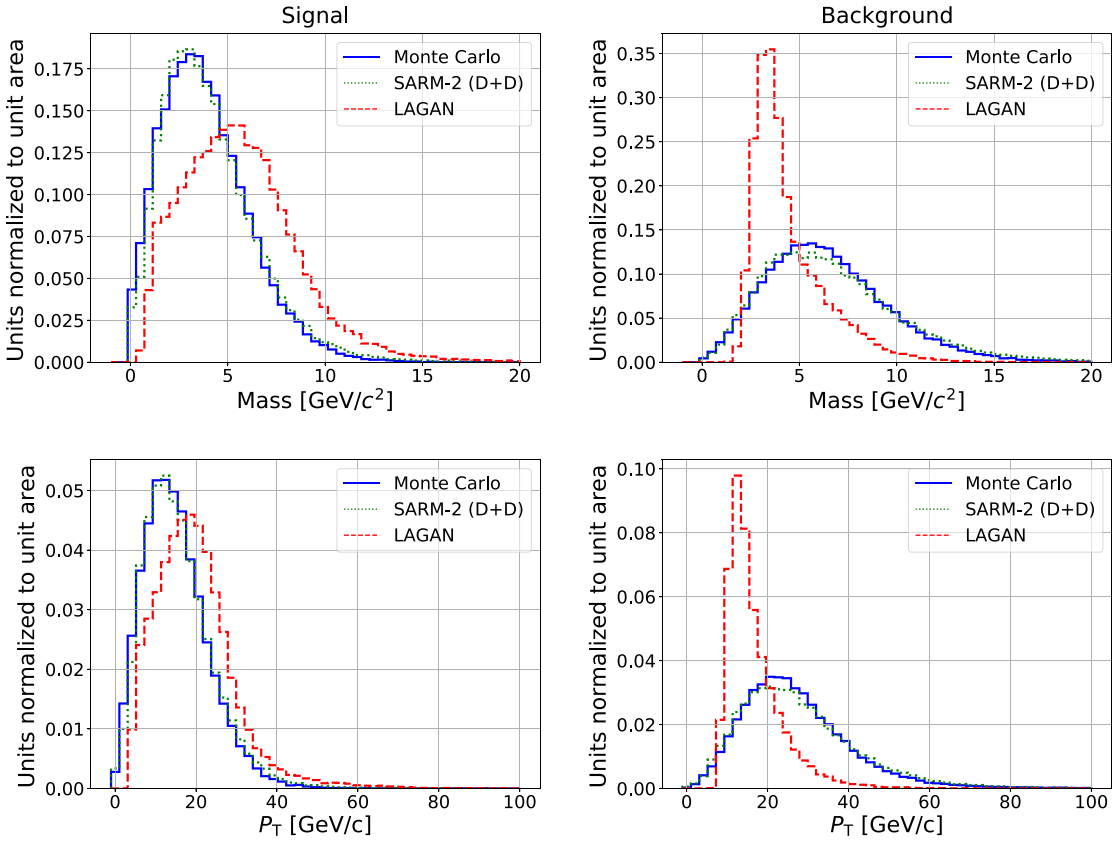

FIG. 19. Comparison of the mass and P_T distributions of the images generated by LAGAN, SARM-2 (D + D), and the Monte Carlo simulations for both signal and background muons.

slightly overestimates pixels in the range 50 to 100 and underestimates them afterward. For high pixel values, Pixel CNN++ strongly overestimates pixels in the range 250–300 while the other models remain reasonably close to PYTHIA. In both cases the models have difficulties learning the high value pixels, which is expected since there are very few pixels in this range in the PYTHIA distribution.

APPENDIX D: FURTHER ANALYSIS OF THE MUON ISOLATION STUDY

1. LAGAN

Despite our best efforts, the LAGAN model performed poorly every time it was trained on the muon isolation dataset. As seen in Figs. 17 and 18 the pixelwise average image does not capture the radial structure present in the dataset and some of the pixels with high values seem to be present in many of the images. This seems to be due to a low amount of variability in the generated images, typical of mode collapse in GANs. This performance is also reflected in the distributions of P_T and mass (Fig. 19) and the respective Wasserstein distances which are 1 order of magnitude worse than the values for the other models (Table VII).

2. SARM vs Pixel CNN++

Figure 20 shows the subtraction between the pixelwise average of the images from each generative model and the pixelwise average from PYTHIA in the muon isolation dataset. For the signal data, all models show very small differences, evenly distributed across the radial structure of the images. In particular, Pixel CNN++ is overrepresenting most of the pixels in the artificial checkerboard pattern noted before. For the background data the errors are slightly higher for all models. The SARM models have more difficulties with the pixels in the

TABLE VII. Wasserstein distance of the physical constituents jet P_T and mass distributions between the original muon images from the Monte Carlo generator and the images created by the generative models. A small distance signifies a good agreement. SARM-2 (D + D) is the two-stage SARM model with a discrete mixture. The boldface is used to highlight the best performances and thereby also the best models.

	P_T		Mass	
	Signal	Background	Signal	Background
LAGAN	4.81	10.88	1.81	2.17
SARM-2 (D + D)	**0.56**	**0.93**	**0.17**	**0.31**

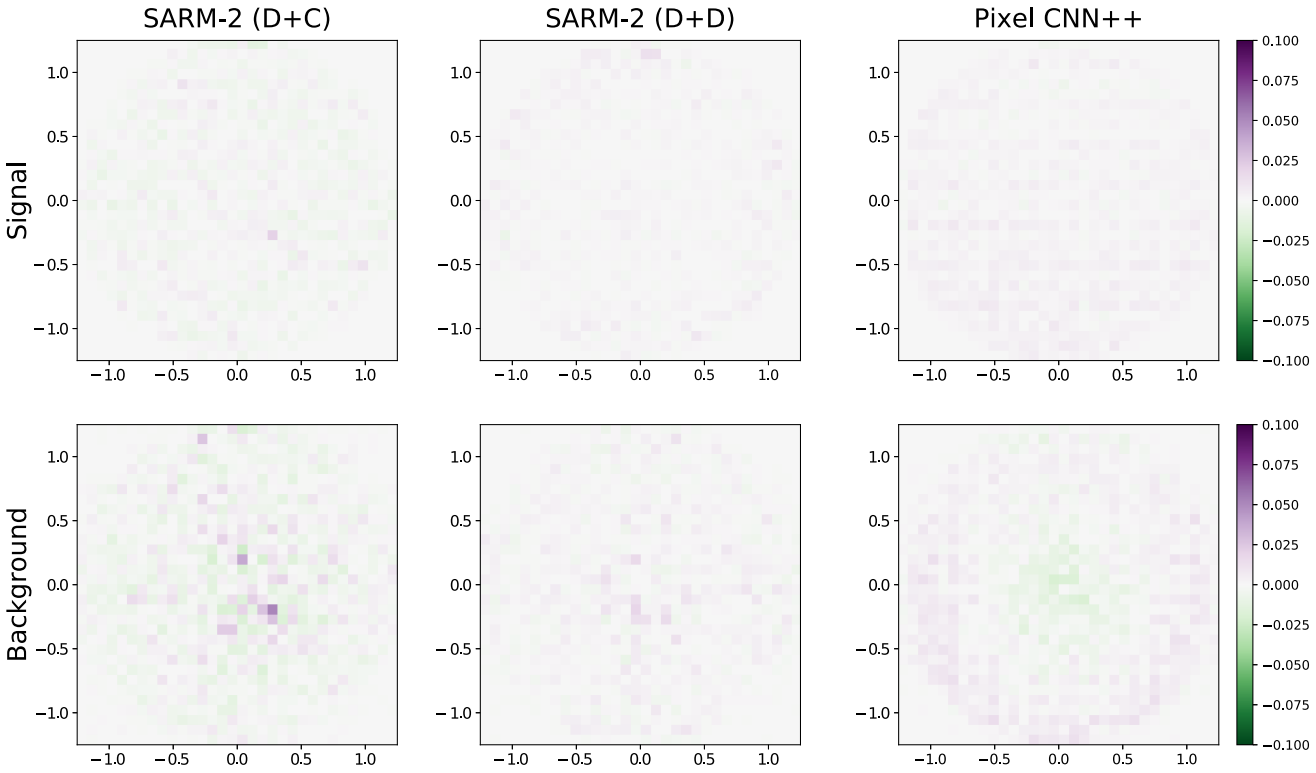

FIG. 20. Subtraction between the pixelwise average of generated images vs Monte Carlo images. The errors are evenly distributed in the signal images, while they are concentrated in the center for the background images. In the center there is larger number of high intensity pixels.

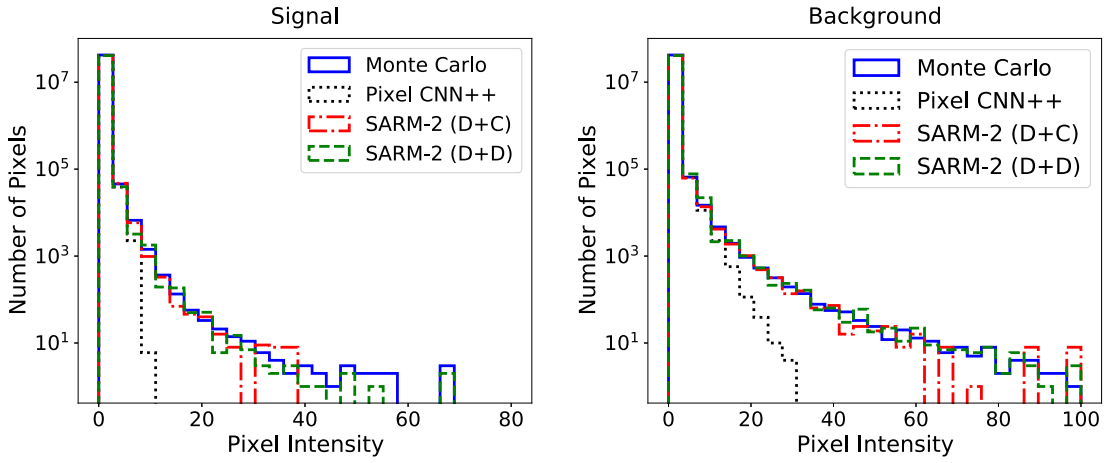

FIG. 21. Distribution of pixel intensity for muon isolation study. Pixel CNN++ underrepresents the distribution while the SARM models miss the high pixel values where there are fewer events.

center and tend to overrepresent them while Pixel CNN++ underrepresents the center and overrepresents the periphery.

Figure 21 shows the distribution of pixel values across all the generated images. For both signal and background the Pixel CNN++ model is underrepresenting pixels with high intensity, while the SARM models match the distribution quite well. Like in the jet substructure study, most of the errors correspond to pixels with high intensity values, which is expected since these values are rare in the training data, making it difficult to correctly learn their distribution.

APPENDIX E: SOFTWARE MODIFICATIONS

1. LAGAN

The code and weights of the original LAGAN model for the jet substructure study dataset are publicly available. This makes it possible to generate new images using the original model's weights for this dataset, but the model needs to be retrained to generate images of a different dataset. The model was retrained for the muon isolation study and it also had to be modified to adapt it to the larger images of 32×32 pixels since it has upsampling layers in the generator part of the GAN.

2. PixelCNN++

As a baseline for autoregressive models we used the Pixel CNN++ [41]. Due to speed and memory restrictions, we had to modify the original model by reducing the number of filters in the masked convolutional layers and the number of residual blocks compared to the original model. Both the number of filters and the number of residual blocks are optimized as hyperparameters using grid search with 5, 10, or 20 filters and 2 or 3 residual blocks. However, we found most hyperparameter combinations to have similar performance. The model with 20 filters and 3 blocks performs slightly better in the jet substructure study, and the model with 10 filters and 5 blocks performs slightly better in the muon isolation study. Even though the models we used are smaller than the original model in [41], they are almost as slow as the traditional Monte Carlo methods (Tables III and VI).

APPENDIX F: ARCHITECTURE AND HYPERPARAMETER OPTIMIZATION

We performed a search over the architectures of the SARMs including the number of hidden layers structure, the size of the central area for the two-stage approach and the size of the intermediate upsampling layer using SHERPA [61]. We also conducted search of the transformation parameter p with values [1, 1.1, 1.2, 1.3, 1.5, 2] for the D + D models. All models were implemented in PyTorch [62], and were trained for 300 epochs with outward spiral (CCW) order using the Adam optimizer [37] with learning rate 3e-4, decreased by half every 100 epochs and minibatch size 128.

For the jet substructure study, the best SARM-2 configuration had a center area of side length 3. For the D + D models, we used five hidden layers with an upsampling layer of size 10 and found that a power transformation with $p = 1.0$ yields slightly better results. For the D + C models, we found that the model with three hidden layers and a mixture of five truncated logistic for the C component works well for both signal and background images. In the generation order experiments, similarly we used SARM-1 (D + D) models with five hidden layers, an upsampling layer of size 10 and a power transformation with $p = 1.0$, effectively no transformation. And all models are trained with identical settings: learning rate of 3e-4, decreased by half every 100 epochs and minibatch size 128. For the LAGAN model we used the publicly available version of LAGAN optimized by the original authors.

For the muon isolation study, the best model we found had five hidden layers, and a center area of side length 7 for both D + D and D + C models. For the SARM-2 (D + D), we used an upsampling layer of size 10 and found that a power transformation with $p = 1.2$ for signal and $p = 1.3$ for background provided the best results. And for the D + C models, we found again that a mixture of five truncated logistic for the C component works well for both signal and background images.

For the classification tasks, we trained five convolutional neural networks with the same structure on each of the datasets. We randomly split the data into a 90% subset for training and a 10% subset for validation. The validation set is used for early stopping during training to avoid overfitting. The convolutional neural network model has two convolutional blocks, two fully connected layers with 100 rectified linear units, and a sigmoid unit at the end to predict the probability of the image being signal. Each convolutional block contains two convolutional layers with 3×3 kernels and 30 filters with rectified linear units followed by a maxpooling layer with 2×2 kernel. All models were trained in PyTorch using the Adam optimizer, with a learning rate of 0.001 and a batch size of 128.

APPENDIX G: COMPLEXITY ANALYSIS

Next we compare the number of parameters for the different models in Table VIII. Note that the original Pixel CNN++ model [41] uses 160 convolutional filters. With all these filters, each forward pass takes more than 1 sec on four NVIDIA TITANX GPU cards, resulting in a generation speed that is 1 order of magnitude slower than the traditional Monte Carlo methods, thus defeating the original purpose. Therefore, in our implementation of the Pixel CNN++ model, we limit the number of its filters to 20 to speed up the generation process and reduce the memory requirements.

TABLE VIII. Model complexity comparison in terms of number of parameters in the Jet substructure study.

Model	Number of parameters
PYTHIA [6]	...
Pixel CNN++	0.7×10^6
SARM-2 (D + D)	6×10^6
SARM-2 (D + C)	7×10^6
LAGAN	5×10^6

APPENDIX H: SAMPLE IMAGES

In this section, we show more generated images from both the jet substructure study and the muon isolation study in Figs. 22 and 23.

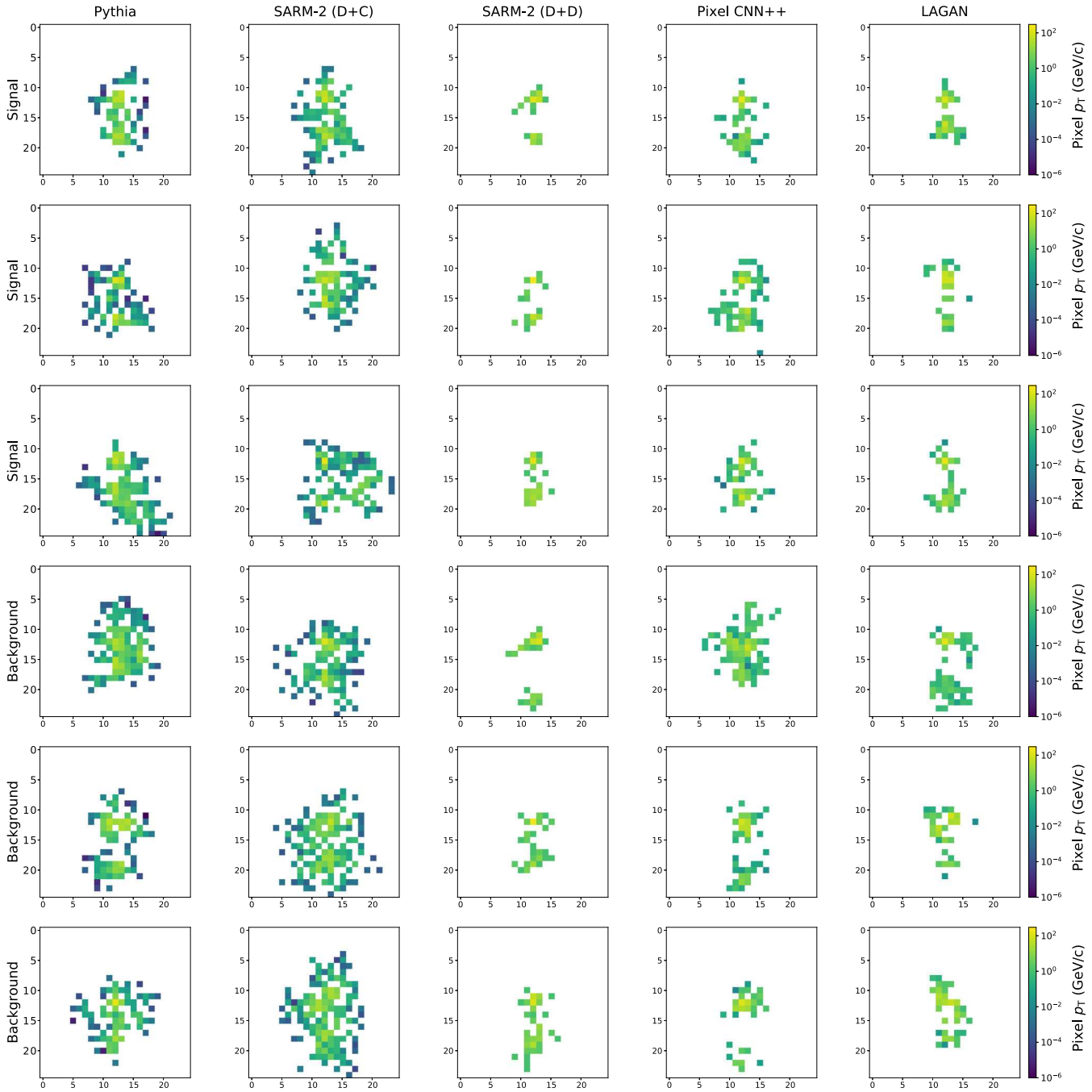

FIG. 22. Additional typical images from the jet substructure study.

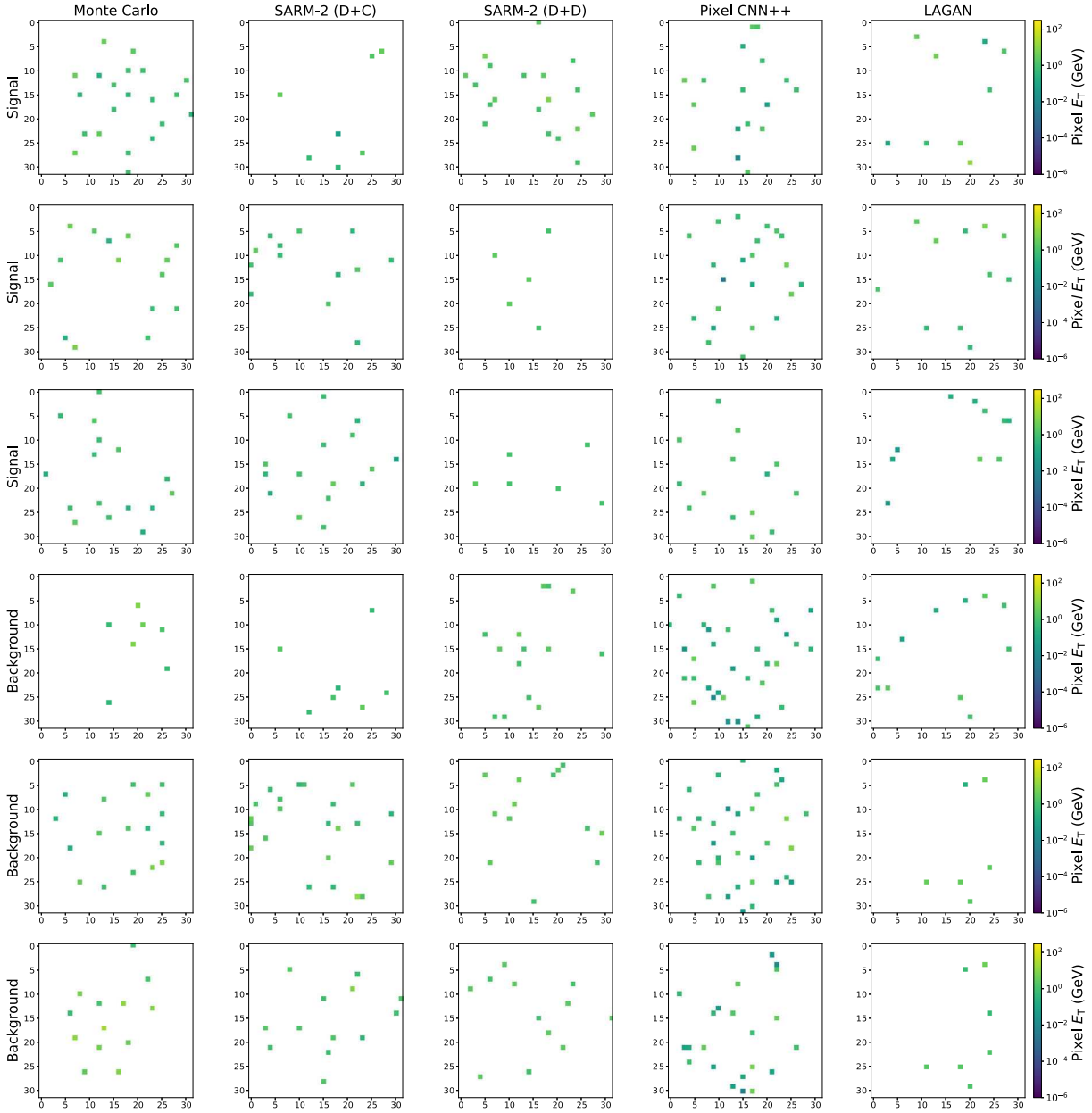

FIG. 23. Additional typical images from the muon isolation study.

[1] S. Agostinelli *et al.* (GEANT4 Collaboration), GEANT4: A simulation toolkit, Nucl. Instrum. Methods Phys. Res., Sect. A **506**, 250 (2003).

[2] G. Aad *et al.* (ATLAS Collaboration), The ATLAS simulation infrastructure, Eur. Phys. J. C **70**, 823 (2010).

[3] R. Rahmat, R. Kroeger, and A. Giammanco, The fast simulation of the CMS experiment, J. Phys. Conf. Ser. **396**, 062016 (2012).

[4] N. Nikiforou (ATLAS Collaboration), Performance of the ATLAS liquid argon calorimeter after three years of LHC operation and plans for a future upgrade, in *Proceedings of the 3rd International Conference on Advancements in Nuclear Instrumentation Measurement Methods and Their Applications (ANIMMA), Marseille, 2013* (IEEE, 2013), https://doi.org/10.1109/ANIMMA.2013.6728060.

[5] LHCb Collaboration, LHCb calorimeters, Technical Design Report No. LHCb-TDR-2 (CERN, Geneva, 2000), https://cds.cern.ch/record/494264.

[6] L. de Oliveira, M. Paganini, and B. Nachman, Learning particle physics by example: Location-aware generative adversarial networks for physics synthesis, Comput. Softw. Big Sci. **1**, 4 (2017).

[7] Y. Lu, J. Collado, K. Bauer, D. Whiteson, and P. Baldi, Sparse image generation with decoupled generative models, in *Proceedings of the 33rd Conference on Neural Information Processing Systems: Machine Learning and the Physical Sciences Workshop*, 2019, https://ml4physicalsciences.github.io/2019/files/NeurIPS_ML4PS_2019_161.pdf.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial networks, in *Advances in Neural Information Processing Systems 27*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Curran Associates, New York, 2014), https://proceedings.neurips.cc/paper/2014.

[9] D. P. Kingma and M. Welling, Auto-encoding variational Bayes, in *Proceedings of the 2nd International Conference on Learning Representations (ICLR), 2014*, arXiv:1312.6114.

[10] A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu, Pixel recurrent neural networks, in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research Vol. 48 (PMLR, New York, 2016), pp. 1747–1756.

[11] J. Zhu, T. Park, P. Isola, and A. A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, arXiv:1703.10593.

[12] A. Brock, J. Donahue, and K. Simonyan, Large scale GAN training for high fidelity natural image synthesis, in *Proceedings of the 7th International Conference on Learning Representations (ICLR), New Orleans, 2019*, https://openreview.net/forum?id=B1xsqj09Fm.

[13] D. P. Kingma and P. Dhariwal, Glow: Generative flow with invertible 1×1 convolutions, in *Advances in Neural Information Processing Systems 31*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, New York, 2018), pp. 10,215–10,224, https://proceedings.neurips.cc/paper/2018.

[14] P. Baldi, P. Sadowski, and D. Whiteson, Searching for exotic particles in high-energy physics with deep learning, Nat. Commun. **5**, 4308 (2014).

[15] S. Delaquis *et al.*, Deep neural networks for energy and position reconstruction in EXO-200, J. Instrum. **13**, P08023 (2015).

[16] C. Shimmin, P. Sadowski, P. Baldi, E. Weik, D. Whiteson, E. Goul, and A. Sgaard, Decorrelated jet substructure tagging using adversarial neural networks, Phys. Rev. D **96**, 074034 (2017).

[17] P. Baldi, J. Bian, L. Hertel, and L. Li, Improved energy reconstruction in NOvA with regression convolutional neural networks, Phys. Rev. D **99**, 012011 (2019).

[18] D. Guest, J. Collado, P. Baldi, S.-C. Hsu, G. Urban, and D. Whiteson, Jet flavor classification in high-energy physics with deep neural networks, Phys. Rev. D **94**, 112002 (2016).

[19] P. Sadowski, J. Collado, D. Whiteson, and P. Baldi, Deep learning, dark knowledge, and dark matter, in *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*, Proceedings of Machine Learning Research Vol. 42 (PMLR, Montreal, 2015), pp. 81–87, http://proceedings.mlr.press/v42/sado14.html.

[20] I. Seong, L. Hertel, J. Collado, L. Li, N. Nayak, J. Bian, and P. Baldi, Convolutional neural networks for energy and vertex reconstruction in DUNE, in *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS): Machine Learning and the Physical Sciences Workshop*, 2019, https://ml4physicalsciences.github.io/2019/files/NeurIPS_ML4PS_2019_77.pdf.

[21] P. Baldi, *Deep Learning in Science: Theory, Algorithms, and Applications* (Cambridge University Press, Cambridge, England, 2020).

[22] M. Mustafa, D. Bard, W. Bhimji, Z. Lukić, R. Al-Rfou, and J. M. Kratochvil, CosmoGAN: Creating high-fidelity weak lensing convergence maps using generative adversarial networks, Comput. Astrophys. Cosmol. **6**, 1 (2019).

[23] P. Musella and F. Pandolfi, Fast and accurate simulation of particle detectors using generative adversarial networks, Comput. Softw. Big Sci. **2**, 8 (2018).

[24] K. Zhou, G. Endrődi, L.-G. Pang, and H. Stöcker, Regressive and generative neural networks for scalar field theory, Phys. Rev. D **100**, 011501 (2019).

[25] G. r. Khattak, S. Vallecorsa, and F. Carminati, Three dimensional energy parametrized generative adversarial networks for electromagnetic shower simulation, in *Proceedings of the 25th IEEE International Conference on Image Processing (ICIP), Athens, 2018* (IEEE, New York, 2018), pp. 3913–3917, https://ieeexplore.ieee.org/document/8451587.

[26] S. Alonso Monsalve and L. Whitehead, Image-based model parameter optimization using model-assisted generative adversarial networks, IEEE Trans. Neural Networks Learn. Syst. **31**, 5645 (2020).

[27] K. Deja, T. Trzciński, and Ł. Graczykowski, Generative models for fast cluster simulations in the TPC for the ALICE experiment, EPJ Web Conf. **214**, 06003 (2019).

[28] F. Carminati, M. P. Gulrukh Khattak, B. H. Amir Farbin, W. Wei, M. Zhang, V. B. Pacela, S. Vallecorsafac, M. Spiropulu, and J.-R. Vlimant, Calorimetry with deep learning: Particle classification, energy regression, and simulation for high-energy physics, *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS): Deep Learning for Physical Sciences Workshop, Long Beach, CA, 2017* (Curran Associates, Red Hook, NY, 2017).

[29] V. Shah, A. Joshi, S. Ghosal, B. S. S. Pokuri, S. Sarkar, B. Ganapathysubramanian, and C. Hegde, Encoding invariances in deep generative models, arXiv:1906.01626.

[30] K. Cranmer, S. Gadatsch, A. Ghosh, T. Golling, D. R. Gilles Louppe, and D. Salamani (G. S. on behalf of the ATLAS Collaboration), Deep generative models for fast shower simulation in ATLAS, *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS): Bayesian Deep Learning Workshop, Montreal*, 2018, http://bayesiandeeplearning.org/2018/papers/24.pdf.

[31] B. Hashemi, N. Amin, K. Datta, D. Olivito, and M. Pierini, LHC analysis-specific datasets with generative adversarial networks, arXiv:1901.05282.
[32] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. R. de Austri, and R. Verheyen, Event generation and statistical sampling for physics with deep generative models and a density information buffer, arXiv:1901.00875.
[33] J. Cogan, M. Kagan, E. Strauss, and A. Schwartzman, Jet-images: Computer vision inspired techniques for jet tagging, J. High Energy Phys. 02 (2015) 118.
[34] M. Paganini, L. de Oliveira, and B. Nachman, CaloGAN: Simulating 3D high energy particle showers in multi-layer electromagnetic calorimeters with generative adversarial networks, Phys. Rev. D **97**, 014021 (2018).
[35] S. Chintala, How to train a GAN? in *Proceedings of the Workshop on Generative Adversarial Networks, Barcelona, 2016*, https://github.com/soumith/ganhacks.
[36] L. Bottou, Large-scale machine learning with stochastic gradient descent, in *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT), Paris*, 2010, https://leon.bottou.org/publications/pdf/compstat-2010.pdf.
[37] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, 2014*.
[38] M. Arjovsky and L. Bottou, Towards principled methods for training generative adversarial networks, in *Proceedings of the 5th International Conference on Learning Representations (ICLR), Toulon, France, 2017*, arXiv:1701.04862.
[39] V. Nagarajan and J. Z. Kolter, Gradient descent GAN optimization is locally stable, in *Advances in Neural Information Processing Systems 30* (Curran Associates, Inc., New York, 2017), pp. 5585–5595.
[40] A. Radford, L. Metz, and S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv:1511.06434.
[41] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, PixelCNN + +: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications, arXiv:1701.05517.
[42] L. G. Almeida, M. Backovi, M. Cliche, S. J. Lee, and M. Perelstein, Playing tag with ANN: Boosted top identification with pattern recognition, J. High Energy Phys. 07 (2015) 086.
[43] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, Jet-images deep learning edition, J. High Energy Phys. 07 (2016) 069.
[44] J. Barnard, E. N. Dawe, M. J. Dolan, and N. Rajcic, Parton shower uncertainties in jet substructure analyses with deep neural networks, Phys. Rev. D **95**, 014018 (2017).
[45] P. T. Komiske, E. M. Metodiev, and M. D. Schwartz, Deep learning in color: Towards automated quark/gluon jet discrimination, J. High Energy Phys. 01 (2017) 110.
[46] T. Sjostrand, S. Mrenna, and P. Z. Skands, PYTHIA6.4 physics and manual, J. High Energy Phys. 05 (2006) 026.
[47] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations, J. High Energy Phys. 07 (2014) 079.
[48] J. de Favereau *et al.* (DELPHES 3 Collaboration), DELPHES3, A modular framework for fast simulation of a generic collider experiment, J. High Energy Phys. 02 (2014) 057.
[49] H. Larochelle and I. Murray, The neural autoregressive distribution estimator, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research Vol. 15 (PMLR, Fort Lauderdale, 2011), pp. 29–37, http://proceedings.mlr.press/v15/larochelle11a/larochelle11a.pdf.
[50] M. Germain, K. Gregor, I. Murray, and H. Larochelle, MADE: Masked autoencoder for distribution estimation, arXiv:1502.03509.
[51] B. Uria, I. Murray, and H. Larochelle, RNADE: The real-valued neural autoregressive density-estimator, in *Advances in Neural Information Processing Systems 26* (Curran Associates, New York, 2013), pp. 2175–2183.
[52] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville, Neural autoregressive flows, in *Proceedings of the 35th International Conference on Machine Learning*, *Proceedings of Machine Learning Research* Vol. 80 (PMLR, Stockholm, 2018), pp. 2078–2087, http://proceedings.mlr.press/v80/huang18d/huang18d.pdf.
[53] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra, Deep autoregressive networks, in *Proceedings of the 31st International Conference on Machine Learning*, *Proceedings of Machine Learning Research* Vol. 32 (PMLR, Beijing, China, 2014), pp. 1242–1250.
[54] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, 2009* (IEEE, New York, 2009), http://www.image-net.org/papers/imagenet_cvpr09.
[55] Y. Rubner, C. Tomasi, and L. J. Guibas, The earth mover's distance as a metric for image retrieval, Int. J. Comput. Vis. **40**, 99 (2000).
[56] P. Baldi, K. Bauer, C. Eng, P. Sadowski, and D. Whiteson, Jet substructure classification in high-energy physics with deep neural networks, Phys. Rev. D **93**, 094034 (2016).
[57] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 2016* (IEEE, New York, 2016), pp. 770–778.
[58] A. Odena, V. Dumoulin, and C. Olah, Deconvolution and checkerboard artifacts, Distill, http://distill.pub/2016/deconv-checkerboard.
[59] See http://mlphysics.ics.uci.edu/.
[60] D. Hendrycks and K. Gimpel, Bridging nonlinearities and stochastic regularizers with Gaussian error linear units, arXiv:1606.08415.
[61] L. Hertel, J. Collado, P. Sadowski, J. Ott, and P. Baldi, Sherpa: Robust hyperparameter optimization for machine learning, SoftwareX **12**, 100591 (2020); software available at https://github.com/sherpa-ai/sherpa.
[62] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, Automatic differentiation in PyTorch, in *Proceedings of the NIPS Workshop on Autodiff, Long Beach, CA*, 2017, https://openreview.net/forum?id=BJJsrmfCZ.