Data Compression using Auto-Encoders for CLAS12 Online

Gagik Gavalian[a]

[a]*Jefferson Lab, Newport News, VA, USA*

Abstract

In this article, we present the results of using Auto-Encoders for compressing the pulse data from CLAS12 detector systems. The raw pulse form from Analog to Digital Convertors (ADC) is compressed to decrease raw data size. The results show that the pulses can be compressed by a factor of 4, without significant loss in the pulse integral (about 1.4%).

1. Introduction

The technological advancements in recent decades lead to more intensive experiments in Nuclear Physics. The experiments are conducted with higher luminosities (higher frequency of beam to target interactions), and more complex detector systems with many more channels. This leads to increased data rates originating from each individual detector component. As a result, the data volume has significantly increased compared to the past experiments. The increased data volume presents challenges in data transfer and persistence, requiring more storage facilities. There are known methods for compressing the experimental raw data using lossless compression algorithms that can reduce the stored data size. However, the lossless compression methods are usually slow and are not suitable for the online data acquisition stage, since they do not provide sufficient throughput. New advances in neural networks have had a significant impact on the field of nuclear physics. Neural networks can be used to solve a variety of problems in nuclear physics, including simulation of nuclear reactions, data analysis, reconstruction of events from detector data, and many more. The use of neural networks in nuclear physics is still in its early stages, but it has the potential to revolutionize the field. As neural networks become more powerful and sophisticated, they will be able to solve even more problems in nuclear physics.

In this paper, we investigate Machine Learning (ML) methods for lossy raw data compression and their applications in online data acquisition.

2. CLAS12 Detector

The CLAS12 detector, located in experimental Hall-B at Jefferson lab, is designed to study electro-induced nuclear and hadronic reactions by providing efficient detection of charged and neutral particles over a large fraction of the full solid angle. The detector is based on a combination of a six-coil torus magnet and a high-field solenoid magnet. The combined magnetic field provides a large coverage in both azimuthal and polar angles; a schematic view is presented in Figure 1.

Some of CLAS12 detector components (such as Time-Of-Flight counters and Electromagnetic Calorimeters) have an FADC readout which is measuring the charge in 4 ns intervals.

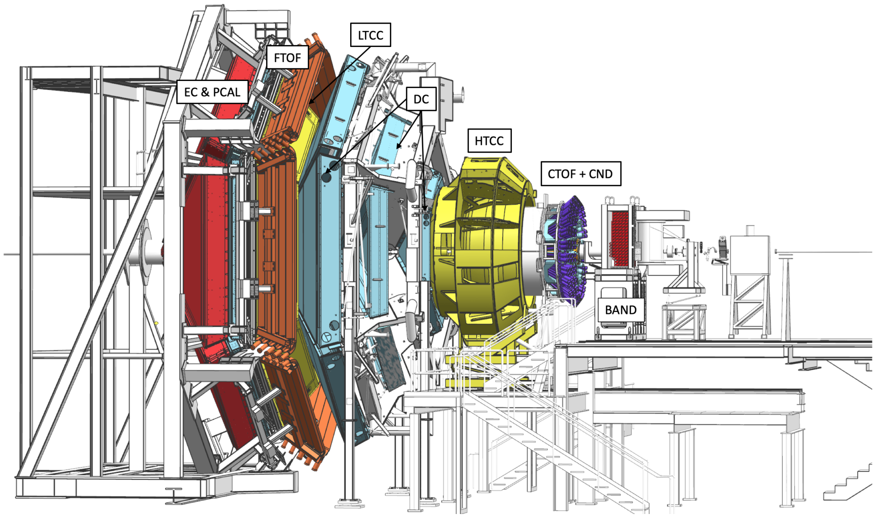

Figure 1: The CLAS12 detector in the Hall B beamline [1]. The electron beam enters from the right and impinges on the production target located in the center of the solenoid magnet shown at the right (upstream) end of CLAS12, where other detector components are also visible. Scattered electrons and forward-going particles are detected in the Forward Detector (FD), consisting of the High Threshold Cherenkov Counter (HTCC) (yellow), followed by the torus magnet (gray), the drift chamber tracking system (light blue), and time-of-flight scintillation counters (brown), and electromagnetic calorimeters (red).

The measured pulses are used in reconstruction to calculate the pulse integral charge and time. The charge integral is used to determine the deposited energy in the detector component such as individual calorimeter scintillating paddles. Having an entire pulse spectrum provides more accuracy in deposited energy measurement since the pedestal subtracted pulse can be obtained from the pulse shape and fitted. The pulse shape also provides timing information which helps in combining hits into clusters. Example pulses from the electormagnetic calorimeter can be seen in Figure 2. However, the accuracy comes at a cost of data size,

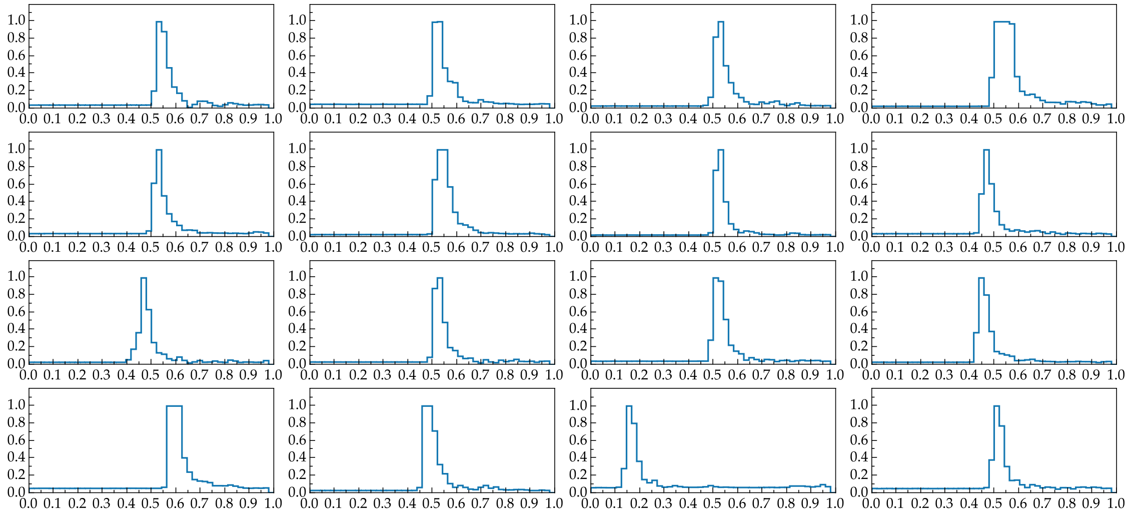

Figure 2: Example pulses from Electromagnetic Calorimeter (EC) of CLAS12 detector. (The pulses are normalized to the peak height)

because instead of keeping one value of integrated charge (one floating number of 4 bytes) the pulse profile is stored in the output data stream consisting of 48 short numbers (96 bytes).

3. Motivation

With upcoming high-luminosity running at CLAS12, the reduction of data can be beneficial for storage and online data transport. In this work, we investigate Machine Learning (ML) approach to reduce the data footprint by compressing the pulse data using auto-encoder type neural networks.

3.1. Auto Encoders

Autoencoders are a type of artificial neural network that is used to learn efficient codings of unlabeled data. They do this by learning two functions: an encoding function that transforms the input data, and a decoding function that recreates the input data from the encoded representation. The autoencoder learns an efficient representation (encoding) for a set of data, typically for dimensionality reduction. Variants exist, aiming to force the learned representations to assume useful properties. Autoencoders are used for a variety of tasks, including:

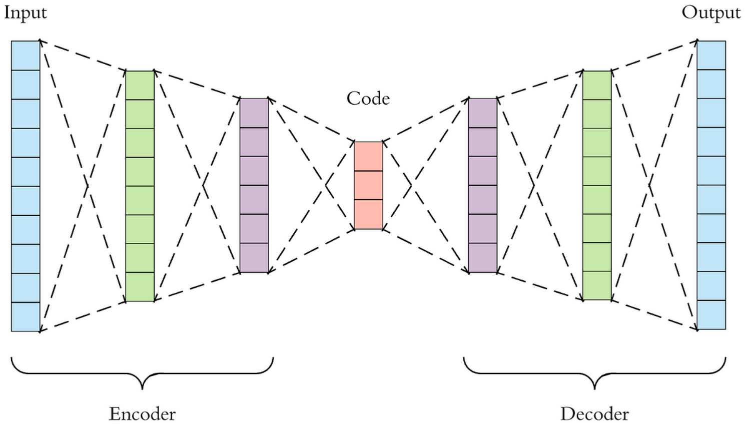

Figure 3: An autoencoder network architecture, consisting of encode part that is used to encode a more compact representation of the input into a latent space (Code) and decoder part that reconstructs the latent space into a different representation of the input.

- Image denoising: Autoencoders can be used to remove noise from images. This is done by first encoding the image into a lower-dimensional representation, and then decoding the representation back to an image. The noise is removed in the encoding process, so the decoded image is cleaner than the original image.

- Image compression: Autoencoders can be used to compress images. This is done by encoding the image into a lower-dimensional representation, which can then be stored more efficiently. When the encoded representation is decoded, the original image can be reconstructed.

- Feature detection: Autoencoders can be used to detect features in data. This is done by learning an encoding representation that captures the most important features of the data. The encoded representation can then be used to identify similar data points.
- Anomaly detection: Autoencoders can be used to detect anomalies in data. This is done by training the autoencoder on normal data. When new data is presented to the autoencoder, if the data is anomalous, the autoencoder will not be able to reconstruct the data as well as it can reconstruct normal data.

Autoencoders are a powerful tool for learning efficient representations of data. They are used for a variety of tasks, and their applications are still being explored. A typical architecture of an autoencoder is show in Figure 3.

3.2. Method

In this paper, we use an autoencoder to compress the ADC pulse shape from the detectors. The input and output to the network are the pulse shape (represented by a vector of length 48), and the middle layer of the autoencoder will represent the compact representation of a pulse. The network is trained to reproduce the pulse in the output with good accuracy given the input pulse. In the compressions stage, this network will be used to run the network on raw data pulses and save the layer (called Code, see Figure 3), which is smaller in size (in our studies factor of 4 smaller). In the decompression stage, the saved vector will be read, and the second half of the network (called decoder, see Figure 3) will run on the vector to reconstruct the full shape of the pulse.

4. Studies with simulated data

4.1. Single Pulse Simulations

The studies with the network were first conducted on simulated data to fine-tune the network before studying the impact on real data. Pulses were generated using a Landau function on top of a flat background.

$$L(x) = e^{-\frac{1}{2}(\frac{x-x_0}{\sigma}+e^{\frac{x-x_0}{\sigma}})} \tag{1}$$

For each sample the position of the peak (x_0) was generated using a pseudo-random generator and the sigma value of 0.015 was used (which is similar to the pulses we see from the detectors). A large training sample was used to train the autoencoder and a validation sample (also randomly generated) was used to compare the output from the network to the original pulse shape. For our initial studies, the network architecture [48,24,12,24,48] was used with **ReLU** activation functions for hidden layers and **Linear** activation function for the output layer. The results are shown in Figure 4.

It can be seen from the figure that the decoder reconstructs the pulses from encoded data with great accuracy, and the stored data is 4 times smaller (vector of the length 12 for the middle layer). In Figure 5 (left) the distribution of the normalized difference of the pulse integral is plotted for the validation sample. As can be seen from the figure the reconstructed from the encoding pulse integral is on average in agreement with the input

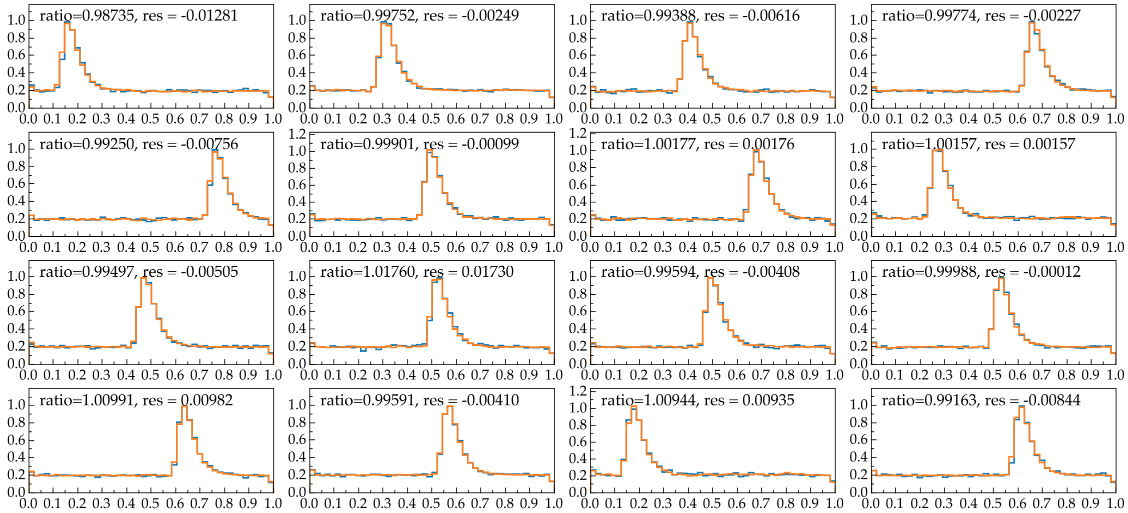

Figure 4: Original pulses plotted with reconstructed pulses overlayed. The data is produced by simulating a single pulse in the 48-bin region.

pulse with a resolution of about 0.7%. In Figure 5 (right) the time difference is calculated for the original pulse and the decompressed pulse. The time is calculated assuming that the 48 bins in the pulse represent a timing window of 48 ns, the resulting difference is multiplied by 1000 and the difference is given in pico-seconds.

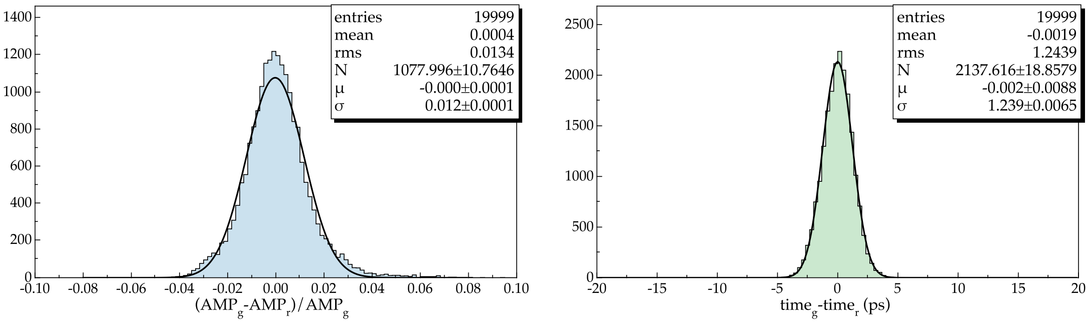

Figure 5: The resolution of the pulse integrals. The integral of the pulse is compared to the integral calculated from decompressed pulses (on the right), and time calculated from the original pulse is compared to the time calculated from the decompressed pulse.

In real experimental conditions, it is not uncommon to have two pulses in a given time window when the electronics are being read. Our next study is to use the same network architecture for training a network for compressing two pulse inputs.

4.2. Double Pulse Simulations

To study the compression of double pulse data we generated a new sample consisting of two pulses with random mean positions and random height ratios, the width of the pulse was kept the same (with $\sigma = 0.015$). The results of compression and decompression for two pulse data are shown in Figure 6. It can be seen from the figure that the network is able to accurately reconstruct the peak positions but in some cases, the heights and the widths of the peaks do not match the input peak, which leads to degraded accuracy for peak integral calculations.

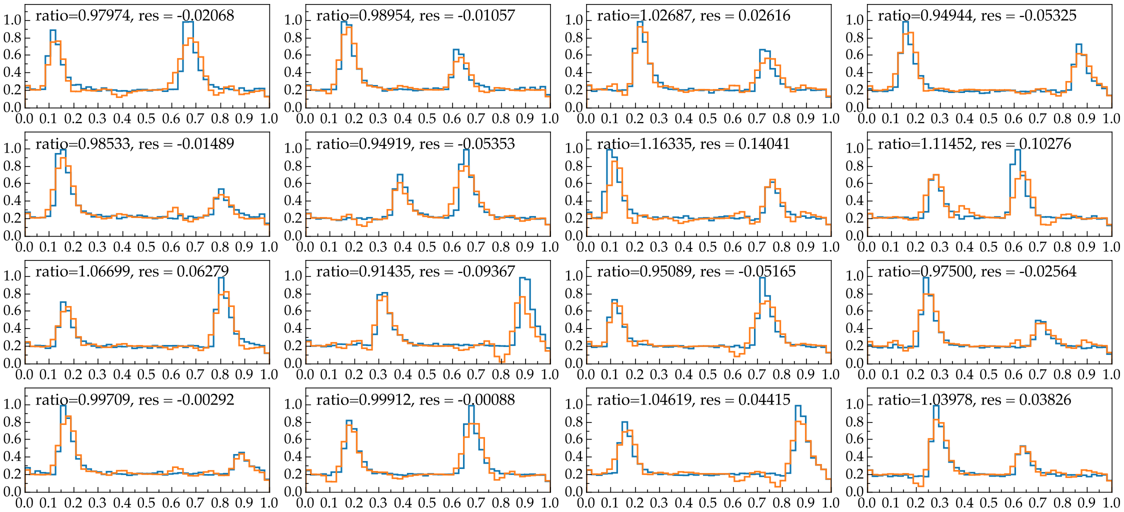

Figure 6: Generated double pulses plotted with the decompressed pulses overlayed. The compression is done with the original auto-encoder [48,24,24,12,24,24,48] architecture. The data is produced by simulating a single pulse in the 48-bin region.

The average resolution for the peak integral reconstruction is shown in Figure 7, and is about 4%. It's worth mentioning that the heights of the peaks being reconstructed with degraded accuracy can lead to inaccurate pulse timing calculations.

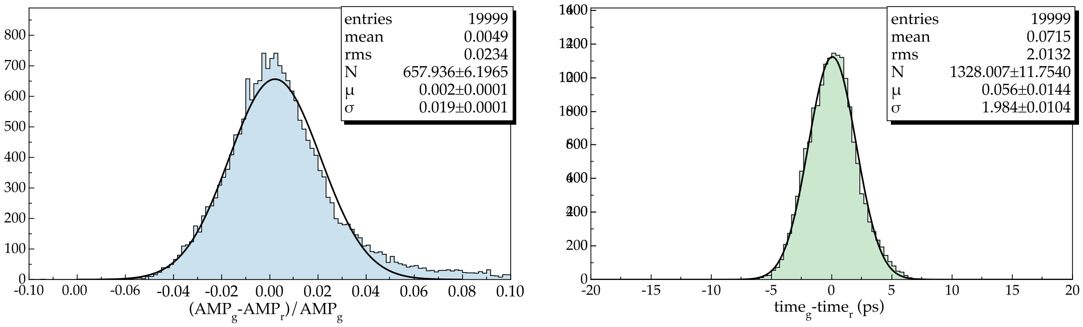

Figure 7: The resolution of the pulse integrals (double pulses). The integral of the pulse is compared to the integral calculated from decompressed pulses (on the right), and the time calculated from the original pulse is compared to the time calculated from the decompressed pulse.

To improve the two-pulse data compression a different network architecture was considered. Traditionally the autoencoders are constructed in a way that each subsequent layer starting from the input layer is smaller in size. This can potentially cause some issues when the input vector is not large enough, where the subsequent layer already has to find a more compact representation of the input. Following the logic from convolutional networks, one would expect that the layer following the input layer should be larger to be able to encode some different features from the input vector into much larger space, before trying to compress it. Inspired by this logic the network architecture was changed to have a larger hidden layer to follow the input layer in the network and changed the architecture to [**48,96,48,24,12,24,48,96,48**] , while keeping it symmetric for both the encoder and decoder sides. The new network was trained on the same training sample with double pulses from our previous test, and the results can be seen in Figure **??**.

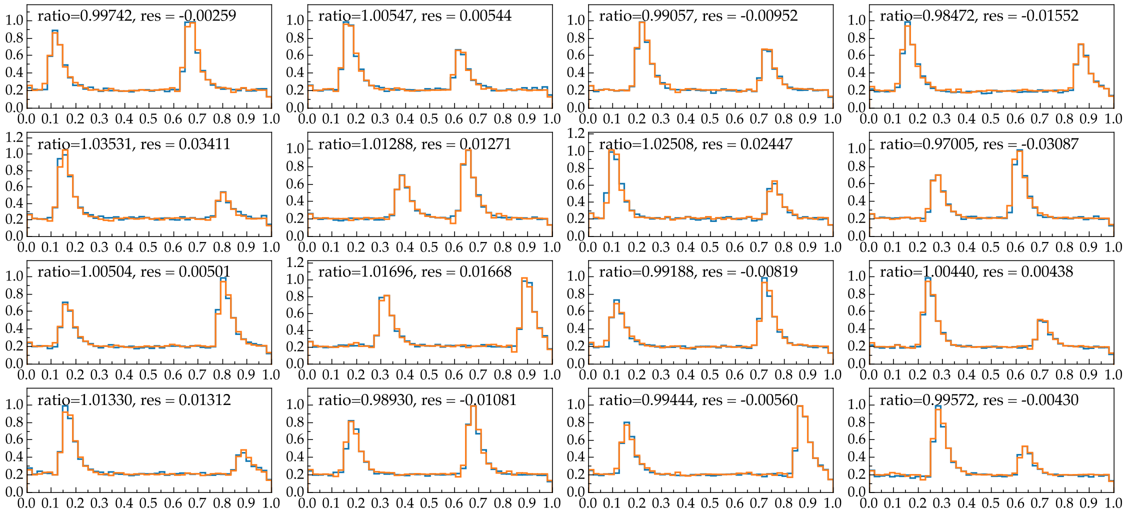

Figure 8: Generated double pulses plotted with the decompressed pulses overlayed. The compression is done with the improved auto-encoder [48,96,48,24,12,24,48.96,48] architecture. The data is produced by simulating a single pulse in the 48-bin region.

The results show significant improvement in peak reconstruction from compressed latent space, where not only the peak positions are well reconstructed, but also the peak heights and withs have very close values to the original raw pulse. And the normalized difference of the peak integrals also shows significant improvements compared to the old network architecture and reaches a resolution of 1.4%.

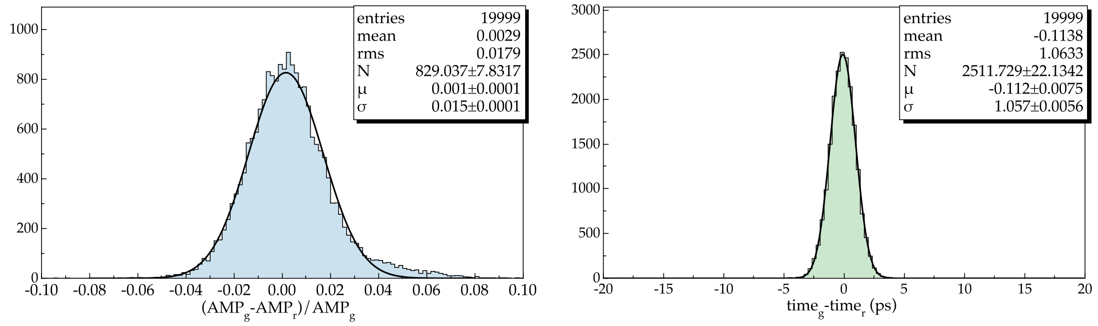

Figure 9: The resolution of the pulse integrals (double pulses). The integral of the pulse is compared to the integral calculated from decompressed (with network [48,96,48,24,12,24,48.96,48]) pulses (on the right), and the time calculated from the original pulse is compared to the time calculated from the decompressed pulse.

5. Studies with Experimental Data

The neural network architecture tested on simulated data was used to train a neural network using real experimental data. As in our simulation studies, the pulses that are measured in one detector are very similar in width due to the same material with the same relaxation time and the same photo-multipliers with the same amplifiers. For our studies, we used pulse data from the Electromagnetic Calorimeter of CLAS12 detector, which makes up about 25% of the data volume produced by CLAS12 data acquisition.

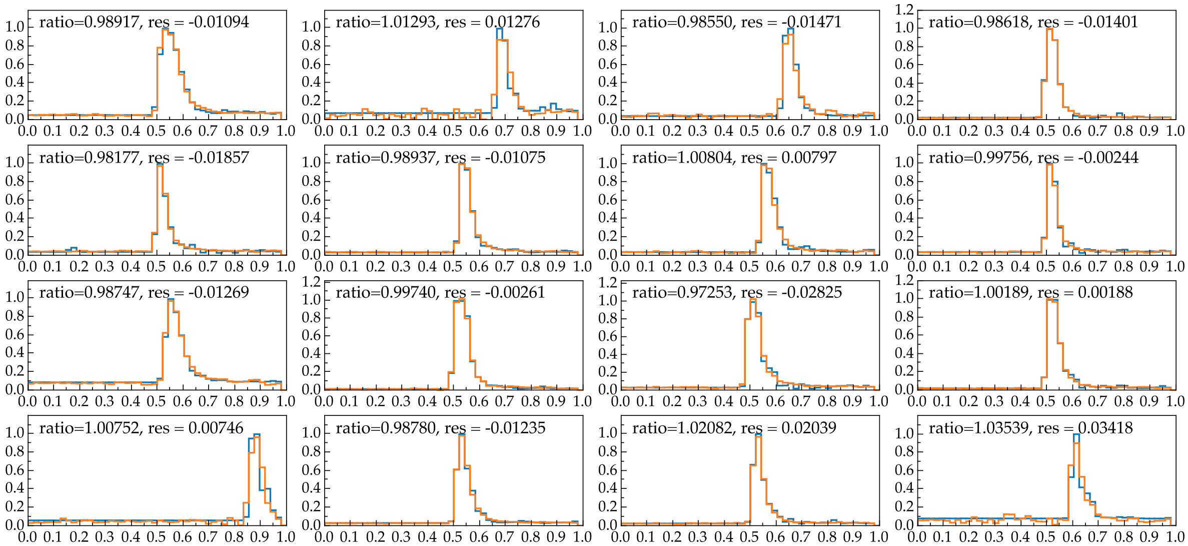

Figure 10: Experimental data pulses plotted with the decompressed pulses overlayed. The compression is done with the improved auto-encoder [48,96,48,24,12,24,48.96,48] architecture.

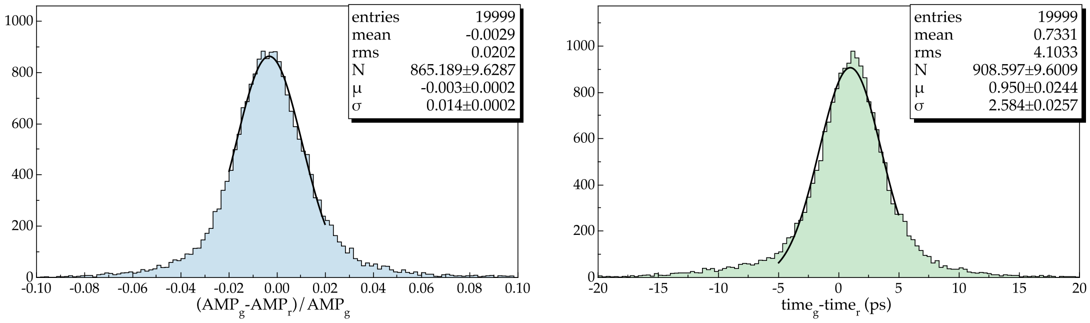

Figure 11: Resolution of the experimental pulses decompressed. Pulse amplitude resolution (on the left) and pulse timing resolution (on the right).

6. Compression Factor Studies

For studies in this article, we used compression factor 4, where the input data with 48 values was compressed into a latent space vector of length 12. However, the latent space vector can be changed to a smaller value to achieve a higher compression factor, but the resolution of the reconstructed pulse integral deteriorates as a function of the compression level. The following studies were done using a latent space layer set to 24, 8 and 6 lengths, increasing the compression ratio to 2, 6 and 8 respectively.

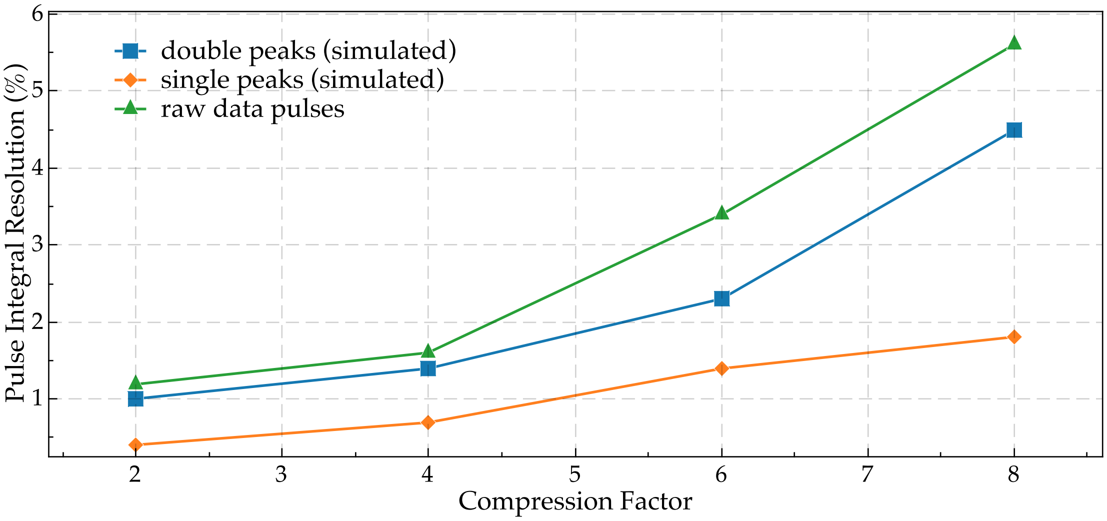

Figure 12: The resolution of decompressed pulse integral as a function of compression factor.

The result for three different data sets are shown in Figure 12, the resolution decrease in single peak simulated data is linear and decreases with the increased compression ratio. In the double peak simulated and raw data the decoded peak resolution decreases rapidly with the compression ratio increase, reaching above 5% for the compression ratio of 8. The studied neural network (autoencoder) is capable of compression pulses into much smaller latent space and preserves very well the position of the peak but has some resolution depending on the compression ratio. Depending on the requirements of the situation different compression ratios can be used.

7. Summary

A new method for raw pulse data compression was developed for CLAS12 using autoencoder neural networks. The neural network is capable of compressing the raw pulse information by a factor of 4, with insignificant loss of resolution (about 2% in real data). The network is capable of compressing multiple pulses in one time window. This method can be used in future high-luminosity runs in CLAS12 where the data volume is large and multi-pulse data is significantly higher in number. This method can be used for other experiments at Jefferson Lab and in future experiments such as EIC where the data rates are expected to be high.

8. Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under contract DE-AC05-06OR23177, and NSF grant no. CCF-1439079 and the Richard T. Cheng Endowment.

References

[1] V. Burkert *et al.*, "The CLAS12 Spectrometer at Jefferson Laboratory," *Nucl. Instrum. Meth. A*, vol. 959, p. 163419, 2020.