Real Time Data Ingestion and Anomaly Detection for Particle Physics

Xinyi Gong* , Lanyu Shang*, Zihao Wang*
Center for Data Science, New York University
{xg555, ls3882, zw1074}@nyu.edu

Abstract—This project is looking for a method to compress the high-dimensional data from high energy particle collisions for storage purpose as well as decompress the record whenever necessary. The potential approach for this is the autoencoder, which can support the online training, comparing to PCA models. The autoencoder model with a good prediction after encoding and decoding, is also expected to detect any anomalous events that happen during the particle collision.

Index Terms—Autoencoder, anomaly detection, deep learning, online training

1 Introduction

In physics, high energy particle is an important topic in current research. The data which most physicists conduct research on is mainly from the particle collision which happens every minute in Large Hadron Collider (LHC). However, every collision during high-energy physics experiments can simultaneously create an enormous amount of high-dimensional data. Thus, researchers want to find feasible solutions to store and track these events in a limited storage.

Currently, researchers track and record events by utilizing particular triggers, which could only record limited amount of data as only information that meets requirements of the trigger can be recorded. Besides, triggers are not held in the database in compiled form which results in recompiling each time a trigger is fired. In this project, a general method would be developed to compress and record all data generated from collisions as well as to decompress any records whenever needed. Under such circumstances, autoencoder can be a potential alternative as its prevalence in feature selection. As long as the feature selection and the model can be improved, not only will it save time and storage memory, but also increase the efficiency of research activities. In addition, PCA is also good at reducing dimensions and selecting features. The comparison between the performance of PCA and autoencoder will also be evaluated.

2 Data and Prepocessing

The data in this project is collected by [1], and it is the record of events that happened at CERN[1]. It runs the world's largest particle physics lab. In a typical experiment, one high-energy particle shower shoots through a pipeline against and perpendicular to the surface of high-granular linear collider detector. The energy then spreads out due to the collision and is detected by the calorimeters with respect to the location. The pattern of energy will be recorded as a high-dimensional vector.

The single-particle energy showers are produced by either photons or neutral pions. There are two calorimeters, which are electromagnetic calorimeter (ECAL) and hadron calorimeter (HCAL), inside the high-granular linear collider detector. The ECAL has 25 layers and 24 x 24 absorbers in each layer. The HCAL locates behind the ECAL, and it is for the detection of the most energetic hadrons. It has 60 layers, and the size of sensor cells is six times larger than that of ECAL. Thus, there are 4 x 4 absorbers in each layer of HCAL. Here is the table of description of every event.

TABLE 1: Description of Data

Keys	Shape	Description
ECAL	24×24×25	Energy detected by ECAL
HCAL	4×4×6	Energy detected by HCAL
target	1×5	Momentum, energy and nature

The events described before are recorded in a bunch of HDF5 files. 10,000 events are packed in a file, and there are 50 files, i.e., there are 500,000 events in total. Each file has 3 keys with 10,000 entries(events). For each event, just as shown in TABLE 1, the keys are labeled as "ECAL", "HCAL" and "target". Each "ECAL" contains a 24x24x25 numpy array standing for the energy detected by ECAL. Each "HCAL" records the energy deposited in HCAL by a 4x4x60 numpy array. The "target" provides the additional information about the momentum of the incoming particle (stored as the p_x, p_y, p_z in the global coordinate system), the energy of the incoming particle and the nature of the incoming particle (1 for photons and 0 for neutral pions). It makes a 1x5 numpy array for each "target".

As the data has the spatial correlation in a 3-dimensional space, which is similar to the image data, the suggested preprocessing of image data are rescaling data to [0,1] and normalization. For preprocessing the data in this project,

* *Team members contributed equally to this project.*

1. CERN is the abbreviation in French of the European Organization for Nuclear Research

events are rescaled to the [0,1] range for easier computation. However, no normalization was applied, because all energy values are in the same unit; otherwise, the sparsity could be lost.

3 APPOARCHES

This project is going to deal with data generated by a simulation. For compression, the main idea is to construct a encoder called $E(x) : \mathrm{R}^n \rightarrow \mathrm{R}^m$ and a decoder called $D(x) : \mathrm{R}^n \rightarrow \mathrm{R}^m$. Here, one way to measure the effect of compression is the compression ratio: $r = m/n$. Because we often require $m << n$. Therefore, after encoding, some information will be lost. To minimize the loss between the original vector and reconstructed vector, a good compressor will try to capture some hidden patterns so that it can describe the original data by using these hidden patterns. Moreover, the main method we use to approach the best encoder is MLP autoencoder because MLP is a universal approximator [4]. Autoencoder is a deep learning model with artificial neural networks that studying unsupervised learning for a generative model. It is a prevalent method for feature selection. Here is the scratch of the autoencoder algorithm: Y is the input and encoded to a low-dimensional representative z. Z can be decoded to the predicted Y later. More hidden layers can be added during both encoding and decoding process in the artificial neural network.

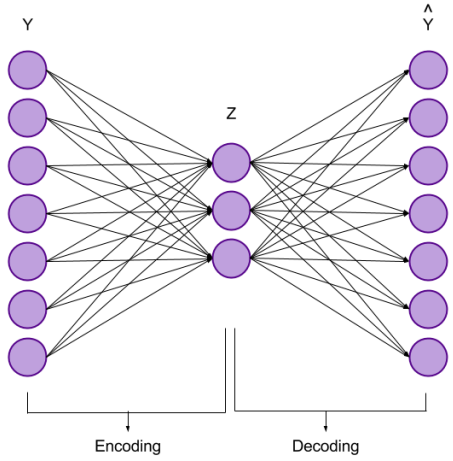

Fig. 1: Autoencoder Algorithm

The input is also the target variable here. It consists of three numpy arrays and is going to be encoded to a low-dimensional representation. Then the representation will be decoded to the predicted target variable later. According to the dataset, many values are small, so some thresholds are set for better prediction during decoding. As the ECAL record has the highest number of dimensions, ECAL record will be mainly focused on being compressing during training. As the data is rescaled into range [0,1] (like a probability), cross entropy loss would be efficient for training this type of data. Therefore, cross entropy loss will be applied to evaluate the performance of the model. L1 regularization is often used to control the sparsity of prediction. As the dataset is sparse, in order to preserve the sparsity during the training process, the L1 regularization is applied for loss penalty. In short, the formula of loss function used in this project is

$$L(X, \hat{X}) = -\Sigma_i X_i \log \hat{X}_i + \lambda \| f_1(W_e X + b_e) \|_1.$$

3.1 One-layer Autoencoder

The idea of one-layer autoencoder is that the input function is first mapped into a hidden variable via a hidden layer with nonlinear activation function then train another hidden layer to remap the hidden variable back to the input space. The formula of one-layer autoencoder is

$$\hat{X} = f_2(W_d f_1(W_e X + b_e) + b_d),$$

where f_1, f_2 are activation function and W_d, W_e are the decode matrix and encode matrix.

3.2 Deep-layer Autoencoder

Comparing to the previous one-layer autoencoder, the deep-layer autoencoder uses more hidden layers during both encoding and decoding process. The MLP (Multilayer Perceptron) is applied to extract deep features. Let

$$f_k(X) = \text{relu}(W_k X + b_k),$$

then the formula of deep autoencoder is

$$\hat{X} = f_N \circ f_{N-1} \circ \ldots \circ f_1(X).$$

Here $f_1 \circ f_2(X) = f_1(f_2(X))$.

3.3 Convolutional Autoencoder

Convolutional autoencoder uses convolution layer to extract features. Since our data is spatial correlated, we can use convolution layer and max-pooling to capture the local connections.

4 RESULTS

The compression ratio of each autoencoder discussed above is 32:14400, that is, every $24 \times 24 \times 25$ vector is encoded into a vector of length 32. Fig.2 contains the visualization of original data and the reconstructed data after decoding, from which, the reconstructed data of one-layer autoencoder and deep-layer kept the pattern of the original data, while that of convolutional autoencoder seemed blur on its pattern.

TABLE 2: R^2 Score of Models

Autoencoder	R^2 score
One-layer	0.9497
Deep-layer	0.9166
Convolutional	0.0967

Along with the visualization of reconstructed data, based on the R^2 scores shown in TABLE 2 above, one-layer autoencoder had the best performance on the dataset. However, the performance on anomaly detection can also be an evaluation scale.

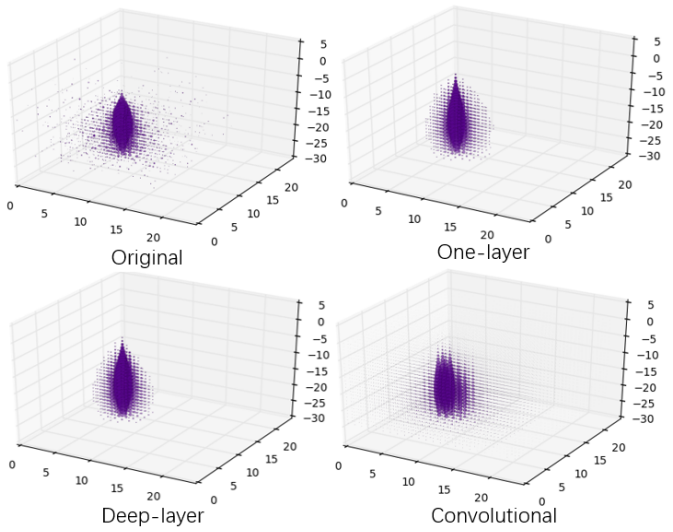

Fig. 2: Original and Reconstructed Data

5 Anomaly Preparation

As one of the goals of this project is to detect the anomaly, however, the chances that anomalous particles occur are really low. Artificial anomalous events are created and added to the test dataset, particularly, to evaluate the ability of models in detecting anomalies as well as analyzing errors. Fig.3 is the visualization of these anomalies.

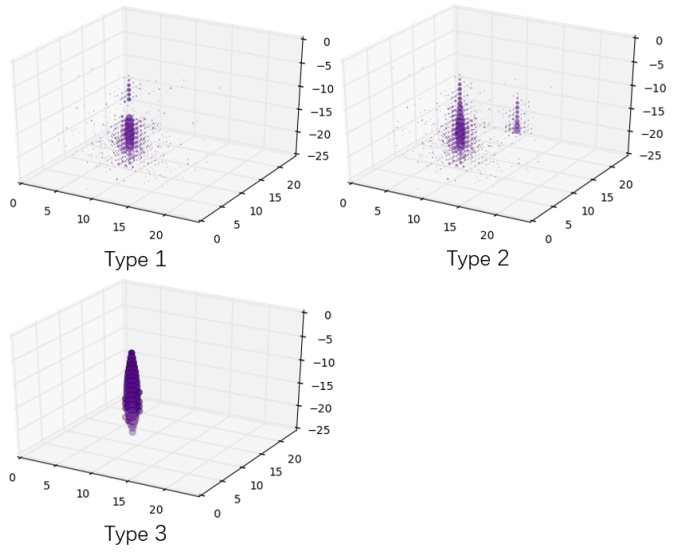

Fig. 3: Anomaly Preparations

Here are three types of anomalies:

- **Type 1: Missing energy records**
  There are some parts of the records being deleted and set to zero. It can be caused by some different particles that can block the magnetic wave.
- **Type 2: Energy records at abnormal position**
  There are some abnormal energy records at a position that do not match the distribution pattern as that of other events.
- **Type 3: Ignoring small energy values**
  All the points with energy values are set to zero if they are less than the threshold. The threshold will be discussed in the later section.

6 Anomaly Detection

For anomaly detection, $\|X - X_{decode}\|_2$ (i.e. the L2 distance between X and X_{decode}) is used to examine the whether the event contains any anomalies. If the event does not contain any anomalies, the distance is expected to be low. Fig.4 is the comparison of the anomaly detection of each autoencoder. Thus, if an autoencoder is sensitive to these anomalies, distinct differences are expected for the L2 distance of each event.

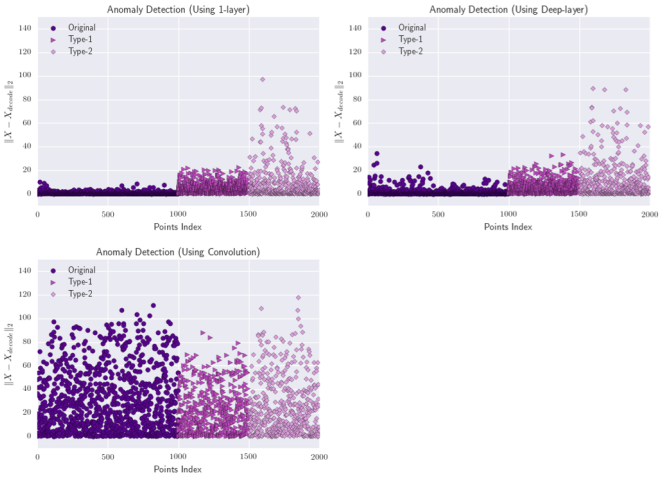

Fig. 4: Anomaly Detection on Type 1 and 2

According to Fig.4, one-layer and deep-layer autoencoders performed equally well on the detection of type 1 and type 2 anomalies.

7 Error Analysis

During the training of autoencoders, there once existed a problem, and it was after decoding, the sparsity did not maintain. Some small-value energy points were predicted to larger values. These points can be considered as noise. Hence, a threshold on ReLU activation on the first layer of autoencoder was added to eliminate these noise, so that the sparsity of data can keep.

From the visualizations of reconstruction data (Fig. 2), especially that of the convolutional autoencoder, it seems the autoencoder is trying to capture the large energy points to ensure the R^2 score, then, the artificial type 3 anomaly is used for error analysis. The threshold here is the 32nd largest energy value. As the dimension of encoder vector is set to be 32, in order to test whether the autoencoder just remembers the large numbers, selecting top 32 large values has become a good choice. It is used to test if the autoencoder just records the larger values. Similar to the criteria of the anomaly detection on type 1 and 2, if the

pattern of distance between X and X_{decode} of type 3 is different from that of original data, the autoencoder performs well on feature extractions on all energy points. Fig. 5 is the comparison of the anomaly detection on type 3 which also known as the error analysis for autoencoder.

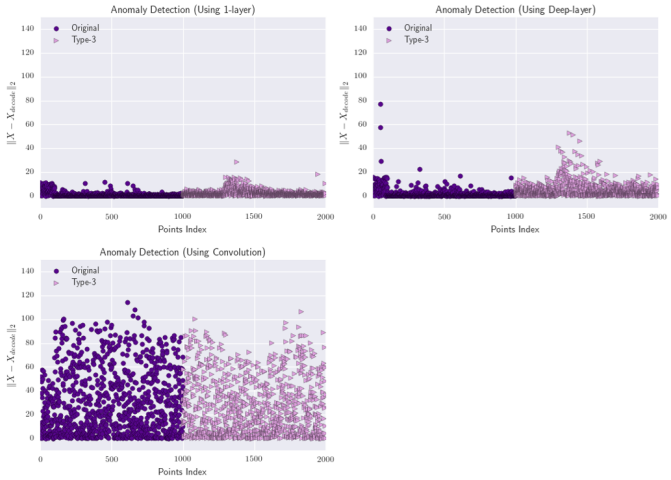

Fig. 5: Error Analysis on Type 3

The one-layer autoencoder and convolutional autoencoder do not illustrate much difference on the original data and type-3 data. Therefore, if these models can preserve the larger values of energy, the loss of prediction would not be high. These two models are not sensitive for detecting this type of anomaly.

Fortunately, for the model with the best performance (deep-layer autoencoder), it not only captures the obvious features (e.g. large values) but also preserves some hidden features. Thus, deep-layer autoencoder is more robust in anomaly detection.

8 Conclusion

The implement of the autoencoder for data of high-energy particle collision events brings mainly two benefits. The first one is the storage can be saved as the autoencoder aims to find a representation for the original data, by reducing the dimension of data (length of 32 after encoding to length of 14400 before). As the algorithm of autoencoder is similar to the algorithm of PCA, the final representative of events will result in a low-dimensional with some most essential features as the result of encoding. Then, the representative vector can be decoded to the predicted vector, which is also known as the reconstructed vector. Another benefit of a well-trained autoencoder is finding anomalies. The basic idea is that if the predicted (reconstructed) vector is far away from the real input vector, it is more likely to explore a new pattern or event in high-energy physics.

The PCA model can perform really well on the sparse data (with R^2 score 0.9905 for the data in this project). However, it is not recommended to be used in practice. The first reason is that PCA cannot be trained online, which means that every time a new training data is added to the existing training data, the whole model needs to be retrained. While autoencoder is an online training method, stochastic gradient descent (SGD) is applied to train the parameters. Another disadvantage of PCA is it is time-consuming. So far, the deep-layer autoencoder is preferred for anomaly detection, and one-layer autoencoder is expected to store the real-time data for its high R^2 score.

Appendix A
Layers of Autoencoders

A.1 One-layer Autoencoder

Layer Type	Parameters
ThresholdedReLU	$\theta = 0.001$
Sigmoid	-
Fully connected	#neurons: 14400
ReLU	-
Fully connected	#neurons: 32
Input	14400 Dimension vector

A.2 Deep-layer Autoencoder

Layer Type	Parameters
ThresholdedReLU	$\theta = 0.001$
Sigmoid	-
Fully connected	#neurons: 14400
ReLU	-
Fully connected	#neurons: 2400
ReLU	-
Fully connected	#neurons: 1200
ReLU	-
Fully connected	#neurons: 600
ReLU	-
Fully connected	#neurons: 128
ReLU	-
Fully connected	#neurons: 32
ReLU	-
Fully connected	#neurons: 128
ReLU	-
Fully connected	#neurons: 600
ReLU	-
Fully connected	#neurons: 1200
ReLU	-
Fully connected	#neurons: 2400
Input	14400 Dimension vector

A.3 Convolutional Autoencoder

Layer Type	Parameters
ThresholdedReLU	$\theta = 0.001$
Sigmoid	-
Convolution3D	#filters: 1, k:$4 \times 1 \times 1$, s:1
UpSampling	k:$2 \times 2 \times 2$
ReLU	-
Convolution3D	#filters: 16, k:$3 \times 3 \times 3$, s:1, same border
UpSampling	k:$2 \times 2 \times 2$
ReLU	-
Convolution3D	#filters: 8, k:$3 \times 3 \times 3$, s:1, same border
Reshape	-
ReLU	-
Fully connected	#neurons: 2016
ReLU	-
Fully connected	#neurons: 32
MaxPooling3D	k:$2 \times 2 \times 2$, s:1, same border
ReLU	-
Convolution3D	#filters: 8, k:$3 \times 3 \times 3$, s:1, same border
MaxPooling3D	k:$2 \times 2 \times 2$, s:2, same border
ReLU	-
Convolution3D	#filters: 16, k:$3 \times 3 \times 3$, s:1, same border
Input	$25 \times 24 \times 24$ HCAL vector

ACKNOWLEDGMENTS

We would like to thank Professor Kyle Cranmer for supervision and guidance and Professor Cladio Silva for instruction and efforts on the Capstone course. We also would like to thank Maurizio Pierini and Jean-Roch Vilmant for providing the data source.

REFERENCES

[1] J. Bendavid, K Dattaa, A. Farbin, N. Howe, J. Mahapatra, M. Pierini, M. Spiropulu, J.R. Vlimant *Imaging calorimeter data for machine learning applications in HEP*, unpublished.

[2] C. Doersch. *Tutorial on variational autoencoders*,2016. arXiv preprint arXiv:1606.05908

[3] Y. LeCun. *Unsupervised learning*, 2016. [Online]. Available: http://cilvr.nyu.edu/lib/exe/fetch.php?media=deeplearning:2016:lecun-20160308-unssupervised-learning-nyu.pdf

[4] K. Hornik. *Multilayer Feedforward Networks are Universal Approximators*, 1989. Available: http://deeplearning.cs.cmu.edu/pdfs/Kornic_et_al.pdf

[5] M. Kuusela, T. Vatanen, E. Malmi, T. Raiko, T. Aaltonen and Y. Nagai. *Semi-supervised anomaly detection – towards model-independent searches of new physics*, 2012. arXiv preprint arXiv:1112.3329

[6] K. Muandet and B. Schölkopf. *One-class support measure machines for group anomaly detection*, 2013. arXiv preprint arXiv:1303.0309

[7] X. Wei. *Must know tips/tricks in deep neural networks*, 2015. [Online]. Available: http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html. Accessed: Dec. 17, 2016.