Deep Learning for photon identification and energy measurement in a highly granular calorimeter

Kaustuv Datta,[1,2] Jayesh Mahapatra,[3,4] Maurizio Pierini,[3] Maria Spiropulu,[2] and Jean-Roch Vlimant[2]

[1] *Reed College**
[2] *California Institute of Technology*
[3] *CERN*
[4] *Veer Surendra Sai University of Technology, India*

Deep learning is increasingly being considered for the development of next generation event reconstruction and particle identification algorithms. We explore the veracity of such machine learning techniques, using a dataset consisting of single-particle energy showers in the barrel of a highly-granular Linear Collider Detector calorimeter with a regular 3D array of cells. Prototyping of multiple neural network topologies was carried out for particle classification - discrimination between γ signals and π^0 background, and energy measurement - prediction of energy of photon hits from calorimeter data. Topologies capable of individually carrying out logistic regression or classification, and others that simultaneously addressed both problems, were explored. The performances of these topologies are discussed with details of work in progress, and possible future directions.

I. INTRODUCTION

Traditionally, cut based selections have been used for online and offline analyses at high-energy physics (HEP) experiments. Some older machine learning techniques have also been used after the turn of the millennium, whereby supervised classification using support vector machines [1] and rudimentary neural networks [2] have been used for particle identification. However, the LHC at CERN produces much more data than the previous highest energy collider - the Tevatron at Fermilab, introducing additional challenges for efficient data collection and analysis. Given the slowdown in Moore's Law, coupled with the fact that future High Luminosity LHC data collection is projected to enter the exabyte (10^{18} bytes) regime, older algorithms would discard a large amount of valuable data. Consequently, it would be very difficult to detect new physics signatures efficiently. New directions have to be probed to develop algorithms that can give at least equivalent physics resolution while increasing computational efficiency. As a result, in recent years, efforts have been made to make machine learning a larger part of the toolkit available to high energy physicists, using new advances in computing. Sophisticated methods such as likelihoods, boosted decision trees[3], and Deep Learning[4, 5] are being actively researched for use, or are in use for analyses, in HEP in the current day.

Our work focuses on using Deep Learning for energy measurement and identification of photon events detected in a highly granular calorimeter. This is done by performing logistic regression on the energy of deposits in calorimeter cells (pixels), and the classification of signals (γ events) against background (π^0 events). Our approach concentrates specifically on utilizing neural networks with varying combinations of 2D and 3D convolutional layers followed by different numbers of fully connected layers, with additional variations in the number of nodes per dense layer. The motivation for this is the crucial capability of neural networks to efficiently model complex, non-linear relationships in the solution of both logistic regression and classification problems. In particular, the highly granular nature of the calorimeters in our simulations played an important role in enabling us to use tried and tested image classification and computer vision ideas after necessary feature engineering and data preprocessing were carried out. The mapping of the cylindrical calorimeter geometries to 3D space allows for a layer of abstraction whereby particle shower information is approximated to an image on/with several calorimeter layers, which serve as an analog to different color channels (like RGB) for images. This powerful abstraction that is possible with deep neural networks motivates our usage of such tools for this analysis. However, this is with the knowledge that optimization of these algorithms, and studies of uncertainties, will be made less intuitive, and more difficult.

Our methodology, explicitly discussed in later sections, involved training and carrying out inferences using networks that were either trained to do logistic regression or classification. A separate set of "hybrid" networks was also designed to simultaneously tackle both problems. In this report, we describe our datasets and the simulation framework that was utilized to generate them. We attempt to comprehensively probe the feasibility of our different deep learning approaches, with considerations of further algorithmic optimization. Full descriptions of the different topologies utilized are presented in the appendices along with links to the GitHub repository for the relevant algorithms that were designed for this project.

II. EXPERIMENTAL SETUP

All neural networks were built and studied using the highly modular and easy to use Keras 1.0.8 Deep Learn-

* dattak@reed.edu

ing libraries, utilizing the Theano 0.8 backend. All code was written in Python 2.7.

Initial prototyping of, and final inferences with, neural network topologies were performed on the Caltech titans system. This system had two available NVIDIA GeForce 900 series GTX TitanX graphics processing units (GPU), built on the 28 nm Maxwell architecture, coupled with an Intel Core i7-5960X CPU. Given the large sizes of the dataset, the memory size and bandwidth of the TitanXs were beneficial in allowing for large batch sizes, and reduced data bottlenecks, while working with large network topologies. The specifications of the GPU are as listed in Table II.

CUDA Cores	Processor Clock (MHz)	Memory Size (GB)	Memory Bandwidth (GB/s)
3072	1000/1075	12	375

TABLE I: Manufacturer specifications of the NVIDIA GeForce GTX TitanX

The latter phase of our work, using more resource intensive network topologies, were conducted on the Piz Daint supercomputer at the Swiss National Supercomputing Centre (CSCS). Due to the availability of a large number of nodes, each equipped with an NVIDIA Tesla K20X GPU and an Intel Xeon E5-2670 CPU, several networks could be trained simultaneously without issues.

CUDA Cores	Processor Clock (MHz)	Memory Size (GB)	Memory Bandwidth (GB/s)
2688	732	6	250

TABLE II: Manufacturer specifications of the NVIDIA Tesla K20X

III. LCD FRAMEWORK AND EVENT REPRESENTATION

The open access DDhep simulation framework was utilized for event generation, using an implementation of the preliminary design of a CLIC detector. This Linear Collider Detector (LCD) package contained a GEANT4 simulation of high granularity calorimeters, whereby particle and detector material interactions were accurately recreated to give the best approximation to real data.

A particle gun approach was used to generate events instead of carrying out full simulations. Individual photons and π^0s were shot in the transverse (x-y) plane of the *global* coordinate system, along the x axis. The z axis corresponded to the beamline of the detector. Each particle travels through the simulated LCD tracker before impinging upon the inner surface of the calorimeter and showering. Our dataset included information of energy deposits from these showers, in cells of both the LCD's high-granularity Electromagnetic Calorimeter (ECAL) and Hadronic Calorimeter (HCAL).

Calorimeter

The ECAL, with a 1.5 m inner radius, consists of 25 tungsten absorber layers, with silicon sensor cells between the layers. Each such cell is a 5 mm×5 mm segment of the detector plane. The HCAL, situated behind the ECAL, consists of 60 steel absorber layers with polystyrene scintillator sensors between the absorbers. Each of the HCAL sensor cells had dimensions 3.0 cm×3.0 cm, corresponding to a 6×6 array of ECAL cells as shown in Fig. 1.

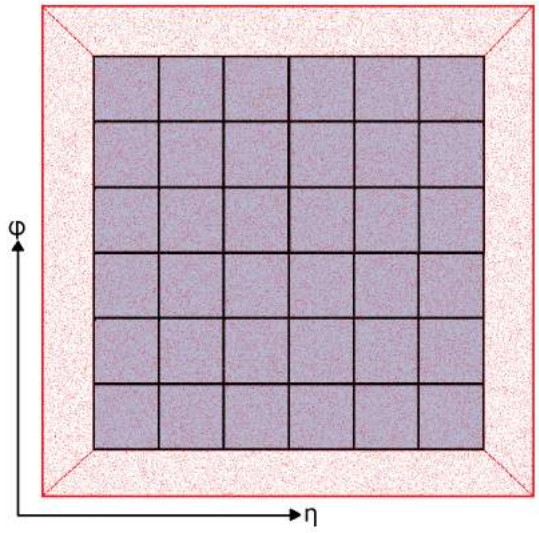

FIG. 1: Pictorial representation of overlay of one HCAL cell (red) corresponding to 6×6 ECAL pixels (grey), in $\eta - \phi$ space.

Barycenter Calculations

Detector cells are characterized by the energy deposits in them, and the cell position indices (iX, iY, iZ), where these three *local* coordinates are represented by signed integer values, with units of the cell size. Thus, in comparison to the *global* coordinate system, iZ is oriented along the x axis, while the iX-iY plane is parallel to the y-z plane. The origin of this *local* coordinate system is set at the point of first contact of the incoming particle and the inner surface of the ECAL.

The barycenter $(i\bar{X}, i\bar{Y})$ of each shower event in the ECAL was calculated by taking a weighted average of

energy deposits in all cells,

$$i\bar{X} = \frac{\sum_{iX} iX \times \sum_{iY,iZ} E(iX,iY,iZ)}{\sum_{iX,iY,iZ} E(iX,iY,iZ)}, \quad (1)$$

$$i\bar{Y} = \frac{\sum_{iY} iY \times \sum_{iX,iZ} E(iX,iY,iZ)}{\sum_{iX,iY,iZ} E(iX,iY,iZ)}, \quad (2)$$

where $E(iX, iY, iZ)$ is the energy deposited in the cell referenced by (iX, iY, iZ). In the most recent iteration of this algorithm, a 25×25 array of cells is chosen around the calculated barycenter (located at $i\bar{X} = i\bar{Y} = 12$), for all 25 sensor planes in the ECAL. This forms the ECAL data array of size $25 \times 25 \times 25$ for any given event. This ECAL barycenter is then mapped to the HCAL, by translating along the line between the origin and ECAL barycenter in terms of the *global* coordinate system. The $4 \times 4 \times 60$ HCAL data array is constructed similarly, by storing information from the 4×4 array of cells around the barycenter location in each of the 60 scintillator planes in the detector. It is to be noted here that a slightly different, previous version of the barycenter calculation algorithm was used for creating datasets for the network topologies that are discussed in this report. In that version, the ECAL data arrays had a shape of $24 \times 24 \times 25$, while the HCAL data still had the same shape as mentioned above.

The barycenter calculation was the penultimate, and a very necessary, step before training network topologies. This feature engineering procedure enabled the abstraction of our particle physics data to a computer vision problem. This step also minimized the size of our datasets by only storing information from a fixed range of cells around the barycenter, as opposed to including null data from calorimeter cells that did not witness the energy shower from a given event.

IV. DATASETS AND DATA FEEDING

Two LCD datasets were used in our study. The first toy dataset was used for gaining experience in working with neural networks in Keras, and gauging the feasibility of different topological approaches. The second was used to formalize our ideas and extensively test our network topologies for classification and regression on a larger and more challenging dataset.

Our dataset is initially in the form of ROOT output files from the LCD simulation. These were then converted to .txt files with the necessary information extracted from the initial outputs. On running the barycenter calculation algorithm, our dataset was then converted into a set of compressed HDF5 files. Two keys were assigned for "images" - the readouts of the shower from the ECAL and HCAL for a given event, as described in the previous section. Corresponding to each event, "target" keys were also assigned which detail, in increasing order of column number, particle ID - 0 for pions and 1 for photons, recorded energy of the hit, and the momentum 3-vector. Given the large number of events in the dataset, the compressed nature of the HDF5 file format was essential in reducing disk usage.

Data Generator

At the time of training, images and targets were fed into the neural networks with the aid of a custom-designed data generator. In working with small datasets, it is often convenient to load an entire dataset onto memory during training, but for large datasets like the ones used in this study, this is not possible. In our case, it was advantageous to use the data-generator to create an archive of pre-processed data files. Three such data generator classes were designed, each specifically tasked with sourcing data to classification, regression and hybrid topologies, respectively. Each of these generators, depending on a necessary argument that can be passed in the function calls, have the capability to reshape data to match with different required input shapes for dense, and 2D and 3D convolutional, input layers.

In general, once provided a list of data files, the data generator takes care of splitting the file-list into training, testing and validation sets by using the train(), test() and validation() methods in the class, respectively. Given a batch size during the training call, events are fed into a batch till the required batch size is satisfied, at which time a single batch is yielded to Keras' fit_generator() function for training and validation. Batches are continuously yielded till the specified training and validation sample sizes per epoch, ie - total number of events for a training iteration, are satisfied. The data generator keeps a record of where it left off after reading from a file, resuming from the same point when it receives the next call from the fit_generator. The data generator iteratively goes through the file-list, yielding batches of specified sizes indefinitely. Thus, at any given time only one out of all the files in the dataset is open, reducing the load on computational resources.

Toy Dataset

The readouts from cells surrounding the barycenter, for different layers of the ECAL, were stored in the .h5 (HDF5) format as $20 \times 20 \times 25$ sized 3D arrays under the "images" key, as per a previous version of the barycenter calculation algorithm. The energy and type of the particle, for the corresponding shower event, were stored under "target" keys. Events were of discrete, integer-valued energies over the range 10-109 GeV. The final merged files each contained 10,000 events, and were labeled according the specific energy of events that they contained. A total of 20 files were created for each energy point with 500 events per file. Two such datasets were created for π^0 and γ data, with a total of two million events.

As a final data preprocessing step, all the files for a given energy were concatenated. The resulting 100 files, with 10000 events, were shuffled 500 times (an adequate number of times to remove possible biases). In every iteration, the shuffling algorithm opened two randomly chosen files, and stored the image and target of each event in both files in two different numpy arrays. Then the two arrays were shuffled using the same random seed, to not lose correspondence between images and targets of a given event. The numpy arrays were then split into half and stored back in their original files. This formed the final dataset of 100 files with gamma events of all energies, present in each file.

CaloImage Dataset

The events presented in this dataset contain information of energy deposits in the barrel region of both the ECAL and HCAL, over a flat energy spectrum between 10-500 GeV, from single particles hitting ECAL's inner surface and subsequently showering. Including HCAL information is important to image the tail of hadron showers, and see more of higher energy electron showers continuing into the HCAL from the ECAL, in addition to providing increased generalization of our methodology. In general, more data, including possibly more identifiable features in data, is extremely important from the deep learning point of view, and justifies the inclusion of HCAL readouts to our dataset.

The energy spectrum of γ and π^0 events in this dataset are shown in Fig. 2 and Fig. 3 respectively. As can be seen in the aforementioned figures the energy spectra are fit well by a uniform distribution function.

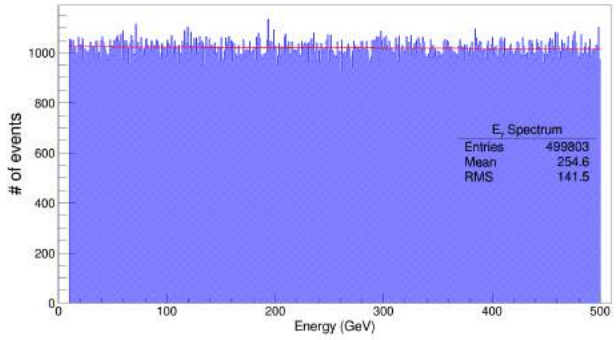

FIG. 2: Photon (γ) event energy spectrum for CaloImage Dataset.

A total of 50 .h5 files, with approximately 10000 events each, were created for both π^0 and γ events. Thus, for each species this dataset contains approximately 500000 events. The π^0 and γ files were concatenated, and processed to form 100 files each with 10000 events, with the events split evenly between the two particle species. Further, file shuffling using an algorithm similar to the one

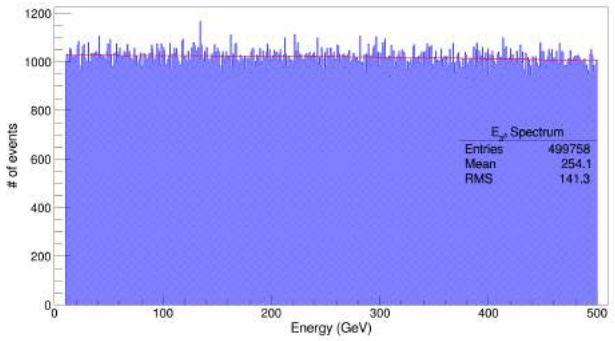

FIG. 3: Neutral pion (π^0) event energy spectrum for CaloImage Dataset.

used on the toy dataset, was carried out 500 times to ensure we had an unbiased dataset with an even mix of particle types.

During the training, validation and testing of classification and hybrid topologies, this shuffled dataset was utilized. For topologies only carrying out regression, the original γ events were used since each file in the dataset contained events with a uniform energy spectrum, minimizing chances of biasing the training towards any specific sub-range of energies in the dataset.

V. NEURAL NETWORKS

Neural network topologies that were trained and tested on the CaloImage Dataset are described here, while the results of those studies are presented in the following section. One of the distinct features of this dataset was that it provided two kinds of input data per event - readouts from both the ECAL and HCAL, each of different array shapes. This had an important impact on the design of our neural networks, requiring that we use branched topologies to simultaneously input both ECAL and HCAL data into the network. The flowcharts with specific details of the network topologies - number of sequential convolutional layers, fully connected layers and number of nodes per layer, are listed in Appendices B, C and D.

Training and Inference

All of these topologies were trained using Keras' fit_generator(), in tandem with our data generator. The training data included the ECAL and HCAL and provided corresponding target value(s) - particle ID and/or energy of the registered hit. Given several samples to train on, the neural networks began to learn what signals at different energies "look" like, or how a γ shower looks compared to π^0 showers. However, it was imperative to prevent over-fitting of the networks, which was achieved

by utilizing Keras' EarlyStopping() function to monitor validation losses and exit training if losses did not decrease beyond a threshold within a given "patience" limit (number of epochs before early stopping occurs).

Inference - prediction with neural networks, was then carried out on relevant test samples, previously unseen to the networks during the training and validation processes, using the test() function of the data generator class. It was imperative that the inferences were carried out on unseen samples both to measure and compare performance of different topologies, and to ensure that overfitting of neural networks did not occur in the training process.

Network Architectures

Similar network topologies were utilized in the solution of both the regression and classification problems, as shown in Fig. 12 - 26. Both 2D and 3D Convolutional layers, followed by 2D and 3D MaxPooling layers respectively, were utilized to compare their relative performances, while varying the number of fully connected layers after the merging of the input branches. Branched models with Dense input layers were also benchmarked. The models for both regression and classification utilized sigmoid as the activation function for all fully connected layers, and rectified linear units (ReLU) for input layers. Stochastic gradient descent was used as the loss optimizer for all of the topologies presented here, with Keras' option of using Nesterov momentum switched off. The output neurons for classification used sigmoid as the activation function, since a sigmoid's output varies between 0 and 1, which were our customized particle ID's for neutral pions and photons, respectively. However, in the regression topologies, a linear activation function was utilized due to the nature of the problem being solved. The loss functions used for classification and regression were binary cross-entropy and mean squared error, respectively.

It must be noted that other activation functions such as ReLU were experimented with in the fully connected layers. However, topologies with this activation performed very poorly due to the "dying ReLU" problem, whereby a large gradient passing through a ReLU activated neuron causes the weights of the neuron to update in a way that the neuron effectively "dies". This essentially implies that such neurons will irreversibly stop activating on input, and always output the same null or zero value. Under these conditions, and given that the stochastic gradient descent was used as the optimizer, once a ReLU neuron dies it would be very unlikely for it recover. This is because the gradient at 0 is 0, leading to no further updating of weights of the neuron. Similar problems were faced while using the randomized ReLU (rRELU) activation as well, while the parametrized ReLU (pReLU) function required more memory than was available to us at the time of training of these networks. The sigmoid activation function conveniently resolved these problems, and was thus used in all our networks.

In addition to the networks that performed only regression and classification, we also attempted the more difficult and somewhat unusual approach of optimizing hybrid networks that tried to solve both problems simultaneously. This hybrid network approach was attempted since it avoids the need to train two similarly structured, but isolated networks. In such a network that attempts both operations, it is expected that the usage of more than one label (both particle ID and energy of hits) for training samples would give the network an advantage in terms of more completely "visualizing" which "images" of signals against background correspond to certain energies of hits, for a specific particle type. The simple motivation for this approach was that it would allow the network to correlate both energy deposits in calorimeter cells and particle classes to input "images", possibly improving physics sensitivity. This approach was not significantly less successful, and achieved their purpose without much decrease in computational efficiency compared to single purpose networks.

VI. ENERGY REGRESSION PERFORMANCE

The performance studies of the different network topologies that were used for regression are detailed in this section. In addition, the energy regression performances of hybrid and single purpose network topologies have been compared to give us further ideas for optimization. Observables such as mean and median prediction, percentage uncertainty of predictions, residual and z-score have been studied as a function of the energy of registered hits.

The variation of these chosen observables with energy give us an idea about the energy regimes the networks perform best in. In addition, they also give us an idea about what tweaks or feature engineering steps should be made to further improve their performance. These observations and means of further optimization are discussed below.

Mean and Median Energy Predictions

The variation of the mean and median energy predictions of photon events, as a function of energy, are studied in Figs. 4 and 5, respectively. The "perfect prediction" line is shown as a dashed light-blue line, to give an idea of how much predictions by different network topologies vary from the actual energy target value of events.

As far as our methodology of measuring performance is concerned, since the CaloImage Dataset was a flat spectrum of energies we compared predictions to target energies rounded off to the nearest whole number (ie - within a ± 0.5 window around integer valued numbers in our range of energies). The mean and median prediction

for each such rounded off the target value was calculated by testing our networks repeatedly on each such energy point corresponding to different events by creating numpy arrays from the imported data from the .h5 files the networks had not seen during the training process. The median (calculated from a sorted array of the data points) and mean predictions were assumed for each energy point between 10-500 GeV as shown below,

$$Mean : \langle \hat{x} \rangle = \frac{1}{N_{event}} \sum_{i=1}^{N_{event}} x_{i,predicted} \tag{3}$$

$$Median : \langle \hat{x} \rangle = \begin{cases} \left(\frac{N+1}{2}\right)^{th} \ term, & N \ is \ odd \\ \frac{\left(\frac{N}{2}\right)^{th} \ term + \left(\frac{N+1}{2}\right)^{th} \ term}{2}, & N \ is \ even \end{cases} \tag{4}$$

where N is the number of events of in the prediction array, for a given energy target in the testing dataset.

As can be seen in Figs. 4 and 5, most network topologies perform well in the energy range $60-400$ GeV energy range. As would have been expected, branched dense neural networks, both hybrid and regression-only, performed worse than the other branched convolutional 2D and 3D topologies. The best performance is shown by the branched topologies with chained covolutional layers - bcnn4_regcl, bcnn4_reg, bcnn5_reg, bcnn5_regcl. Decreased performance in the higher energies between 400-500 GeV, points to the difficulty of neural networks in discriminating those kinds of events. At higher energies specifically, this might be because the images become increasingly similar to each other due to the increased collimation of the particle showers in the calorimeters.

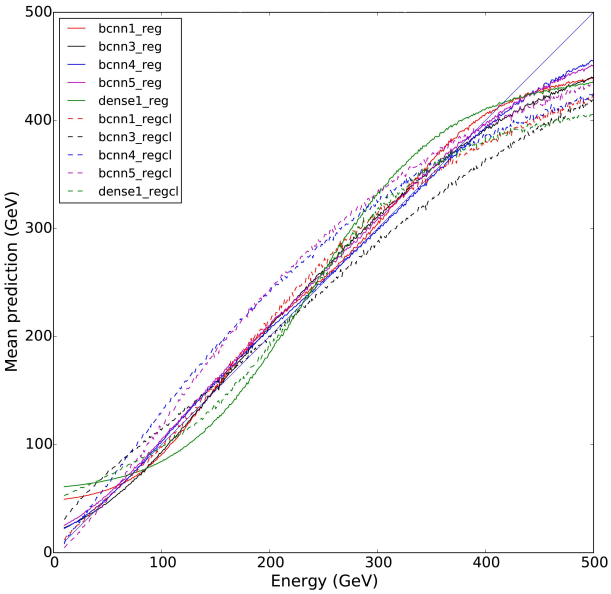

FIG. 4: Variation of mean prediction of energy on unseen data, as a function of energy.

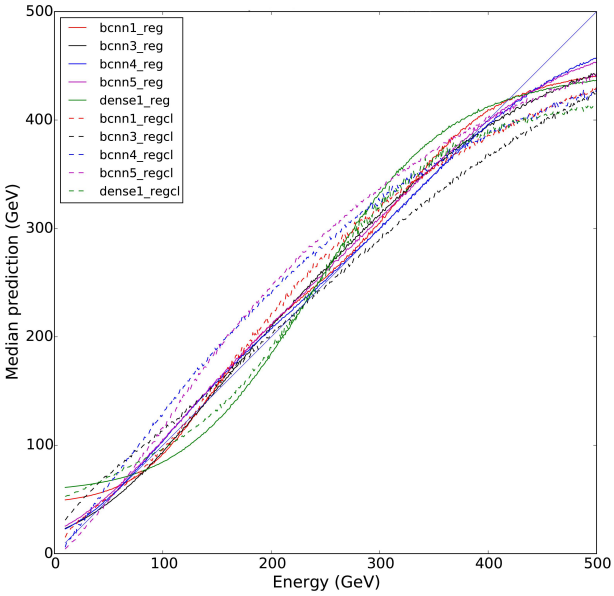

FIG. 5: Variation of median prediction of energy on unseen data, as a function of energy.

Residual Calculations from Predictions

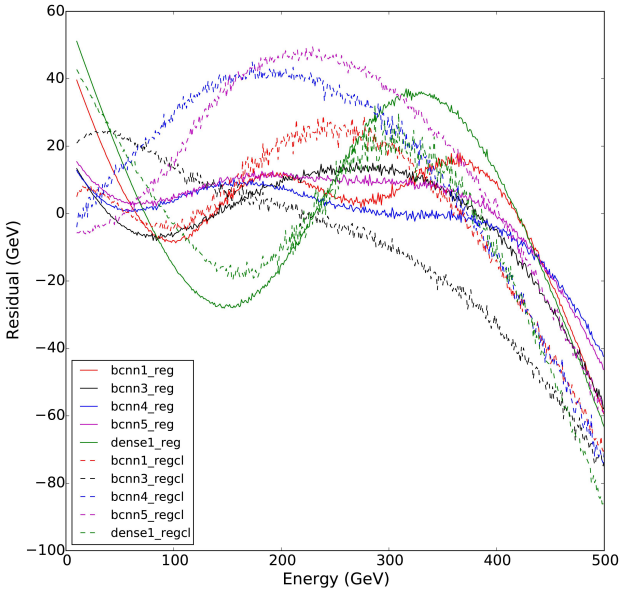

FIG. 6: Variation of difference between target and median predicted energies, as a function of energy.

In regression analysis, the difference between the observed value of a dependent variable and the predicted value is called the residual (e). The residual was used to quantify the deviation of the median of predictions,

at each energy point, from the actual target energy by calculating as follows,

$$e = \langle \hat{x} \rangle - x_{target} \tag{5}$$

where, $\langle \hat{x} \rangle$ is the median of the array of predictions for a given energy. This was useful in studying the relative error in different energy sub-ranges of our dataset. As seen clearly, in Fig. 6, all of the network topologies performed most accurately in the 250-400 GeV subrange of energies from our dataset. It is also easy to identify that the hybrid topologies performed significantly worse than the single-purpose regression topologies, over the entire range of energies. However, noticeably, we can see that the branched, chained convolution 2D and 3D topologies of bcnn4_reg and bcnn5_reg tail off the least at higher energies, while showing the least deviation in the lower energy ranges. This allows us to conclude that chaining convolutional layers might be key to improving performance in the context of the regression problem.

Percentage Uncertainty in Predictions

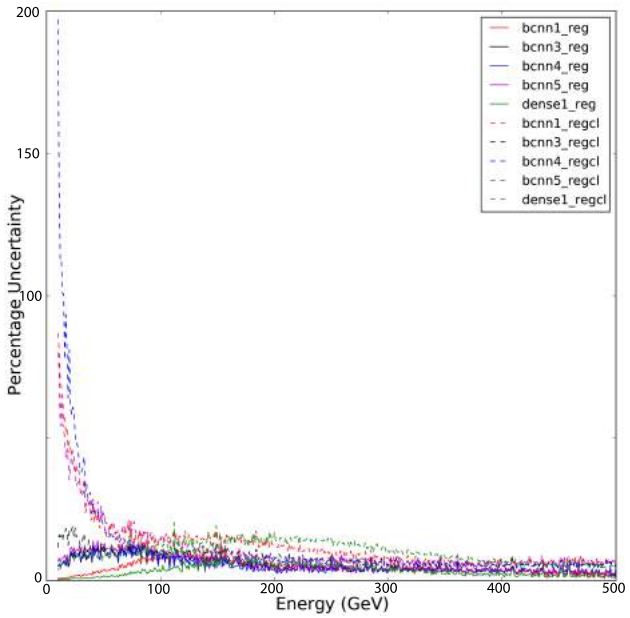

FIG. 7: Variation of percentage uncertainty in prediction of energy, as a function of energy.

The percentage uncertainty ($\Delta x\%$) in predictions is another quantitative measurement of the accuracy of our networks, where the fractional uncertainty calculation is essentially weighted or normalized by the energy value in the denominator as shown,

$$\Delta \hat{x}\% = \frac{\sigma_x}{\langle \hat{x} \rangle} \times 100\%, \tag{6}$$

where σ_x is the standard deviation of predictions, and $\langle \hat{x} \rangle$ is the median of predictions for a given energy point. From Fig. 7, it is noticeable that the low-energy performance of the hybrid topologies, even those with chained convolutional layers is quite poor.

Z-Score of Predictions

The Z-score (z), also referred to as the normal deviate, is a numerical measurement that is a common statistical tool for standardizing data, to enable quantitative comparisons.

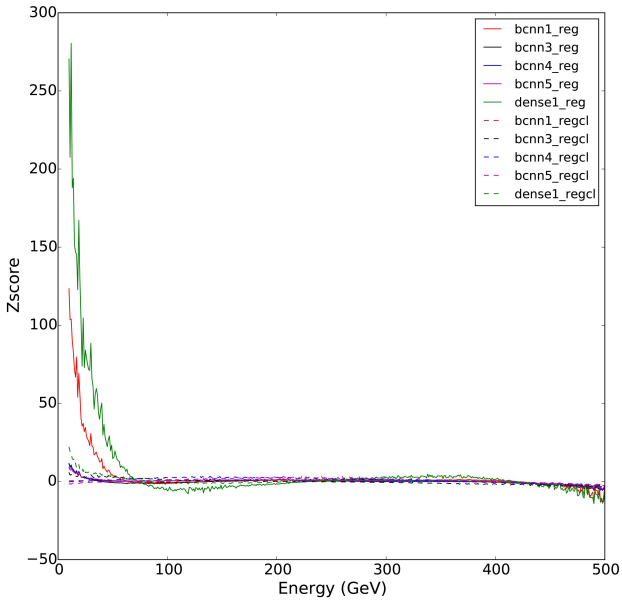

FIG. 8: Variation of z-score of energy predictions, as a function of energy.

The Z-score of predictions for our neural networks, at a given energy point was calculated by,

$$z = \frac{\langle \hat{x} \rangle - x_{target}}{\sigma_x} \tag{7}$$

$$= \frac{e}{\sigma_x}, \tag{8}$$

where σ_x is the standard deviation of predictions, e is the residual at that energy point, and $\langle \hat{x} \rangle$ is the median of predictions.

A zero Z-score is ideal, indicating that the median score equals the target. This score may also be positive or negative, indicating the prediction is above or below the target value, respectively. Thus, it can be noted from Fig. 8, that while a few models do have problems with predicting much higher energies than the actual targets, at lower energy levels, the majority of networks have predictions lower than the targets in the higher energy ranges and perform at a similar level to each other for the middle-range of energies.

VII. PARTICLE CLASSIFICATION PERFORMANCE

The performance studies of the different network topologies that were used for classification are detailed in this section. It should be noted, in Figs. 10 and 11, we were unable to plot the area under the curve or variation of false positive rate for all hybrid topologies, and the bcnn5_cl single-purpose topology, due various technical issues including an unfortunate corruption of the files with training weights saved for these networks. However, the receiver operator characteristics were studied, as shown in Fig. 9, before the loss of the weight files, allowing for at least a qualitative comparison of discrimination power of hybrid and single-purpose topologies.

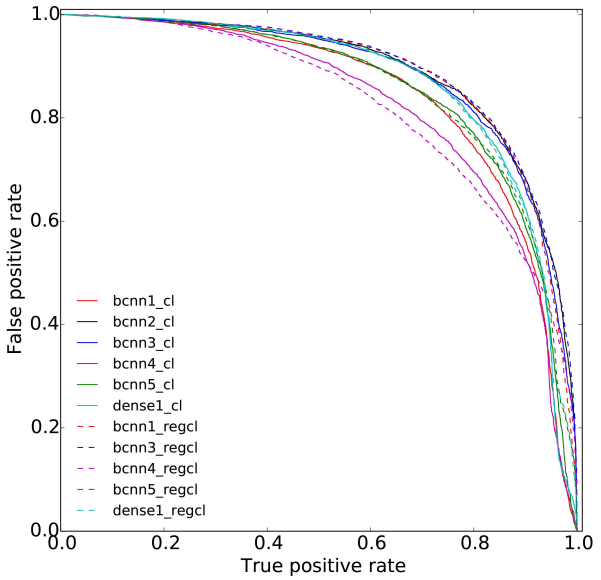

FIG. 9: ROC curves for particle classification for both single-purpose and hybrid network topologies.

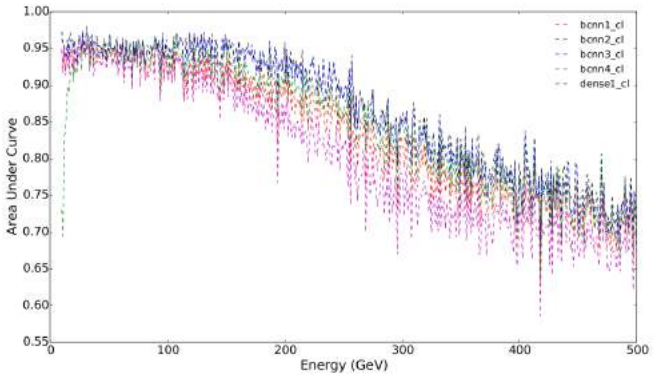

FIG. 10: Variation of area under the ROC curve as a function of energy, for single-purpose classification topologies.

The receiver operator characteristics (ROC) curve was plotted, to study the ability of the network topologies to discriminate γ signals from π^0 background. In Fig.9, single-purpose and hybrid networks with similar architectures show similar classification efficiency. This is indicative of the fact that increasing number of fully connected layers or the number of nodes gives us diminishing returns. This allows us to then determine that optimizing number of convolutional layers, prior to the fully connected layers, is key to maximizing the discrimination power of topologies.

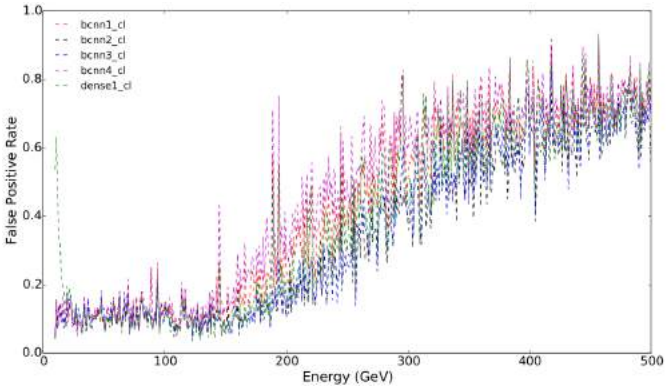

FIG. 11: Variation in false positive rate at 90% true positive rate, as a function of energy, for single-purpose classification topologies.

The area under the ROC curve (AUC) for the classification only models was also studied, as a function of energy, to quantify the variation in discrimination power with the energy of registered hits on the calorimeters. Since the area under a ROC curve is at most 1, high AUC values at any given energy is indicative of better discriminating power. From Fig. 10, it is easy to observe that the single-purpose topologies performed well in the lower energy regimes, with discrimination efficiency tailing off significantly for energies greater than 200 GeV. This increased mis-identification of signals at higher energies is further visualized in Fig. 11, where the false positive rate of identification of background as signals is plotted against energy, at a fixed 90% true positive rate of identification. This helps us narrow down the problem to an increased mis-classification of π^0s at higher energies, in addition to reduced efficiency of identification of γ signals at those energies.

VIII. CONCLUSIONS AND FUTURE WORK

The network topologies and their performances, as presented, were our first attempts at using deep learning on a dataset of particle showers in high-granularity calorimeters. The results of our studies were mixed. All network topologies showed reduced efficiency at solving the regression and classification problems at high energies. In

addition, most topologies used for regression, were also not within a desirable range of accuracy at lower energies. On the other hand, all networks performed similarly well in the middle range of energies (approximately $100 - 400$ GeV). Our attempt at hybrid networks yielded similar or poorer results to single-purpose networks with similar architectures, in both regression and classification. While this is not enough reason to stop pursuing work with hybrid topologies, different optimizations are being attempted to increase their performance. We have recently also generated new samples with more kinds of particles - charged pions and electrons. Working with multiple kinds of background events, might help increase classification efficiency, in addition to making the problems at hand more physically realistic.

In terms of further optimization of our deep learning approaches, fundamentally different network topologies are being considered, in addition to tweaking hyperparameters of the best performing branched, chained convolutional topologies (bcnn4 and bcnn5). One of the new techniques being considered is Upsampling, which is essentially a form of backwards strided convolution. Depending on whether we are working with a 2D or 3D convolutional network, the HCAL data ($5 \times 5 \times 60$) can be upsampled by size=(5,5) such that the first and second dimension of the HCAL data is upscaled by 5 each. If that is then merged with ($25 \times 25 \times 25$) ECAL data, we would then be feeding in ($25 \times 25 \times (60 + 25)$) sized arrays per event to the neural network. If the merged data is fed into a single convolutional input layer, we would bypass the need to feed in data from the ECAL and HCAL to two different input branches. The convolutional layer will "see" one image per event, instead of two. One could, similarly, downsample ECAL data to HCAL dimensions and proceed, but this is potentially with the loss of generality, since there less empty pixels in the ECAL arrays than in the HCAL ones. This procedure is used frequently in image processing applications to process high-resolution pictures that can then be blown up even further without pixelation since the neural nets approximately fill in the gaps where pixelation/smearing would normally occur. This would also potentially be very useful with the new samples - where we look at electrons, and charged pions, and need to have HCAL data to fully describe events. This would also allow us to benchmark the optimizations of our previous approaches against upsampling/downsampling procedures.

In addition to further work on the regression and classification problems, with the view to publish our results, we also plan to publicly release a modified version of the CaloImage Dataset, with new electron and charged pion samples included. This is in collaboration with the CERN OpenData initiative, with the expectation that our dataset would serve as a HEP equivalent to the MNIST handwriting dataset that is widely used as a starting point for those new to deep learning. This would also afford machine learning enthusiasts, and people in the industry, an opportunity to benchmark the performances of advanced deep learning reconstruction techniques with HEP data, without necessarily being a part of the HEP community. Work is also being conducted by other CERN-affiliated researchers to provide publicly available visualizations of our particle shower data.

IX. ACKNOWLEDGMENTS

I would like to thank Prof. Spiropulu for her help and encouragement, and for giving me this opportunity to work with a group of incredible individuals from whom I've learned a lot. I am indebted to Maurizio for guiding our development of this project and generating the LCD datasets that were utilized. I am extremely grateful for Jean-Roch's patience in explaining new Deep Learning methods to me, especially since I was a newcomer to that field at the start of the summer. I would also like to thank both Maurizio and Jean-Roch for the constant deluge of ideas that they had, which not only taught me how to think about our problems in a creative and analytical way, but also guided the development of our work as I tried to incorporate these new thoughts into our methodology. I want to thank Jayesh for his constant help with basic Deep Learning and programming concepts, and our constant exchange of physics and computer science ideas as we tackled this project together.

I am also extremely grateful for Prof. Newman and Josh Bendavid's illuminating questions about our work, which definitely helped streamline our studies. I would also like to thank Dustin, Adi, Federico, Aytaj, Olga and all of the others working with the Caltech CMS group for a wonderful summer experience at CERN.

Last, but definitely not least, I would like to thank Reed College for awarding me the President's Summer Fellowship 2016 grant which enabled this wonderful opportunity.

[1] D. O. Whiteson and N. A. Naumann, Neurocomputing **55**, 251-264 (2003).
[2] V. M. Abazov *et al.*, [D0 Collaboration], Phys. Lett. B **517**, 282 (2001).
[3] T. Likhomanenko, et al., Journal of Phys.: Conf. Series, Vol. 664.
[4] P. Baldi *et al.*, Nat. Commun. **5**, 4308 (2014).
[5] P. Baldi *et al.*, Eur. Phys. J. C **76**, 235 (2016).

APPENDICES

All scripts that were used in our work, and of relevance to those wishing to attempt deep learning approaches to working with our dataset, have been listed in Appendix A. Network topologies used for classification and regression are visualized as flowcharts in Appendices B, C, and D.

Appendix A: Scripts

The GitHub links for scripts required for data reading and preprocessing, in a Python 2.7 environment, are listed below:

1. HDF5 data file reader:
   `https://github.com/kaustuvdatta/CaloImageMacros/blob/master/h5read.py`
2. Barycenter calculation algorithm:
   `https://github.com/kaustuvdatta/CaloImageMacros/blob/master/Centroid_h5.py`
3. File shuffling script, for reduction of biases in data:
   `https://github.com/kaustuvdatta/CaloImageMacros/blob/master/File_Shuffle.py`
4. Data Generator class, with separate generators customized for Convolution 2D, 3D and Dense layer input formats:
   `https://github.com/kaustuvdatta/CaloImageMacros/blob/master/Generators.py`
5. Datasets will eventually be publicly accessible from the CERN OpenData portal:
   `http://opendata.cern.ch/`
   (OpenData access details to be updated when datasets and visualization software goes online).

Appendix B: Regression Topologies

The following topologies were used for logistic regression on energy deposits in calorimeter cells. In the flowcharts below, branched convolutional neural network (bcnn) topologies of type 1, 3, 4, 5 and a single branched dense network are visualized. In bcnn 1 and 3, the branched inputs were processed by the 3D convolutional layers and then merged and fed into fully connected layers, these networks then differed in the number of fully connected layers and number of nodes in dense layers. In bcnn 4 and 5 there were two chained 2D and 3D convolutional and maxpooling units, and the fully connected layers after merging were constituted of the same number of layers and nodes per layer. The dense1 topology consisted of chained dense layers in the input branches and after the inputs were merged, they were fed into another set of dense layers. These architectures were exactly the same for classification and regression, and hybrid networks, each with a single output node to return the classification or regression value, or two output nodes for the hybrid networks.

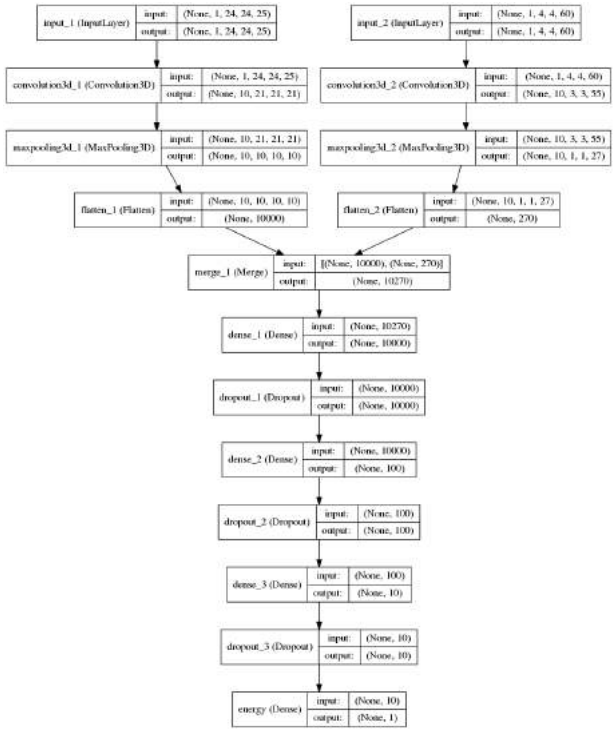

FIG. 12: bcnn1_reg network topology.

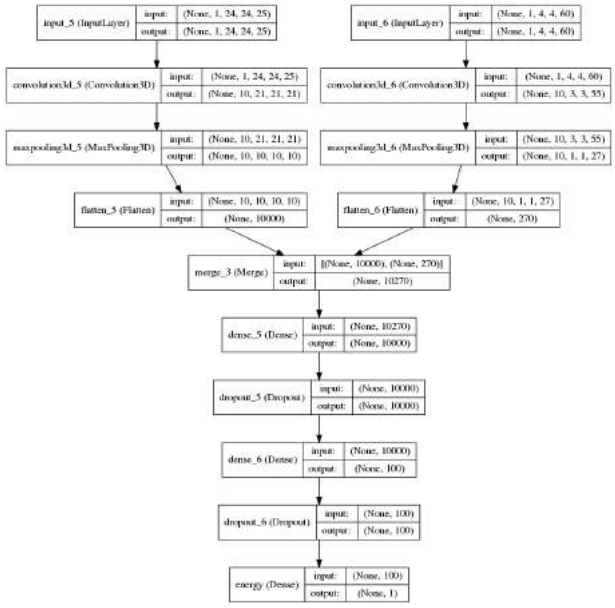

FIG. 13: bcnn3_reg network topology.

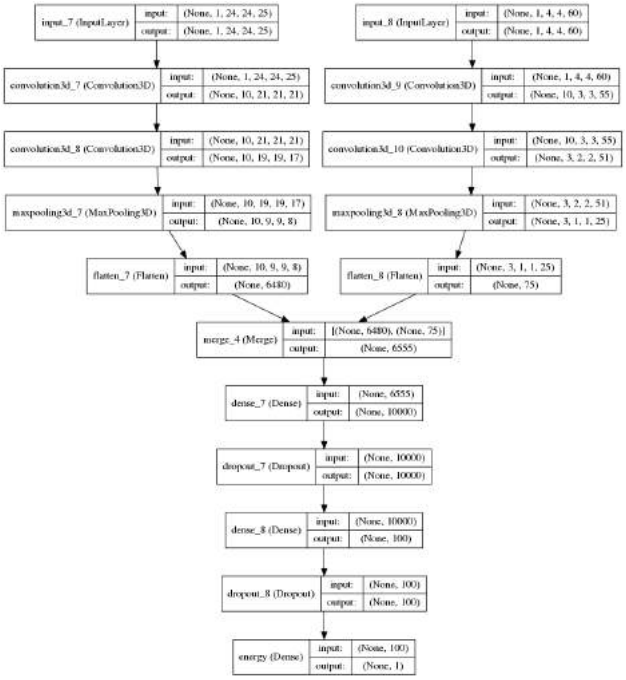

FIG. 14: bcnn4_reg network topology.

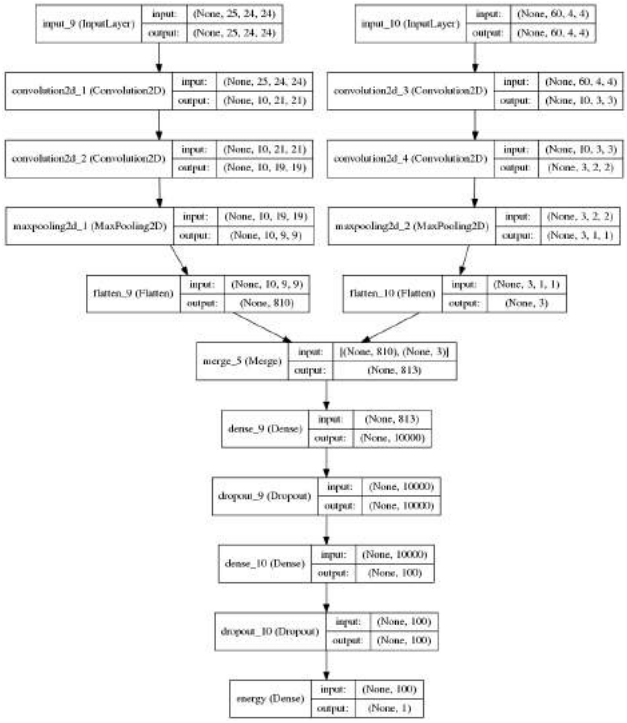

FIG. 15: bcnn5_reg network topology.

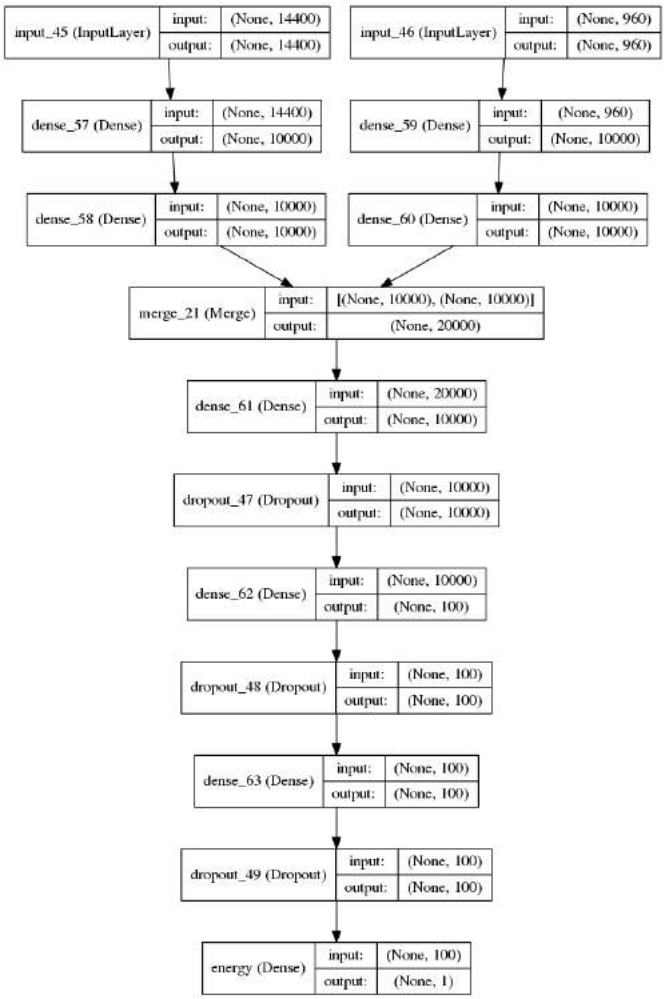

FIG. 16: dense1_reg network topology.

Appendix C: Classification Topologies

The following topologies were used for particle classification using the CaloImage Dataset.

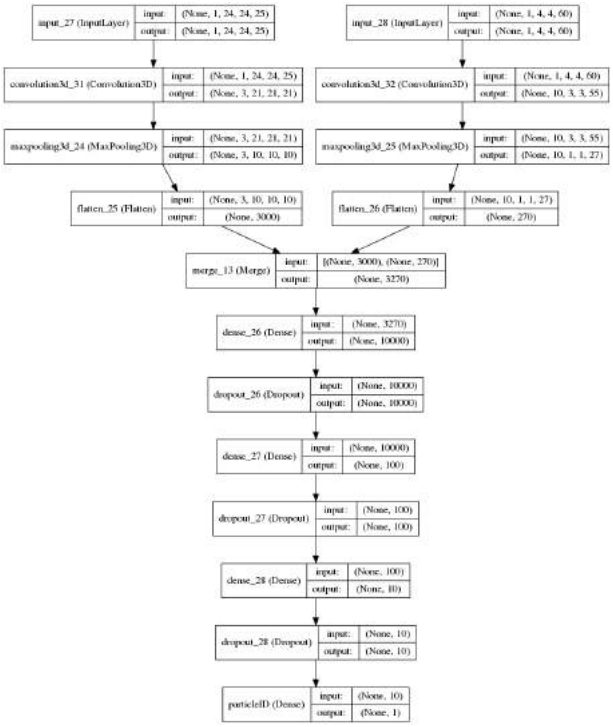

FIG. 17: bcnn1_cl network topology.

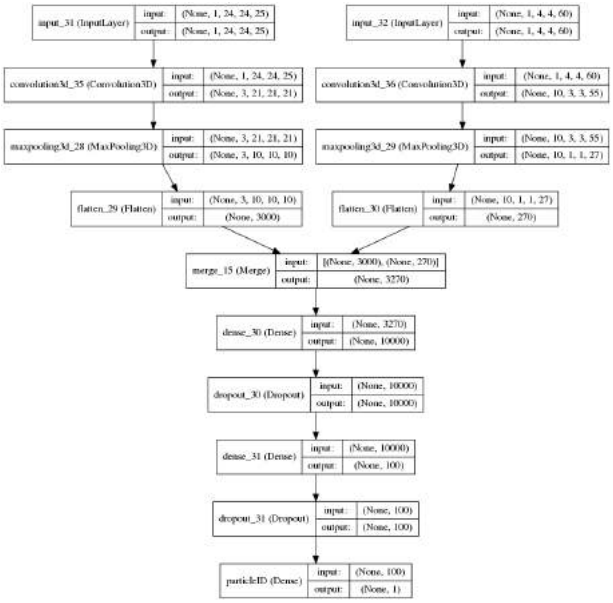

FIG. 18: bcnn3_cl network topology.

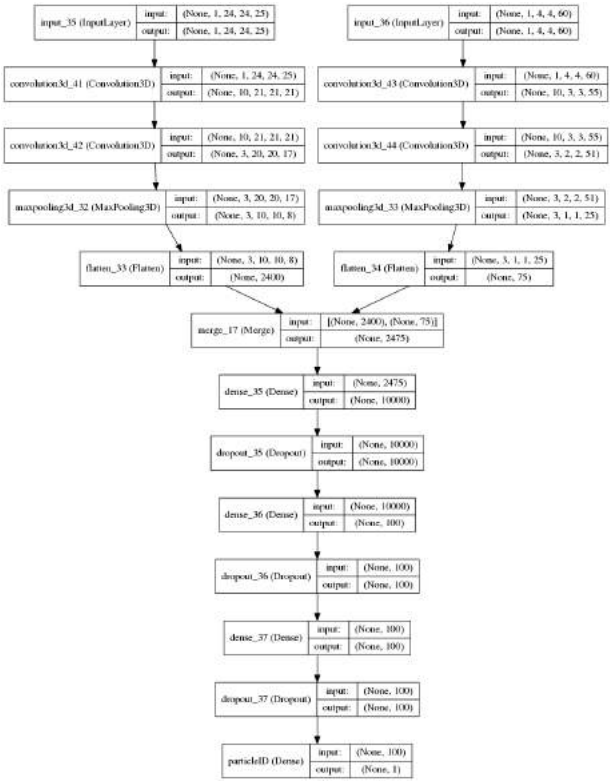

FIG. 19: bcnn4_cl network topology.

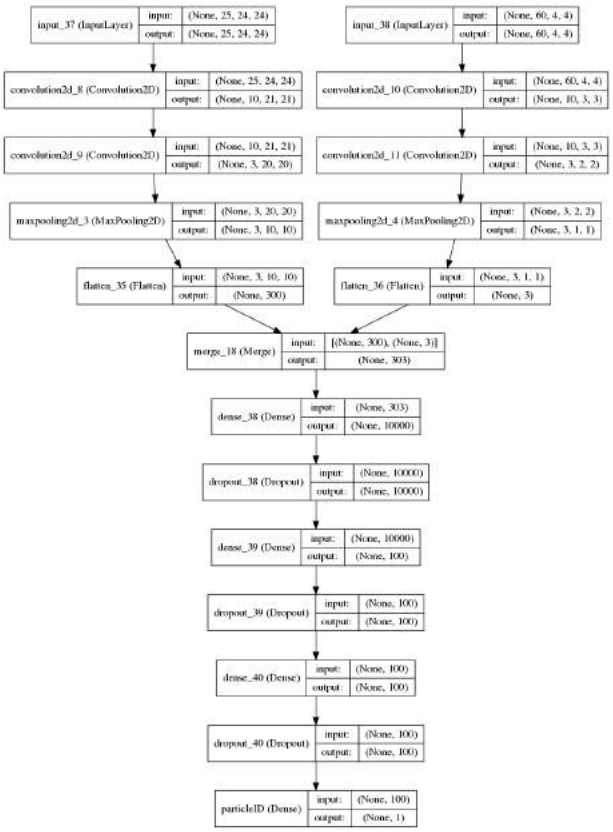

FIG. 20: bcnn5_cl network topology.

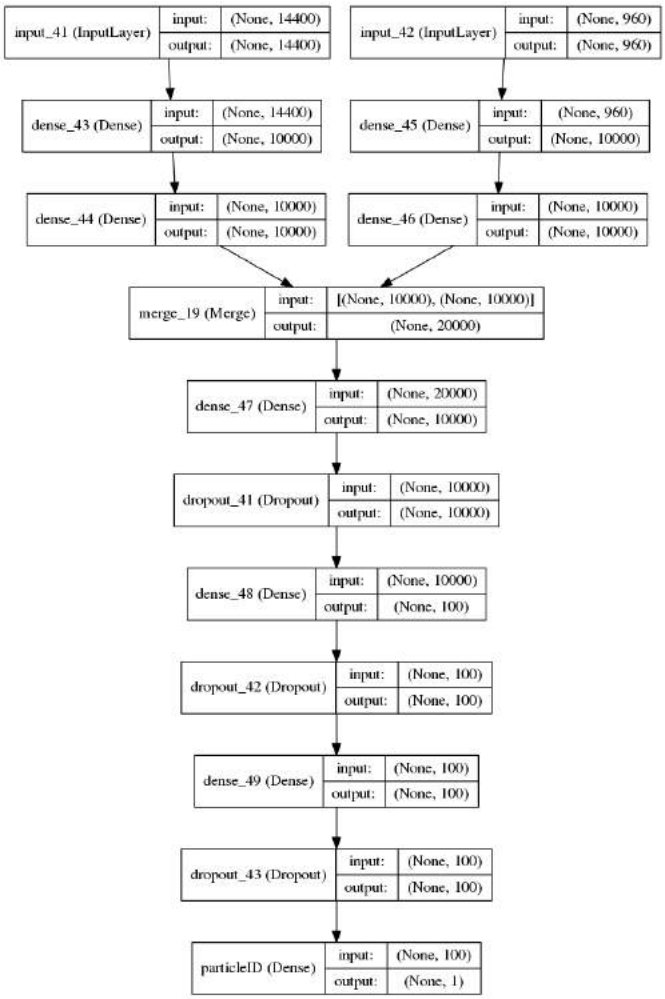

FIG. 21: dense1_cl network topology.

Appendix D: Hybrid Topologies

The following topologies were used for simultaneous energy regression and particle classification using the CaloImage Dataset.

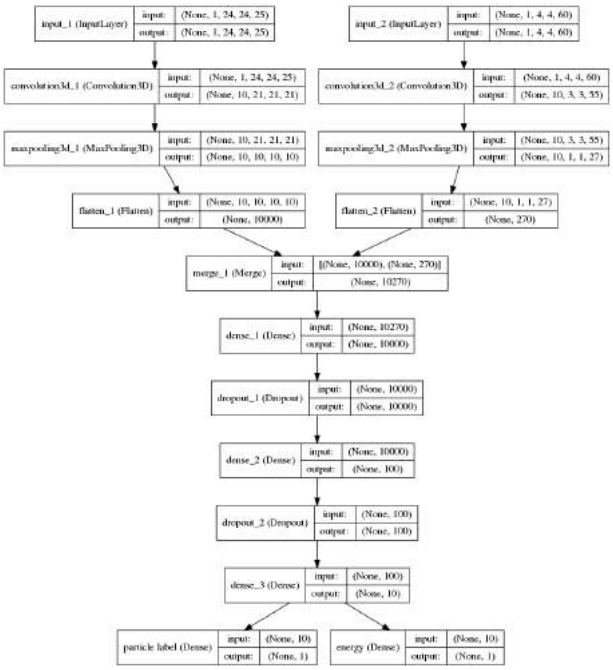

FIG. 22: bcnn1_regcl network topology.

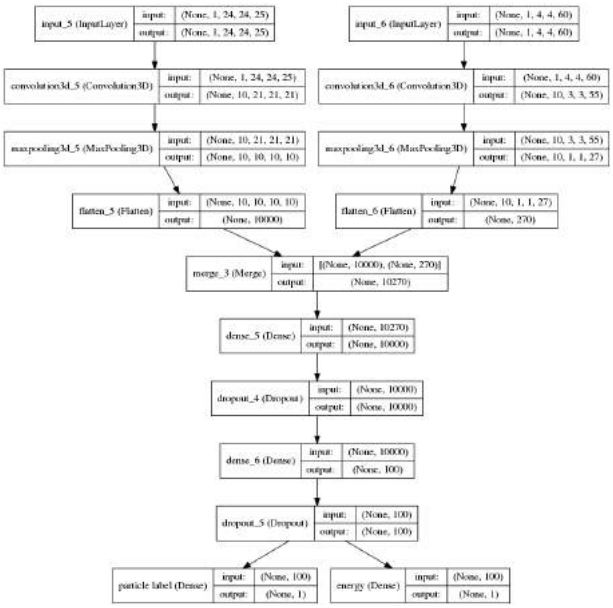

FIG. 23: bcnn3_regcl network topology.

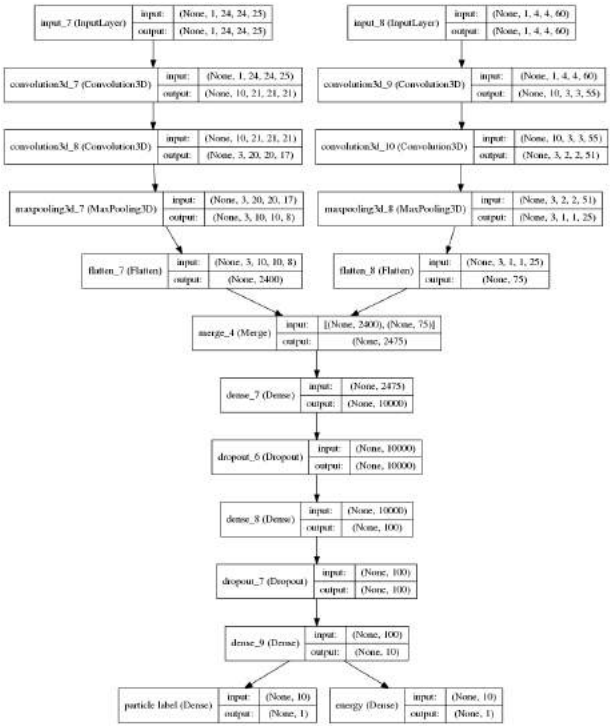

FIG. 24: bcnn4_regcl network topology.

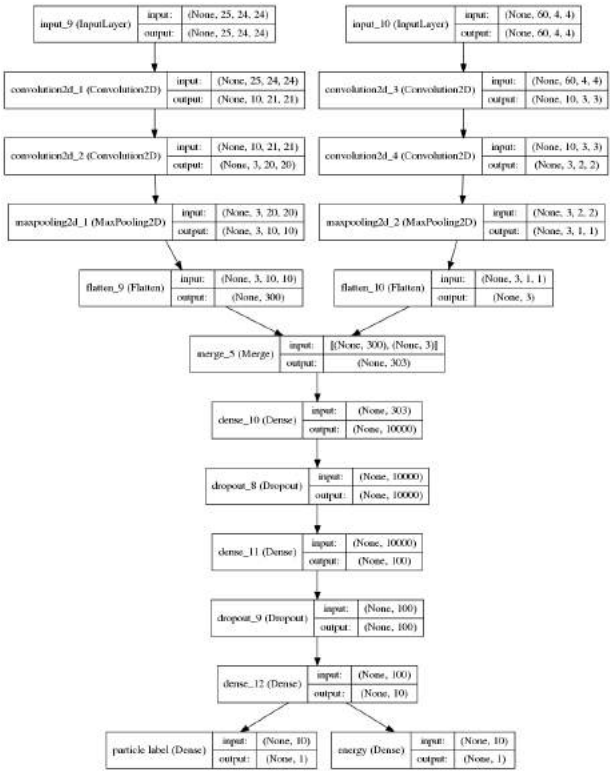

FIG. 25: bcnn5_regcl network topology.

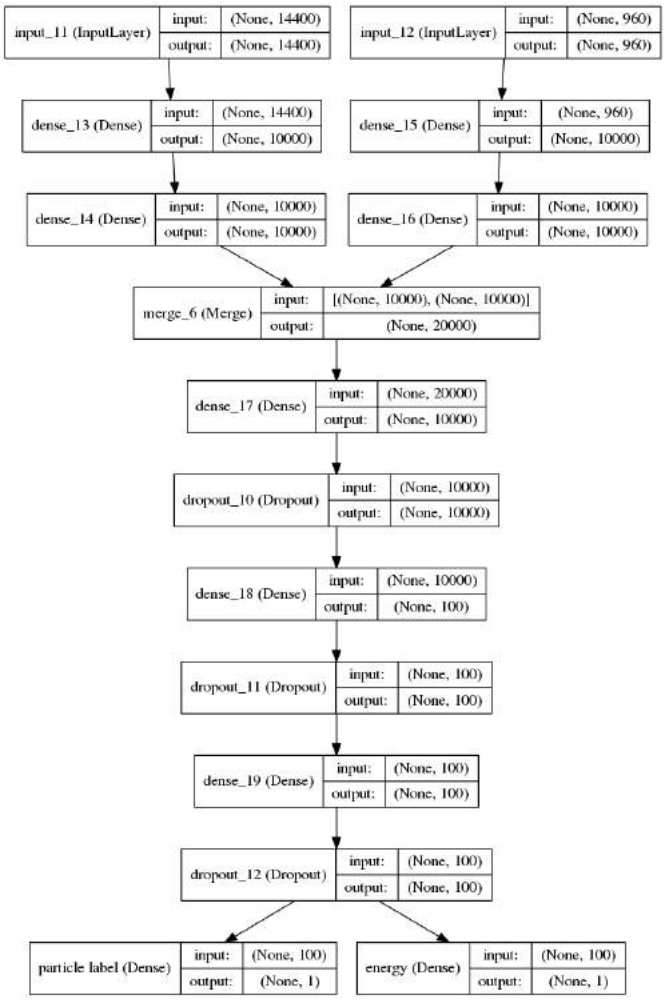

FIG. 26: dense1_regcl network topology.