# Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics

Dawit Belayneh[1], Federico Carminati[2], Amir Farbin[3], Benjamin Hooberman[4], Gulrukh Khattak[25], Miaoyuan Liu[6], Junze Liu[4], Dominick Olivito[7], Vitória Barin Pacela[8], Maurizio Pierini[2], Alexander Schwing[4], Maria Spiropulu[9], Sofia Vallecorsa[2], Jean-Roch Vlimant[9], Wei Wei[4], and Matt Zhang[a4]

[1] Univ. of Chicago
[2] European Organization for Nuclear Research (CERN)
[3] Univ. of Texas Arlington
[4] Univ. of Illinois at Urbana-Champaign
[5] UET Peshawar
[6] Fermi National Accelerator Laboratory
[7] Univ. of California, San Diego
[8] Univ. of Helsinki
[9] California Institute of Technology

**Abstract.** Using detailed simulations of calorimeter showers as training data, we investigate the use of deep learning algorithms for the simulation and reconstruction of particles produced in high-energy physics collisions. We train neural networks on shower data at the calorimeter-cell level, and show significant improvements for simulation and reconstruction when using these networks compared to methods which rely on currently-used state-of-the-art algorithms. We define two models: an end-to-end reconstruction network which performs simultaneous particle identification and energy regression of particles when given calorimeter shower data, and a generative network which can provide reasonable modeling of calorimeter showers for different particle types at specified angles and energies. We investigate the optimization of our models with hyperparameter scans. Furthermore, we demonstrate the applicability of the reconstruction model to shower inputs from other detector geometries, specifically ATLAS-like and CMS-like geometries. These networks can serve as fast and computationally light methods for particle shower simulation and reconstruction for current and future experiments at particle colliders.

## 1 Overview

In high energy physics (HEP) experiments, detectors act as imaging devices, allowing physicists to take snapshots of decay products from particle collision "events". Calorimeters are key components of such detectors. When a high-energy primary particle travels through dense calorimeter material, it deposits its energy and produces a shower of secondary particles. Detector "cells" within the calorimeter then capture these energy depositions, forming a set of voxelized images which are characteristic of the type and energy of the primary particle.

The starting point of any physics analysis is the identification of the types of particles produced in each collision and the measurement of the momentum carried by each of these particles. These tasks have traditionally used manually-designed algorithms, producing measurements of physical features such as shower width and rate of energy loss for particles traversing calorimeter layers. In the last few years, researchers have started realizing that machine learning (ML) techniques are well suited for such tasks, e.g. using boosted decision trees (BDTs) on calculated features for doing particle classification. Indeed, ML has long been applied to various other tasks in HEP [1,2,3], including the 2012 discovery of the Higgs boson [4,5] at the ATLAS [6] and CMS [7] experiments at the Large Hadron Collider (LHC).

In the next decade, the planned High Luminosity Large Hadron Collider (HL-LHC) upgrade [8] will enhance the experimental sensitivity to rare phenomena by increasing the number of collected proton-proton collisions by a factor of ten. In addition, many next-generation detector components, such as the sampling calorimeters proposed for the ILC [9], CLIC [10], and CMS [11] detectors, will improve physicists' ability to identify and measure particles by using much finer 3D arrays of voxels. These and future accelerator upgrades will lead to higher data volumes and pose a variety of technological and computational challenges in tasks such as real-time particle reconstruction.

[a] corresponding author, mzhang60@illinois.edu

In addition to actual collision data, physics analyses typically require extremely detailed and precise simulations of detector data, generated using software packages such as GEANT4[12]. These simulations are used to develop and test analysis techniques. They rely on calculations of the micro-physics governing the interaction of particles with matter, and are generally very CPU intensive. In some cases, such as the ATLAS experiment, simulation currently requires roughly half of the experiment's computing resources[13]. This fraction is expected to increase significantly for the HL-LHC. These challenges require novel computational and algorithmic techniques, which has prompted recent efforts in HEP to apply modern ML to calorimetry [14,15,16,17].

With this work, we aim to demonstrate the applicability of neural-network based approaches to reconstruction and simulation tasks, looking at a real use case. To do this, we use fully simulated calorimeter data for a typical collider detector to train two models: (i) a network for end-to-end particle reconstruction, receiving as input a particle shower and acting both as a particle identification algorithm and as a regression algorithm for the particle's energy; (ii) a generative adversarial network (GAN) [18] for simulating particle showers, design to return calorimeter-cell voxelized images like those generated buy GEANT4. Both models aim to preserve the accuracy of more traditional approaches while drastically reducing the required computing resources and time, thanks partly to a built-in portability to heterogeneous CPU+GPU computing environments.

This paper is a legacy document summarizing two years of work. It builds upon initial simulation, classification, and regression results which we presented at the 2017 NeurIPS conference. Those results were derived using simplified problem formulations [19]. For instance, we only used particles of a single fixed energy for classification, and had only considered showers produced by particles traveling perpendicularly to the calorimeter surface. The results presented in this paper deal with a more realistic use case and supersede the results in Ref. [19].

For the studies presented in this paper, we used two computing clusters: at the University of Texas at Arlington (UTA), and at the Blue Waters supercomputing network, located at the University of Illinois at Urbana Champaign (UIUC). The UTA cluster has 10 GTX Titan GPU's with 6 GB of memory each. Blue Waters uses Nvidia Kepler GPU's, also with 6 GB of memory each.

GAN models were implemented and trained using Keras [20] and Tensorflow [21]. Reconstruction models were implemented and trained using PyTorch [22]. The sample generation and training frameworks were both written in Python, with the sample generation codebase at `https://github.com/UTA-HEP-Computing/CaloSampleGeneration` and the TriForce reconstruction training framework at `https://github.com/BucketOfFish/Triforce_CaloML`.

This document is structured as follows: In Section 2, we describe how we created and prepared the data used in these studies. Sections 3 introduces the two physics problems, particle simulation and reconstruction. Sections 4 and 5 describe the corresponding models, how they were trained, and the performances they reached. In particular, Section 5 compares our results to those of more traditional approaches, and also extends those comparisons to simulated performances on detector geometries similar to those of the ATLAS and CMS calorimeters. Conclusions are given in Section 6.

## 2 Dataset

This study is based on simulated data produced with GEANT4 [12], using the geometric layout of the proposed Linear Collider Detector (LCD) for the CLIC accelerator [23]. We limit the study to the central region (barrel) of the LCD detector, where the electromagnetic calorimeter (ECAL) consists of a cylinder with inner radius of 1.5 m, structured as a set of 25 silicon sensor planes, segmented in $5.1 \times 5.1$ mm$^2$ square cells, alternated with tungsten absorber planes. In the barrel region, the hadronic calorimeter (HCAL) sits behind the ECAL, at an inner radius of 1.7 m. The HCAL consists of 60 layers of polystyrene scintillators, segmented in cells with $3 \times 3$ cm$^2$ area and alternated with layers of steel absorbers.

The event simulation considers the full detector layout, including the material in front of the calorimeter and the effect of the solenoidal magnetic field. From the full data we take slices centered around the barycenter of each ECAL energy deposit and we represent the ECAL and HCAL slices as 3D arrays of cell energy deposits.

We consider four kinds of particles (electrons $e$, photons $\gamma$, charged pions $\pi$, and neutral pions $\pi^0$) with energies uniformly distributed between 10 and 510 GeV, and with incident angles uniformly distributed between a polar angle $\theta$ between 1.047 and 2.094 radians with respect to the beam direction.

We get the barycenter of a shower from taking the 2D projection of its energy deposit on the ECAL inner surface. Then, knowing the point of origin of the incoming particle, we use the barycenter to estimate the particle's polar and azimuthal angles $\theta$ and $\phi$. The estimated pseudorapidity $\eta$ is then computed as $\eta = -\log[\tan\frac{\theta}{2}]$. Each single-shower event is prepared by taking a slice of the ECAL in a window around the shower barycenter, as well as the corresponding HCAL slice behind. Depending on the task, we take:

– **GEN dataset**: A $51 \times 51 \times 25$ cell window in the ECAL, for electrons in the energy range $100 - 200$ GeV. For use in the shower generation task.
– **REC dataset**: A $25 \times 25 \times 25$ cell slice of the electromagnetic calorimeter (ECAL) and a corresponding $11 \times 11 \times 60$ cell slice of the hadronic calorimeter (HCAL), for $e$, $\gamma$, $\pi$, or $\pi^0$ in the energy range $10 - 510$ GeV and with $\eta$ from $-0.524 - 0.524$. For use in the particle reconstruction task.

Examples of an electron shower and a charged-pion shower can be seen in Figure 1. The incoming particles enter from the top ($z = 0$), at the center of the $(x, y)$ transverse plane ($x = y = 25$). The electron event has left

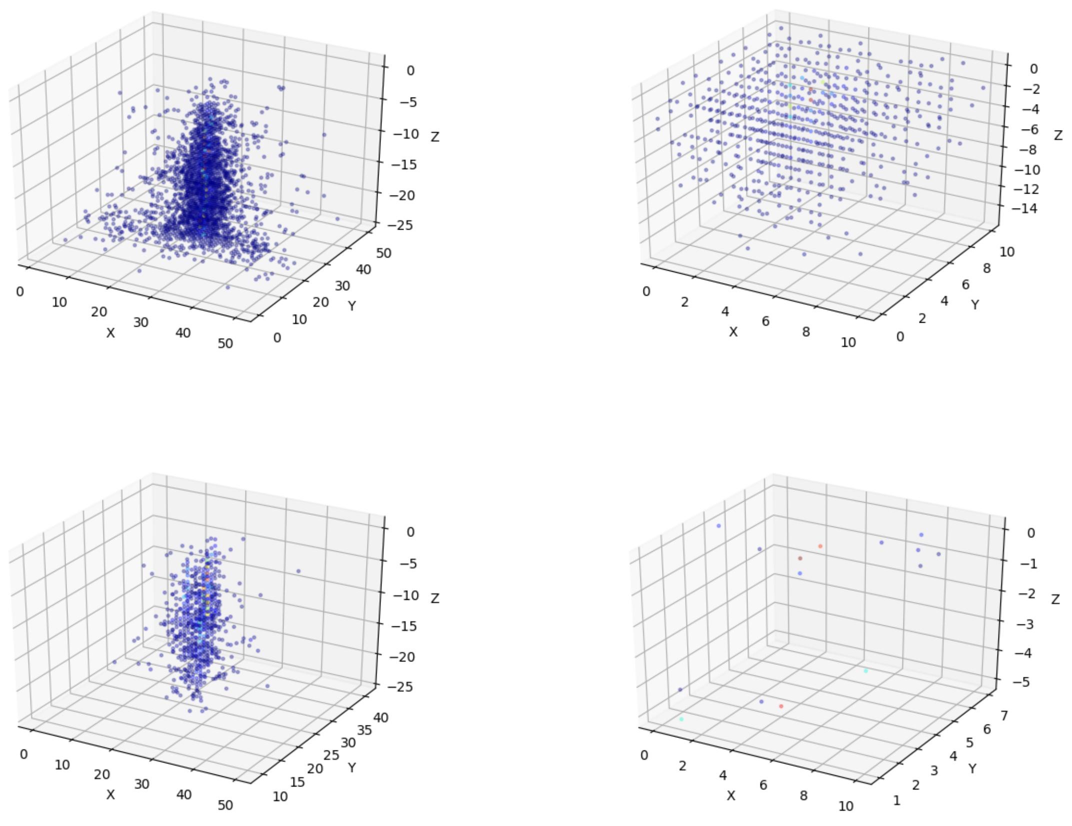


**Fig. 1.** 3D image of a photon (top) and neutral pion (bottom) shower in ECAL (left) and HCAL (right).

more hits in both the ECAL and HCAL. We can also see the presence of two subtracks in the neutral pion event.

The window size for the GEN dataset has been defined in order to contain as much of the shower information as practically possible. Motivated by the need of reducing the memory footprint for some of the models, we used a smaller window size for the REC dataset. When training classification models on these data, a negligible accuracy increase was observed when moving to larger windows, as described in Appendix A.

We apply a task-dependent filtering of the REC dataset, in order to select the subset of examples for which the task at hand is not trivial. For instance, distinguishing a generic pion from a generic electron is an easy task, and can be accomplished with high accuracy by looking at the HCAL/ECAL energy ratio. On the other hand, it is difficult to distinguish a generic electron from a pion that has a small HCAL/ECAL energy ratio. Thus we ignore charged-pion showers with a large HCAL/ECAL energy ratio. To be more specific, we see in Figure 2 that the ratio of total ECAL energy to total HCAL energy is very different for electrons and charged pions, with the heavier charged pions tending to leave little energy in the ECAL. In order to make the particle-identification task more challenging, we only consider showers with HCAL/ECAL $< 0.1$ cut. The results

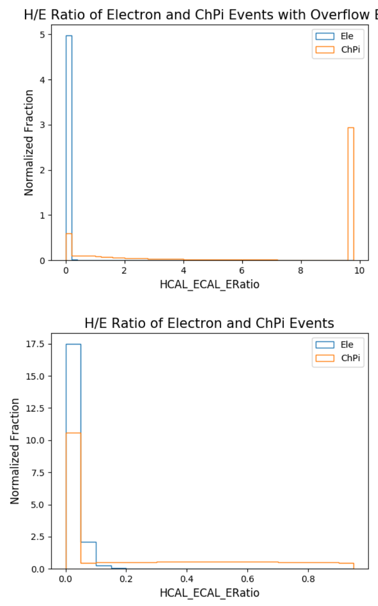


**Fig. 2.** HCAL/ECAL energy ratios for electrons and charged pions. The bottom plot is a zoomed-in version of the top plot.

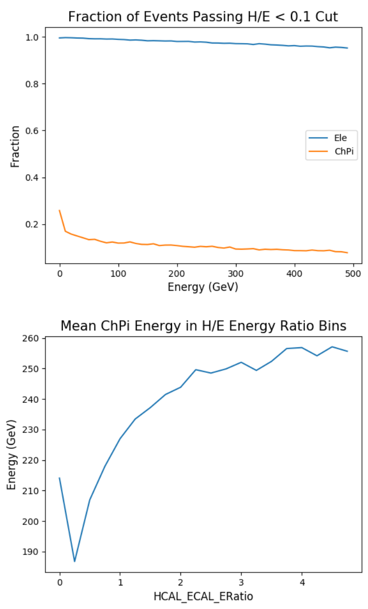


**Fig. 3.** Fractions of electrons and charged pions passing a HCAL/ECAL energy selection at various particle energies (top). The mean charged pion energy at each energy ratio (bottom).

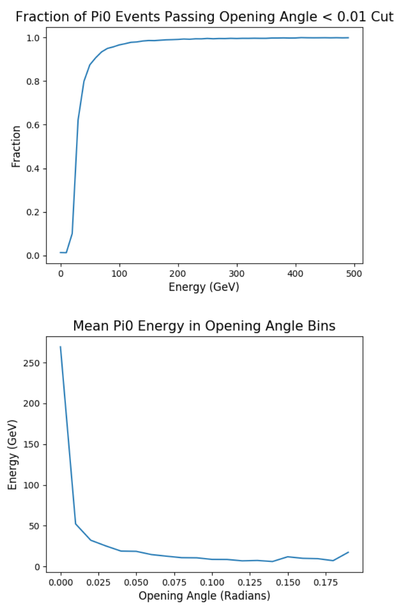


**Fig. 5.** The fraction of photons and neutral pions passing an opening angle $< 0.01$ radian selection at various particle energies (top). The mean neutral pion energy at each opening angle (bottom).

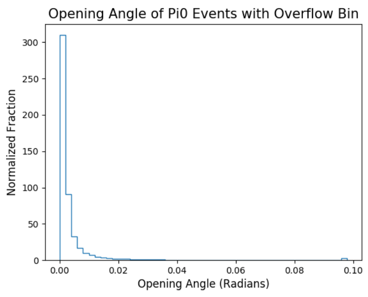


**Fig. 4.** Opening angle distribution for neutral pions decaying into two photons.

of this selection are shown in Figure 3. We can see that this selection favors mostly low-energy charged pions, which tend to leave more of their energy in the ECAL rather than punching through to the HCAL. Discriminating accurately between electrons and charged pions in this range is thus crucial for compressed-mass physics analyses, where we search for decay products with low energy.

Photons and neutral pions are more similar to each other. The easily distinguishable events are mostly due to the fact that neutral pions decay into two photons which are separated by a small angle. If the pion has a low energy, the opening angle between the two photons is larger and the shower is easily identified as originating from a neutral pion. High-energy neutral pions produce more collimated photon pairs, which are more easily mistaken as a single high-energy photon. The opening angle distribution for neutral pions is shown in Figure 4. In order to limit the study to the most challenging case, we filter the neutral-pion dataset by requiring the opening angle between the two photons to be smaller than 0.01 radian. The effect of this requirement on the otherwise uniform energy distribution is shown in Figure 5. As expected, the selection mostly removes low-energy neutral pions.

The ECAL and HCAL 3D arrays are passed directly to our neural networks. We also compute a set of expert features, as described in Ref. [24]. These features are used to train alternative benchmark algorithms (see Appendices C and D), representing currently-used ML algorithms in HEP.

## 3 Benchmark Tasks

In this section, we introduce the two benchmark tasks that we aim to solve with DL algorithms:

– Particle reconstruction: starting from raw detector hits, determine the nature of a particle and its momentum.
– Particle simulation: starting from a generator-level information of an incoming particle, generate the detector response (raw detector hits) using random numbers to model the stochastic nature of the process.

Both tasks represent heavy loads for central computing systems of large-scale high-energy physics experiments. A sizable acceleration of these processes in terms of resource reduction and execution time would generate a resource saving that could be invested in new opportunities.

### 3.1 Simulation

It is common in HEP to generate large amounts of accurate synthetic data from Monte Carlo simulations. These simulated data allow physicists to determine the expected outcome of a given experiment based on known physics. Having at hand this prior expectation, one can reveal the presence of new phenomena by observing an otherwise inexplicable difference between real and simulated data. An accurate simulation of a detector response is a computationally heavy task, currently taking a significant fraction of the overall computing resources. Thus we also investigate the use of DL algorithms to speed up the event simulation process. In particular, we build a generative model to simulate detector showers, similar to those on which we train the end-to-end reconstruction algorithm. Such a DL-based generator could drastically reduce the computing needs of any collider-physics experiment, by drastically reducing the turnaround time for a generation workflow. Potentially, this could turn a Monte Carlo event generation campaign into an on-demand task.

In order to create realistic calorimetric shower data, we train a generative adversarial network (GAN) on the GEN dataset defined in Section 2. We restrict the study to ECAL showers for incoming electrons with energy between 100 and 200 GeV. The task is to create a model that can take an electron's energy and flight direction as inputs and generate a full ECAL shower, represented as a $51 \times 51 \times 25$ array of energy deposits along the trajectory of the incoming electron. The advantage of using a GAN is that it's much faster and less computationally intense than traditional Monte Carlo simulation, and the results may more accurately reproduce physical behavior if the GAN is trained on real data.

## 3.2 Reconstruction

At particle-collider experiments, data consist of sparse sets of hits recorded by various detector components at beam collision points. A typical analysis begins with a complex reconstruction algorithm that processes these raw data to produce a set of physics objects (jets, electrons, muons, etc.), which are then used further down the line. Traditionally, the reconstruction software consists of a set of rule-based algorithms that are designed based on physics knowledge of the specific problem at hand (e.g., the bending of particles in a solenoidal magnetic field, due to the Lorentz force). Over the past decade or so, machine-learning algorithms have been integrated into certain aspects of particle reconstruction (e.g., particle identification). One example is the identification of electrons and photons via a BDT, taking as input for each event a set of high-level features quantifying the shape of the energy cluster deposited in a calorimeter shower [25].

Event reconstruction is one of the most CPU-intensive tasks at the LHC. In order to reduce the resource needs, one could imagine using deep learning (DL) techniques to extract the required information directly from the raw data, without first computing high-level features. Following this idea, we investigate here an end-to-end DL model based on computer vision techniques, treating the calorimeter input as a 3D image. Using a combined architecture, the model is designed to simultaneously perform particle identification and energy measurement.

When dealing with particle reconstruction, one is interested in identifying a particle's type (electron, photon, etc.) and its momentum. An end-to-end application aiming to provide a full reconstruction of a given particle should thus be able to simultaneously solve a multi-class classification problem and a regression problem. In our study, we filter the REC dataset to make the classification task non-trivial, as described in Section 2. Since differentiating charged and uncharged particles is trivial, we judged the classification of our model on its ability to distinguish electrons from charged pions, and photons from neutral pions.

Our reconstruction networks were thus given the following three tasks:

- **Identify electrons over a background of charged pions**: Charged pions are the most abundant particles produced in LHC collisions. They are typically arranged into jets, which are collimated sprays resulting from the showering and hadronization processes of quarks and gluons. On the other hand, electrons are rarely produced, and their presence is typically an indication of an interesting event occurring in the collision. A good electron identification should aim at misidentifying at most 1 in 10,000 pions as an electron. In our problem, we consider the background as originating from single pions, which is a case more typical of electron-positron colliders. Initial studies of dense neural network (DNN) based classification of unfiltered events containing the four particles types in our simulated samples yielded extremely good results, with area under curve (AUC) of receiver-operator curves (ROC) near 1, when using equally sized and unbiased samples of each particle class. In order to test DL capabilities with a more challenging problem and to approach the kind of task that one faces at the LHC, we perform this binary classification on the REC dataset of Section 2, after applying an HCAL/ECAL energy ratio cut.
- **Identify photons over a background of neutral pions**: At particle colliders, the main background to photon identification comes from neutral pions decaying to a photon pair. The two photons from the $\pi^0$ decays are produced with an angular distance that tends to zero in the limit of high $\pi^0$ momentum. In general, a generic $\gamma/\pi^0$ classification task is relatively easy, since the presence of two nearby clusters is a clear signature of $\pi^0$. On the other hand, at high $\pi^0$ momentum the energy deposits from the two photons merge into a single cluster, and $\gamma$ and $\pi^0$ become very similar. In this paper, we focus on this case, running the binary classification task on the REC dataset of Section 2, considering only $\pi^0$s with an opening angle $< 0.01$ radian.
- **Momentum measurement: do we regress energy or momentum?** Once the particle is identified, it is very important to accurately determine its momentum, since this allows physicists to calculate all the relevant high-level features, such as the mass of new particles that generated the detected particles when decaying.

In this study, we address this problem on the same dataset used for the classification tasks, restricting the focus to range of energies from 10 to 510 GeV, and at various incident angles ($\eta$). Regression results using various neural network architectures were compared with results from linear regression, comparing both resolution and bias. The models we consider are designed to return the full particle momentum (energy, $\eta$, and $\phi$) of the incoming particle momentum. At this stage, this functionality is not fully exploited and only the energy determination is considered. An extension of our work to include the determination of $\eta$ and $\phi$ could be the matter of future studies.

## 4 Generative Model

Generative Adversarial Networks are composed of two networks, a discriminator and a generator. Our model, 3DGAN, implements an architecture inspired by the auxiliary classifier GAN [26]. The generator takes as input a specific particle type, flight direction, and energy, and generates the 3D image of an energy deposit using an auxiliary input vector of random quantities (latent vector). The output has the same format as the 3D array of ECAL hits in the GEN sample (see Section 2). The discriminator network receives as input an ECAL 3D array and classifies it as *real* (coming from the GEANT4-generated GEN dataset) or *fake* (produced by the generator).

Our initial 3DGAN prototype [24] successfully simulated detector outputs for electrons which were orthogonally incident to the calorimeter surface. In addition, the discriminator performed an auxiliary regression task on the input particle energy. This task was used to cross check the quality of the generation process.

In this study, we consider a more complex dataset, e.g., due to the variable incident angle of the incoming electron on the inner ECAL surface. To monitor this additional complexity, we add additional components to the loss function, related to the regression of the particle direction and the pixel intensity distribution (energy deposition in cells). This will be described in more detail below.

Before training our GAN, we pre-processed the GEN dataset by replacing each cell energy content $E$ with $E^{\alpha}$, where $\alpha < 1$ is a fixed hyperparameter. This pre-processing compensates for the large energy range (about 7 orders of magnitude) covered by individual cell energies, and mitigates some performance degradation we previously observed at low energies. After testing for different values of $\alpha$, we observed optimal performance for $\alpha = 0.85$.

### 4.1 GAN Architecture

The 3DGAN architecture is based on 3-dimensional convolutional layers, as shown in Figure 6. The generator takes as input a vector with a desired particle energy and angle, and concatenates a latent vector of 254 normally distributed random numbers. This goes through a set of alternating upsampling and convolutional layers. The first convolution layer has 64 filters with $6 \times 6 \times 8$ kernels. The next two convolutional layers have 6 filters of $5 \times 8 \times 8$ and $3 \times 5 \times 8$ kernels, respectively. The last convolutional layer has a single filter with a $2 \times 2 \times 2$ kernel. The first three layers are activated by leaky ReLU functions, while ReLU functions are used for the last layer. Batch normalization and upscaling layers were added after the first and second convolutional layers.

The discriminator takes as input a $51 \times 51 \times 25$ array and consists of four 3D convolutional layers. The first layer has 16 filters with $5 \times 6 \times 6$ kernels. The second, third, and fourth convolutional layers each have 8 filters with $5 \times 6 \times 6$ kernels. There are leaky ReLU activation functions in each convolutional layer. Batch normalization and dropout layers are added after the second, third, and fourth convolutional layers. The output of the final convolution layer is flattened and connected to two output nodes: a classification node, activated by a sigmoid and returning the probability of a given input to be true or fake; and a regression node, activated by a linear function and returning the input particle energy. The 3DGAN model is implemented in KERAS [20] and Tensorflow [21].

### 4.2 Training and Results

The 3DGAN loss function

$$L_{Tot} = W_G L_G + W_P L_P + W_A L_A + W_E L_E + W_B L_B \quad (1)$$

is built as a weighted sum of several terms: a binary cross entropy ($L_G$) function of the real/fake probability returned by the discriminator, mean absolute percentage error terms (MAPE) related to the regression of the primary-particle energy ($L_P$) , the total deposited energy ($L_E$) and the binned pixel intensity distribution ($L_B$), and a mean absolute error (MAE) for the incident angles measurement ($L_A$). The binary cross entropy term, percentage errors and absolute error are weighted by 3.0, 0.1 and 25 respectively. The weights $W$ are tuned to balance the relative importance of each contribution. The predicted energy and incident angle provide a feedback on the conditioning of the image. The binned pixel intensity distribution loss compares the counts in different bins of pixel intensities.

The model training is done using the RMSprop [27] optimiser. We alternately train the discriminator on a batch of real images and a batch of generated images, applying label switching. We then train the generator while freezing the discriminator weights.

Figure 7 shows a few events from the GEN data set. The events were selected to cover both ends of the primary-particle energy and angle spectrum. Figure 8 presents the corresponding generated events with the same primary particle energy and angle as the GEN events in Figure 7. Initial visual inspection shows no obvious difference between the original and GAN generated images. A detailed validation based on several energy-shape related features confirms these results. We discuss a few examples below.

The top row in Figure 9 shows the ratio between the total energy deposited in the calorimeter and the primary

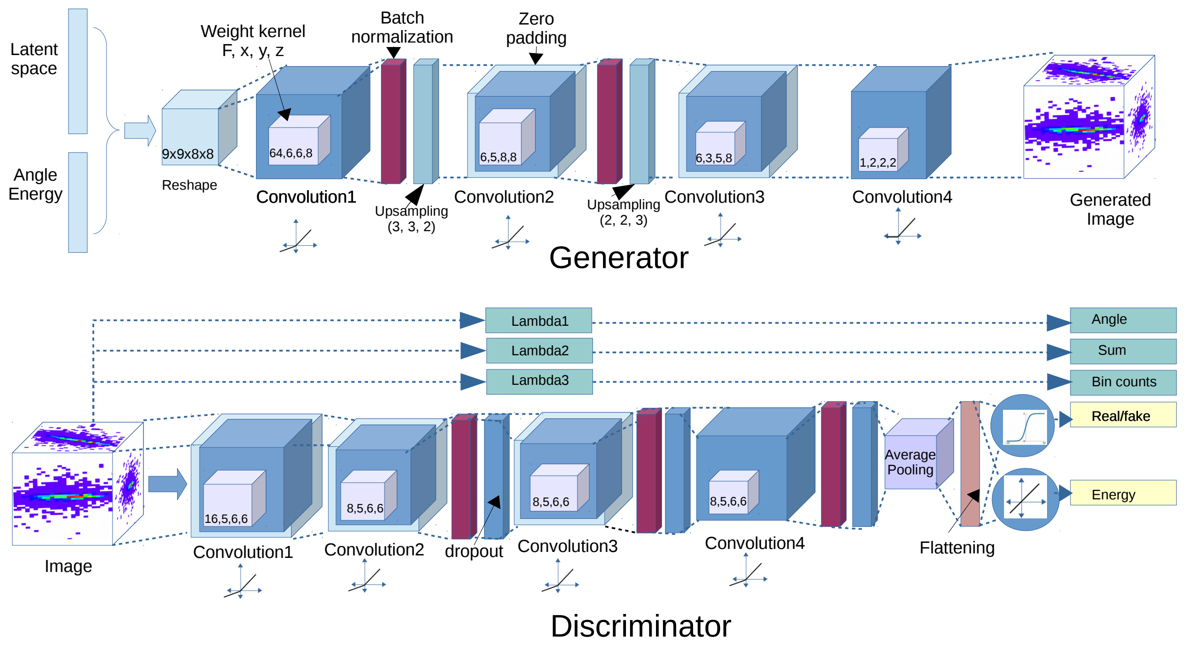


**Fig. 6.** 3DGAN generator and discriminator network architectures

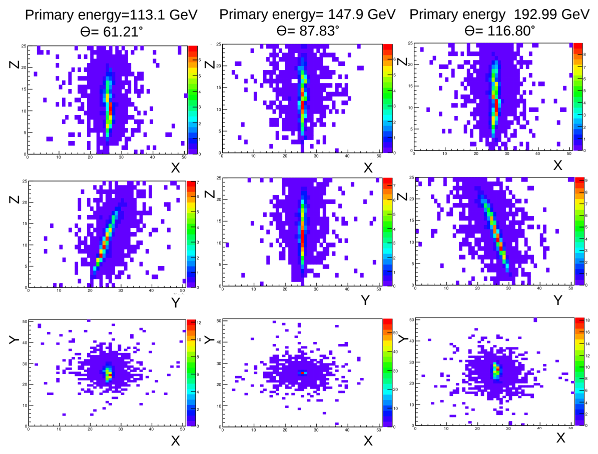


**Fig. 7.** GEN sample: electrons with different primary particle energies and angles.

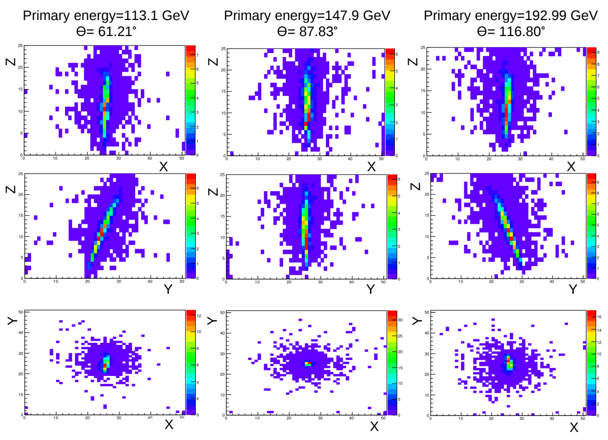


**Fig. 8.** GAN generated electrons with primary energies and angles corresponding to the electrons showed in Figure 7.

particle energy as a function of the primary particle energy (we refer to it as "sampling fraction") for different angle values. 3DGAN can nicely reproduce the expected behaviour over the whole energy spectrum. The second row in Figure 9 shows the number of hits above a $3 \times 10^{-4}$ MeV threshold. Figure 9 also shows the x, y, z "energy shapes", i.e. the amount of energy deposited along different axes (x and y on the transverse plane and z along the calorimeter depth). The agreement is very good, and in particular 3DGAN is able to mimic the way the energy distributions changes with incident angle. Figure 10 shows some additional features aimed at defining the shape of the deposited energy distribution. In particular the second moments along the x,y and z axes are shown on the first column, measuring the width of the deposited energy distribution along those axes. The second column shows the way the energy is deposited along the depth of the calorimeter, by splitting the calorimeter in three parts along the longitudinal direction and measuring the ratios between the energy deposited in each third and the total deposited energy. Finally, the third column in Figure 10 highlights the tails of the "energy shapes". It can be seen that, while the core of the distribution is perfectly described by 3DGAN, the network tends to overestimate the amount of energy deposited at the edges of the volume. It should be noted however that energy depositions in those cells are very sparse.

The 3DGAN training runs in around 1.5 hours per epoch on a single NVIDIA GeForce GTX 1080 card for 60 epochs. The simulation time on a Intel Xeon 8180 is about 13 ms/particle and it goes down to about 4 ms/par-

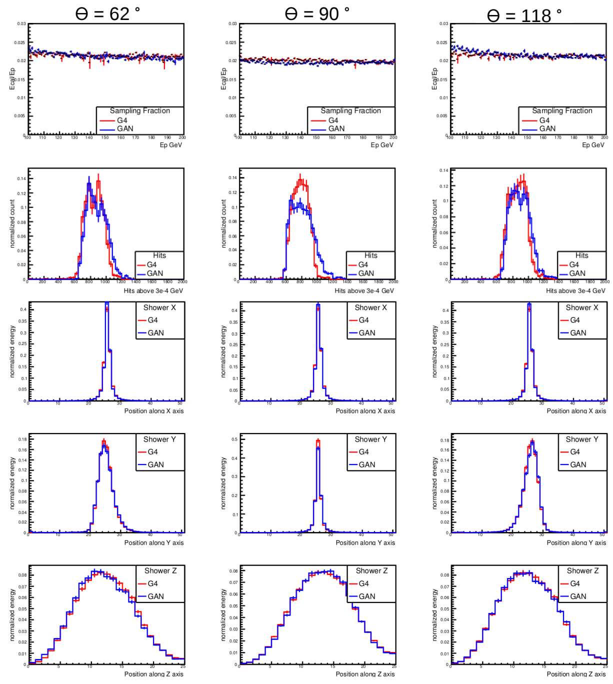


**Fig. 9.** GEANT4 vs. GAN comparison for sampling fraction, number of hits and shower shapes along x,y,z axis for different angle bins with 100-200 GeV primary particle energies.

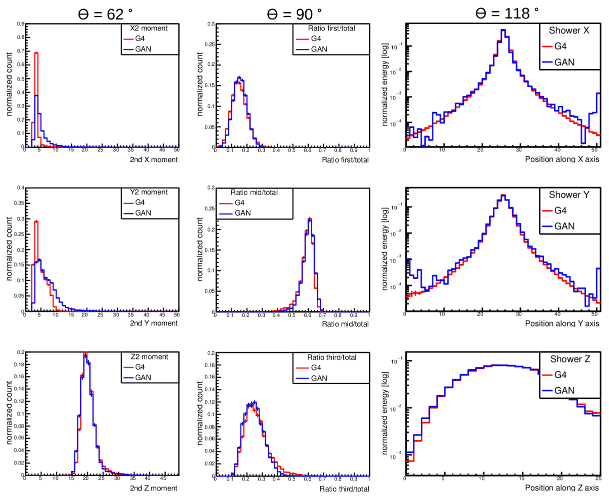


**Fig. 10.** GEANT4 vs. GAN comparison for shower width (second moment) in x,y,z, ratio of energy deposited in parts along direction of particle traversal to total energy and shower shapes along x,y,z axis in log scale for 100-200 GeV primary particle energies and 60-120 degrees $\theta$.

ticle on a NVIDIA GeForce GTX 1080. For comparison GEANT4 simulation takes about 17 seconds per particle on a Intel Xeon 8180 (currently it is not possible to run a full GEANT4-based simulation on GPUs). Thus our GAN represents a potential simulation speedup of over 4,000 times for this specific aspect of the event simulation.

When given as input to a particle regression and reconstruction model (see section 5), this dataset produces the same output as the original GEANT4 sample, as described in Appendix B.

## 5 End-to-End Particle Reconstruction

This section describes the use of a deep neural network to accomplish an end-to-end particle reconstruction task. The model consists of a neural architecture which simultaneously performs both particle classification and energy regression. This combined network is trained using the ECAL and HCAL cell arrays as well as the total ECAL energy and total HCAL energy as inputs. The training loss function is written as the sum of a binary cross entropy for particle identification and a mean-square error loss for energy regression. Through experimentation, we found that multiplying the energy loss by a factor of 200 gave the best results, as it was easier to quickly achieve low loss values for energy regression.

We compare three different architectures for our reconstruction model, each trained using calorimeter cell-level information as inputs:

– A dense (i.e, fully connected) neural network (DNN).
– A 3D convolutional network (CNN).
– A network based on GoogLeNet (GN) [28], using layers of inception modules.

In order to compare the model performance to a typical state-of-the-art particle reconstruction algorithm, we also consider the following alternatives:

– A feature-based BDT (see Appendix C) for the classification task.
– A linear regression for the regression task.
– A BDT for the regression task (for more info on regression baselines see Appendix D).

In a previous study [24], we compared the classification accuracy obtained with a neural model taking as input the energy cells, a feature-based neural models, and a feature-based BDTs. In that context, we demonstrated that feature-based BDTs and neural networks perform equally well, and are both equally capable of correctly classify particles from a small set of calculated features. We do not compare feature-based neural networks in this paper, and use feature-based BDTs to represent the current state-of-the-art classification algorithms.

### 5.1 Deep Network Models

The three DL models take as input the ECAL and HCAL 3D energy arrays of the REC dataset (see Section 2), together with the total energies recorded in ECAL and in HCAL (i.e., the sum of the values stored in the 3D arrays), as well as the estimated $\phi$ and $\eta$ angles of the incoming particle, calculated using the collision origin and the barycenter of the event. The architecture of each model is defined with a number of floating parameters (e.g. number of hidden layers), which are refined through a hyperparameter optimization, as described in Section 5.2. Each model returns three numbers. After applying a softmax activation, two of these elements are interpreted as the classification probabilities of the current two-class problem. The third output is interpreted as the energy of the particle.

Here we describe in detail the three model architectures:

- In the DNN model we first flatten our ECAL and HCAL inputs into 1D arrays. We then concatenate these array along with the total ECAL energy, total HCAL energy, estimated $\phi$, and estimated $\eta$, for an array of total size $25 \times 25 \times 25 + 11 \times 11 \times 60 + 4 = 22889$ inputs. This array is fed as input to the first layer of the DNN, followed by a number of hidden layers each followed by a ReLU activation function and a dropout layer. The number of neurons per hidden layer and the dropout probability are identical for each relevant layer. The number of hidden layers, number of hidden neurons per layer, and dropout rate are hyperparameters, tuned as described in the next session. Finally, we take the output from the last dropout layer, append the total energies and estimated angles again, and feed the concatenated array into a final hidden layer, which results in a three-element output.
- The CNN architecture consists of one 3D convolutional layer for each of the ECAL and HCAL inputs, each followed by a ReLU activation function and a max pooling layer of kernel size $2 \times 2 \times 2$. The number of filters and the kernel size in the ECAL convolutional layer are treated as optimized hyperparameter (see next session). The HCAL layer is fixed at 3 filters with a kernel size of $2 \times 2 \times 6$. The two outputs are then flattened and concatenated along with the total ECAL and HCAL energies, as well as the estimated $\phi$ and $\eta$ coordinates of the incoming particle. The resulting 1D array is passed to a sequence of dense layers each followed by a ReLU activation function and dropout layer, as in the DNN model. The number of hidden layers and the number of neurons on each layer are considered as hyperparameters to be optimized. The output layer consists of three numbers, as for the DNN model. We found that adding additional convolutional layers to this model beyond the first had little impact on performance. This may be because a single layer is already able to capture important information about localized shower structure, and reduces the dimensionality of the event enough where a densely connected net is able to do the rest.
- The third model uses elements of the GoogLeNet [28] architecture. This network processes the ECAL input array with a 3D convolutional layer with 192 filters, a kernel size of 3 in all directions, and a stride size of 1. The result is batch-normalized and sent through a ReLU activation function. This is followed by a series of inception and MaxPool layers of various sizes, with the full architecture described in Appendix E. The output of this sequence is concatenated to the total ECAL energy, the total HCAL energy, the estimated $\phi$ and $\eta$ coordinates, and passed to a series of dense layers like in the DNN architecture, to return the final three outputs. The number of neurons in the final dense hidden layer is the only architecture-related hyperparameter for the GN model. Due to practical limitations imposed by memory constraints, this model does not take the HCAL 3D array as input. This limitation has a small impact on the model performance, since the ECAL array carries the majority of the relevant information for the problems at hand (see Appendix F).

On all models, the regression task is facilitated by using skip connections to directly append the input total ECAL and HCAL energies to the last layer. The impact of this architecture choice on regression performance is described in Appendix G. In addition to using total energies, we also tested the possibility of using 2D projections of the input energy arrays, summing along the $z$ dimension (detector depth). This choice resulted in worse performance (see Appendix H) and was discarded.

## 5.2 Hyperparameter Scans

In order to determine the best architectures for the end-to-end reconstruction models, we scanned over a hyperparameter space for each architecture. Learning rate and decay rate were additional hyperparameters for each architecture. For simplicity, we used classification accuracy for the $\gamma$ vs. $\pi^0$ problem as a metric to determine the overall best hyperparameter set for each architecture. This is because a model optimized for this task was found to generate good results for the other three tasks as well, and because $\gamma$ vs. $\pi^0$ classification was found to be the most difficult problem.

Each hyperparameter point was scanned ten times, in order to obtain an estimate of the uncertainty associated with each quoted performance value. For each scan point, the DNN and CNN architectures trained on 400,000 events, using another sample of 400,000 events for testing. DNN and CNN scan points trained for three epochs each, taking about seven hours each. GN trained on 100,000 events and tested on another 100,000. Due to a higher training time, each GN scan point only trained for a single epoch, taking about twenty hours.

For CNN and DNN training, we used batches of 1,000 events when training. However, due to GPU memory limitations, we could not do the same with GN. Instead, we split each batch into 100 minibatches of ten events each. A single minibatch was loaded on the GPU at a time, and gradients were added up after back-propagation. Only after the entire batch was calculated did we update network weights using the combined gradients.

The best settings were found to be as follows:

- For DNN, 4 hidden layers, 512 neurons per hidden layer, a learning rate of 0.0002, decay rate of 0, and a dropout probability of 0.04.
- For CNN, 4 hidden layers and 512 neurons per hidden layer, a learning rate of 0.0004, decay rate of 0, a dropout probability of 0.12, 6 ECAL filters with a kernel size of $6 \times 6 \times 6$.
- For GN, 1024 neurons in the hidden layer, 0.0001 learning rate, and 0.01 decay rate.

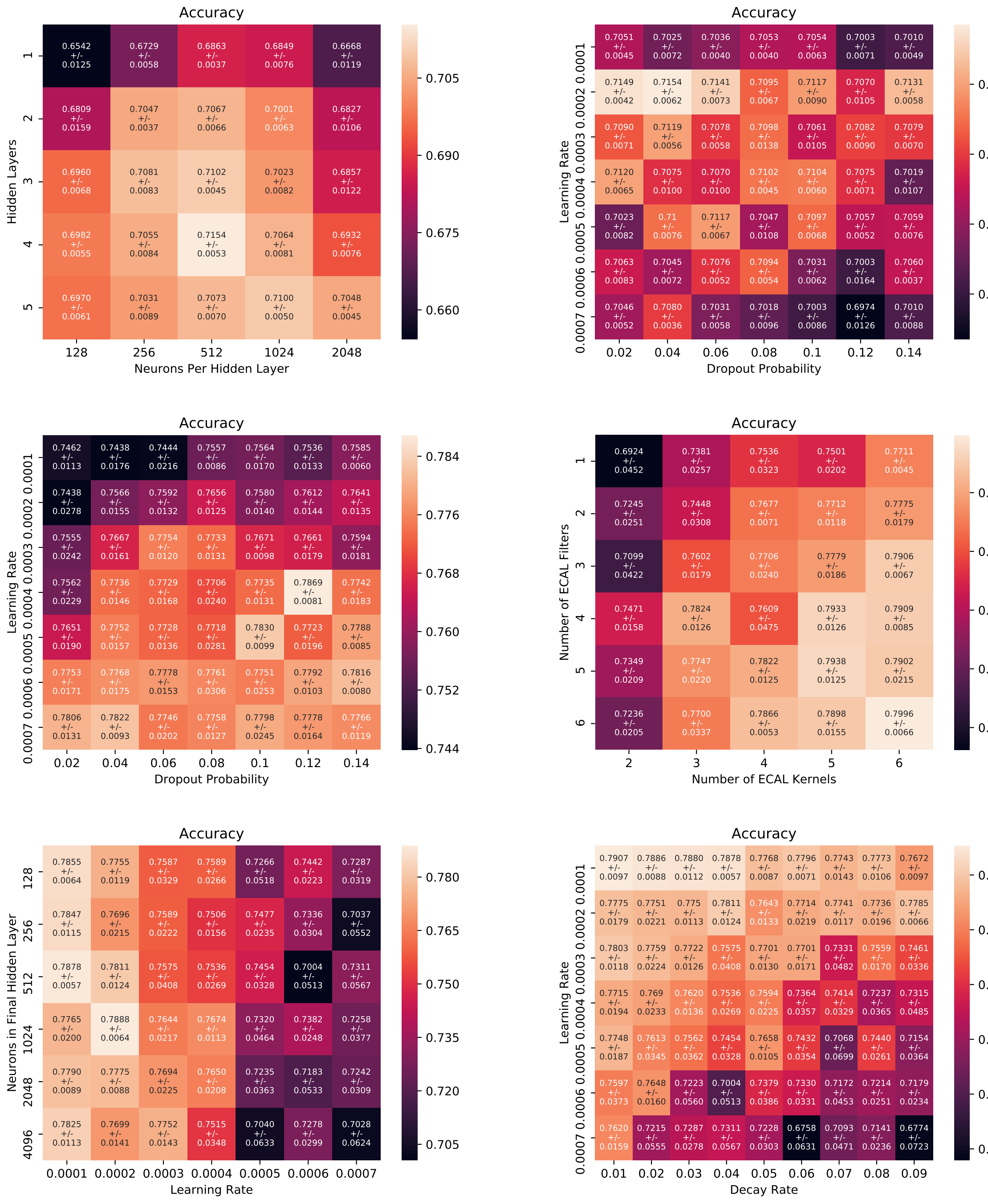
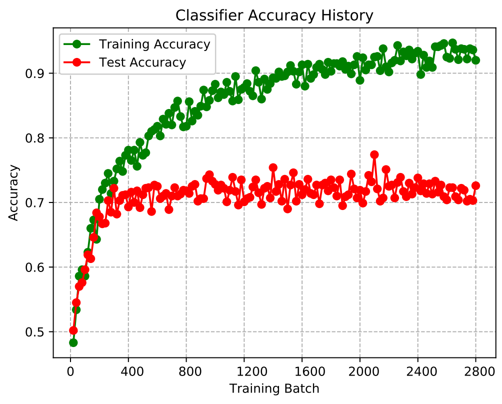
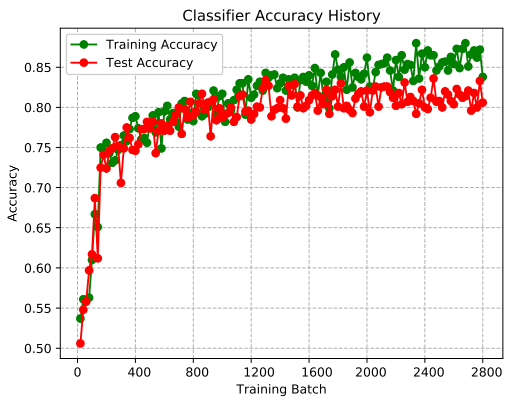


**Fig. 11.** Selected hyperparameter scan results for DNN (top), CNN (center), and GN (bottom). In each figure, the classification accuracy is displayed as a function of the hyperparameters reported on the two axes.

Selected hyperparameter scan slices are shown in Figure 11. These 2D scans were obtained setting all values besides the two under consideration (i.e., those on the axes) to be fixed at default values: a dropout rate of 0.08, a learning rate of 0.0004, a decay rate of 0.04, three dense layers for CNN and DNN, and 512 neurons per hidden layer. For GN, the default number of ECAL filters was 3, with a kernel size of 4.

After performing the hyperparameter scan, we trained each architecture using its optimal hyperparameters for a greater number of epochs. The evolution of the training and validation accuracy as a function of the batch number for these extended trainings is shown in Figure 12.





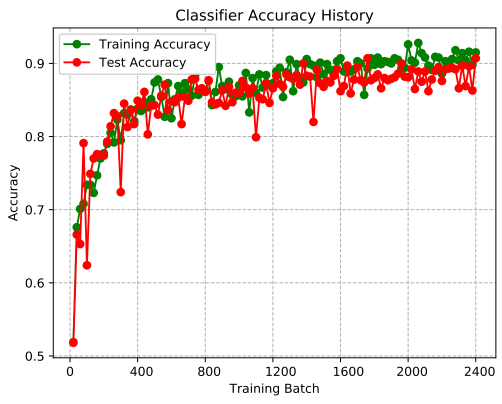


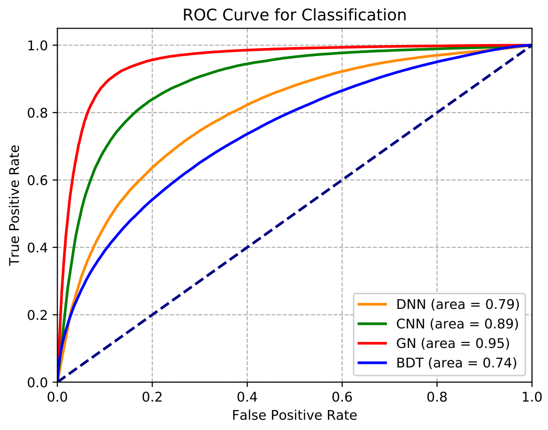
**Fig. 12.** Training curves for best DNN (top), CNN (middle), and GN (bottom) hyperparameters, trained on variable-angle $\gamma/\pi^0$ samples. We see that the DNN over-trains quickly and saturates at a relatively low accuracy, while the CNN takes longer to over-train and reaches a higher accuracy, and GN performs best of all. Each 400 batches corresponds to a single epoch.

## 5.3 Results

We apply the best architectures described in the previous section separately to our electron vs. charged pion and photon vs. neutral pion reconstruction problems.

### 5.3.1 Classification Performance

Figure 13 shows ROC curve comparisons for the two classification tasks. As expected, the electron vs. charged pion classification problem was found to be a simple task, resulting in an area under the curve (AUC) close to 100%. For a baseline comparison, the curve obtained for a BDT (see Appendix C) is also shown. This BDT was optimized using the *scikit-optimize* package [29], and was trained using high-level features computed from the raw 3D arrays. It represents the performance of current ML approaches on these problems.



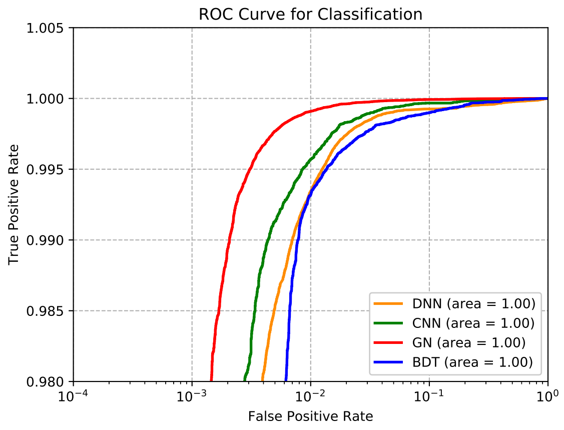


**Fig. 13.** ROC curve comparisons for $\gamma$ vs. $\pi^0$ (top) and $e$ vs. $\pi^{\pm}$ (bottom) classification using different neural network architectures. Samples include particle energies from 10 to 510 GeV, and an inclusive $\eta$ range.

The DL models outperform the BDT, with the GN reaching the best classification performance on both problems. Figure 14 shows the best-model performance as a function of the energy and $\eta$ of the incoming particle, for the photon vs. neutral pion and the electron vs. charged pion problems. These figures show that classification accuracy is maintained over a wide range of particle energies and angles. The models appear to perform a bit worse at higher energies for the photon vs. neutral pion case, due

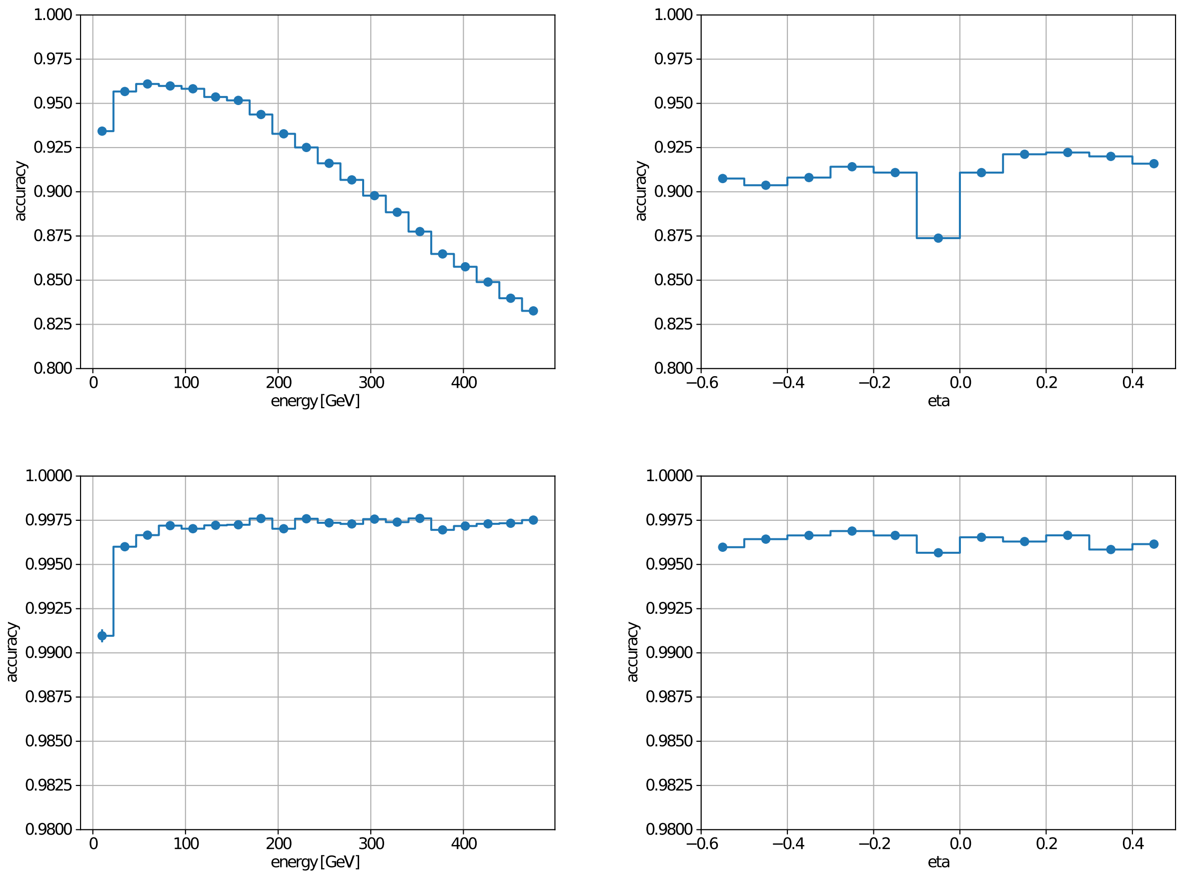


**Fig. 14.** Classification accuracy of best performing network for $\gamma$ vs. $\pi^0$ (top) and $e$ vs. $\pi^\pm$ (bottom), in bins of energy (left) and $\eta$ (right).

to the fact that the pion to two photon decay becomes increasingly collimated at higher energies. Similarly, the performance is slightly worse when particles impact the detector perpendicularly than when they enter at a wide angle, because the shower cross section on the calorimeter inner surface is reduced at 90°, making it harder to distinguish shower features.

### 5.3.2 Regression Performance

Figure 15 shows the energy regression performance for each particle type, obtained from the end-to-end reconstruction architectures. In this case, we compare against a linear regression algorithm and a BDT (labelled as "XGBoost") representing the current state-of-the-art, as described in Appendix D.

Since the energy regression problem is not as complex as the classification problem, the three architectures (DNN, CNN, GN) perform fairly similarly, with the exception of the GN performance on $\pi^\pm$, which is a bit worse. The performance is overall worse for $\pi^\pm$, both with the networks and with the benchmark baselines (linear regression and XGBoost).

A closer look at the performance boost given by each network can be obtained examining the case of particles entering the calorimeter inner surface at 90°, i.e. with $\eta = 0$ [1]. In this case, the problem is more constrained and both the networks and the baseline algorithms are able to perform accurately. The results for fixed angle samples are shown in Appendix I.

We have also tested the result of training on one class of particle and performing regression on another. These results can be seen in Appendix J. In addition, we have looked at the effect on energy regression of increasing the ECAL and HCAL window sizes. This can be seen in Appendix K.

## 5.4 Resampling to ATLAS and CMS Geometries

In addition to the results presented so far, we show in this section how the end-to-end reconstruction would perform on calorimeters with granularity and geometry similar to those of the ATLAS and CMS calorimeters. Since the REC dataset (see Section 2) is generated using the geometry of

[1] For these additional fixed-angle regression plots, we did not train GoogLeNet architectures.

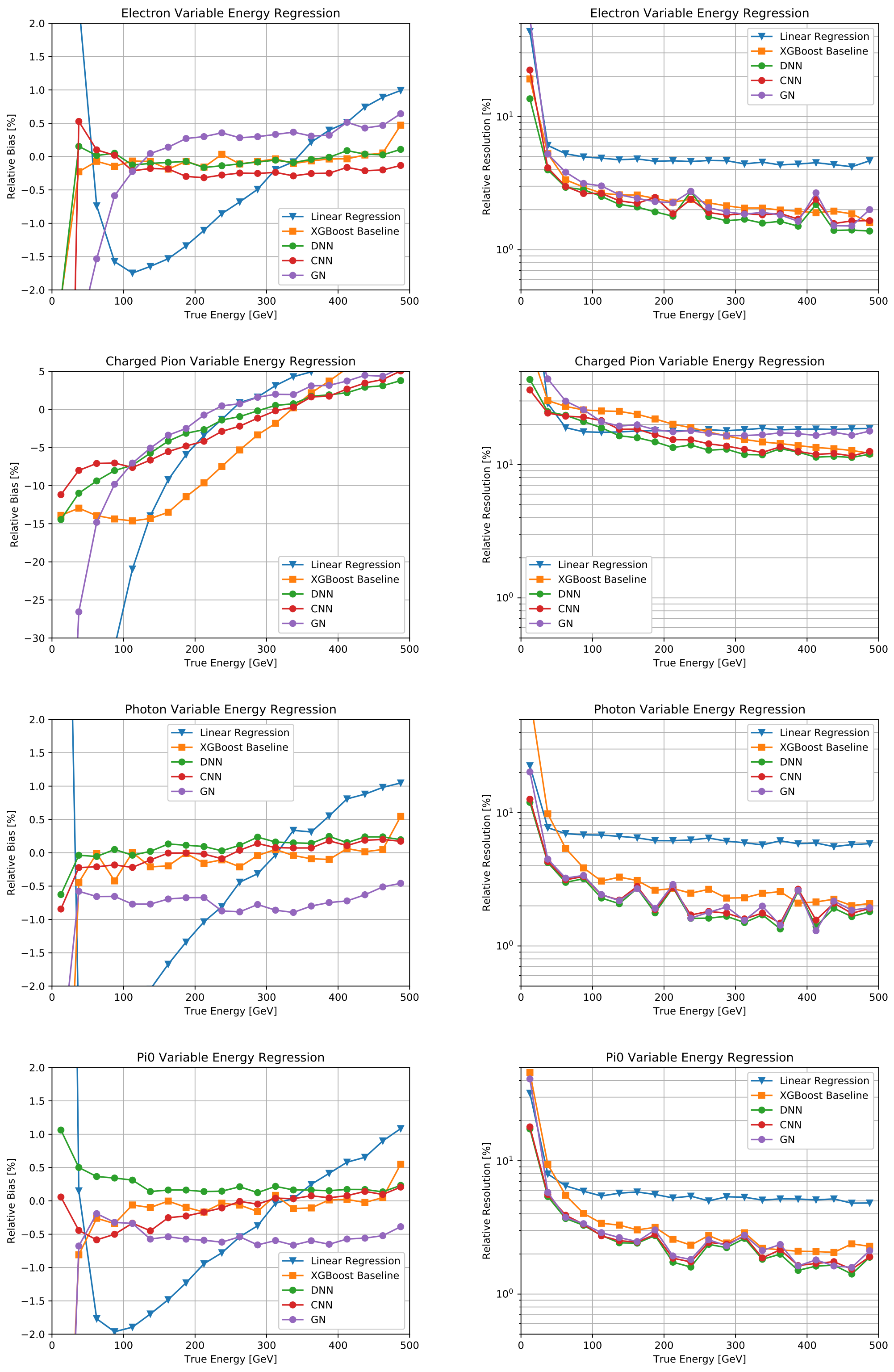


**Fig. 15.** Regression bias (top) and resolution (bottom) as a function of true energy for energy predictions on the REC dataset with variable-angle incident angle. From top to bottom: electrons, charged pions, photons, and neutral pions.

**Table 1.** Detailed description of the three detector geometries used in this study: the baseline CLIC ECAL detector and the ATLAS and CMS calorimeters.

| Parameter | **CLIC** | **ATLAS** 1st layer | 2nd layer | 3rd layer | **CMS** |
|---|---|---|---|---|---|
| $\Delta\eta$ | 0.003 | 0.025 /8 | 0.025 | 0.5 | 0.0175 |
| $\Delta\phi$ | 0.003 | 0.1 | 0.025 | 0.025 | 0.0175 |
| Radiation Length [cm] | 0.3504 | 14 | 14 | 14 | 0.8903 |
| Moliere radios [cm] | 0.9327 | 9.043 | 9.043 | 9.043 | 1.959 |

the proposed LCD detector, it has a much higher granularity than the current-generation ATLAS and CMS detectors. To visualize how our calorimeter data would look with a coarser detector, we linearly extrapolate the contents of each event to a different calorimeter geometry, using a process we have termed "resampling". To keep the resampling procedure simple, we discard the HCAL information and consider only the ECAL 3D array.

A not-to-scale example of the full procedure is shown in Figure 16. In this example, we resample the input to a regular square grid with lower granularity than the input data. The operation is simplified in the figure, in order to make the explanation easy to visualize. The actual ATLAS and CMS calorimeter geometries are more complex than a regular array, as described in Table 1.

In the resampling process, we first extrapolate each energy value from the grid of CLIC cells to a different geometry. To do so, we scale the content of each CLIC cell to the fraction of overlap area between the CLIC cell and the cell of the target geometry. When computing the overlap fraction, we take into account the fact that different materials have different properties (Moliere radius, interaction length, and radiation length). For instance, CLIC is more fine-grained than CMS or ATLAS detectors, but the Moliere radius of the CLIC ECAL is much smaller than in either of those detectors. This difference determines an offset in the fine binning. Thus, when applying our resampling procedure we normalize the cell size by the detector properties. The Moliere radius is used for $x$ and $y$ re-binning, and radiation length is used for the $z$ direction. At this point we have a good approximation for how the event would look in a calorimeter with the target geometry.

To complete the resampling process, we invert the procedure to go back to our original high-granularity geometry. This last step allows us to keep using the model architectures that we have already optimized. It adds no additional information that would not be present in the low-granularity geometry. This up-sampling also allows us to deal with the irregular geometry of the ATLAS calorimeter by turning it into a neat grid. With no up-sampling, it would not be possible to apply the CNN and GN models.

The resampling procedure comes with a substantial simplification of the underlying physics process. First of all, the information at the edge of the grid is imperfectly translated during the resampling process, leading to worse performance than what could theoretically be achieved in the actual CMS and ATLAS detectors. Also, this simple geometrical rescaling doesn't capture many other detector characteristics. For example, the CMS ECAL detector has

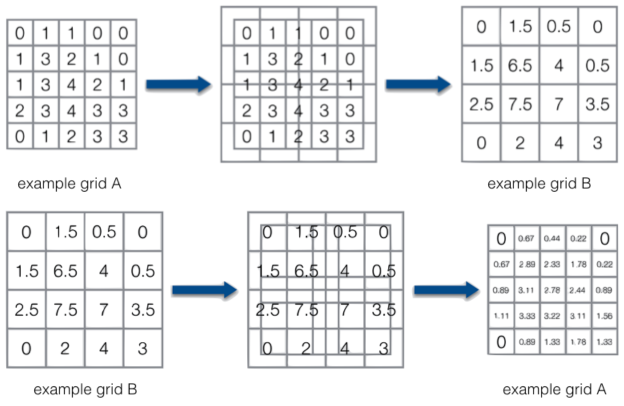
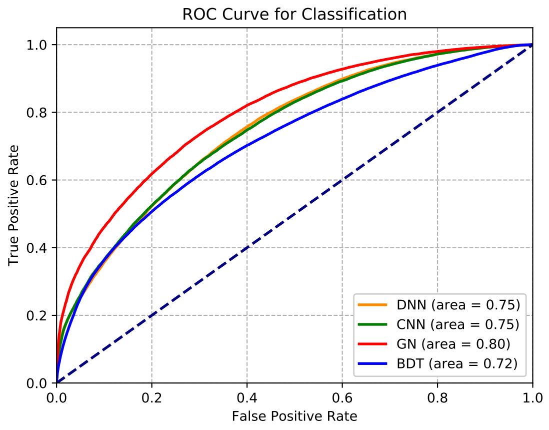


**Fig. 16.** Example of the resampling procedure used to emulate CLIC data on a different detector geometry (the example shown here is simply a larger grid). First, we extrapolate hit information from one geometry to another (top). Next, we extrapolate back to the original geometry (bottom). This allows us to emulate the rougher granularity of the second geometry, while keeping data array sizes constant and enabling us to use the models we have already developed for the CLIC dataset. Note that some information is lost at the edges.

no depth information, but being homogeneous it provides a very precise energy measurement. Our resampling method only captures geometric effects, and would not be able to model the improvement in energy resolution. Furthermore, we are unable to include second-order effects such as gaps in the detector geometries. Despite these limitations, one can still extract useful information from the resampled datasets, comparing the classification and regression performances of the end-to-end models defined in Sections 5.3.1 and 5.3.2 on different detector geometries.

Comparisons of classification ROC curves between network architectures and our BDT baseline are shown in Figure 17 for ATLAS-like and CMS-like geometries. Here we can see that the previously observed performance ranking still holds true. The GN model performs best, followed by the CNN, then the DNN. All three networks outperform the BDT baseline. The effect is less pronounced after the CMS-like resampling, due to the low granularity and the single detector layer in the z direction.

Regression results are shown in Figure 18 and 19, for photons and neutral pions (we did not train electrons or charged pions for this comparison). Here we have included the regression baselines, DNN networks, and CNN networks, but not GN (which we did not train on resampled



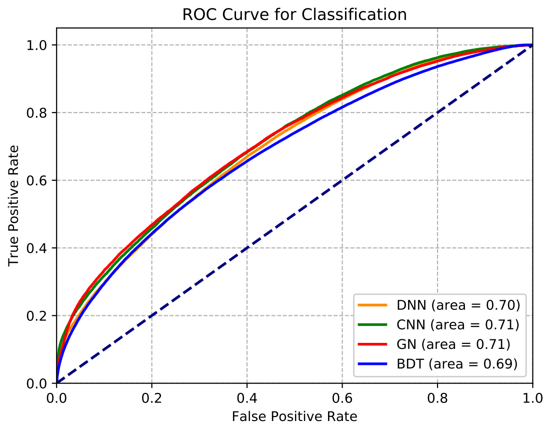


**Fig. 17.** ROC curve comparisons for variable-angle $\gamma/\pi^0$ classification on data resampled to ATLAS-like (top) and CMS-like (bottom) geometries.

data). The results obtained for the ATLAS-like resampling match those on the REC dataset, with DNN and CNN matching the BDT outcome in terms of bias and surpassing it in resolution. With the CMS-like resampling the neural networks match but do not improves over the BDT energy regression resolution. Once again, this is due to the low spatial resolution in the CMS-like geometry, especially due to the lack of $z$ segmentation. We are unable to model the improved energy resolution from the actual CMS detector, so these energy regression results are based on geometry only.

!

## 6 Conclusion and Future Work

This paper shows how deep learning techniques could outperform traditional and resource-consuming techniques in tasks typical of physics experiments at particle colliders, such as particle shower simulation and reconstruction in a calorimeter. We consider several model architectures, notably 3D convolutional neural networks, and we show competitive performance, matched to short execution time. In addition, this strategy comes with a GPU-friendly computing solution and would fit the current trends in particle physics towards heterogeneous computing platforms.

We confirm findings from previous studies of this kind. On the other hand, we do so utilizing a fully accurate detector simulation, based on a complete GEANT4 simulation of a full particle detector, including several detector components, magnetic field, etc. In addition, we design the network so that different tasks are performed by a single architecture, optimized through an hyperparameter scan.

We look forward to the development of similar solutions for current and future particle detectors, for which this kind of end-to-end solution could be extremely helpful.

## 7 Acknowledgemnts

The authors thank Daniel Weitekamp for providing us with the event generator used in regression training. We also thank Andre Sailer from the CERN CLIC group, for guiding us on how to generate the single-particle samples.

This project is partially supported by the United States Department of Energy, Office of High Energy Physics Research under Caltech Contract No. DE-SC0011925. JR is partially supported by the Office of High Energy Physics HEP-Computation. M. P. is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement n$^o$ 772369).

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the State of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

Part of this work was conducted at "*iBanks*", the AI GPU cluster at Caltech. We acknowledge NVIDIA, SuperMicro and the Kavli Foundation for their support of "*iBanks*". The authors are grateful to Caltech and the Kavli Foundation for their support of undergraduate student research in cross-cutting areas of machine learning and domain sciences.

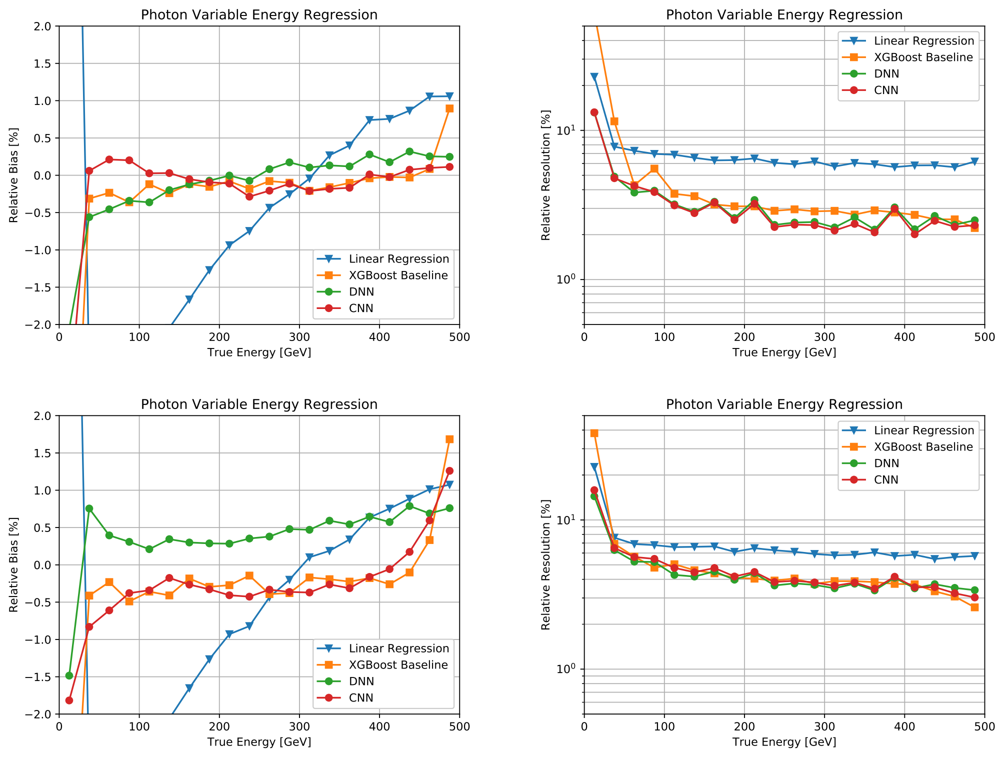


**Fig. 18.** Bias (left) and resolution (right) as a function of true energy for energy predictions for photons, on variable-angle samples resampled to ATLAS-like (top) and CMS-like (bottom) geometries.

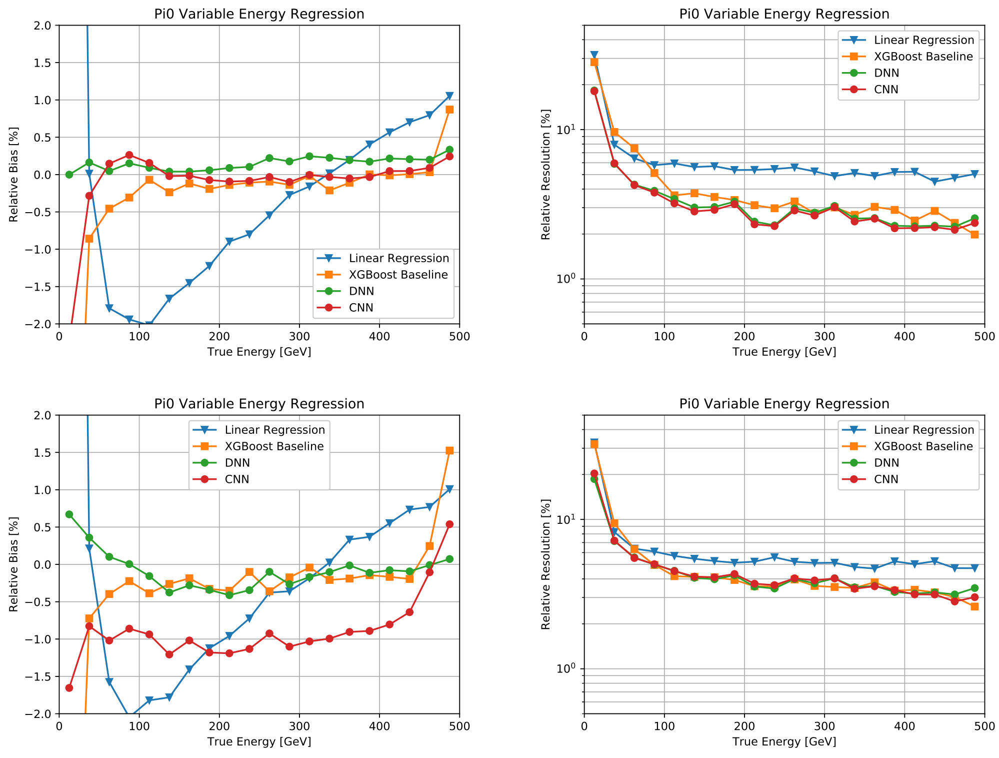


**Fig. 19.** Bias (left) and resolution (right) as a function of true energy for energy predictions for $\pi^0$, on variable-angle samples resampled to ATLAS-like (top) and CMS-like (bottom) geometries.

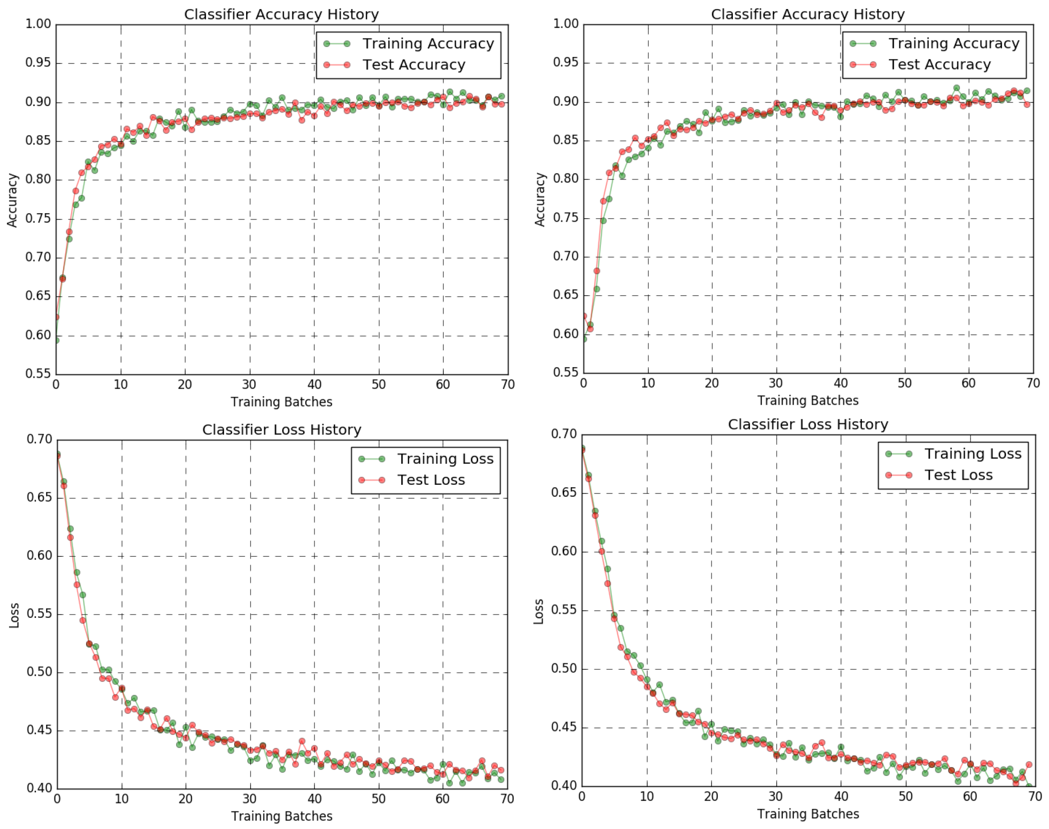


**Fig. 20.** Training history for different choices of the input 3D array zise: Accuracy (top) and loss (bottom) as a function of the training batch for photon/neutral pion classification, using a 25x25x25 (left) and 51x51x25 (right) ECAL window size.

## A Calorimeter Window Size

The optimal window size to store for ECAL and HCAL is an important issue, since this impacts not only sample storage size, but also training speed and the maximum batch sizes which we could feed to our GPUs.

From examinations of our generated samples, we found that an ECAL window of 25x25x51 and an HCAL window of 11x11x60 looked reasonable. To test this hypothesis, we performed training using the samples and classification architectures described in our previous studies [24], but with different-sized input samples. The architecture was altered to accommodate larger windows simply by increasing the number of neurons on the input layer. Results trained using an ECAL window of size 25x25x25 and 51x51x25 are shown in Figure 20. From the similarity of these curves, we have decided that an expanded ECAL window size does not contain much additional useful information, and is thus not necessary for our problems.

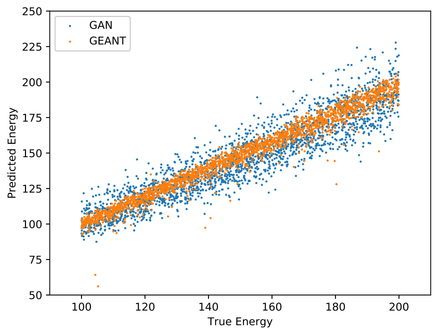


**Fig. 21.** Predicted vs. true particle energy for GAN and GEANT images. Predictions were made using the reconstruction tool described in section 5. This plot was made using 2213 electron events of each type (GAN and GEANT).

## B End-to-end reconstruction of the ECAL showers produced by the 3DGAN

In order to further validate the GAN image quality we run the 3D CNN reconstruction network described in section 5 on the 3DGAN output and compare the response to the results obtained by running the tool on Monte Carlo data. Figure 21 shows a comparison of the energy resolution obtained on GAN and GEANT4 images. The predicted energy shows a reasonable agreement for the mean while the resolution for GAN images seems to be broader than for GEANT4 images. The classification accuracy presented in Figure 22 is very high (close to 100%) for both GAN and GEANT4 events.

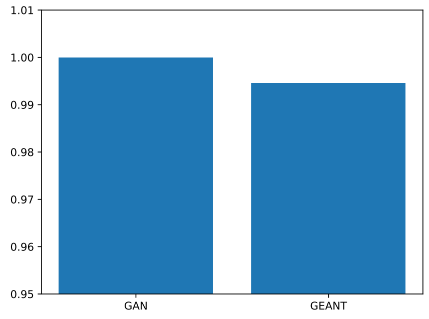


**Fig. 22.** Predicted particle type (electron vs. charge pions) for GAN and GEANT images. There were 2213 electron events for each type.

## C Classification Baseline

Boosted Decision Trees were chosen as the baseline of comparison for our classification task, due to their popularity with HEP experiments. A BDT is effective in processing high-level features, performing complex and optimized cut-based classification in the multi-dimensional space of the input quantities.

The features we use for our baseline BDT classification model, introduced in Ref. [24], are commonly used to characterize particle showers. One additional feature we added is R9, i.e., the fraction of energy contained in a 3x3 window of the $(x, y)$ projection of the shower centered around the energy barycenter. This quantity provides a measure of the "concentration" of a shower within a small region. For values near 1, the shower is highly collimated within a single region, as in electromagnetic showers. Smaller values are typical of more spread out showers, as for hadronic and multi-prong showers. A comparison of R9 values between photons and neutral pions can be seen in Figure 23. After training, the discriminating power of various features can be seen in Figure 24.

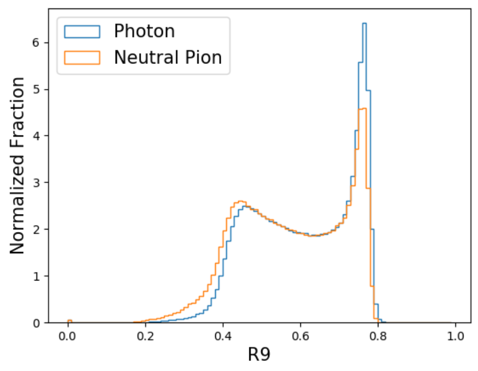


**Fig. 23.** Comparison of R9 distributions between photon and neutral pion events. In both cases, we have some events where energy is centralized, and some events where energy is "half-localized" (maybe split into two regions). Photons tend to deposit their energy more in a single location.

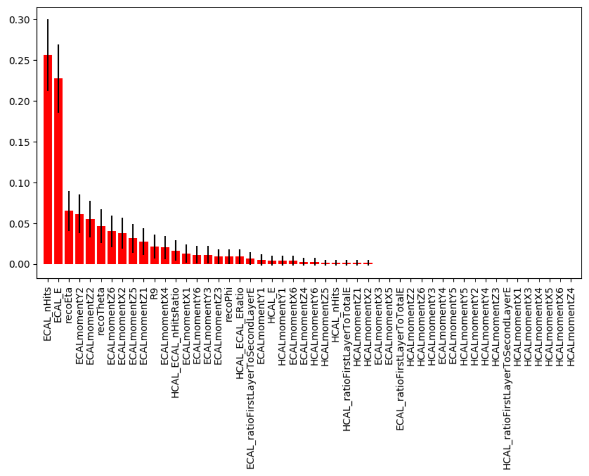


**Fig. 24.** Feature importances for inputs used in BDT training. Values shown are gini importances [30].

## D Energy Regression Baseline

We use linear regression with ECAL and HCAL total energy as one of our baseline methods to compare to machine learning results (seen in Eq. 2).

$$E = a \cdot E_{ECAL} + b \cdot E_{HCAL} + c \tag{2}$$

Updated results for each of the particle types are shown in Figure 25. In all the resolution plots shown, the points have been fitted with the expected resolution function of Eq. 3, and the fitted function is plotted as a line.

$$\frac{\sigma(\Delta E)}{E_{\text{true}}} = \frac{a}{\sqrt{E_{\text{true}}}} \oplus b \oplus \frac{c}{E_{\text{true}}} \tag{3}$$

It is already typical for basic ML methods like BDTs to be used for energy regression in the LHC experiments, in cases where the best resolution is critical (e.g., to study $H \to \gamma\gamma$ decays). We tried a BDT with a few summary features as input to form an improved baseline for comparing more advanced ML techniques. The XGBoost package was used in python, with the following hyperparameters.

- maximum 1000 iterations, with early stopping if loss doesn't improve on the test set in 10 iterations
- maximum tree depth of 3
- minimum child weight of 1 (default)
- learning rate $\eta = 0.3$ (default)

Varying the hyperparameters led to either worse results or negligible changes.

The following features gave good performance for electrons, photons, and $\pi^0$:

- total ECAL energy
- total HCAL energy
- mean $z$ coordinate of the ECAL shower

Adding the mean $z$ coordinate to the ECAL and HCAL total energies improved the energy resolution for all energy values, but in particular at high energy. This is shown in Figure 26 for electrons.

For $\pi^{\pm}$, adding the following variables gave an improved result:

- RMS in the $x$ direction of the ECAL shower
- RMS in the $(x, y)$ plane of the HCAL shower
- mean $z$ coordinate of the HCAL shower

In addition, for $\pi^{\pm}$, around 0.5% of events were found to have almost no reconstructed energy in the selected calorimeter window. Including these events adversely affected the algorithm training, so they were removed for all the results shown in this and the following sections. Specifically, the raw ECAL+HCAL energy is required to be at least 30% of the true generated energy.

The results of the XGBoost baseline are shown in Figure 27, where they are compared to linear regression results. The performance of XGBoost on electrons, photons, and $\pi^0$ is similar, achieving relative resolutions of about 6–8% at the lowest energies and 1.0–1.1% at the highest energies. Compared to the baseline linear regression, the resolution improves by a factor of about two at low energy and three to four at high energy. For $\pi^{\pm}$, the resolution after XGBoost regression ranges between 20 and 5.4%, with a relative improvement over linear regression of up to 40% at high energy.

One drawback of using a BDT algorithm in a real-world setting is that it can not be used for energy values outside the range of the training set. That is, most tree algorithms do not perform extrapolation. This can be mitigated by increasing the range of values present in the training set, or by using another algorithm like a neural network. A small DNN was trained with the features above and was able to achieve similar performance to the BDT, with the same performance at high energy and slightly worse performance at the lowest energies.

## E GoogLeNet Model Architecture Details

In our GoogLeNet architecture, we use inception modules. In these modules, inputs go through four separate branches and are then concatenated together. For an inception layer denoted as Inception(A, B, C, D, E, F, G) the branches are defined as follows:

- Branch 1: A simple $1\times1\times1$ convolution, taking A input channels to B output channels. This is followed by a batch normalization and a ReLU activation function.
- Branch 2: A $1\times1\times1$ convolution followed by a $3\times3\times3$ convolution. The first convolution takes A input channels to C output channels, followed by batch normalization and ReLU. This then goes to the next convolution layer, which outputs D channels using a kernel of size $3\times3\times3$. This is again followed by batch normalization and ReLU.
- Branch 3: A $1\times1\times1$ convolution followed by a $5\times5\times5$ convolution. The details are the same as for the other branches, but the first convolution takes A input channels to E output channels, and the next convolution outputs F channels.

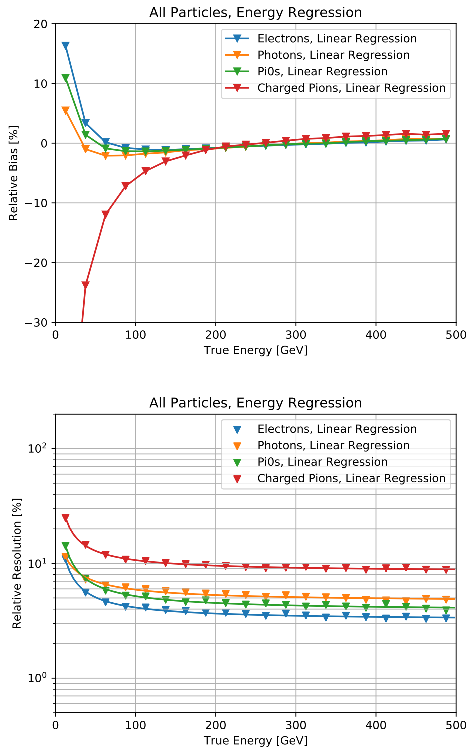


**Fig. 25.** Bias (top) and resolution (bottom) as a function of true energy for linear regression predictions of particle energy for the different particle types, trained on fixed-angle samples.

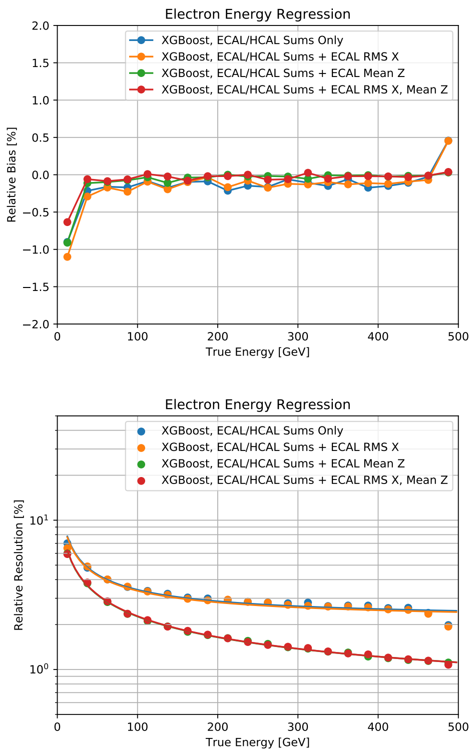
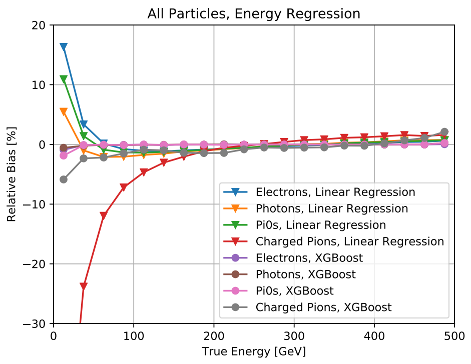


**Fig. 26.** Bias (top) and resolution (bottom) as a function of true energy for the XGBoost regression predictions of particle energy, using different input features for electrons.



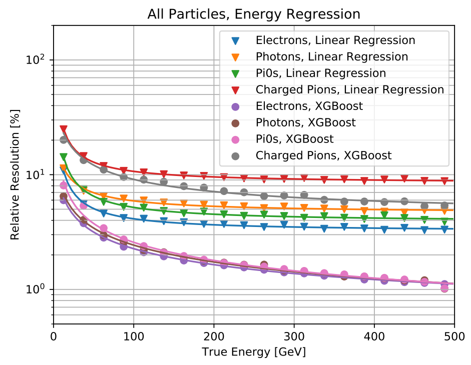


**Fig. 27.** Bias (top) and resolution (bottom) as a function of true energy for linear regression and XGBoost predictions of particle energy for the different particle types.

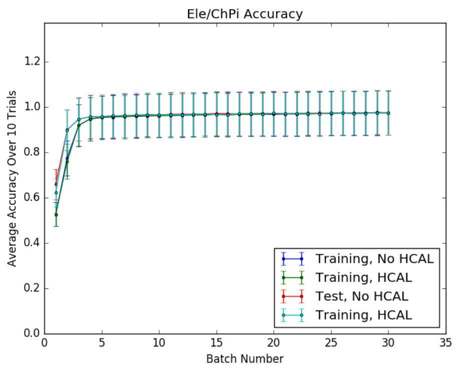
– Branch 4: A max pool of kernel size $3 \times 3 \times 3$ is followed by a convolution of kernel size $1 \times 1 \times 1$ that takes A input channels to G output channels. This is followed once again by batch normalization and ReLU.

Here are full details for each layer of the GoogLeNet-based architecture:

– Apply instance normalization to ECAL input.
– Convolution with 3D kernel of size 3, going from 1 input channel to 192 channels, with a padding of 1. This is followed by batch normalization and ReLU.
– Inception(192, 64, 96, 128, 16, 32, 32)
– Inception(256, 128, 128, 192, 32, 96, 64)
– Max pooling with a 3D kernel of size 3, a stride of 2, and padding of 1.
– Inception(480, 192, 96, 208, 16, 48, 64)
– Inception(512, 160, 112, 224, 24, 64, 64)
– Inception(512, 128, 128, 256, 24, 64, 64)
– Inception(512, 112, 144, 288, 32, 64, 64)
– Inception(528, 256, 160, 320, 32, 128, 128)
– Max pooling with a 3D kernel of size 3, a stride of 2, and padding of 1.
– Inception(832, 256, 160, 320, 32, 128, 128)
– Inception(832, 384, 192, 384, 48, 128, 128)
– Average pooling with a 3D kernel of size 7 and a stride of 1.
– The output array is flattened and concatenated with input $\phi$, $\eta$, total ECAL energy, and total HCAL energy.
– A densely connected layer with 1024 outputs, followed by ReLU.
– The output array is once again concatenated with the same input values.
– A final densely connected layer outputs 5 values, as in the architectures of the other two models.

## F Use of HCAL in Classification

Since the GoogLeNet architecture was quite large and required significant memory usage and computational power, we decided to investigate the possibility of leaving out HCAL cell-level information, since most of the particle shower occurs in the ECAL. Using our best-performing DNN architecture, we ran ten training sessions with HCAL information, and ten training sessions without HCAL. Averaged training curves from these runs are shown in Figures 28 and 29. These studies demonstrated that including the HCAL caused little to no improvement in classification accuracy. For memory purposes, we thus kept HCAL cell-level information out of our GN architecture. Summed HCAL energy was still fed as an input to the combined classification-regression net, for use in energy regression.



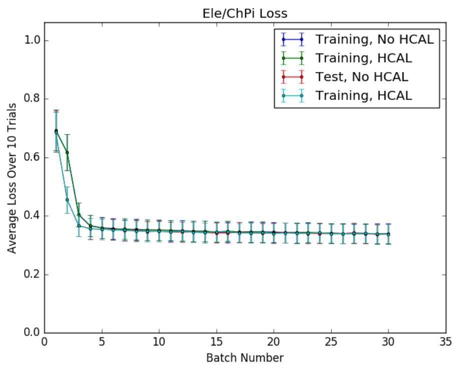


**Fig. 28.** Accuracy and loss curves for electron/charged pion classification, with and without HCAL cells, using best DNN architecture.

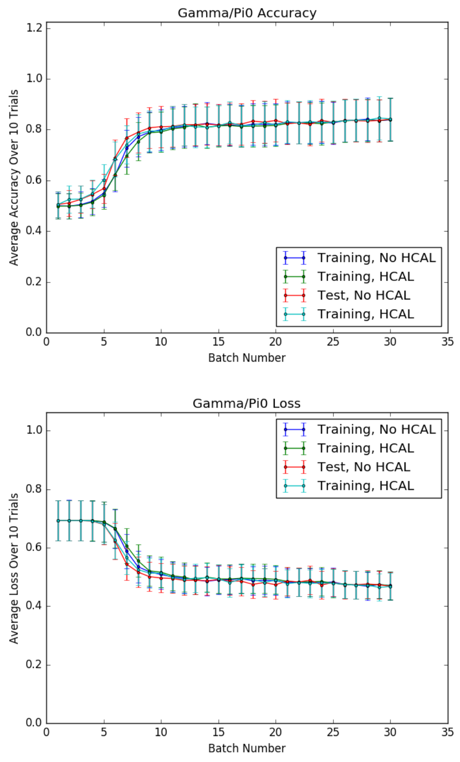


**Fig. 29.** Accuracy and loss curves for photon/neutral pion classification, with and without HCAL cells, using best DNN architecture.

## G Skip Connections for Regression

A design choice that improved convergence time, and improved performance for the CNN, is including "skip connections" for the total ECAL and HCAL energies in the network. In addition to the individual cell energy values, the total ECAL and HCAL energy values are given as inputs to both the first dense layer and to the last output layer. The weights for these energy values are initialized to 1, as linear regression with coefficients near 1 is observed to reasonably reproduce the true energy values. The impact of adding skip connections on performance using a CNN architecture for a fixed number of 5 training epochs is shown in Figure 30.

## H Training for Regression Using Energy Summed in $z$

For regression, we tried using only the energy summed in layers in the $z$ direction, instead of the full array of cell energies, as the mean $z$ coordinate was seen to be the most important additional feature in the XGBoost baseline. The performance is better than the XGBoost baseline at high energies but worse than using the full cell-level information, as shown in Figure 31.

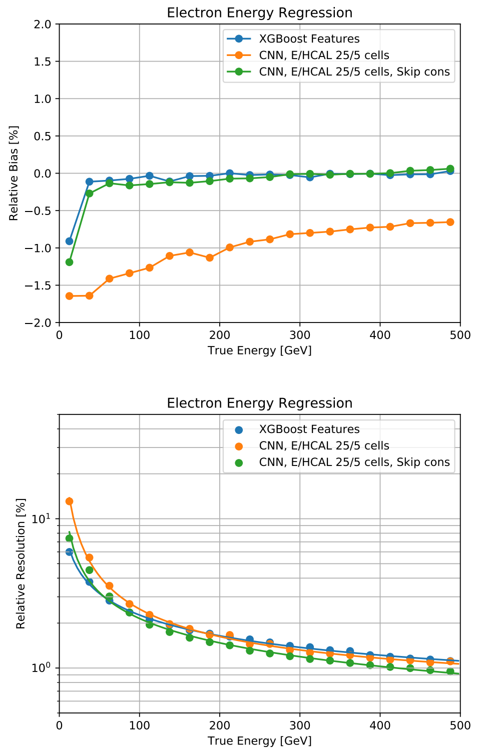


**Fig. 30.** Bias (top) and resolution (bottom) as a function of true energy for CNN energy predictions for electrons, with or without skip connections in the architecture.

## I Energy Regression at Fixed Angles

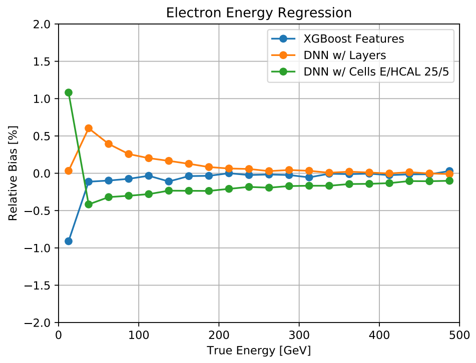
In Figure 32 we show energy regression results when particles impact the calorimeter inner surface at a fixed angle of 90°. All neural architectures and baseline algorithms are able to perform with great accuracy in this regime.

Furthermore, in Figure 33 we summarize performance results on fixed-angle samples for all particle types with the XGBoost baseline and the CNN model.

## J Regression performance training on a different particle type

All the tests so far have assumed that we know exactly what type of particle led to the reconstructed energy deposits. In a real world situation, the particle identities are not known with complete confidence. To see how the algorithms above would cope with that situation, we tried training each algorithm on an input sample of electron events, and then we used the trained algorithm to predict the energies for other particle types.

The results are shown in Figure 34 for predicting photon energies and Figure 35 for predicting $\pi^0$ energies, and are compared to algorithms that are both trained and tested on the same particle type. In each case, a DNN or CNN trained on electrons is able to achieve the same resolution as a CNN trained on photons or $\pi^0$. The bias is slightly larger in some cases.



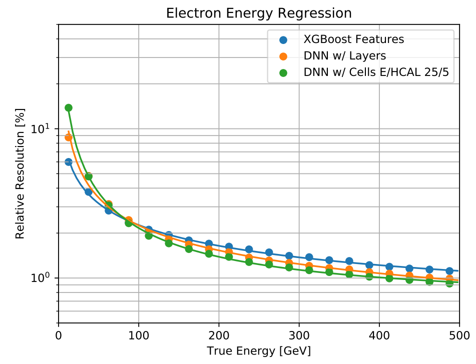


**Fig. 31.** Bias (top) and resolution (bottom) as a function of true energy for DNN energy predictions for electrons, using as input either the energy summed in layers of $z$, or the full cell information.

Models trained on electrons, photons, or $\pi^0$ were found to not describe $\pi^\pm$ well at all. This is not surprising given that $\pi^\pm$ have a hadronic shower, with a large fraction of energy deposited in the HCAL, compared to the other particles depositing almost all of their energy in the ECAL.

We also checked whether the energy regression was different for photons that have converted into an $e^+e^-$ pair through interaction with the detector material. These conversion photons comprise about 9% of the photon sample. We tried training and/or evaluating regression models separately on converted photons compared to all photons (which are dominated by unconverted). The results are shown for XGBoost in Figure 36 and for CNN/DNN models in Figure 37. Worse resolution is seen in each case for converted photons below around 100 GeV, which can be attributed to the subsequent electrons forming two showers instead of one in the calorimeter. With XGBoost, the resolution remains the same for converted photons when training on the full sample, while for CNN or DNN, the resolution is worse below around 100 GeV. The bias is also worse for converted photons at lower energy when training on all photons.

## K Regression Studies with Large Sample Windows

The studies in this section were performed using the full large window samples, of size 51x51x25 in ECAL and 11x11x60 in HCAL. The samples consist of approximately 800,000 events for each particle type. 2/3 of the events were used for training and 1/3 of the events were used for testing.

The most important design choice found for the DNN/CNN networks is the size of the window used as input. For both DNN and CNN, to achieve the best performance for energies above 150 GeV, a minimum $(x, y)$ size of 25x25 in the ECAL and 5x5 in the HCAL is needed. For energies below 150 GeV, the optimal performance is observed for a window size of 51x51 in the ECAL and 11x11 in the HCAL. This is presumably due to wider showers at low energy. The impact of the choice of window size is shown for DNN in Figure 38, with the results for CNN being similar. Drawbacks to the larger window size, however, include larger files, more memory usage, and that training takes about 5 times longer per epoch.

Showers for $\pi^\pm$ were observed to be wider than the other particle types, especially at low energies, and so we compare the effect of the calorimeter window size choice for $\pi^\pm$ in Figure 39. The wider window of 51x51 in $(x, y)$ in the ECAL and 11x11 in the HCAL gives better performance, especially at the lowest energies where the resolution is improved by a factor of about 2 over the smaller window size (25x25 ECAL, 5x5 HCAL).

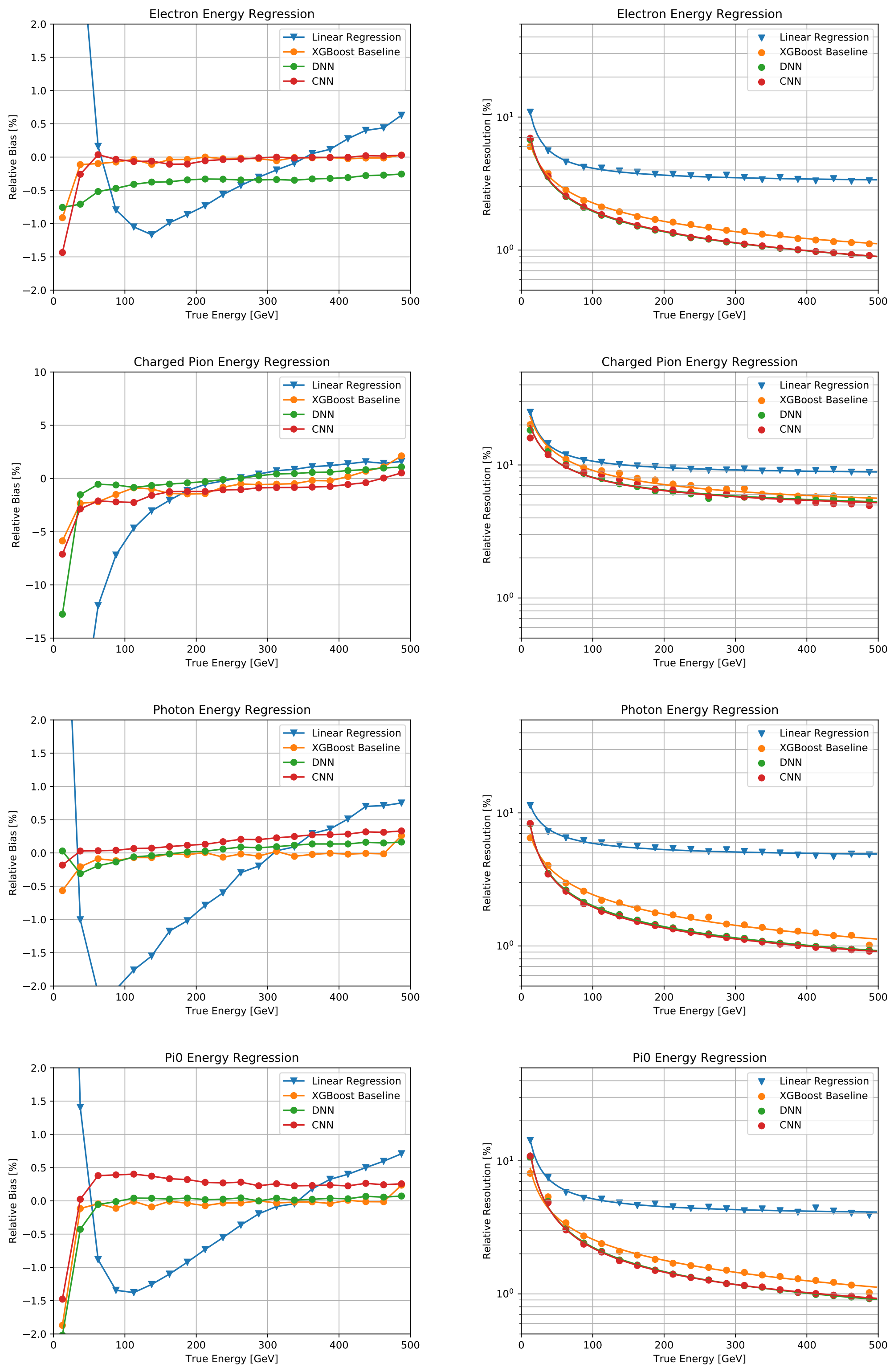


**Fig. 32.** Regression bias (top) and resolution (bottom) as a function of true energy for energy predictions on the REC dataset with fixed incident angle (90°). From top to bottom: electrons, charged pions, photons, and neutral pions.

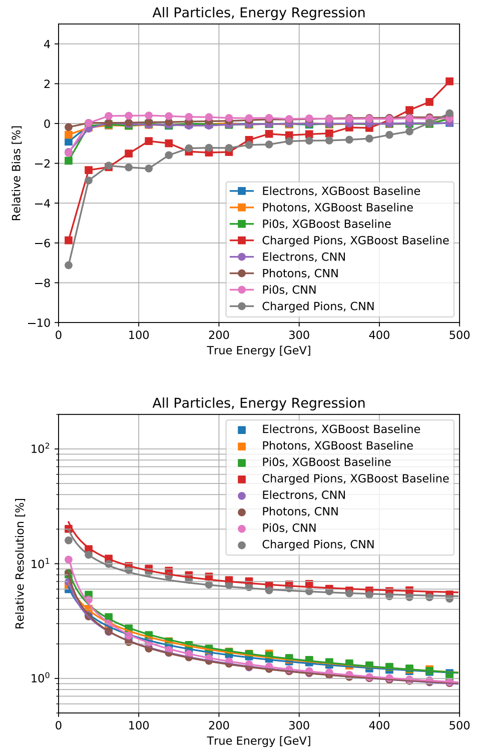


**Fig. 33.** Regression bias (top) and resolution (bottom) as a function of true energy for all particles, comparing the XGBoost baseline with the best CNN model on fixed-angle samples.

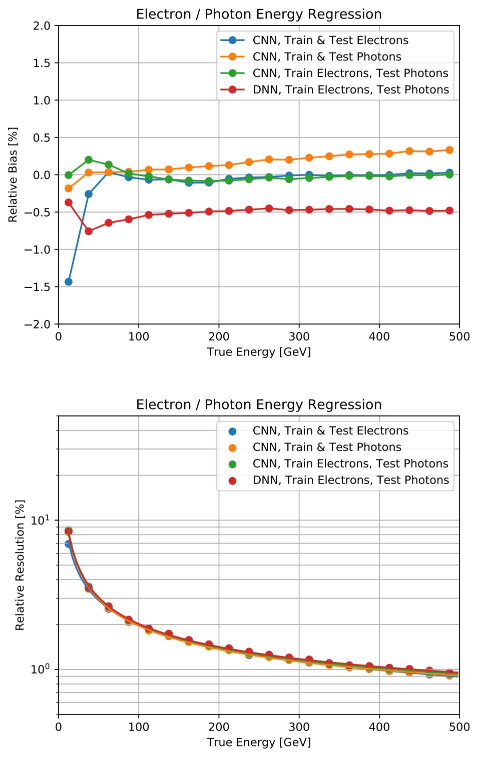


**Fig. 34.** Bias (top) and resolution (bottom) as a function of true energy, for electrons and photons. The particles used to train and test each algorithm are given in the legend.

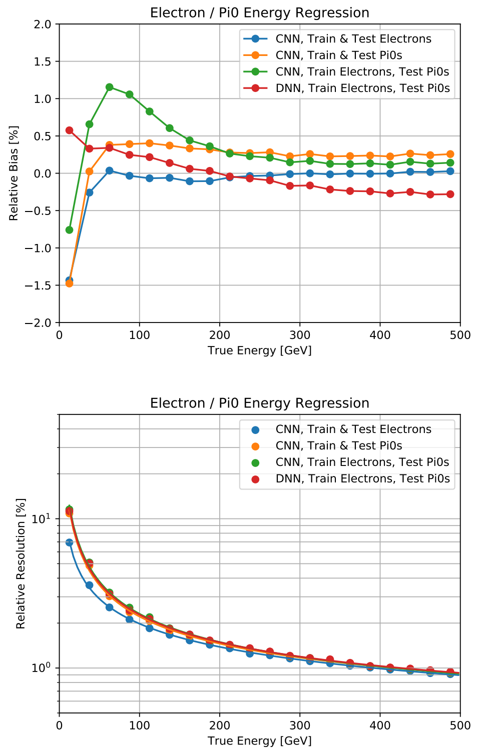


**Fig. 35.** Bias (top) and resolution (bottom) as a function of true energy, for electrons and $\pi^0$. The particles used to train and test each algorithm are given in the legend.

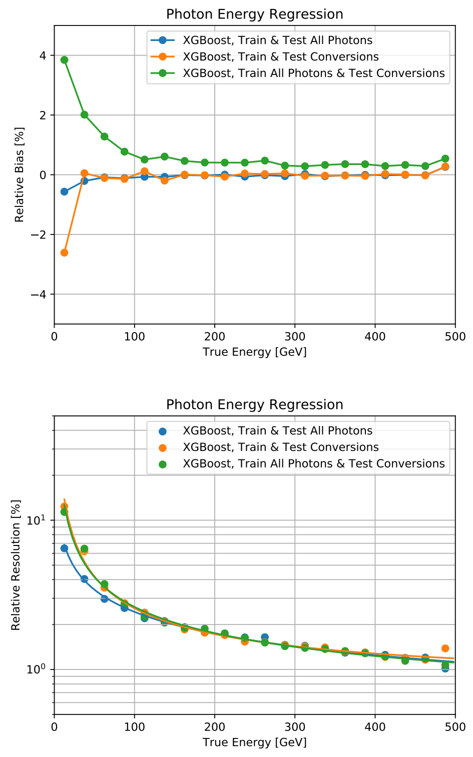


**Fig. 36.** Bias (top) and resolution (bottom) as a function of true energy, for photons using XGBoost regression. We look at the photon sample when split up by conversions.

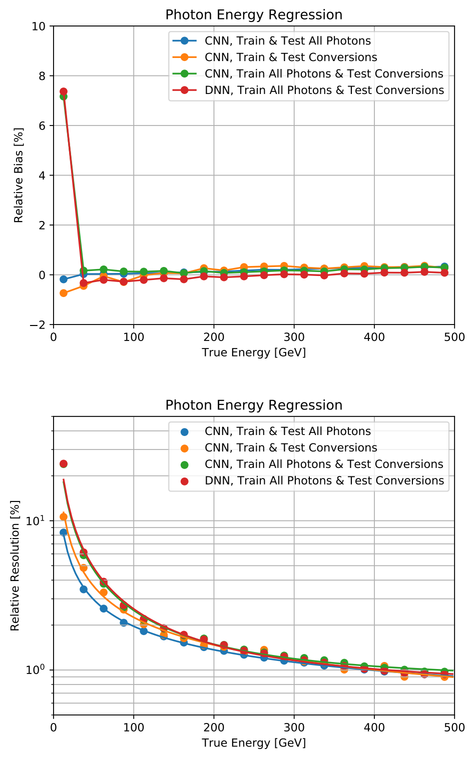


**Fig. 37.** Bias (top) and resolution (bottom) as a function of true energy, for photons using CNN or DNN regression. We look at the photon sample when split up by conversions.

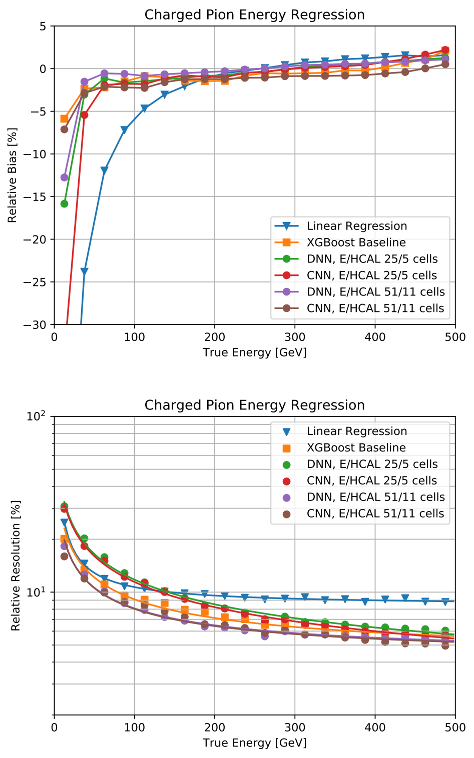


**Fig. 39.** Bias (top) and resolution (bottom) as a function of true energy for energy predictions for $\pi^{\pm}$, comparing calorimeter window sizes for the CNN and DNN models.

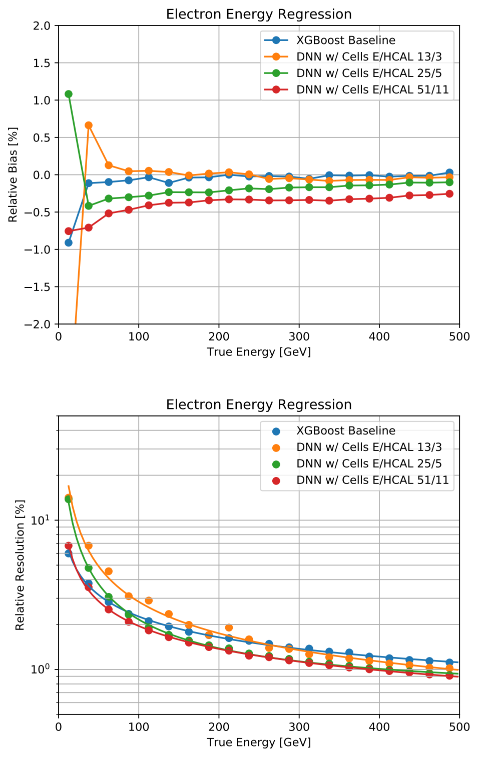


**Fig. 38.** Bias (top) and resolution (bottom) as a function of true energy for DNN energy predictions for electrons, with varying input window sizes.

## References

1. Bruce H. Denby. Neural Networks and Cellular Automata in Experimental High-energy Physics. *Comput. Phys. Commun.*, 49:429–448, 1988.
2. Carsten Peterson. Track Finding With Neural Networks. *Nucl. Instrum. Meth.*, A279:537, 1989.
3. P. Abreu et al. Classification of the hadronic decays of the Z0 into b and c quark pairs using a neural network. *Phys. Lett.*, B295:383–395, 1992.
4. Georges Aad et al. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett.*, B716:1–29, 2012.
5. Serguei Chatrchyan et al. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys. Lett.*, B716:30–61, 2012.
6. G. Aad et al. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, 3:S08003, 2008.
7. S. Chatrchyan et al. The CMS Experiment at the CERN LHC. *JINST*, 3:S08004, 2008.
8. Apollinari G., Béjar Alonso I., Brüning O., Fessia P., Lamont M., Rossi L., and Tavian L. *High-Luminosity Large Hadron Collider (HL-LHC): Technical Design Report V. 0.1*. CERN Yellow Reports: Monographs. CERN, Geneva, 2017.
9. Ties Behnke, James E. Brau, Brian Foster, Juan Fuster, Mike Harrison, James McEwan Paterson, Michael Peskin, Marcel Stanitzki, Nicholas Walker, and Hitoshi Yamamoto. The International Linear Collider Technical Design Report - Volume 1: Executive Summary. 2013.
10. L. Linssen, A. Miyamoto, M. Stanitzki, and H. Weerts. Physics and Detectors at CLIC: CLIC Conceptual Design Report. *ArXiv e-prints*, February 2012.
11. V. Khachatryan et al. Technical Proposal for the Phase-II Upgrade of the CMS Detector. 2015.
12. S. Agostinelli et al. GEANT4: A Simulation toolkit. *Nucl. Instrum. Meth.*, A506:250–303, 2003.
13. Roland Jansky. The ATLAS Fast Monte Carlo Production Chain Project. *J. Phys.: Conf. Ser. 664 072024*, 2015.
14. Luke de Oliveira, Michael Kagan, Lester Mackey, Benjamin Nachman, and Ariel Schwartzman. Jet-images — deep learning edition. *JHEP*, 07:069, 2016.
15. Luke de Oliveira, Michela Paganini, and Benjamin Nachman. Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis. *Comput. Softw. Big Sci.*, 1(1):4, 2017.
16. Michela Paganini, Luke de Oliveira, and Benjamin Nachman. CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks. 2017.
17. Josh Cogan, Michael Kagan, Emanuel Strauss, and Ariel Schwarztman. Jet-Images: Computer Vision Inspired Techniques for Jet Tagging. *JHEP*, 02:118, 2015.
18. I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.
19. Federico Carminati, Gulrukh Khattak, Maurizio Pierini, Sofia Vallecorsafa, Amir Farbin, Benjamin Hooberman, Wei Wei, Matt Zhang, Barin Pacela, Vitorial, Maria Spiropulu, and Jean-roch Vlimant. Calorimetry with Deep Learning : Particle Classification , Energy Regression , and Simulation for High-Energy Physics. In *NIPS*, 2017.
20. François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.
21. Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
22. Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.
23. P. Lebrun, L. Linssen, A. Lucaci-Timoce, D. Schulte, F. Simon, S. Stapnes, N. Toge, H. Weerts, and J. Wells. The CLIC Programme: Towards a Staged e+e- Linear Collider Exploring the Terascale : CLIC Conceptual Design Report. 2012.
24. Federico Carminati, Amir Farbin, Benjamin Hooberman, Gulrukh Khattak, Vitória Barin Pacela, Maurizio Pierini, Maria Spiropulu, Sofia Vallecorsafac, Jean-Roch Vlimant, Wei Wei, and Matt Zhang. Calorimetry with deep learning : Particle classification , energy regression , and simulation for high-energy physics. 2017.
25. Nicolas Palm Perez. Electron identification using machine learning in the atlas experiment with 2016 data, 2017.
26. A. Odena, C. Olah, and J. Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. *ArXiv e-prints*, October 2016.
27. Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Lecture 6a overview of mini–batch gradi-ent descent. 2012.
28. Christian Szegedy et al. Going deeper with convolutions, 2014.
29. Tim Head et al. scikit-optimize/scikit-optimize: v0.5.2, March 2018.
30. Friedman Breiman. *Classification and regression trees*. 1984.