

Baler - Machine Learning Based Compression of Scientific Data

F. Bengtsson¹ C. Doglioni² P.A. Ekman¹ A. Gallén¹ P. Jawahar² A. Orucevic-Alagic¹
M. Camps Santasmas² N. Skidmore² O. Woolland²

¹*Lund University*

²*University of Manchester*

ABSTRACT: Storing and sharing increasingly large datasets is a challenge across scientific research and industry. In this paper, we document the development and applications of Baler - a Machine Learning based data compression tool for use across scientific disciplines and industry. Here, we present Baler’s performance for the compression of High Energy Physics (HEP) data, as well as its application to Computational Fluid Dynamics (CFD) toy data as a proof-of-principle. We also present suggestions for cross-disciplinary guidelines to enable feasibility studies for machine learning based compression for scientific data.

1 Introduction

Many different fields of science share a common issue; storing ever-growing datasets. By the end of the next decade, the Large Hadron Collider (LHC) experiments will have over an order of magnitude more data to analyze than currently [1–3]; the Square Kilometre Array (SKA) experiment is expected to record 8.5EB of data over its 15-year lifespan [4] and fields such as Computational Fluid Dynamics (CFD) rely on TB-sized simulation samples that need to be stored and shared. Without significant R&D, the datasets expected to be collected by big-data science experiments are projected to exceed the available storage resources (see e.g. Fig. 2 of Ref. [1] for the case of the ATLAS experiment at the LHC). This cross-disciplinary issue is not limited to scientific research and extends to industrial operations [5].

1.1 Lossy data compression in high energy physics

A common mitigation strategy to this problem involves compressing data using lossless algorithms, see e.g. Refs. [6–8]. Once the storage limit is reached, one is forced to discard parts of the dataset, or only save certain features of the data. Generally, this can be done without impacting the overall scientific program of the experiments, for example by using a data selection system called *trigger* that only stores data satisfying certain pre-determined characteristics that ensure the dataset will be aligned with the experiment’s main scientific goals. However, saving only a subset of data is not ideal for processes where additional

statistical power is necessary, e.g. for rare signals buried in high-rate backgrounds. In these cases, one can foresee using *lossy* compression algorithms that reduce the data size ideally beyond what lossless compression algorithms can do [9], using approximation and partial data discarding, at the expense of data fidelity. One limitation of lossy compression is that to obtain high compression ratios with low data loss, the compression algorithm must be tailored to the input data; for instance, MP3 [10] is an example of a lossy compression algorithm that uses techniques specifically suited for waves and frequencies. Thereby, a general solution to this cross-disciplinary problem is hard to obtain by traditional methods. As a solution to this problem, we present Baler, a lossy data compression tool based on the machine learning autoencoder architecture, which tailors the compression to the user’s dataset. It is also important to note that for such a tool to be usable in a scientific experiment, the loss in data quality must be controlled and it must also be deemed to be tolerable/negligible with respect to other sources of experimental jitter.

1.2 Autoencoders for lossy data compression

Autoencoders (AEs) [11] are a class of unsupervised deep neural networks characterized by an encoder, a central latent space, a decoder, and a target space of the same dimensionality as the input space, as illustrated in Figure 1. The encoder, is a neural network that maps each input, x , to an abstract latent point z , generally of lower dimensionality than the input. The decoder then extrapolates the latent space back to the same dimensions as the input to give the reconstructed output, \hat{x} . AEs can therefore be trained to reconstruct the various features of the input data, while their bottleneck structure prevents them from simply learning the identity map. The dimensionality of the latent space is of particular importance, as it determines the amount of compression achieved, with the latent points being the compressed data and the decoder acting as the decompression algorithm.

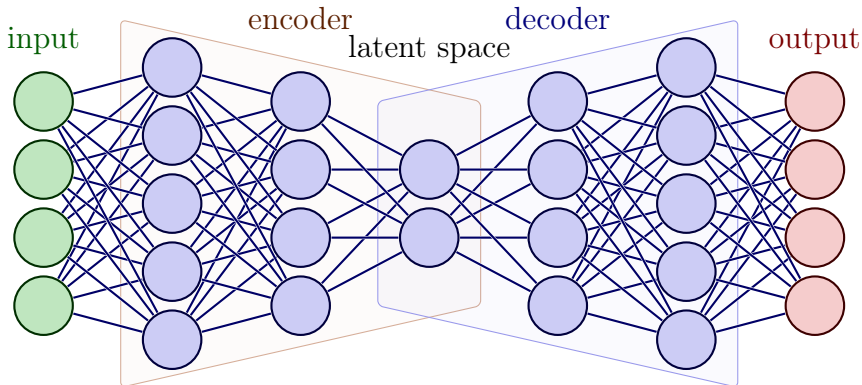


Figure 1: Illustration of an autoencoder consisting of an input and output layer. In between the input and output, there are hidden layers and a latent space, where the dimensionality of the latent space is less, equal, or greater to the input and output layers. Modified from Ref. [12].

AE based compression of scientific data has shown promising results for multiple fields of study such as meteorology, cosmology, computational fluid dynamics, crystallography etc. [13–19]. The use of AEs for data compression in High Energy Physics (HEP) has been also shown to be promising in previous studies [20–23]. A number of these studies focus on the compression of objects directly as the data is taken (*online* compression), which would technically require to train a model on a dataset and using it to compress a different dataset with the same input characteristics. *Offline* compression on the other hand corresponds to the case where the model is trained to compress a dataset and is used to compress that dataset only. In this work, we deal with offline compression as a stepping stone toward online compression and leave the latter for future studies.

2 Baler methodology

AE based compression workflows generally consist of data pre-processing, model architecture selection, model training, compression and decompression using a trained model, and performance evaluation via selected metrics. Baler is an intuitively packaged, modular tool that allows for easy modifications in any component of the workflow. Baler is available in its open-source software repository [24]. In this section, we discuss the setup for HEP data compression as a working example.

2.1 HEP model design

The benchmark HEP AE is built with 3 fully connected neural network layers in both the encoder and decoder. The encoder layers have 200, 100, and 50 nodes respectively, while the decoder has a symmetrically inverted layer structure. We train our models by minimizing the loss function:

$$L_{total} = (1 - \beta)L_{reco} + \beta L_{sparse} \quad (2.1)$$

where, β is a free hyperparameter that controls the contribution of each term to the net loss, L_{reco} is a suitable reconstruction error metric and L_{sparse} is a L_1 -type regularization term to enforce sparsity in the AE weights. In this work we choose L_{reco} to be the mean squared error (MSE) summed over each mini-batch, defined as,

$$\text{MSE} = \frac{1}{n} \sum_{j=1}^m \sum_{i=1}^n (X_i - \hat{X}_i)_j^2$$

Here, m is the batch-size and n is the number of variables in each data entry. X is the vector of batched model inputs, and \hat{X} is the vector corresponding model reconstruction. L_{sparse} introduces sparsity to the model and reduces its storage size, thereby reducing the overheads as discussed in Sec. 2.2. L_{sparse} is defined as,

$$L_{sparse} = \sum_i |w_i| \quad (2.2)$$

and has previously been shown to perform better with regards to HEP data compression [25].

2.2 HEP training setup and evaluation metrics

The models are built using the `PyTorch` [26] framework and optimized using the Adam minimizer [27]. A learning rate of 10^{-3} is used in combination with a learning rate scheduler, namely the `ReduceLROnPlateau` method from `PyTorch`. The scheduler uses a patience of 50 epochs, a reduction factor of 0.5, and a minimum learning rate of 10^{-6} . We train for 1000 epochs with a batch size of 512, and an early stopping strategy with a patience of 100 epochs. L_{total} converges to an order of 10^{-5} .

We choose the residual and response as our performance evaluation metrics, defined as,

$$\text{residual} = \hat{X} - X \tag{2.3}$$

$$\text{response} = \frac{\text{residual}}{X}, \tag{2.4}$$

with X being the original data and \hat{X} being the data reconstructed from the compressed file. Another important metric for compression is the compression ratio, defined as,

$$R = \frac{\text{size(input file)}}{\text{size(compressed file)}}. \tag{2.5}$$

However, unlike some traditional compression algorithms, AE compression requires auxiliary files to be saved. Auxiliary files mainly include decoder weights and biases along with the corresponding `PyTorch` metadata required to load the model when decompressing. This information is saved using the save functionality provided by `PyTorch`. Auxiliary files can also include auxiliary data such as normalization features and data headers. Taking this into account, the actual compression ratio is

$$R_{\text{actual}} = \frac{\text{size(input file)}}{\text{size(decompressed file + auxiliary file(s))}} \tag{2.6}$$

To avoid the results being skewed due to a single specific seed being used in the training, the training was done using 10 different seeds, and the performance evaluation was done on the 5 best performing seeds. For the offline compression case studied here, choosing the best seed is considered as a type of hyperparameter optimization. The impact of different seeds will be studied in the future.

2.3 HEP implementation

For the results presented in this work, Baler release v1.0.0 [28] was used. The computations used to obtain the results presented in this work were performed on the AURORA cluster

hosted by LUNARC [29] at Lund University. The nodes on the cluster are running CentOS 7.2 x86_64 and consist of 2 Intel Xeon E5-2650 v3 (2.3 GHz, 10-core) with 64 GB of Memory (3.2 GB/core). By using the CUDA implementation in PyTorch the training of the AE was done using NVIDIA TESLA K80 GPUs. The prototyping and developing of the GPU implementation of Baler were performed with the help of the Blakett Computing Facility [30] at the University of Manchester.

3 Baler input data

Baler supports NumPy [31] arrays as input and output. This format was chosen because NumPy arrays are an easy-to-handle data format that is already widely used across various scientific disciplines. Also, since the PyTorch [26] library at the core of Baler uses tensors and conversion from the user's original file format is necessary and simple with NumPy arrays. In this initial study, we will focus on HEP data, and touch on preliminary studies using data from CFD.

3.1 HEP input data

Processes involving the strong force dominate proton-proton collision interactions at the LHC. Therefore, one of the most commonly occurring observable objects at experiments like ATLAS and CMS are the collimated showers of particles resulting from these strong processes, reconstructed into *jets* [32].

To showcase Baler's performance on HEP data, we use a subset of the jet data recorded by the CMS experiment at the LHC in 2012, released as open data under the Creative Commons CC0 waiver [33]. In this dataset, each entry is a proton-proton collision *event*, and each event can contain multiple jets. Each collision event is independent from other events and there is no time-dependency in this data. In the data, jets are represented as 4-vectors (p_T, η, ϕ, m) . Where p_T is the momentum of the jet perpendicular to the direction of the colliding proton beams, η is a quantity related to the angle between the particle momentum and the beam, ϕ is the azimuthal angle measured around the beam axis, and m is the mass of the jet. Collectively, these variables are called the jet's four-momentum which are the most relevant variables for LHC measurements and analyses involving jets. Each jet has several other associated variables. For example "jet area" is a measure of the footprint size of the jet. The full list of variables and further information about the content of the dataset we use for testing Baler can be found in Ref. [34]. At this stage, it is not clear whether it would be recommended to use a lossy compression algorithm on the four-momenta, but we include them in the bench-marking of the algorithm for this initial study.

3.2 HEP data pre-processing

For a simpler use of the HEP data in Baler, the data is pre-processed. First, the data is flattened as the original hierarchical data structures of the input data are not supported by current machine learning frameworks such as `PyTorch`. This means that each jet in the dataset is considered independently from the others and correlations within events are lost*. Secondly, features of the data which are non-numerical, are dropped. Both these pre-processing steps are limitations in the applicability of this compression method that can be overcome at a later stage.

This pre-processing step removes nine variables only containing zeroes, and further truncates 15% of the data, yielding a final dataset with a size of 116.9 MB consisting of 24 variables and 608,978 entries. A list of the 24 variables can be found in Appendix A.2.

4 Baler performance on HEP data

As described in Section 2.2, we perform multiple training runs on the same dataset with different random seeds to account for statistical variations introduced in the model by seeds. We visualize the performance of Baler by looking at the 4-momentum variable distributions and their response distribution for one seed, as well as the mean and root mean squared (RMS) values of the response and residual distributions averaged across the five different seeds.

Figure 2 shows, for one seed, the distribution of the 4-momentum variables after compression and decompression with $R = 1.7$. Figure 3 shows the corresponding plot with $R = 6$. Alongside each distribution is a histogram of the response, defined in Eq. 2.4, for that variable. These figures show that Baler achieves good reconstruction of the overall distributions whilst retaining a tight distribution of jet-by-jet variation for these variables at $R = 1.7$, whilst $R = 6$ shows wider distributions and worse overall reconstruction. For example, Figure 2 shows that at $R = 1.7$ the average p_T response is 0.0018. The remaining 20 variables are presented in Appendix A.1.

Regarding the 4-momentum variables only, Table 1 shows the mean and RMS of the residual and response for 5 different seeded runs at $R = 1.7$ and $R = 6$, with an added statistical error of two standard deviations. Appendix A.2 shows the same for the remaining 20 variables.

The difference between R and R_{actual} for HEP data is negligible. The model size on disk reaches approximately 500 KB and, including normalization features plus headers, the total

*For early studies of the impact of considering individual jets versus jets from the same event, see Ref. [35]

auxiliary file size doesn't exceed 550 KB. This means that for our case, where our input file size is 116.9 MB, we obtain: $R = 1.7 \rightarrow R_{\text{actual}} \approx 1.59$, and $R = 6 \rightarrow R_{\text{actual}} \approx 5.8$. As the auxiliary file sizes for HEP do not increase with the number of entries they become negligible.

Table 1: Residual and response distribution means and RMS values for the 4-momentum variables. These values are presented at two different compression ratios, and all values have been averaged over 5 runs of different random seeds.

Variable	Metric	$R = 1.7$		$R = 6$	
		Mean	RMS	Mean	RMS
p_T	Residual	$-1.44 \times 10^{-2} \pm 1.04 \times 10^{-1}$	$2.12 \times 10^{-1} \pm 5.29 \times 10^{-2}$	$-5.60 \times 10^{-2} \pm 1.52 \times 10^{-1}$	$1.17 \times 10^1 \pm 3.13$
	Response	$-1.07 \times 10^{-3} \pm 1.34 \times 10^{-2}$	$2.09 \times 10^{-2} \pm 3.56 \times 10^{-3}$	$9.08 \times 10^{-2} \pm 2.37 \times 10^{-2}$	$3.67 \times 10^{-1} \pm 4.17 \times 10^{-2}$
η	Residual	$-1.12 \times 10^{-3} \pm 2.67 \times 10^{-3}$	$2.09 \times 10^{-3} \pm 1.45 \times 10^{-3}$	$-2.14 \times 10^{-3} \pm 6.21 \times 10^{-3}$	$1.47 \times 10^{-1} \pm 3.67 \times 10^{-2}$
	Response	$3.75 \times 10^{-4} \pm 6.11 \times 10^{-4}$	$8.12 \times 10^{-1} \pm 1.17$	$-5.42 \times 10^{-2} \pm 3.34 \times 10^{-1}$	$8.28 \times 10^1 \pm 1.32 \times 10^2$
ϕ	Residual	$2.45 \times 10^{-4} \pm 1.80 \times 10^{-3}$	$9.91 \times 10^{-4} \pm 1.12 \times 10^{-3}$	$2.52 \times 10^{-4} \pm 1.46 \times 10^{-3}$	$9.92 \times 10^{-3} \pm 2.12 \times 10^{-2}$
	Response	$3.44 \times 10^{-4} \pm 8.64 \times 10^{-4}$	$1.93 \times 10^{-1} \pm 4.32 \times 10^{-1}$	$1.14 \times 10^{-4} \pm 1.52 \times 10^{-3}$	$6.63 \times 10^{-1} \pm 8.32 \times 10^{-1}$
mass	Residual	$-8.05 \times 10^{-3} \pm 2.51 \times 10^{-2}$	$3.98 \times 10^{-2} \pm 1.42 \times 10^{-2}$	$1.22 \times 10^{-2} \pm 2.55 \times 10^{-2}$	$1.86 \pm 1.94 \times 10^{-1}$
	Response	$2.39 \times 10^{-1} \pm 7.87$	$4.38 \times 10^3 \pm 4.47 \times 10^3$	$-1.34 \times 10^1 \pm 5.05 \times 10^1$	$5.95 \times 10^4 \pm 3.42 \times 10^4$

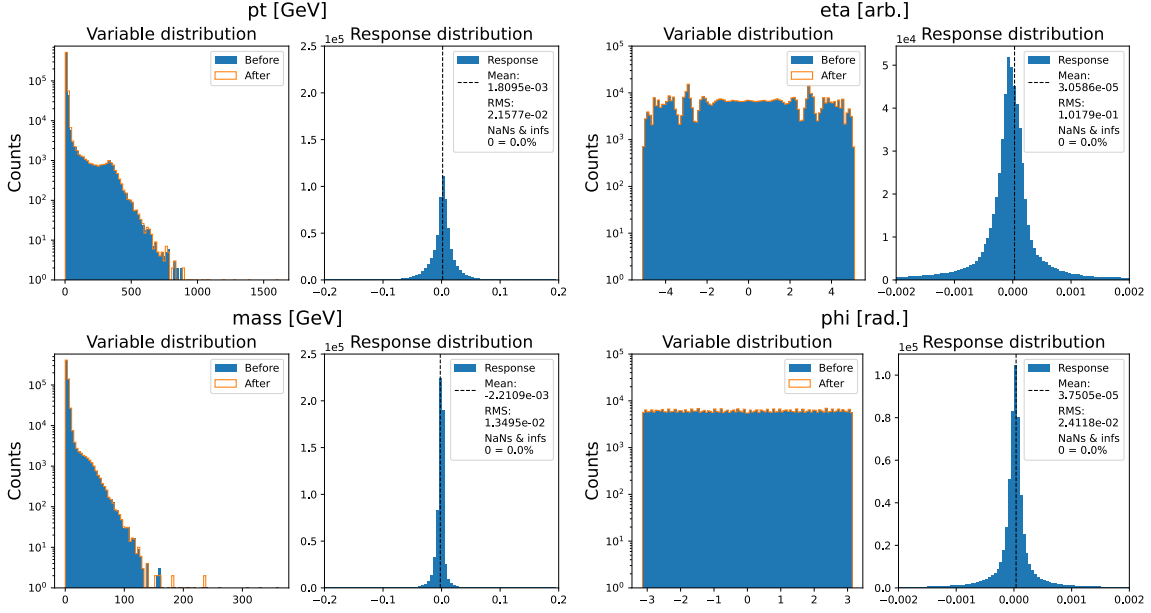


Figure 2: Distributions of the variables making up the 4-momentum of the jets. Alongside each distribution is a histogram of the response for that variable after compression with $R = 1.7$.

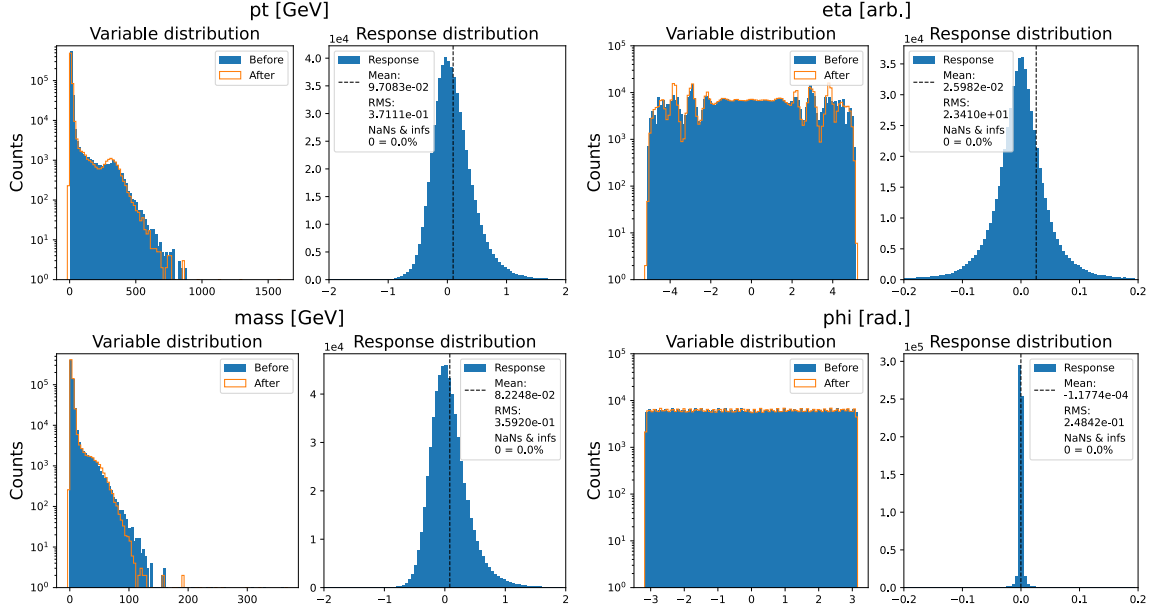


Figure 3: Distributions of the variables making up the 4-momentum of the jets. Alongside each distribution is a histogram of the response for that variable after compression with $R = 6$.

5 Baler application in other scientific fields

Since a major goal of Baler is to investigate the feasibility of AE compression in different fields of science, Baler was also tested on simulated toy data from CFD. The simulated dataset used for this test was the x-component velocity of air flowing over a cube mounted to a wall. For simplicity, we only considered one slice in 3D space, making the compression of the 2D data simple using a convolutional-AE model where the encoder and decoder are Convolutional Neural Networks [36].

Figure 4a and 4b show the 2D data before and after compression and decompression, with $R = 88$. Figure 4c shows the difference between the two which is on a scale four orders of magnitude smaller. As these results show Baler’s wider applications to multiple scientific disciplines.

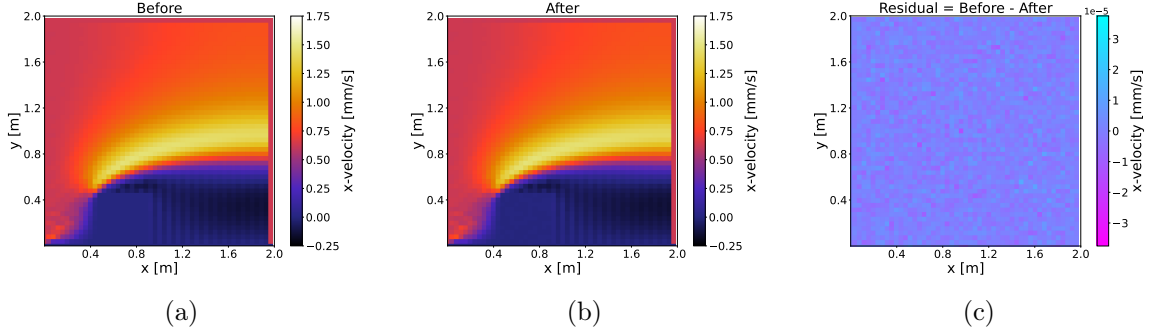


Figure 4: A Computational Fluid Dynamics toy simulation showing x-component of air velocity before compression with $R = 88$ (a), after decompression (b), and the difference between the two (c).

6 Comparison to gzip

Previous studies report a typical `gzip` [37] compression ratio of 1.5 - 2 for scientific data [38, 39]. It has also been shown that an AE based compression can reach compression ratios of 100-500 with a point-wise relative error bound to 0.01 [38].

As shown in Figure 2 and Figure 3, Baler can compress and decompress the HEP data with $R = 1.7$ more accurately than $R = 6.0$. However, for this specific dataset, `gzip` can perform a lossless compression of $R = 4.0$. Furthermore, applying `gzip` to the Baler output, which is already compressed to $R = 1.7$, does not significantly reduce the size of the Baler output. This is because as Baler decreases the dimensionality of the dataset, very few repeating values remain once compressed, and therefore methods like `gzip` are ineffective at compressing the Baler output further. This behavior is true even for larger file sizes: 250, 500, 750, 1000, 1250, and 1500 MB, as shown in Appendix A.3.

On the contrary, for the CFD data shown in Figure 4, `gzip` achieves $R = 2.2$ whilst Baler achieves $R = 88$. The dilemma here is that the decoder Baler produces as part of the auxiliary files is 0.6 GB, making $R_{\text{actual}} < 1$. This is however not a major concern, since other studies have shown good performance with respect to the overhead [38], and we expect to achieve similar lightweight models in the future. It is also important to note that the CFD dataset used in this work is a small toy since this is a proof-of-concept study. With AE compression, a model of fixed size can compress both small and relatively larger datasets, with the model overhead being large in the first case and much smaller in the second. We see this attenuating model overhead trend with HEP data, and we expect to see the same for CFD data, pending further studies with larger datasets.

7 Conclusions and Outlook

In this work, we motivate the need for effective data compression strategies as a solution to the growing storage issues related to large data volumes across many disciplines of scientific research. We present Baler as a modular solution to leverage machine learning based lossy data compression. We identify and define two major use cases of the tool namely, online and offline data compression. We evaluate performance for offline compression of HEP data using autoencoders and provide a guide to performing similar feasibility studies for other disciplines of research using the tool we have developed. We also study the compression of CFD data as a proof-of-concept to demonstrate Baler’s flexibility.

Near future extensions to this work include assessing performance variations related to dataset sizes and support for error-bound compression where a specified error metric is used as a limit on the maximum possible compression ratio. Though we provide guidelines for using Baler to perform feasibility studies for a given dataset, there is currently no method to quickly project the likelihood of a dataset being suitable for compression with Baler. A potential way to implement this is to calculate a coefficient of variation for a given dataset as described in [38] and use this as a likelihood metric. This implementation along with studies on other potential solutions for this problem are marked as features for future releases of Baler.

To deal with variations across dataset sizes we plan to perform follow-up studies by exploring different HEP datasets and input representations that may involve the use of low-level detector data. Another related limitation is compressing files larger than RAM, where we plan to test industry standards such as optimal caching of objects in memory. These solutions are viable for offline compression since there are no associated latency or resource constraints in this case.

However, online compression is a major area of study we intend to carry out given its high potential to be used in large HEP experiments that generate data at very high rates. To tackle the problem of online compression we would need better generalization capabilities within the machine learning models and a potential way to achieve this with unsupervised learning is to use probabilistic generative models such as variational autoencoders and normalizing flows.

Acknowledgments

This work was supported by the Turing-Manchester Feasibility Project Funding.

References

- [1] ATLAS Collaboration. *ATLAS Software and Computing HL-LHC Roadmap*. Tech. rep. Geneva: CERN, 2022. URL: <http://cds.cern.ch/record/2802918>.
- [2] CMS Offline Software and Computing. *CMS Phase-2 Computing Model: Update Document*. Tech. rep. Geneva: CERN, 2022. URL: <http://cds.cern.ch/record/2815292>.
- [3] LHCb Collaboration. *Computing Model of the Upgrade LHCb experiment*. Tech. rep. Geneva: CERN, 2018. DOI: [10.17181/CERN.QOP4.570N](https://doi.org/10.17181/CERN.QOP4.570N). URL: <http://cds.cern.ch/record/2319756>.
- [4] A. M. M. Scaife. In: *Phil. Trans. R. Soc. A*. 378 (2166 2020).
- [5] Maqbool Khan, Xiaotong Wu, Xiaolong Xu, and Wanchun Dou. “Big data challenges and opportunities in the hype of Industry 4.0”. In: *2017 IEEE International Conference on Communications (ICC)*. 2017, pp. 1–6. DOI: [10.1109/ICC.2017.7996801](https://doi.org/10.1109/ICC.2017.7996801).
- [6] Oksana Shadura, Brian Paul Bockelman, Philippe Canal, Danilo Piparo, and Zhe Zhang. “ROOT I/O compression improvements for HEP analysis”. In: *EPJ Web of Conferences* 245 (2020), p. 02017. DOI: [10.1051/epjconf/202024502017](https://doi.org/10.1051/epjconf/202024502017). URL: <https://doi.org/10.1051/5C%2Fepjconf%5C%2F202024502017>.
- [7] Christian Patauner. “Lossy and lossless data compression of data from high energy physics experiments”. Presented 2011. 2011. URL: <https://cds.cern.ch/record/1433839>.
- [8] Arjun Rawal. “Exploiting Domain-specific Data Properties to Improve Compression for High Energy Physics Data”. Presented 2020. 2020. URL: <https://newtraell.cs.uchicago.edu/research/publications/techreports/TR-2020-03>.
- [9] Khalid Sayood. *Introduction to data compression*. Morgan Kaufmann, 2017.
- [10] Karlheinz Brandenburg. “MP3 and AAC explained”. In: *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. Audio Engineering Society. 1999.

- [11] Mark A. Kramer. “Nonlinear principal component analysis using autoassociative neural networks”. In: *AIChE Journal* 37.2 (1991), pp. 233–243. DOI: [10.1002/aic.690370209](https://doi.org/10.1002/aic.690370209). eprint: <https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690370209>. URL: <https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209>.
- [12] Izaak Neutelings. *Neural Networks*. [Online; accessed 02-May-2023; Last edited 11 September 2022]. 2021. URL: https://tikz.net/neural_networks/.
- [13] Tong Liu, Jinzhen Wang, Qing Liu, Shakeel Alibhai, Tao Lu, and Xubin He. “High-Ratio Lossy Compression: Exploring the Autoencoder to Compress Scientific Data”. In: *IEEE Transactions on Big Data* 9.1 (2023), pp. 22–36. DOI: [10.1109/TBDATA.2021.3066151](https://doi.org/10.1109/TBDATA.2021.3066151).
- [14] Jinyang Liu, Sheng Di, Kai Zhao, Sian Jin, Dingwen Tao, Xin Liang, Zizhong Chen, and Franck Cappello. “Exploring Autoencoder-based Error-bounded Compression for Scientific Data”. In: *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. 2021, pp. 294–306. DOI: [10.1109/Cluster48925.2021.00034](https://doi.org/10.1109/Cluster48925.2021.00034).
- [15] Nan Wang, Tong Liu, Jinzhen Wang, Qing Liu, Shakeel Alibhai, and Xubin He. “Locality-based transfer learning on compression autoencoder for efficient scientific data lossy compression”. In: *Journal of Network and Computer Applications* 205 (2022), p. 103452.
- [16] Riccardo La Grassa, Cristina Re, Gabriele Cremonese, and Ignazio Gallo. “Hyperspectral Data Compression Using Fully Convolutional Autoencoder”. In: *Remote Sensing* 14.10 (2022), p. 2472.
- [17] Yi Huang, Yihui Ren, Shinjae Yoo, and Jin Huang. “Efficient Data Compression for 3D Sparse TPC via Bicephalous Convolutional Autoencoder”. In: *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2021, pp. 1094–1099. DOI: [10.1109/ICMLA52953.2021.00179](https://doi.org/10.1109/ICMLA52953.2021.00179).
- [18] Jaemoon Lee, Qian Gong, Jong Choi, Tania Banerjee, Scott Klasky, Sanjay Ranka, and Anand Rangarajan. “Error-bounded learned scientific data compression with preservation of derived quantities”. In: *Applied Sciences* 12.13 (2022), p. 6718.
- [19] S Sriram, Arun K Dwivedi, P Chitra, V Vijay Sankar, S Abirami, SJ Rethina Durai, Divya Pandey, and Manoj K Khare. “DeepComp: A Hybrid Framework for Data Compression Using Attention Coupled Autoencoder”. In: *Arabian Journal for Science and Engineering* 47.8 (2022), pp. 10395–10410.
- [20] Eric Wulff. *Deep Autoencoders for Compression in High Energy Physics*. eng. Student Paper. 2020. URL: <http://lup.lub.lu.se/student-papers/record/9004751>.
- [21] Erik Wallin. *Tests of Autoencoder Compression of Trigger Jets in the ATLAS Experiment*. eng. Student Paper. 2020. URL: <http://lup.lub.lu.se/student-papers/record/9012882>.

- [22] Jack H. Collins, Yifeng Huang, Simon Knapen, Benjamin Nachman, and Daniel Whiteson. “Machine-Learning Compression for Particle Physics Discoveries”. In: (Oct. 2022). arXiv: [2210.11489](https://arxiv.org/abs/2210.11489) [[hep-ph](#)].
- [23] Constantin Weisser and Mike Williams. *"Autoencoders for LHCb", presented at the Reconstruction, Trigger, and Machine Learning for the HL-LHC" MIT workshop*. eng. Student Presentation. 2018. URL: https://indico.cern.ch/event/714134/contributions/2964667/attachments/1641424/2621410/Autoencoder_MIT_Weisser.pdf.
- [24] Baler Collaboration. *Baler*. <https://github.com/baler-collaboration/baler>. 2023.
- [25] Dialektakis George. *Deep Autoencoders for ATLAS Data Compression - George Dialektakis - Google Summer of Code 2021 Project*. Version Version 1. Sept. 2021. DOI: [10.5281/zenodo.5482611](https://doi.org/10.5281/zenodo.5482611). URL: <https://doi.org/10.5281/zenodo.5482611>.
- [26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [27] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: [10.48550/ARXIV.1412.6980](https://arxiv.org/abs/1412.6980). URL: <https://arxiv.org/abs/1412.6980>.
- [28] Per Alexander Ekman, Axel Gallén, Pratik Jawahar, Fritjof Bengtsson, Oliver Woolland, Caterina Doglioni, Marta Camps Santasmasas, Nicola Skidmore, and Alma Orucevic-Alagic. *baler-collaboration/baler: v1.0.0*. Version v1.0.0. Apr. 2023. DOI: [10.5281/zenodo.7817467](https://doi.org/10.5281/zenodo.7817467). URL: <https://doi.org/10.5281/zenodo.7817467>.
- [29] *Aurora / LUNARC*. [Accessed: 2023-04-11]. 2019. URL: <https://www.lunarc.lu.se/resources/hardware/aurora/>.
- [30] *Blackett Computing Facility*. [Accessed: 2023-04-11]. 2001. URL: <https://www.blackett.manchester.ac.uk/>.
- [31] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. “Array programming with NumPy”. In: *Nature* 585:7825 (Sept.

- 2020), pp. 357–362. DOI: [10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2). URL: <https://doi.org/10.1038/s41586-020-2649-2>.
- [32] G. P. Salam. “Elements of QCD for hadron colliders”. In: *CERN Yellow Rep. School Proc.* 5 (2020). Ed. by M. Mulders and J. Trần Thanh Vân, pp. 1–56. DOI: [10.23730/CYRSP-2020-005.1](https://doi.org/10.23730/CYRSP-2020-005.1).
- [33] CMS collaboration. *JetHT primary dataset in AOD format from Run of 2012*. Tech. rep. 2017. DOI: [DOI:10.7483/OPENDATA.CMS.KL8H.HFVH](https://doi.org/10.7483/OPENDATA.CMS.KL8H.HFVH). URL: <http://opendata.cern.ch/record/6010>.
- [34] CMS Collaboration. “*CMS Physics Objects 2015*”. eng. Online. 2015. URL: <http://opendata.cern.ch/docs/cms-physics-objects-2015>.
- [35] Sten Aastrand. *Autoencoder Compression in High Energy Physics*. eng. Student Paper. 2022. URL: <http://lup.lub.lu.se/student-papers/record/9004751>.
- [36] Yann LeCun, Yoshua Bengio, et al. “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.
- [37] J.-l. Gailly. *gzip: The data compression program*. Apr. 2022. URL: <https://www.gnu.org/software/gzip/manual/gzip.pdf>.
- [38] Tong Liu, Jinzhen Wang, Qing Liu, Shakeel Alibhai, Tao Lu, and Xubin He. “High-Ratio Lossy Compression: Exploring the Autoencoder to Compress Scientific Data”. In: *IEEE Transactions on Big Data* 9.1 (2023), pp. 22–36. DOI: [10.1109/TBDATA.2021.3066151](https://doi.org/10.1109/TBDATA.2021.3066151).
- [39] Tao Lu, Qing Liu, Xubin He, Huizhang Luo, Eric Suchyta, Jong Choi, Norbert Podhorszki, Scott Klasky, Mathew Wolf, Tong Liu, and Zhenbo Qiao. “Understanding and Modeling Lossy Compression Schemes on HPC Scientific Data”. In: *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2018, pp. 348–357. DOI: [10.1109/IPDPS.2018.00044](https://doi.org/10.1109/IPDPS.2018.00044).

A Additional Plots

A.1 Variable distribution plots

Here, the distribution of the 24 variables compressed is displayed against their decompressed counterpart at different compression ratios. Note that the definition of the response leads to inf and NaN values which are not represented in these response distributions. The extent of this is presented in the legend of the following figures.

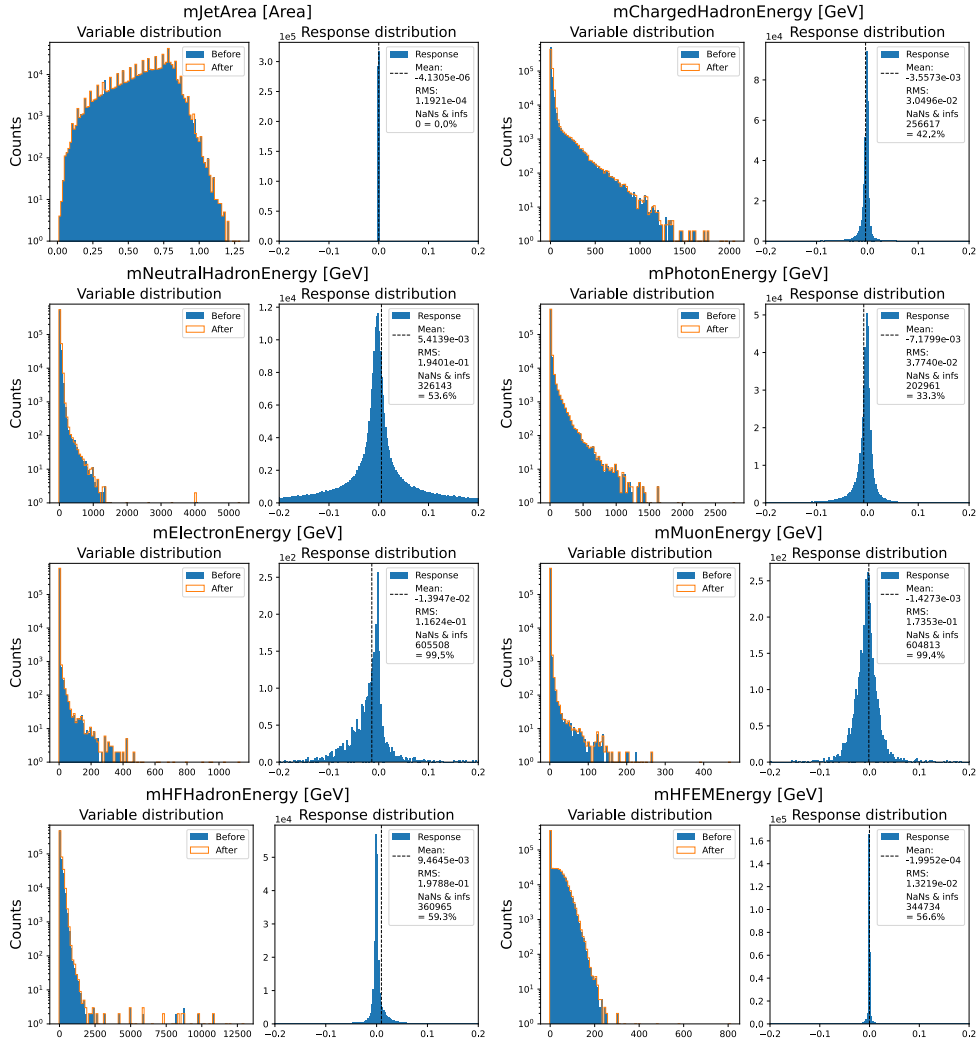


Figure 5: Original distribution and reconstructed distributions of eight jet variables after compression with $R = 1.7$. Alongside each distribution is a histogram of the response for that variable.

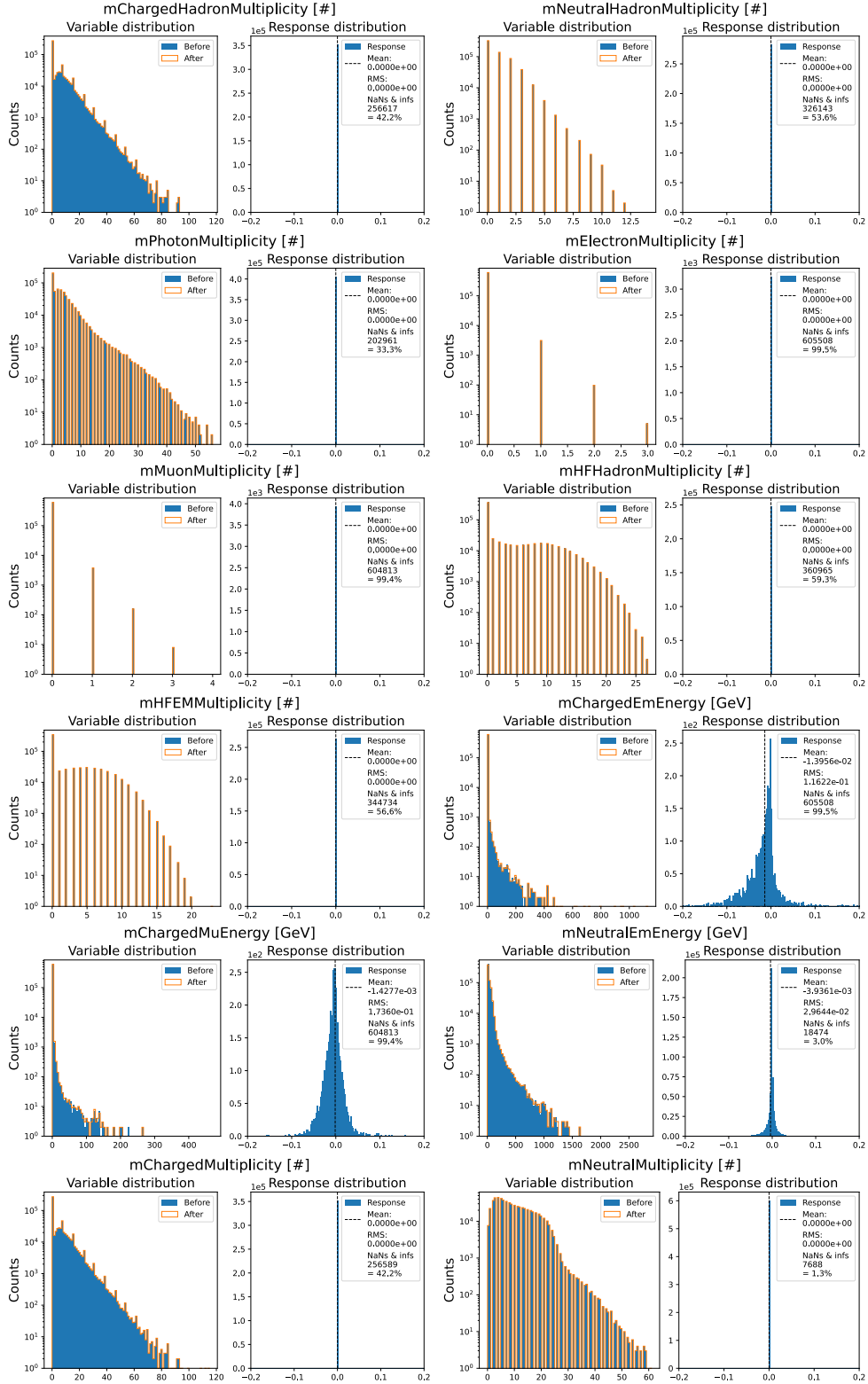


Figure 6: Original distribution and reconstructed distributions of twelve jet variables after compression with $R = 1.7$. Alongside each distribution is a histogram of the response for that variable.

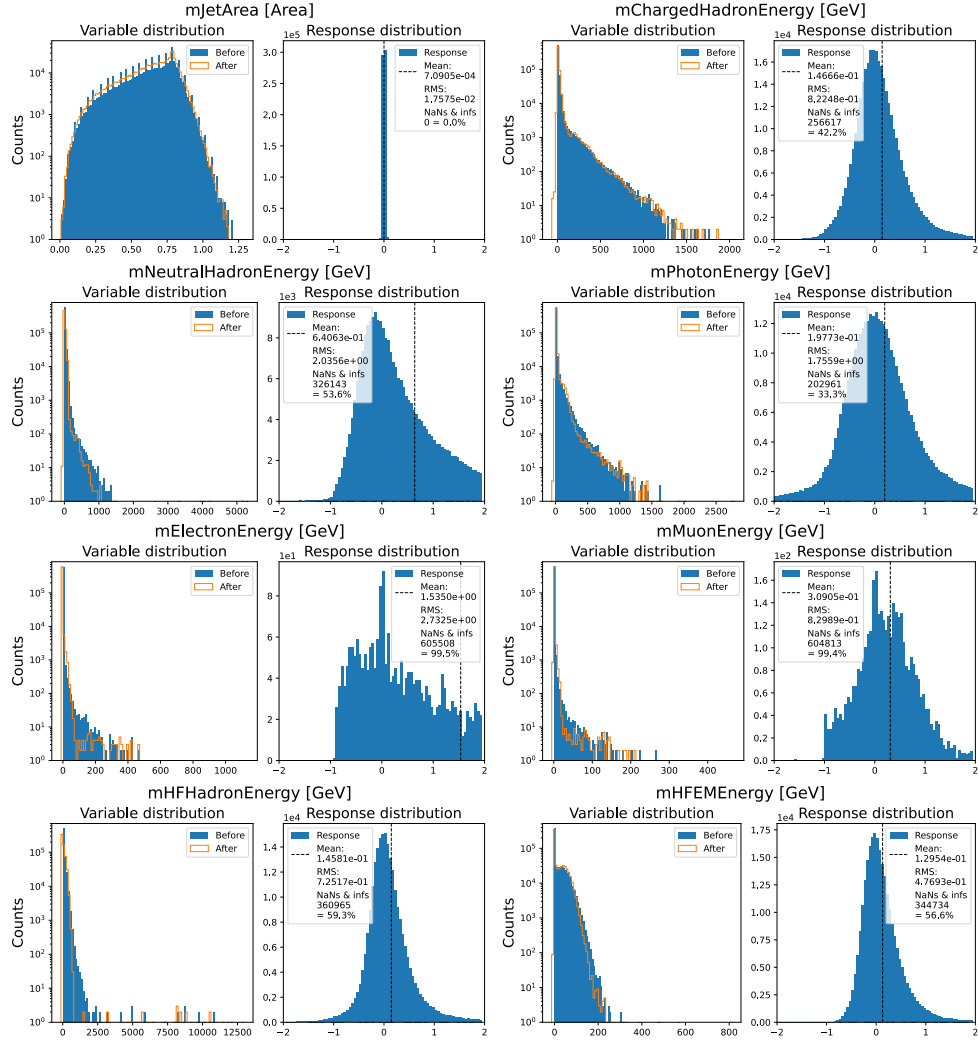


Figure 7: Original distribution and reconstructed distributions of eight jet variables after compression with $R = 6$. Alongside each distribution is a histogram of the response for that variable.

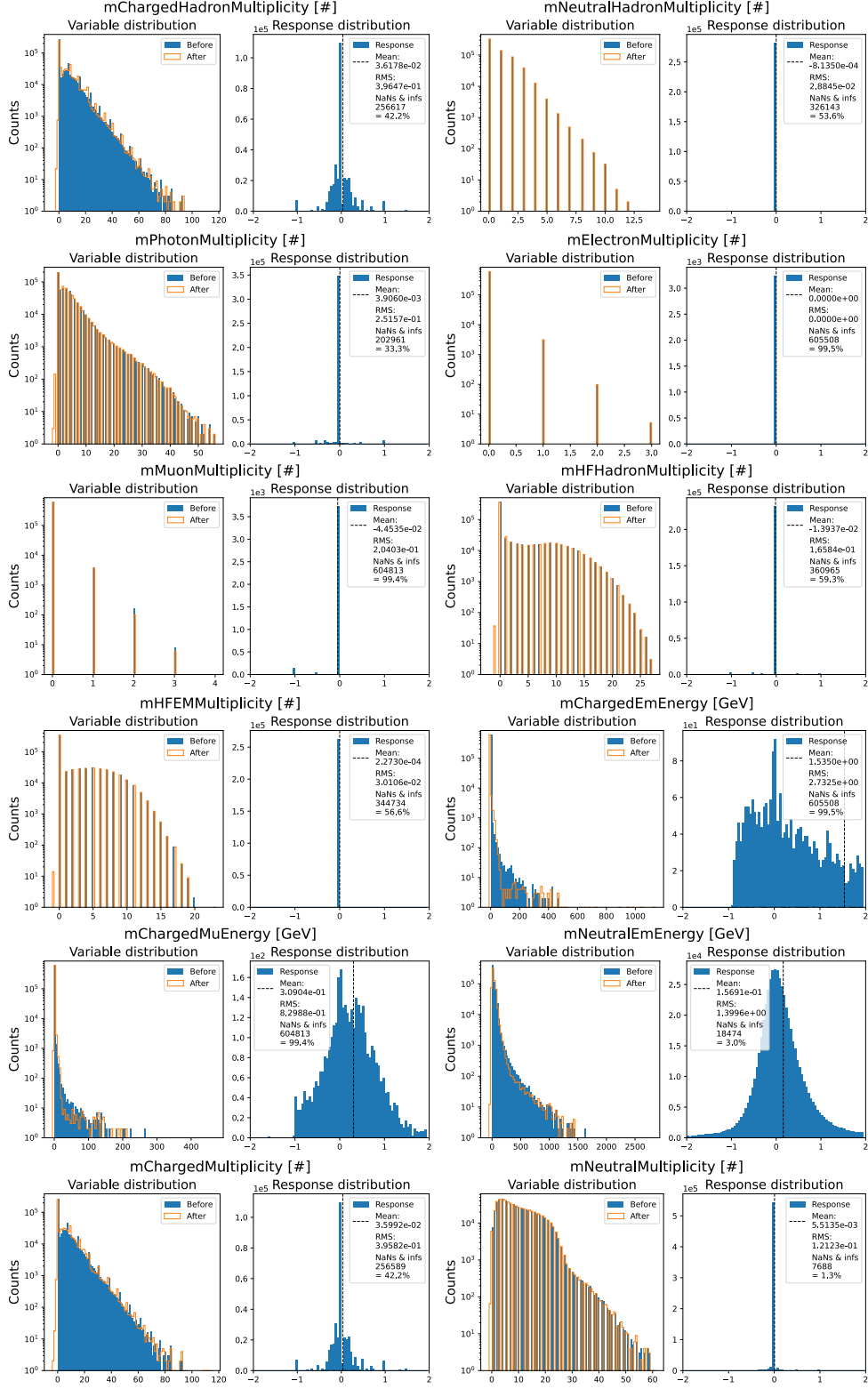


Figure 8: Original distribution and reconstructed distributions of twelve jet variables after compression with $R = 6$. Alongside each distribution is a histogram of the response for that variable.

A.2 Variable evaluation metrics

Here, the final averaged evaluation metrics of all variables are presented. Table 2 shows the values for $R = 1.7$, while Table 3 shows it for $R = 6$.

Table 2: Residual and Response distribution means and RMS values for all variables in the dataset. These values are presented at $R = 1.7$, and all values have been averaged over 5 runs, with an added statistical error of two standard deviations.

Variable ($R = 1.7$)	Response		Residual	
	Mean	RMS	Mean	RMS
p_T	$-1.07 \times 10^{-3} \pm 1.34 \times 10^{-2}$	$2.09 \times 10^{-2} \pm 3.56 \times 10^{-3}$	$-1.44 \times 10^{-2} \pm 1.04 \times 10^{-1}$	$2.12 \times 10^{-1} \pm 5.29 \times 10^{-2}$
η	$3.75 \times 10^{-4} \pm 6.11 \times 10^{-4}$	$8.12 \times 10^{-1} \pm 1.17$	$-1.12 \times 10^{-3} \pm 2.67 \times 10^{-3}$	$2.09 \times 10^{-3} \pm 1.45 \times 10^{-3}$
ϕ	$3.44 \times 10^{-4} \pm 8.64 \times 10^{-4}$	$1.93 \times 10^{-1} \pm 4.32 \times 10^{-1}$	$2.45 \times 10^{-4} \pm 1.80 \times 10^{-3}$	$9.91 \times 10^{-4} \pm 1.12 \times 10^{-3}$
mass	$2.39 \times 10^{-1} \pm 7.87$	$4.38 \times 10^3 \pm 4.47 \times 10^3$	$-8.05 \times 10^{-3} \pm 2.51 \times 10^{-2}$	$3.98 \times 10^{-2} \pm 1.42 \times 10^{-2}$
mJetArea	$6.12 \times 10^{-5} \pm 1.81 \times 10^{-4}$	$3.13 \times 10^{-4} \pm 1.48 \times 10^{-4}$	$3.21 \times 10^{-5} \pm 8.90 \times 10^{-5}$	$1.10 \times 10^{-4} \pm 5.77 \times 10^{-5}$
mChargedHadronEnergy	$1.58 \times 10^{-3} \pm 1.70 \times 10^{-2}$	$2.85 \times 10^{-2} \pm 1.30 \times 10^{-2}$	$1.68 \times 10^{-2} \pm 1.43 \times 10^{-1}$	$1.71 \times 10^{-1} \pm 7.33 \times 10^{-2}$
mNeutralHadronEnergy	$7.05 \times 10^{-2} \pm 9.88 \times 10^{-2}$	$2.22 \times 10^{-1} \pm 6.59 \times 10^{-2}$	$2.77 \times 10^{-1} \pm 5.23 \times 10^{-1}$	$6.94 \times 10^{-1} \pm 2.26 \times 10^{-1}$
mPhotonEnergy	$-2.75 \times 10^{-2} \pm 7.48 \times 10^{-2}$	$6.84 \times 10^{-2} \pm 1.09 \times 10^{-1}$	$-8.00 \times 10^{-2} \pm 1.87 \times 10^{-1}$	$1.52 \times 10^{-1} \pm 1.77 \times 10^{-1}$
mElectronEnergy	$-7.71 \times 10^{-2} \pm 1.05 \times 10^{-1}$	$1.44 \times 10^{-1} \pm 7.47 \times 10^{-2}$	$1.71 \times 10^{-2} \pm 5.32 \times 10^{-2}$	$8.40 \times 10^{-2} \pm 4.15 \times 10^{-2}$
mMuonEnergy	$1.29 \times 10^{-2} \pm 1.97 \times 10^{-2}$	$8.04 \times 10^{-2} \pm 9.77 \times 10^{-2}$	$1.18 \times 10^{-2} \pm 1.46 \times 10^{-2}$	$3.15 \times 10^{-2} \pm 7.05 \times 10^{-3}$
mHFHadronEnergy	$-1.10 \times 10^{-2} \pm 4.66 \times 10^{-2}$	$1.77 \times 10^{-1} \pm 2.48 \times 10^{-2}$	$-3.15 \times 10^{-1} \pm 1.07$	$1.85 \pm 7.31 \times 10^{-1}$
mHFEMEnergy	$1.78 \times 10^{-3} \pm 7.40 \times 10^{-3}$	$1.41 \times 10^{-2} \pm 3.63 \times 10^{-3}$	$1.22 \times 10^{-2} \pm 8.26 \times 10^{-2}$	$6.93 \times 10^{-2} \pm 5.54 \times 10^{-2}$
mChargedHadronMultiplicity	$-1.00 \times 10^{-3} \pm 5.04 \times 10^{-3}$	$4.48 \times 10^{-3} \pm 4.90 \times 10^{-3}$	$-3.13 \times 10^{-3} \pm 1.82 \times 10^{-2}$	$9.68 \times 10^{-3} \pm 1.50 \times 10^{-2}$
mNeutralHadronMultiplicity	$-1.22 \times 10^{-4} \pm 1.29 \times 10^{-3}$	$8.76 \times 10^{-4} \pm 9.42 \times 10^{-4}$	$-1.19 \times 10^{-4} \pm 1.51 \times 10^{-3}$	$9.89 \times 10^{-4} \pm 1.20 \times 10^{-3}$
mPhotonMultiplicity	$-1.14 \times 10^{-3} \pm 3.62 \times 10^{-3}$	$2.72 \times 10^{-3} \pm 4.14 \times 10^{-3}$	$-2.69 \times 10^{-3} \pm 7.44 \times 10^{-3}$	$4.92 \times 10^{-3} \pm 7.12 \times 10^{-3}$
mElectronMultiplicity	$1.07 \times 10^{-3} \pm 3.87 \times 10^{-3}$	$2.37 \times 10^{-3} \pm 2.37 \times 10^{-3}$	$-1.54 \times 10^{-5} \pm 9.96 \times 10^{-5}$	$2.11 \times 10^{-4} \pm 1.75 \times 10^{-4}$
mMuonMultiplicity	$1.12 \times 10^{-3} \pm 1.22 \times 10^{-3}$	$2.51 \times 10^{-3} \pm 6.69 \times 10^{-4}$	$5.67 \times 10^{-5} \pm 1.16 \times 10^{-4}$	$2.41 \times 10^{-4} \pm 6.35 \times 10^{-5}$
mHFHadronMultiplicity	$-1.34 \times 10^{-3} \pm 1.84 \times 10^{-3}$	$2.53 \times 10^{-3} \pm 1.94 \times 10^{-3}$	$-2.67 \times 10^{-3} \pm 3.33 \times 10^{-3}$	$4.44 \times 10^{-3} \pm 4.05 \times 10^{-3}$
mHFEMMultiplicity	$2.41 \times 10^{-4} \pm 2.51 \times 10^{-3}$	$1.98 \times 10^{-3} \pm 1.33 \times 10^{-3}$	$5.98 \times 10^{-4} \pm 4.16 \times 10^{-3}$	$3.08 \times 10^{-3} \pm 2.95 \times 10^{-3}$
mChargedEmEnergy	$-7.72 \times 10^{-2} \pm 1.05 \times 10^{-1}$	$1.44 \times 10^{-1} \pm 7.48 \times 10^{-2}$	$1.72 \times 10^{-2} \pm 5.30 \times 10^{-2}$	$8.40 \times 10^{-2} \pm 4.15 \times 10^{-2}$
mChargedMuEnergy	$1.29 \times 10^{-2} \pm 1.97 \times 10^{-2}$	$8.05 \times 10^{-2} \pm 9.78 \times 10^{-2}$	$1.18 \times 10^{-2} \pm 1.46 \times 10^{-2}$	$3.15 \times 10^{-2} \pm 7.07 \times 10^{-3}$
mNeutralEmEnergy	$-1.73 \times 10^{-2} \pm 5.42 \times 10^{-2}$	$5.89 \times 10^{-2} \pm 8.87 \times 10^{-2}$	$-6.70 \times 10^{-2} \pm 2.57 \times 10^{-1}$	$1.75 \times 10^{-1} \pm 1.81 \times 10^{-1}$
mChargedMultiplicity	$-9.83 \times 10^{-4} \pm 5.04 \times 10^{-3}$	$4.46 \times 10^{-3} \pm 4.88 \times 10^{-3}$	$-3.07 \times 10^{-3} \pm 1.83 \times 10^{-2}$	$9.74 \times 10^{-3} \pm 1.51 \times 10^{-2}$
mNeutralMultiplicity	$-8.97 \times 10^{-4} \pm 1.42 \times 10^{-3}$	$1.56 \times 10^{-3} \pm 1.93 \times 10^{-3}$	$-5.36 \times 10^{-3} \pm 7.37 \times 10^{-3}$	$7.34 \times 10^{-3} \pm 6.60 \times 10^{-3}$

Table 3: Residual and Response distribution means and RMS values for all variables in the dataset. These values are presented at $R = 6$, and all values have been averaged over 5 runs, with an added statistical error of two standard deviations.

Variable ($R = 6$)	Response		Residual	
	Mean	RMS	Mean	RMS
p_T	$9.08 \times 10^{-2} \pm 2.37 \times 10^{-2}$	$3.67 \times 10^{-1} \pm 4.17 \times 10^{-2}$	$-5.60 \times 10^{-2} \pm 1.52 \times 10^{-1}$	$1.17 \times 10^1 \pm 3.13$
η	$-5.42 \times 10^{-2} \pm 3.34 \times 10^{-1}$	$8.28 \times 10^1 \pm 1.32 \times 10^2$	$-2.14 \times 10^{-3} \pm 6.21 \times 10^{-3}$	$1.47 \times 10^{-1} \pm 3.67 \times 10^{-2}$
ϕ	$1.14 \times 10^{-4} \pm 1.52 \times 10^{-3}$	$6.63 \times 10^{-1} \pm 8.32 \times 10^{-1}$	$2.52 \times 10^{-4} \pm 1.46 \times 10^{-3}$	$9.92 \times 10^{-3} \pm 2.12 \times 10^{-2}$
mass	$-1.34 \times 10^1 \pm 5.05 \times 10^1$	$5.95 \times 10^4 \pm 3.42 \times 10^4$	$1.22 \times 10^{-2} \pm 2.55 \times 10^{-2}$	$1.86 \pm 1.94 \times 10^{-1}$
mJetArea	$2.77 \times 10^{-4} \pm 9.83 \times 10^{-4}$	$2.01 \times 10^{-2} \pm 4.58 \times 10^{-3}$	$-1.64 \times 10^{-5} \pm 5.66 \times 10^{-4}$	$1.16 \times 10^{-2} \pm 2.13 \times 10^{-3}$
mChargedHadronEnergy	$1.29 \times 10^{-1} \pm 2.32 \times 10^{-2}$	$8.37 \times 10^{-1} \pm 1.09 \times 10^{-1}$	$-9.49 \times 10^{-2} \pm 2.37 \times 10^{-1}$	$1.58 \times 10^1 \pm 2.09$
mNeutralHadronEnergy	$6.51 \times 10^{-1} \pm 3.75 \times 10^{-2}$	$2.02 \pm 1.35 \times 10^{-1}$	$8.35 \times 10^{-2} \pm 2.99 \times 10^{-1}$	$1.74 \times 10^1 \pm 1.09$
mPhotonEnergy	$2.02 \times 10^{-1} \pm 1.38 \times 10^{-1}$	$1.71 \pm 1.93 \times 10^{-1}$	$-2.70 \times 10^{-1} \pm 3.79 \times 10^{-1}$	$1.52 \times 10^1 \pm 1.51$
mElectronEnergy	$1.52 \pm 2.06 \times 10^{-1}$	$2.70 \pm 2.67 \times 10^{-1}$	$-3.08 \times 10^{-3} \pm 2.08 \times 10^{-2}$	$2.12 \pm 3.23 \times 10^{-1}$
mMuonEnergy	$3.53 \times 10^{-1} \pm 1.13 \times 10^{-1}$	$9.15 \times 10^{-1} \pm 1.01 \times 10^{-1}$	$-9.21 \times 10^{-3} \pm 1.14 \times 10^{-2}$	$7.81 \times 10^{-1} \pm 1.85 \times 10^{-1}$
mHFHadronEnergy	$1.43 \times 10^{-1} \pm 6.72 \times 10^{-2}$	$7.15 \times 10^{-1} \pm 1.64 \times 10^{-1}$	$-1.99 \times 10^{-2} \pm 3.71 \times 10^{-1}$	$3.59 \times 10^1 \pm 8.51 \times 10^{-1}$
mHFEMEnergy	$1.35 \times 10^{-1} \pm 2.05 \times 10^{-2}$	$4.91 \times 10^{-1} \pm 4.87 \times 10^{-2}$	$9.50 \times 10^{-2} \pm 9.11 \times 10^{-2}$	$8.70 \pm 7.63 \times 10^{-1}$
mChargedHadronMultiplicity	$2.60 \times 10^{-2} \pm 2.32 \times 10^{-2}$	$3.59 \times 10^{-1} \pm 7.28 \times 10^{-2}$	$7.48 \times 10^{-3} \pm 3.39 \times 10^{-2}$	$1.14 \pm 3.21 \times 10^{-1}$
mNeutralHadronMultiplicity	$6.98 \times 10^{-3} \pm 2.29 \times 10^{-2}$	$7.91 \times 10^{-2} \pm 9.70 \times 10^{-2}$	$7.03 \times 10^{-4} \pm 2.75 \times 10^{-3}$	$1.09 \times 10^{-1} \pm 9.42 \times 10^{-2}$
mPhotonMultiplicity	$4.37 \times 10^{-3} \pm 1.44 \times 10^{-2}$	$2.24 \times 10^{-1} \pm 6.04 \times 10^{-2}$	$-5.24 \times 10^{-3} \pm 2.14 \times 10^{-2}$	$4.71 \times 10^{-1} \pm 1.12 \times 10^{-1}$
mElectronMultiplicity	$8.09 \times 10^{-4} \pm 3.07 \times 10^{-3}$	$2.06 \times 10^{-2} \pm 1.90 \times 10^{-2}$	$-4.56 \times 10^{-5} \pm 1.33 \times 10^{-4}$	$1.76 \times 10^{-3} \pm 1.56 \times 10^{-3}$
mMuonMultiplicity	$-2.05 \times 10^{-2} \pm 4.75 \times 10^{-2}$	$1.10 \times 10^{-1} \pm 1.77 \times 10^{-1}$	$1.68 \times 10^{-5} \pm 1.77 \times 10^{-4}$	$1.10 \times 10^{-2} \pm 1.81 \times 10^{-2}$
mHFHadronMultiplicity	$-1.93 \times 10^{-2} \pm 2.89 \times 10^{-2}$	$1.56 \times 10^{-1} \pm 1.57 \times 10^{-2}$	$6.87 \times 10^{-4} \pm 4.38 \times 10^{-3}$	$2.16 \times 10^{-1} \pm 5.30 \times 10^{-2}$
mHFEMMultiplicity	$6.34 \times 10^{-4} \pm 3.67 \times 10^{-3}$	$5.84 \times 10^{-2} \pm 2.48 \times 10^{-2}$	$4.87 \times 10^{-4} \pm 7.10 \times 10^{-3}$	$9.81 \times 10^{-2} \pm 2.35 \times 10^{-2}$
mChargedEmEnergy	$1.52 \pm 2.06 \times 10^{-1}$	$2.70 \pm 2.67 \times 10^{-1}$	$-3.17 \times 10^{-3} \pm 2.07 \times 10^{-2}$	$2.12 \pm 3.23 \times 10^{-1}$
mChargedMuEnergy	$3.53 \times 10^{-1} \pm 1.13 \times 10^{-1}$	$9.15 \times 10^{-1} \pm 1.01 \times 10^{-1}$	$-9.16 \times 10^{-3} \pm 1.13 \times 10^{-2}$	$7.81 \times 10^{-1} \pm 1.85 \times 10^{-1}$
mNeutralEmEnergy	$1.64 \times 10^{-1} \pm 8.33 \times 10^{-2}$	$1.38 \pm 1.51 \times 10^{-1}$	$-8.00 \times 10^{-2} \pm 3.08 \times 10^{-1}$	$1.76 \times 10^1 \pm 1.44$
mChargedMultiplicity	$2.59 \times 10^{-2} \pm 2.32 \times 10^{-2}$	$3.58 \times 10^{-1} \pm 7.34 \times 10^{-2}$	$7.50 \times 10^{-3} \pm 3.39 \times 10^{-2}$	$1.14 \pm 3.21 \times 10^{-1}$
mNeutralMultiplicity	$4.23 \times 10^{-3} \pm 6.92 \times 10^{-3}$	$9.32 \times 10^{-2} \pm 4.73 \times 10^{-2}$	$-2.91 \times 10^{-3} \pm 2.04 \times 10^{-2}$	$4.32 \times 10^{-1} \pm 1.17 \times 10^{-1}$

A.3 Comparison to Zip

To make sure that relationship between the zipped Baler output and zipped input is not dependent on file size, we tested this for different input file sizes: 250, 500, 750, 1000, 1250, and 1500 MB. All originating from the same data source [33]. Figure 9 shows the output file size of the different scenarios tested as a function of the input file size.

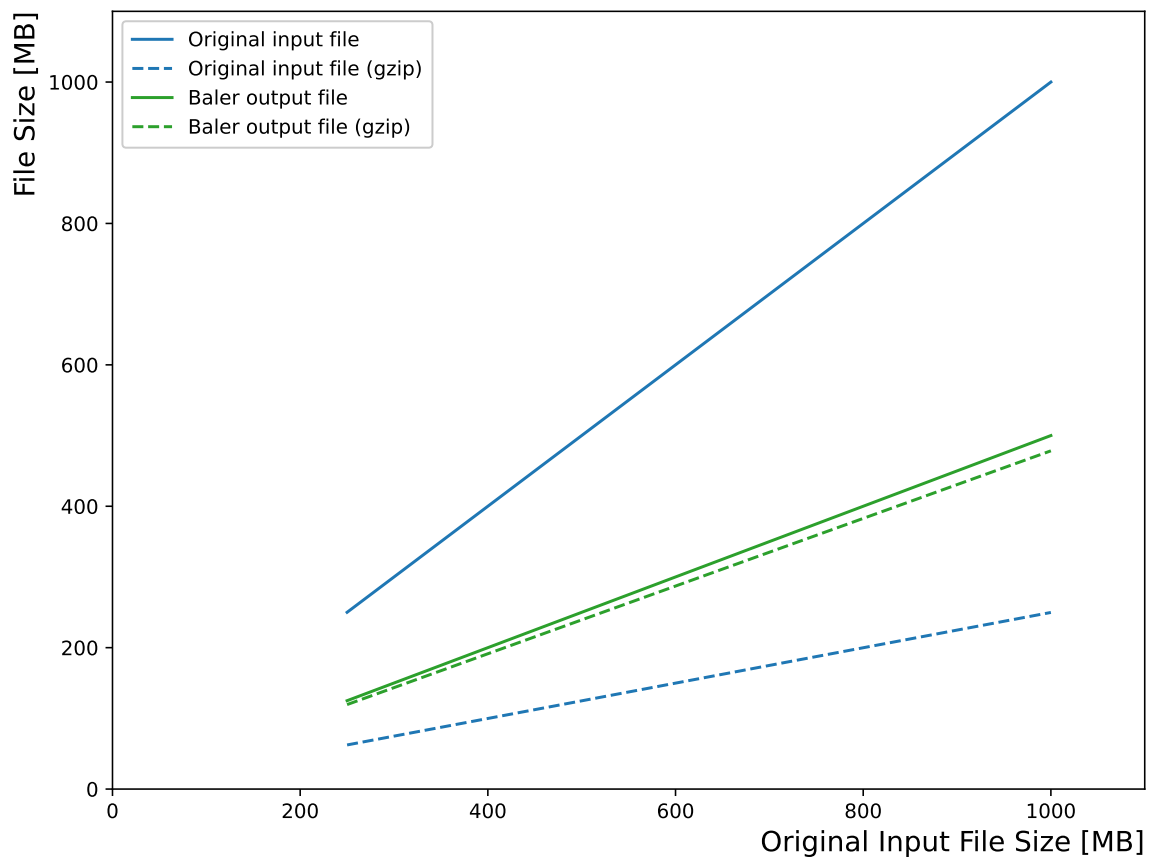


Figure 9: The output file size of the different scenarios tested as a function of the input file size. Note that in this scenario, Baler used a $R = 2.0$