TOWARDS DESIGNING AND EXPLOITING GENERATIVE NETWORKS FOR NEUTRINO PHYSICS EXPERIMENTS USING LIQUID ARGON TIME PROJECTION CHAMBERS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we show that a *hybrid* approach to generative modeling via combining the decoder from an autoencoder together with an explicit generative model for the latent space is a promising method for producing images of particle trajectories in a liquid argon time projection chamber (LArTPC). LArTPCs are a type of particle physics detector used by several current and future experiments focused on studies of the neutrino. We implement a Vector-Quantized Variational Autoencoder (VQ-VAE) and PixelCNN which produces images with LArTPC-like features and introduce a method to evaluate the quality of the images using a semantic segmentation that identifies important physics-based features.

1 INTRODUCTION

Liquid argon time projection chambers (LArTPC) (Rubbia (1977); Chen et al. (1976)) are a class of detector playing a prominent role in current and future experiments studying the neutrino, one of the fundamental particles (Amerio et al. (2004); Anderson et al. (2012); Acciarri et al. (2017a;a); Abi et al. (2018)). Through precision measurements of the behavior of the neutrino, the experiments aim to further our understanding of the physical laws that govern our universe. LArTPCs are an appealing choice of detector technology because they can be built to large sizes – kiloton-scale detectors with O(10 m) dimensions – while also economically instrumented to capture the features of charged particle trajectories at the O(mm) scale. This combination enables LArTPC experiments to record a large number of high resolution observations of neutrino interactions.

The data produced by LArTPCs can be naturally arranged into an image-like format which capture projections of particle trajectories. Charged particles traversing the detector create clouds of ionization electrons along their path. The amount of ionization created is proportional to the amount of energy lost by the particle. The pattern of ionization and the amount of energy lost in the detector depends on the particle type and momentum. This makes it possible to analyze the images and infer the sets of particles and their energies. Figure 1(b) shows some examples of images produced.

The need for efficient image analysis has motivated the the use of deep convolutional neural networks (LeCun et al. (1998); Krizhevsky et al. (2017)) to identify key features or objects within LArTPC images for physics analyses (Acciarri et al. (2017b); Collaboration et al. (2019); Abratenko et al. (2020a;b); Drielsma et al. (2020); Abi et al. (2020)). Recent efforts have focused primarily towards mapping an image or portions within it to quantities such as different classes of particle trajectories (Abratenko et al. (2020b)), categories of neutrino interactions (Abi et al. (2018)), or identify individual particles (Abratenko et al. (2020a)).

Less explored, however, are generative models for LArTPCs. The ultimate goal for these models would be to receive a list of particles, their position, and momentum and produce an image containing their trajectories through the detector, thereby providing a faster alternative to the detailed physics-based simulation of the detector. There have been some efforts in producing particle physics data via generative networks. In Alonso-Monsalve & Whitehead (2020) images with tracks, similar to what might be found in a LArTPC, are generated through the help of an explicit physics model. Nevertheless, this preliminary approach is not suitable towards capturing the rich shower-like patterns, varying track-like, and mixed patterns that are present in LArTPC images. To this end we

Popular Models/Methods	Type
Normalizing Flows Kobyzev et al. (2020, To Appear); Papamakarios et al. (2019)	Explicit
Pixel-CNN Van den Oord et al. (2016); Salimans et al. (2017)	Explicit
Variational Auto-Encoders (VAEs) Kingma & Welling (2019); Bousquet et al. (2017)	Implicit
Generative Adversarial Networks Goodfellow et al. (2014); Arjovsky et al. (2017) Gulrajani et al. (2017); Li et al. (2017); An et al. (2019)	Implicit
Vector Quantized-VAE, Probabilistic AE (PAE) Van Den Oord et al. (2017); Böhm & Seljak (2020)	Explicit Latent + Implicit Decoder

Table 1: Table summarizing popular approaches for generative modeling. To emphasize the pertinent differences, we categorize into explicit, implicit, and hybrid models. Hybrid models utilize explicit model to generate a low-dimensional latent with an implicit model that comes from the decoder of an autoencoder.

revisit the recent developments in generative modeling and argue for a particular type of model that exhibits promising behavior.

There are two ways to specify a probability distribution, viz., explicit vs implicit. In explicit models, a parametric form of distribution is specified, say $\mathsf{P}_{\boldsymbol{X}}(\boldsymbol{x};\Theta)$, where Θ is set of parameters. In implicit models, the main idea is that if a random variable $\boldsymbol{X}$ has distribution P, this distribution is implicitly specified via a transformation. That is, $\boldsymbol{X} = G(\boldsymbol{Z})$ where G is a map and $\boldsymbol{Z} \sim \mathsf{P}_{\boldsymbol{Z}}$. Given $G, \mathsf{P}_{\boldsymbol{Z}}$ it is possible to *compute* $\mathsf{P}_{\boldsymbol{X}}$, but it is hard when G is complex, say a deep neural network, and especially when $\mathsf{P}_{\boldsymbol{Z}}$ is assumed to be *simple* and lower-dimensional compared to $\boldsymbol{X}$. On the other hand, this allows one to generate IID *samples* from $\mathsf{P}_{\boldsymbol{X}}$ via IID samples from $P_{\boldsymbol{Z}}$. Hence the name generative modeling. Table 1 provides a rough and apologetically incomplete (for lack of space) literature survey in this context. The main point we want to highlight here is that hybrid models may behave better towards modeling images from LArTPC experiments compared to fully implicit or fully explicit models. We single out the Vector Qunatized(VQ)-VAE Van Den Oord et al. (2017) and Probabilistic Autoencoder (PAE) Böhm & Seljak (2020) as two recent models that are combine implicit modeling with an explicit model towards an overall generative network. Between VQ-VAE and PAE, the main difference is in the way the latent space is regularized. The latent space in VQ-VAE consists of a *finite* set of quantization points, while in PAE it is a continuous subset of $\mathbf{R}^d$. Of these two, in this paper we work with VQ-VAE since the pixel CNN approach that explicitly models the quantized latent space, in spirit, also models the time evolution a particle trajectory through the detector. A full comparison between the two and various tradeoffs is on-going and will be reported in a future manuscript.

Why generative models for LArTPCs? - Generative networks enable computationally efficient means to generate trajectory examples, thereby bypassing and/or compliment the traditional simulation chain consisting of particle transport and detector signal modeling. This would make it easier to meet the demand for example data required by physics analyses.

Generative models also open the path towards a complimentary approach to event reconstruction. With models that can produce trajectory examples, conditional on parameters such as momentum, one can extract quantities like momenta or particle ID by comparing generated images produced by different physical parameters and choosing the best match by a likelihood function or possibly a learned loss function implemented via a neural network. One would iterate until new hypotheses fail to improve the loss. Concretely, given a data image $\boldsymbol{d}$ and a set of generative models $G_p, p = 1, .., P$, one approach inspired by recent use of deep networks in inverse problems Bora et al. (2017); Jalal et al. (2020) would be to solve for,

$$\hat{p} = \arg\min_{p} \left\{ \min_{\boldsymbol{z}} \|G_p(\boldsymbol{z}) - \boldsymbol{d}\|^2 + \lambda \log \mathsf{P}_{\boldsymbol{Z}}(\boldsymbol{z}) \right\} \tag{1}$$

Another motivation for studying generative networks is understanding ways to represent the data that enable different applications. For example, developing good representations of the data can lead to a compression scheme with tolerable losses. A tolerable level of mistakes in a compression algorithm might be defined to be the same level of changes to the raw wire signals coming from the

range of kernel parameters choices for deconvolving wire signals. If a compression scheme can be achieved, this alleviates IO bottle necks in executing physics analyses on the large data sets produced by neutrino experiments.

2 METHODS

The training of a the generative model proceeds in two phases and follows the work in Van Den Oord et al. (2017). The first phase is to train a VQ-VAE network to properly reconstruct images. Through this process, the VQ-VAE network learns a map from detector images to a latent "code" image where each pixel is assigned the index of one of k-vectors in a d-dimensional feature embedding space. In the next phase, a set of training images are mapped into a code image. A PixelCNN network Salimans et al. (2017) is then trained to learn the prior over the latent code indices of these images. Once trained, the PixelCNN can be used to generate a novel code image. This code image is then passed into the decoder of the VQ-VAE in order to generate a detector image. For the training data, we used publicly available examples of LArTPC images[1]. The images contain trajectories from one of five possible particle species: e^-, γ, μ^-, π^+, or proton (p). Please see Section A.1 in the supplement for details of the implementation.

In order to provide a measure of image quality, we studied the output of a semantic segmentation network (SSNet) trained to classify individual pixels as examples from one of two categories: track or shower. These categories come from physics of how particles travel through matter. Heavier charged particles, which include the pion, proton, and muon, travel primarily along a linear path often referred to as a "track." Electrons, which have a much smaller mass, are more easily deflected by electromagnetic (EM) interactions with atoms. Furthermore, EM interactions can induce the creation of photons which, being electrically neutral, do not produce a visible trajectory for some distance before possibly interacting and producing a new electron or an electron-positron pair. This can repeat and result in a cascade of trajectories referred to as an EM shower, or "shower". Track and shower labels are useful to LArTPC analyses and already some form of this network is currently in use by experiments. Therefore, we use the similarity in the SSNet output between real and generated images as a proxy for how well the generative model can reproduce LArTPC image features and thus a proxy of the "visual quality" of generated images. We implement a network based on the work in Abratenko et al. (2020b). Details of this implementation can be found in Section A.5 of the supplement.

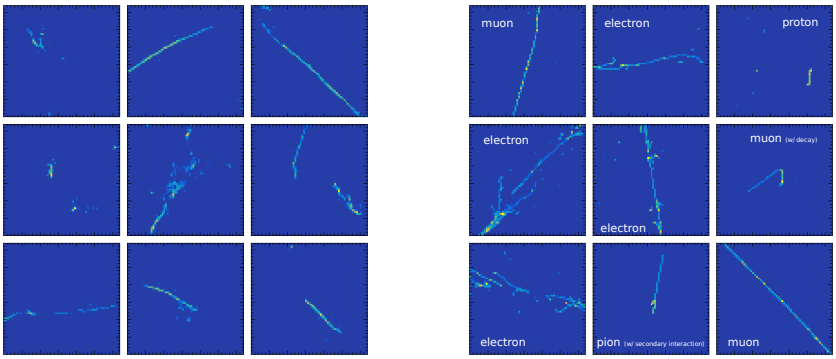
3 RESULTS

Images generated by the PixelCNN+VQ-VAE decoder contain patterns of charge that resemble both track- and shower-type trajectories in LArTPCs. Figure 1 provide examples of both generated and training images. There are more samples in Appendix D. Visual inspection leads to the following observations. Local patches of the generated images are beginning to resemble those from LArTPCs. In particular, "v"-shape or branching structures characteristic of shower trajectories are visible. Extended lines without branching, characteristic of "tracks", are also observed. Isolated shower-like trajectories we believe could fool an expert. However, taken as a whole, the generated trajectories still have clear flaws. Overall, there seems to be a bias towards producing shower-like features. The network, furthermore, has trouble producing extended structure for longer tracks and larger showers. Tracks seem to be shorter in the generated than test images. Larger shower-like regions often do not exhibit the structure one might expect.

We compliment visual inspection with a study of a track-shower pixel labeling network. Figure 2 shows the fraction of pixels with a given label score for both generated and test images. Ideally, if the generated images were indistinguishable from the test images, there would be little different in the histograms. We see, however, an increased number of pixels labeled as shower-like in the generated images and a relative deficit of pixels categorized as track-like. This correlates with what was visually observed. We also compared two configurations where the number of quantized vectors was halved to 256. We found the visual quality to be reduced. This was corroborated in the SSNet

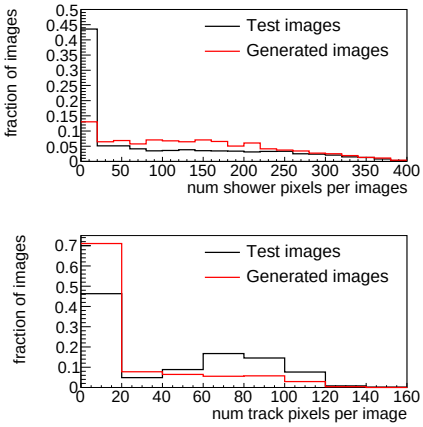
[1]Identifying the source of the images would likely unblind the authors and so is withheld but will be provided in a future version of this manuscript.

scores through: 1) less track pixels per image, 2) a further excess in shower pixels, and 3) a larger population of lower confidence shower pixels.

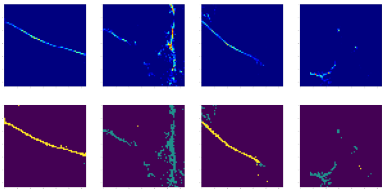


(a) Generated images (b) Training images

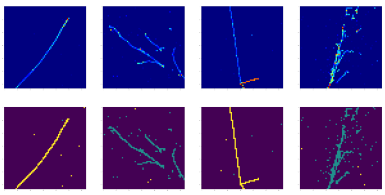
Figure 1: Examples of images generated by a network (left) and from the training data set (right).



(a) Comparison of track-shower pixels per images



(b) Track/shower labels on generated images



(c) Track/shower labels on test images

Figure 2: Results from studies using track-shower semantic segmentation network. (a) Comparison of the number of labeled shower (top) and track (bottom) pixels per image for generated (red) and test (black) images. Examples of labels on (b) generated images and (c) test images. For both, there is an input image (top row) and corresponding label image (bottom row). The label image indicates one of three classes: background (dark purple), track (yellow), and shower (cyan).

4 Conclusions

We present work towards a generative model which can produce convincing LArTPC images. As far as we know, this is the first demonstration of a generative network that produces shower trajectories and the first that produces track trajectories without an underlying physics-based model. The model produces patterns that do resemble track and shower trajectories, but there is clear need for improvement. Primarily, features at larger scales are difficult for the network. There are avenues, such as the use of hierarchical code maps at different image scales in Razavi et al. (2019), that could improve these. Furthermore, the way that the PixelCNN calculates the probabilities can better capture the time-evolution of particle trajectories using tools such as those in Jain et al. (2020), e.g. convolutions proceed from the center out rather in the current raster-scan order. We also plan to move towards conditional generation where the particle species and momenta can be specified. This gets us closer towards applying these models to event reconstruction. Finally, the types of features found in LArTPC images have a different nature than the common data sets that the machine learning community develops on. This, we believe, makes LArTPC images an interesting data set for studying different approaches in generative modeling.

REFERENCES

B Abi, R Acciarri, MA Acero, M Adamowski, C Adams, D Adams, P Adamson, M Adinolfi, Z Ahmad, CH Albright, et al. The DUNE far detector interim design report volume 1: Physics, technology and strategies. *arXiv:1807.10334*, 2018.

B Abi, R Acciarri, MA Acero, G Adamov, D Adams, M Adinolfi, Z Ahmad, J Ahmed, T Alion, S Alonso Monsalve, et al. Neutrino interaction classification with a convolutional neural network in the dune far detector. *Physical Review D*, 102(9):092003, 2020.

P Abratenko, M Alrashed, R An, J Anthony, J Asaadi, A Ashkenazi, S Balasubramanian, B Baller, C Barnes, G Barr, et al. A convolutional neural network for multiple particle identification in the microboone liquid argon time projection chamber. *arXiv preprint arXiv:2010.08653 (Submitted to PRD)*, 2020a.

P Abratenko, M Alrashed, R An, J Anthony, J Asaadi, A Ashkenazi, S Balasubramanian, B Baller, C Barnes, G Barr, et al. Semantic segmentation with a sparse convolutional neural network for event reconstruction in microboone. *arXiv preprint arXiv:2012.08513*, 2020b.

R Acciarri, C Adams, R An, A Aparicio, S Aponte, J Asaadi, M Auger, N Ayoub, L Bagby, B Baller, et al. Design and construction of the MicroBooNE detector. *Journal of Instrumentation*, 12(02): P02017, 2017a.

R Acciarri, C Adams, R An, J Asaadi, M Auger, L Bagby, B Baller, G Barr, M Bass, F Bay, et al. Convolutional neural networks applied to neutrino events in a liquid argon time projection chamber. *Journal of instrumentation*, 12(03):P03011, 2017b.

Saúl Alonso-Monsalve and Leigh H Whitehead. Image-based model parameter optimization using model-assisted generative adversarial networks. *IEEE transactions on neural networks and learning systems*, 31(12):5645–5650, 2020.

S Amerio, S Amoruso, M Antonello, P Aprili, M Armenante, F Arneodo, A Badertscher, B Baiboussinov, M Baldo Ceolin, G Battistoni, et al. Design, construction and tests of the ICARUS T600 detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 527(3):329–410, 2004.

Dongsheng An, Yang Guo, Na Lei, Zhongxuan Luo, Shing-Tung Yau, and Xianfeng Gu. Ae-ot: a new generative model based on extended semi-discrete optimal transport. *ICLR 2020*, 2019.

C Anderson, M Antonello, B Baller, T Bolton, C Bromberg, F Cavanna, E Church, D Edmunds, Antonio Ereditato, S Farooq, et al. The ArgoNeuT detector in the NuMI low-energy beam line at fermilab. *Journal of Instrumentation*, 7(10):P10019, 2012.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Vanessa Böhm and Uroš Seljak. Probabilistic auto-encoder. *arXiv preprint arXiv:2006.05479*, 2020.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 537–546, 2017.

Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl-Johann Simon-Gabriel, and Bernhard Schoelkopf. From optimal transport to generative modeling: the vegan cookbook. *arXiv preprint arXiv:1705.07642*, 2017.

H.H. Chen, P.E. Condon, B.C. Barish, and F.J. Sciulli. Fermilab proposal p-496. Technical report, 1976.

MicroBooNE Collaboration, C Adams, M Alrashed, R An, J Anthony, J Asaadi, A Ashkenazi, M Auger, S Balasubramanian, B Baller, et al. Deep neural network for pixel-level electromagnetic particle identification in the microboone liquid argon time projection chamber. *Physical Review D*, 99(9):092001, 2019.

DeepLearnPhysics Collaboration. Deep learn physics open data, 2017a. URL http://deeplearnphysics.org/DataChallenge/.

DeepLearnPhysics Collaboration. Browsing classification data set (v0.1.0), 2017b. URL http://deeplearnphysics.org/Blog/2017-12-29-BrowsingClassificationData_v0.1.0.html#2017-12-29-BrowsingClassificationData_v0.1.0.

François Drielsma, Laura Dominé, Dae Heun Koh, and Kazuhiro Terao. Data reconstruction using deep neural networks for particle imaging neutrino detectors. 2020.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Ajay Jain, Pieter Abbeel, and Deepak Pathak. Locally masked convolution for autoregressive models. In *Conference on Uncertainty in Artificial Intelligence*, pp. 1358–1367. PMLR, 2020.

Ajil Jalal, Liu Liu, Alexandros G Dimakis, and Constantine Caramanis. Robust compressed sensing using generative models. *Advances in Neural Information Processing Systems*, 33, 2020.

Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

I. Kobyzev, S. Prince, and M. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020, To Appear. doi: 10.1109/TPAMI.2020.2992934.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *NIPS*, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv e-prints*, pp. arXiv–1912, 2019.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems*, pp. 14866–14876, 2019.

Carlo Rubbia. The liquid-argon time projection chamber: a new concept for neutrino detectors. Technical report, 1977.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with PixelCNN decoders. *Advances in neural information processing systems*, 29:4790–4798, 2016.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6306–6315, 2017.

SUPPLEMENTARY to "Towards Designing and Exploiting Generative Networks for Neutrino Physics Experiments using Liquid Argon Time Projection Chambers."

A Additional details on Methods

A.1 VQ-VAE Implementation Details

The VQVAE's structure is best explained as a two-part network. The first network is much like a classical autoencoder, the caveat being that vector quantization is applied to the latent codes, and that extra losses are computed to optimize the quantization. The second network is an autoregressive generative model that generates latent vectors one component at a time.

For the encoder and decoder, we use convolutional and deconvolutional neural networks respectively. The CNN's are arranged into blocks. Each block contains a convolutional layer with ReLU activation, a batch-normalization layer, and finally another ReLU activation. Our experience has been that ReLU-BN-ReLU blocks produce sets of quantization vectors that lead to more realistic generation (compared to just BN-ReLU). This should be investigated in future work. In the convolutional encoder, all blocks except the last downsample the input image by a factor of two. In the deconvolutional decoder, all blocks except the first upsample the image by a factor of two. Encoder outputs are fed through a vector quantization layer before being passed to the decoder.

For the autoregressive model used to model the distribution of quantization vectors over the latent vector, we employ a masked and gated PixelCNN model.

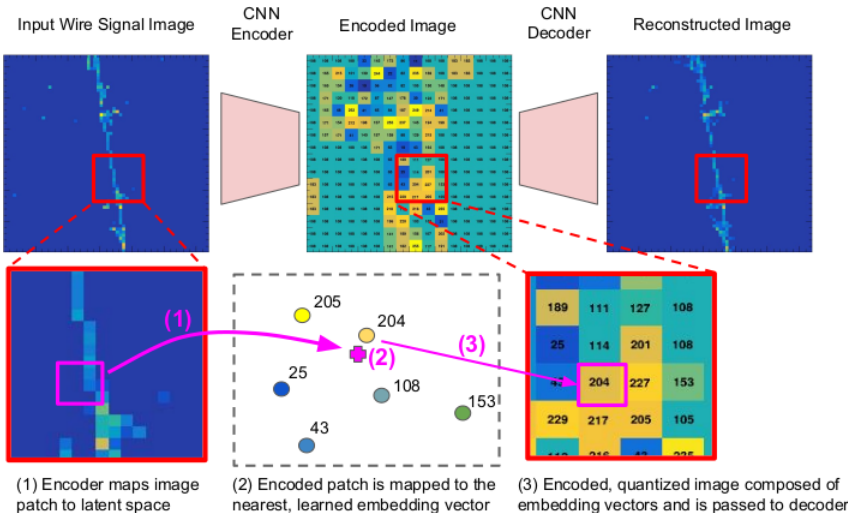


Figure 3: Illustration of image encoding and decoding by the Vector-Quantizied Variational Autoencoder (VQ-VAE) network. The VQ-VAE encoder maps patches of the image to an embedding vector space. The values of this vector are then assigned to the values of the nearest quantized-vector. The values of the k quantized vectors are learned. In this diagram, the index of the assigned quantized vector is displayed. The quantized feature tensor is then passed into the decoder, which reconstructs the image.

A.2 PixelCNN Implementation Details

Our PixelCNN network is composed of six gated, masked, convolutional blocks. Each block is composed of a horizontal and vertical stack. Each stack is composed of masked convolution, a gating layer, and a residual layer. Information from the vertical stack is passed to the horizontal stack. For the vertical stack, gating occurs after the residual layer, while in the horizontal stack gating occurs first.

The convolutional blocks are followed by two convolutional layers with an amount of output filters equal to the number of quantization vectors. The activations are passed into a Softmax function to approximate the likelihood of each quantization vector at that component of the latent code.

Our VQVAE implementation is based on the works of Ken Leidal and Amelie Royer, which can be found at `https://github.com/kkleidal/GatedPixelCNNPyTorch` and `https://github.com/ameroyer/ameroyer.github.io` respectively.

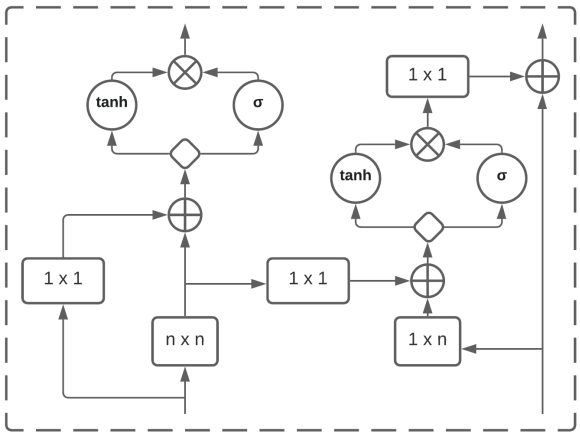


Figure 4: PixelCNN block – rectangles represent convolutions, diamonds represent split operations on incoming filters. The left input is the vertical stack and begins with an nxn masked convolution. The right input is the horizontal stack and begins with a $1xn$ masked convolution.

A.3 Training Data Details

We used publicly available examples of LArTPC images produced by the DeepLearnPhysics collaboration DeepLearnPhysics Collaboration (2017a). Among the available data from this source, we used the 50k single-particle image data set to train the VQ-VAE. We use the separate 40K single-particle image data set to test the reconstructions of the VQ-VAE. Images in both the train and test set consist of a 256x256 image DeepLearnPhysics Collaboration (2017b). The particle generated for each image is chosen among five species: e^-, γ, μ^-, π^+, and protons (p). The momentum of each particle is chosen from a uniform distribution between the following ranges (e^-) 35.5 to 800 MeV/c, (γ) 35 to 800 MeV/c, (μ^-) 90 to 800 MeV/c, (π^+) 105 to 800 MeV/c, and (p) 105 to 800 MeV/c. The particle is simulated inside a large volume of argon. It is propagated via Geant4. Afterwards, a 2.56 m^3 box is chosen in 3D that maximizes the particle's trajectory within the volume and recorded in the file. The 2D images are created as 2D projections (xy, yz, zx) of the 3D charge depositions. All images included in the data set are guaranteed to have 2D projection images with at least 10 non-zero pixels. We use the zx projections.

A.4 Generator Model Training Procedure Details

Num. Quantization Vectors	Quant. Vector Dim.	Enc. Filters	Dec. Filters.	Batch Size
512	8	[16, 32]	[32, 16]	512

Table 2: VQVAE Meta Parameters

Num. PixelCNN Blocks	Num. Filters per Block	Batch Size
6	128	512

Table 3: PixelCNN Meta Parameters

We train the network in two parts. First, the autoencoder is trained on a set of fifty thousand, 64x64, single channel LArTPC images. Training is conducted with a batch size of 512 images and with the Adam optimizer initialized with a learning rate of 3e-4.

For a single training loop, three losses are computed. First is the mean squared error between the reconstructed image and the true image. This loss is propagated through the decoder and encoder. Because the vector quantization process contains an argmin operation, which gradient can not pass through, gradients are copied from the beginning of the first layer of the decoder to the last layer of the encoder.

Bypassing the quantization layer means that the codebook must be learned independently of the reconstruction error. We use an L2 loss between the encoder's unquantized outputs and quantized outputs (with a stop-gradient operation applied to the unquantized outputs) to learn the quantization vectors. To ensure that the encoder commits to a specific set of quantization vectors, a second L2 loss is introduced between the unquantized outputs and the quantized outputs, with the stop-gradient being applied this time to the quantized outputs.

The PixelCNN network is trained using the cross entropy loss between the quantization vector predictions and the true quantization vector. We use the Adam optimizer and initialize it with a learning rate of 1e-3.

A.5 Training of the Track/Shower labeling model

The quality of the generated images is quantified using a convolutional semantic segmentation network (SSNet) modeled after the network described in Abratenko et al. (2020b). The architecture of the track/shower semantic segmentation network is the same in Abratenko et al. (2020b) with the one exception being that dense convolutions are used instead of sparse submanifold convolutions. The network is structured as a U-Net with ResNet layers. Four pairs of down-sampling and up-sampling layers are combined before a convolution layer outputs three classes: background, shower, and track.

The images used to train the track/shower network is related to the data used to train the generative network. The two data sets were created using the same traditional simulation chain. The track/shower data set is different because it contains the pixel-wise truth labels. The training sample consisted of 15k 256x256 examples. A test set with 10k was used to monitor the training of the network for over-fitting. The network was trained using Adam with a batch size of 16, momentum of 0.9, and a weight-decay of 1.0e-4. The network was trained for 20 epochs with a starting learning rate of 1.0e-3. The learning rate was cut in half every 4 epochs. No significant difference in train versus test set loss and accuracy was seen, so the last saved checkpoint is used for the studies in this work.

Data augmentation techniques were applied to the training set. This included flipping the image along the horizontal and vertical axis, transposing the image, and scaling the pixel values across an individual images by a random value between 0.90 and 1.1 drawn uniformly. The pixel-wise classification accuracy after training was 98.5% per pixel.

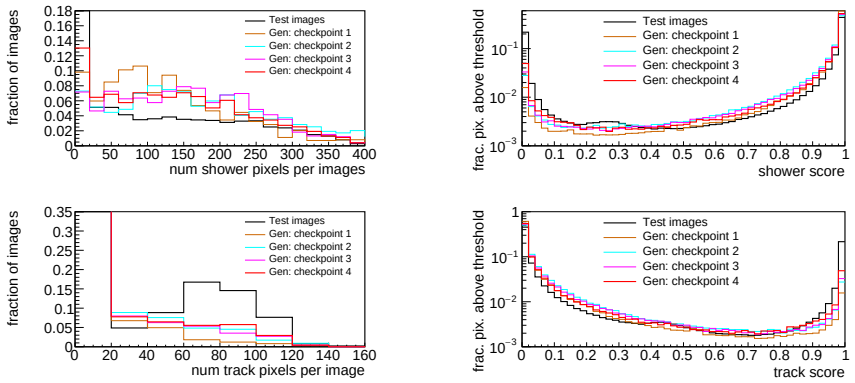
B Additional SSNet Study

As described in the main text, the output of SSNet is used as a metric to evaluate and compare image quality. One validation study we did for this metric was to compare the SSNet output on generated samples produced by the same model after several epochs of training. Our expectation is that the comparison to the output on test images should improve with increasing epoch. Figure 5 shows two distributions. The first is a distribution of the number of pixels labeled as shower or track per image. The second is a distribution of the class score for pixels above threshold. The first tells us how often pixels within patches with shower-like or track-like features are produced. The second tells us how confident the network is that the pixel is part of a region that matches track or shower features. Both plots compare distributions computed for generated images for three successive PixelCNN training checkpoints to the distribution computed over test images (i.e. not generated). We find that the generated distributions better match the test distribution as the model is trained longer. In

Checkpoint	Num. Shower Pix.	Shower Score	Num. Track Pix.	Track Score
#1: 100 epochs	0.36	0.22	0.41	0.21
#2: 200 epochs	0.42	0.19	0.22	0.19
#3: 300 epochs	0.37	0.17	0.22	0.17
#4: 400 epochs	0.24	0.13	0.18	0.13

Table 4: KL-divergence between SSNet output distributions calculated from generated images versus test images. Smaller values are better. This table shows the metric for successive PixelCNN training checkpoints.

order to provide a score to see improvement, we calculate the KL divergence between the generated distributions and the test distribution.



(a) Comparing frequency of ssnet labels per image

(b) Comparing distribution of class scores for above threshold pixels

Figure 5: SSNet output versus training epoch.

C PUBLICLY AVAILABLE CODE, MODELS, AND GENERATED IMAGE SETS

The code implementing all the models discussed here can be found on github. Model weights for the VQ-VAE, PixelCNN, and track/shower semantic segmentation network can be found on Zenodo. A sample of generated images are also provided on Zenodo. URLs are not given here in order to respect the double-blind review process, but will be made available after the review.

D ADDITIONAL EXAMPLE IMAGES

In this section, we provide additional image samples to view.

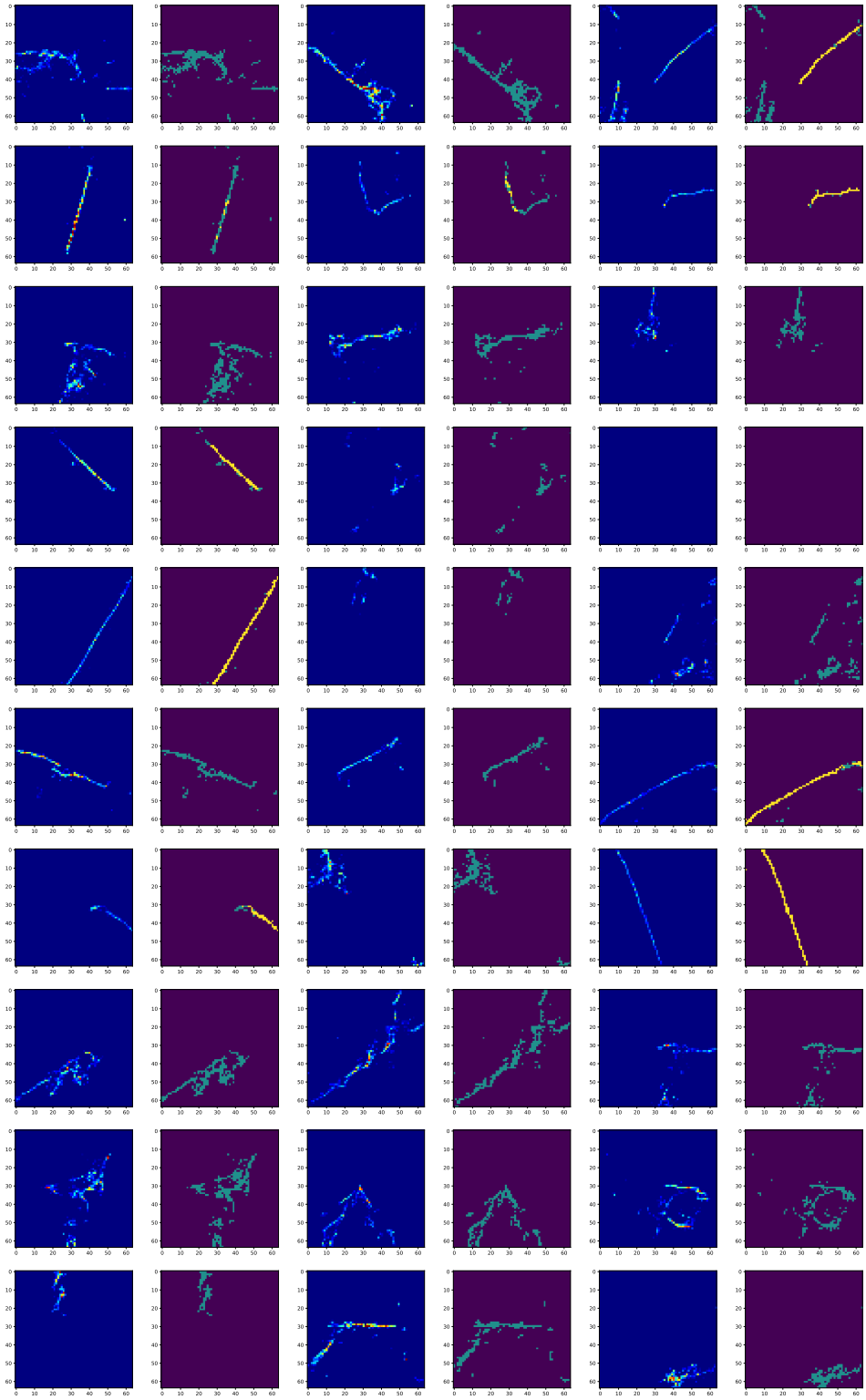


Figure 6: Randomly selected examples showing the labels provided by a track-shower segmentation network on *generated* images. There are 30 image pairs in total. For each pair, the image to the left displays the pixel values; the image to the right indicates the class with the largest score as calculated by the track/shower semantic segmentation network (SSNet): background (dark purple), track (yellow), and shower (cyan).

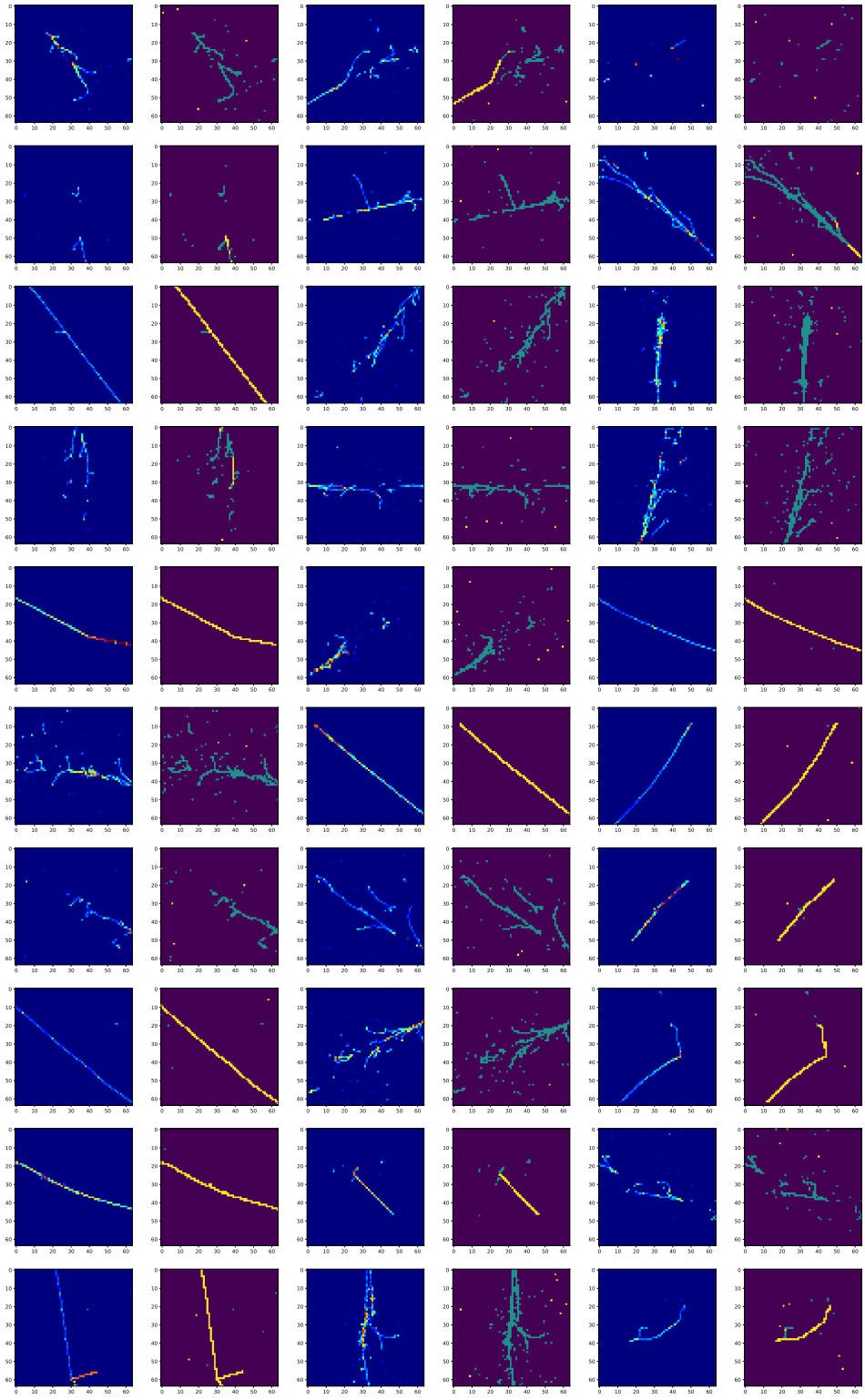


Figure 7: Randomly selected examples showing the labels predicted by a track-shower segmentation network on test images (non-generated) for the VQ-VAE network. There are 30 image pairs in total. For each pair, the image to the left displays the pixel values; the image to the right indicates the class with the largest score as calculated by the track/shower semantic segmentation network: background (dark purple), track (yellow), and shower (cyan).