Baler - Machine Learning Based Compression of Scientific Data

Fritjof Bengtsson Folkesson[1,*], *Caterina* Doglioni[2,**], *Per Alexander* Ekman[1,***], *Axel* Gallén[3,****], *Pratik* Jawahar[2,†], *Marta* Camps Santasmasas[2,‡], and *Nicola* Skidmore[2,§]

[1]Lund University
[2]University of Manchester
[3]Uppsala University

Abstract. A common and growing issue in scientific research and industry is that of storing and sharing ever-increasing datasets. In this paper we document the development and applications of Baler - a Machine Learning based tool for tailored compression of data across multiple disciplines.

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 8.5 EB 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

*e-mail: fritjof.folkeson@gmail.com
**e-mail: caterina.doglioni@manchester.ac.uk
***e-mail: alexander.ekman@hep.lu.se
****e-mail: axel.lars.gallen@cern.ch
†e-mail: pratik.jawahar@cern.ch
‡e-mail: marta.campssantasmasas@manchester.ac.uk
§e-mail: nicola.skidmore@cern.ch

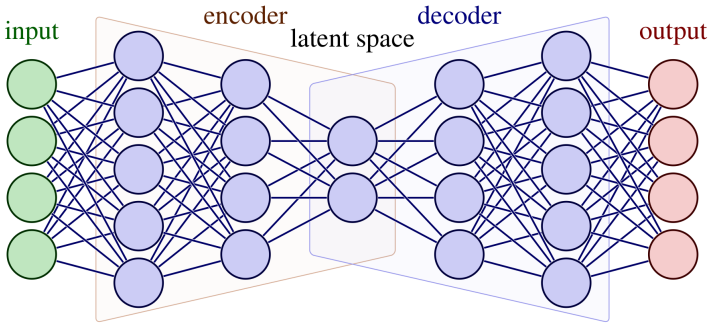


Figure 1: Illustration of an autoencoder consisting of an input and output layer. In between the input and output, there are hidden layers and a latent space. For compression the dimensionality of the latent space is less than the input and output layers. Modified from Ref. [12].

power is necessary, e.g. for rare signals buried in high-rate backgrounds. In these cases, one can consider 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] which is an example of a lossy compression algorithm tailored to audio waves. Thereby, a general solution to this cross-disciplinary problem is hard to obtain. To address this 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.

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 also shown promising results 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 requires training 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 and 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:

$$L_{total} = (1-\beta)L_{reco} + \beta L_1 \tag{1}$$

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_1 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 where $x \in X$ and $\hat{x} \in \hat{X}$. L_1 introduces sparsity to the model and reduces its storage size, thereby reducing the overheads as discussed in Sec. 2.2. L_1 is defined as,

$$\text{L}_1 = \sum_i |w_i| \tag{2}$$

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 consider the mean and RMS of the residual and response as our evaluation metrics, where,

$$\text{residual}_i = \hat{x}_i - x_i, \qquad \text{response}_i = \frac{\text{residual}_i}{x_i} \tag{3}$$

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

$$R = \frac{\text{size(input file)}}{\text{size(compressed file)}}. \tag{4}$$

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

$$R_{\text{actual}} = \frac{\text{size(input file)}}{\text{size(compressed file + auxiliary file(s))}} \tag{5}$$

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.

3 Baler input data

Baler supports `NumPy` [28] 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, 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* [29]. 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 [30]. 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 jet 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. [31]. 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

To simplify the use of 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`. Therefore each jet in the dataset is independent from the others and so correlations within events are lost[1]. 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.

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 by seeds. Baler's performance on a certain dataset is visualized and evaluated by looking at the variable distributions together with the distribution of responses and/or residuals. In Figure 2 we show, for one seed, the distribution of four selected variables before and after compression using $R = 1.7$; together with the response distribution for each variable. The mean and root mean squared (RMS) of the response distributions are presented in the figure. In [33, 34], Baler's performance on the remaining 20 variables are presented at both $R = 1.7$ and $R = 6$.

The difference between R and R_{actual} for HEP data is negligible. On disk, the total auxiliary file size reaches at the very most 550 KB. This means that for our case, where our input file size is 116.9 MB, we obtain: $R = 1.7 \rightarrow R_{\text{actual}} \approx 1.59$. As the auxiliary file sizes for HEP do not increase with the number of entries they become negligible.

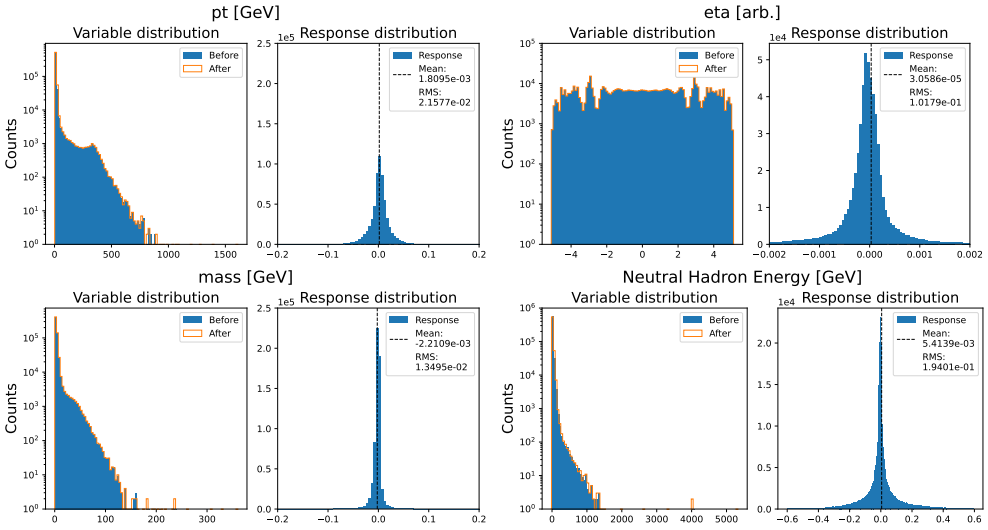
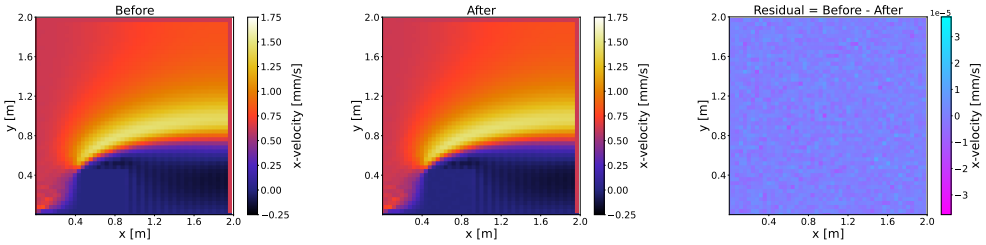


Figure 2: Distributions of four selected jet variables. Alongside each distribution is a histogram of the response for that variable after compression with $R = 1.7$.

[1]For early studies of the impact of considering individual jets versus jets from the same event, see Ref. [32]

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 of velocity for 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 [35]. Figure 3a and 3b show the 2D data before and after compression and decompression, with $R = 88$. Figure 3c shows the difference between the two which is on a scale four orders of magnitude smaller. These results show Baler's wider applications to multiple scientific disciplines.



(a) (b) (c)

Figure 3: 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 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 and CFD data as a proof-of-concept to demonstrate Baler's flexibility. The auxiliary file size produced and the achievable compression ratio are dataset dependant. We note that for the specific HEP dataset used, gzip outperforms Baler [33, 34]. However, for the CFD case we observe that Baler outperforms gzip in terms of compression ratio [33], with the added trade-off of increased auxiliary file size and with future implementation of online compression, the auxiliary file size will be of less significance.

Near future extensions to this work include assessing performance variations related to dataset sizes and support for error-bound compression. 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 method is to calculate a coefficient of variation for a given dataset as described in [36] 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 exploring different HEP datasets and input representations that may involve low-level detector data.

Another related limitation is compressing files larger than RAM, 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 investigate given its high potential 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.

Acknowledgements

We would like to thank the Alan Turing Institute and the University of Manchester for their feasibility grant allowing us to package and better distribute our software publicly. We would also like to thank the Helmholtz Association and the HELIOS Graduate School for funding our attendance at CHEP 2023 (Project number HIRS-0018). PJ is part of SMARTHEP which received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement n. 956086.

References

[1] ATLAS Collaboration, Tech. rep., CERN, Geneva (2022), http://cds.cern.ch/record/2802918

[2] CMS Offline Software, Computing, Tech. rep., CERN, Geneva (2022), http://cds.cern.ch/record/2815292

[3] LHCb Collaboration, Tech. rep., CERN, Geneva (2018), http://cds.cern.ch/record/2319756

[4] A.M.M. Scaife, Phil. Trans. R. Soc. A. **378** (2020)

[5] M. Khan, X. Wu, X. Xu, W. 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

[6] O. Shadura, B.P. Bockelman, P. Canal, D. Piparo, Z. Zhang, EPJ Web of Conferences **245**, 02017 (2020)

[7] C. Patauner, *Lossy and lossless data compression of data from high energy physics experiments* (2011), presented 2011, https://cds.cern.ch/record/1433839

[8] A. Rawal, *Exploiting Domain-specific Data Properties to Improve Compression for High Energy Physics Data* (2020), presented 2020, https://newtraell.cs.uchicago.edu/research/publications/techreports/TR-2020-03

[9] K. Sayood, *Introduction to data compression* (Morgan Kaufmann, 2017)

[10] K. Brandenburg, *MP3 and AAC explained*, in *17th International Conference: High-Quality Audio Coding* (Audio Engineering Society, 1999)

[11] M.A. Kramer, AIChE Journal **37**, 233 (1991), https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690370209

[12] Izaak Neutelings, *Neural networks* (2021), [Online; accessed 02-May-2023; Last edited 11 September 2022], https://tikz.net/neural_networks/

[13] T. Liu, J. Wang, Q. Liu, S. Alibhai, T. Lu, X. He, IEEE Transactions on Big Data **9**, 22 (2023)

[14] J. Liu, S. Di, K. Zhao, S. Jin, D. Tao, X. Liang, Z. Chen, F. Cappello, *Exploring Autoencoder-based Error-bounded Compression for Scientific Data*, in *2021 IEEE International Conference on Cluster Computing (CLUSTER)* (2021), pp. 294–306

[15] N. Wang, T. Liu, J. Wang, Q. Liu, S. Alibhai, X. He, Journal of Network and Computer Applications **205**, 103452 (2022)

[16] R. La Grassa, C. Re, G. Cremonese, I. Gallo, Remote Sensing **14**, 2472 (2022)

[17] Y. Huang, Y. Ren, S. Yoo, J. 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

[18] J. Lee, Q. Gong, J. Choi, T. Banerjee, S. Klasky, S. Ranka, A. Rangarajan, Applied Sciences **12**, 6718 (2022)

[19] S. Sriram, A.K. Dwivedi, P. Chitra, V.V. Sankar, S. Abirami, S.R. Durai, D. Pandey, M.K. Khare, Arabian Journal for Science and Engineering **47**, 10395 (2022)

[20] E. Wulff, *Deep autoencoders for compression in high energy physics* (2020), student Paper, http://lup.lub.lu.se/student-papers/record/9004751

[21] E. Wallin, *Tests of autoencoder compression of trigger jets in the atlas experiment* (2020), student Paper, http://lup.lub.lu.se/student-papers/record/9012882

[22] J.H. Collins, Y. Huang, S. Knapen, B. Nachman, D. Whiteson (2022), 2210.11489

[23] C. Weisser, M. Williams, *"autoencoders for lhcb", presented at the reconstruction, trigger, and machine learning for the hl-lhc" mit workshop* (2018), student Presentation, 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] D. George, *Deep Autoencoders for ATLAS Data Compression - George Dialektakis - Google Summer of Code 2021 Project* (2021), https://doi.org/10.5281/zenodo.5482611

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., *Pytorch: An imperative style, high-performance deep learning library* (2019), https://arxiv.org/abs/1912.01703

[27] D.P. Kingma, J. Ba, *Adam: A method for stochastic optimization* (2014), https://arxiv.org/abs/1412.6980

[28] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith et al., Nature **585**, 357 (2020)

[29] G.P. Salam, CERN Yellow Rep. School Proc. **5**, 1 (2020)

[30] CMS collaboration, Tech. rep. (2017), http://opendata.cern.ch/record/6010

[31] CMS Collaboration, *"cms physics objects 2015"* (2015), online, http://opendata.cern.ch/docs/cms-physics-objects-2015

[32] S. Aastrand, *Autoencoder compression in high energy physics* (2022), student Paper, http://lup.lub.lu.se/student-papers/record/9004751

[33] F. Bengtsson, C. Doglioni, P.A. Ekman, A. Gallén, P. Jawahar, A. Orucevic-Alagic, M.C. Santasmasas, N. Skidmore, O. Woolland, *Baler – machine learning based compression of scientific data* (2023), 2305.02283

[34] Gallén, Axel, *An Open-Source Autoencoder Compression Tool for High Energy Physics* (2023), Student Paper

[35] Y. LeCun, Y. Bengio et al., The handbook of brain theory and neural networks **3361**, 1995 (1995)

[36] T. Liu, J. Wang, Q. Liu, S. Alibhai, T. Lu, X. He, IEEE Transactions on Big Data **9**, 22 (2023)