Baler: Deep Autoencoders for Scientific Data Compression

Final Report - Google Summer of Code 2023

Aman Singh Thakur[1,*], *Per Alexander* Ekman[2,**], and *Caterina* Doglioni[3,***]

[1]University of Massachusetts Amherst, USA
[2]Lund University, Sweden
[3]University of Manchester, England

Abstract. Today the experiments at CERN output roughly one petabyte of data per day [1]. After the planned upgrades to its main experiments, in 2032 these will produce 2-5 times more data than the available storage resources [2]. To tackle this challenge, an exploration of lossy compression methods is necessary. Aiming to address this issue we present Baler, an Autoencoder-based lossy compression tool currently under development at the universities of Lund, Manchester, and Warwick. This report investigates improvements of Baler, with a focus on 2D Convolutional SZ Autoencoders, and compares its performance to its existing Dense network and leading off-the-shelf SZ3[3] lossy compression tool. Finally, we conclude by providing substantial evidence indicating that Dense networks outperform Convolutional models in both offline and online compression scenarios, while concurrently preserving an similar compression ratio to that of SZ3.

*e-mail: amansinghtha@umass.edu
**e-mail: alexander.ekman@hep.lu.se
***e-mail: caterina.doglioni@manchester.ac.uk

Contents

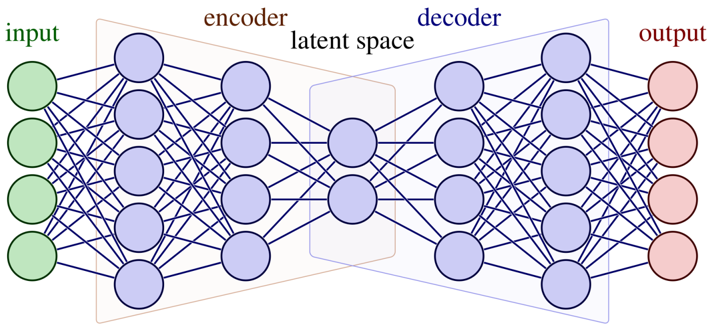


Figure 1: This is a simplified diagram of an autoencoder, comprising an input layer, hidden layers, and an output layer. The lower-dimensional latent space serves as a compression mechanism reducing AE dimensionality compared to the input and output layers. Modified from Website [7].

1 Introduction

1.1 Loss-less Data Compression

When data is compressed and decompressed using lossless techniques, the original data is fully reconstructed without any degradation in quality. This type of compression is called Lossless Data compression [4]. Lossless compression finds widespread application in tasks such as file archiving and data transmission. Prominent examples of lossless compression techniques include Huffman Coding, Run-Length Encoding (RLE), PNG, and MPEG-4 [5].

Classic lossless compression methods rely on the repetition of data for its bulk compression and they typically achieve a compression ratio, $size_{before}/size_{after}$, of 3x depending on the dataset. It is constrained by the total number of unique elements within the dataset, which becomes problematic for scientific data with very accurate float values or no zeros as this leaves few repeating values.

1.2 Lossy Data Compression and AutoEncoders

The large experiments at CERN, ATLAS, ALICE, LHCb, and CMS, are currently generating an astonishing 1 Petabyte of data every second[6]. Although, not all of it is archived, it presents significant challenges in terms of available bandwidth and storage capacity. Consequently, there is a compelling need for selective data retention, even in the face of substantial volume reduction efforts. The impending tenfold increase in particle collisions, anticipated with the High Luminosity LHC upgrade scheduled for 2029, only exacerbates the ongoing data storage crisis [2]. Traditional lossless compression techniques, including those employed with highly compressed data formats like ROOT and ZIP, prove inadequate to address this crisis.

In this context, the adoption of lossy data compression emerges as a pragmatic solution, offering the potential to achieve compression ratios ranging from 10 to 1000 times the original data size. While lossy compression entails some sacrifices in data precision, it facilitates the storage of a significantly larger data volume. In this case, the gain in statistical precision can outweigh the loss in precision form the compression technique.

Some well-known compression methods such as quantization, transform coding Principal Component Analysis (PCA), and predictive coding, primarily embody the principles of reducing dimensionality at a broad level. In this report, our main focus is on a type of Neural Network architecture known as Autoencoders which also performs dimensionality reduction.

Autoencoders (AE) were first introduced by Kramer [8] in 1991 and fall into the category of unsupervised deep neural networks. They are distinguished by their structure, shown in Figure 1. They consist of an encoder, a central latent space, a decoder, and a target space with the same dimensionality as the input space. The encoder serves as a neural network that maps each input to an abstract latent point. Typically, this latent space has a lower dimension compared to the input space. Subsequently, the decoder reconstructs the latent space back to the original input dimensions, resulting in an output that closely resembles the input.

The primary function of autoencoders is to learn the essential features of the input data for reconstruction, with their bottleneck structure preventing them from simply copying the input as-is. The dimensionality of the latent space is a crucial parameter, as it dictates the level of compression, with the latent points serving as the compressed data and the decoder acting as the decompression algorithm. Autoencoder-based data compression has exhibited promising results in various scientific domains, including meteorology, cosmology, computational fluid dynamics, crystallography, and others [9].

1.3 Baler - Open Source Tool Development

Baler's third-time participation in the Google Summer of Code (GSoC) underscores its prominent position in the software development landscape. In 2021, Dialektakis[10] utilized the GSoC platform to investigate the comparative performance of standard autoencoders and variational autoencoders. More recently, Manas[11] employed Baler to estimate the energy cost of scientific software.

Baler, originally a collaboration between Lund University and the University of Manchester, offers three primary working modes - Train, Compress, and Decompress, featuring a robust neural network that efficiently handles tabular data, images, and videos without necessitating intricate user adjustments.

Notably, Baler serves as an open-source tool catering to both industry professionals and academia. Collaborative endeavors have facilitated the acquisition of diverse datasets, enabling comprehensive testing and validation of Baler's capabilities, resulting in impressive performance outcomes. This cooperative effort highlights the tool's commitment to advancing collective knowledge and practices in the compression domain. The source code for Baler is accessible at url[1].

2 Baler Datasets

All Baler Datasets can be found in their respective workspace/data folder on the original repository [2]. All datasets are visualized in Figure 2.

2.1 Scientific Datasets

CERN's Computational Fluid Dynamics (CFD) team supports CERN engineers, physicists, and scientists by providing CFD tools and techniques for flow and thermal analysis, addressing fluid flow and heat transfer challenges [12]. The CFD dataset used here is a simpler proof-of-concept dataset of a mineral oil flowing over a wall-mounted cube. CFD datasets like this one is amenable to lossy compression due to minimal motion in both the background

[1]https://github.com/baler-collaboration/baler

[2]https://github.com/baler-collaboration/baler/tree/1f8b/workspaces

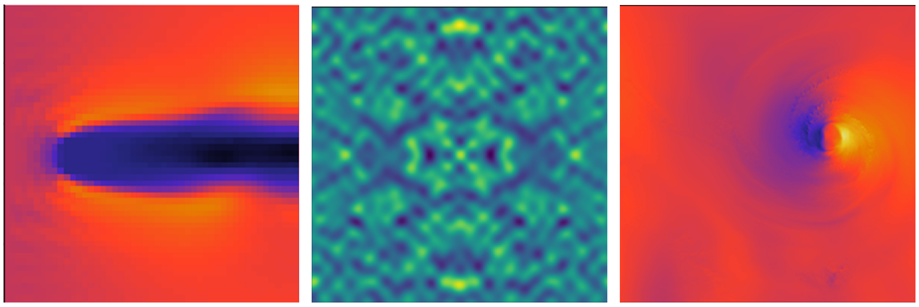

(a) CFD Dataset (b) Exafel Dataset (c) Hurricane Isabel Dataset

Figure 2: Scientific Datasets tested successfully with Baler

and subject, typically following a predictable pattern. We assessed Baler's performance on two distinct data types: the CFD Dataset, consisting of a 2D image time series capturing subjects under fluid flow. The CFD Dataset is 1.14MB, containing 60 pieces of 50x50 pixel frames.

SLAC's Linac Coherent Light Source (LCLS) users employ rigorous computational analysis to process data obtained from ultrafast x-ray pulses, allowing them to capture atom and molecule behavior as "stop-action movies." LCLS's high repetition rate and brightness facilitate precise determination of individual molecule structures, revealing variations in shape and flexibility. This understanding underpins advancements in biological, material, and energy sciences. The Exafel dataset [13], notable for its rapid and intricate pattern changes, presents challenges for lossy compression, but Baler successfully compresses the 12.8MB dataset with 151 frames of 151x151 pixels each.

2.2 Hurricane Isabel Dataset

This report utilizes a public dataset obtained from the National Center for Atmospheric Research in the United States, simulating the 2003 Hurricane Isabel [14]. The dataset comprises various time-dependent scalar and vector variables with significant fluctuations. However, its substantial size presents challenges for interactive analysis. To enhance the dataset's suitability for our research, we have excluded the final 10 frames, which depict landmass. This exclusion is motivated by our current limitations for accurately reconstructing zero value data points. In this report, the Hurricane Isabel dataset consists of 90 frames, each measuring 500x500 pixels, where each (X, Y) coordinate corresponds to a specific longitude and latitude. These frames provide X-velocity data at the 20th hour of the hurricane [15].

We would like to thank Bill Kuo, Wei Wang, Cindy Bruyere, Tim Scheitlin, and Don Middleton of the U.S. National Center for Atmospheric Research (NCAR), and the U.S. National Science Foundation (NSF) for providing the Weather Research and Forecasting (WRF) Model simulation data of Hurricane Isabel.

3 Methodology

3.1 Previous Work on Baler

In May 2023, Baler was introduced as a proof of concept[16], designed for compatibility with numpy datasets. Baler employed a 3-layer Dense AutoEncoder Network (200X100X50X (z-dim) X50X100X200) with MSE loss and L1 regularization during training. Baler was successfully able to achieve 20x compression on its CFD dataset, with completion in under 1-2 minutes, attributed to the dataset's size and network efficiency.

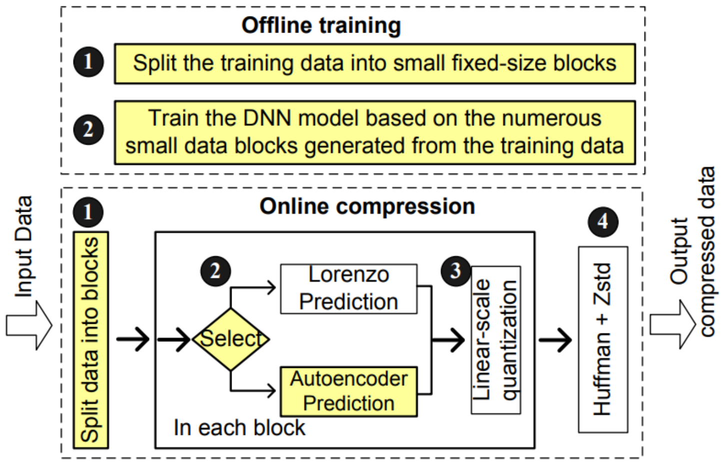

Offline training
1
Split the training data into small fixed-size blocks
2
Train the DNN model based on the numerous small data blocks generated from the training data
Input Data
Online compression
1
Split data into blocks
2
Select
Lorenzo Prediction
Autoencoder Prediction
In each block
3
Linear-scale quantization
4
Huffman + Zstd
Output compressed data

Figure 3: Baler enhancements based on the design of AE-SZ discussed in Reference Paper[17]

3.2 Literature Review

Recent advancements in deep autoencoders, particularly Variational Autoencoders (VAEs) as introduced in "An Introduction to Variational Autoencoders" [18], have become potent tools for compressing scientific data, offering efficient data compression while preserving crucial features. Autoencoders, as emphasized in "Autoencoders, Unsupervised Learning, and Deep Architectures" [19], play a pivotal role in feature learning and representation for scientific data compression. Deep architectures, including stacked autoencoders, enhance compression performance by modeling complex relationships in scientific datasets. Additionally, the transition "From Variational to Deterministic Autoencoders" [20] highlights the adaptability of autoencoders in scientific data compression, with deterministic autoencoders offering precise and non-probabilistic data reconstruction. In summary, deep autoencoders are versatile and practical for scientific data compression, spanning diverse domains and accommodating various compression requirements while ensuring data preservation and anomaly handling.

In this report, we predominantly center our attention on paper "Exploring Autoencoder-based Error-bounded Compression for Scientific Data" [17], which currently represents the state-of-the-art in the realm of Deep Autoencoders applied to the compression of scientific data. It is important to note that it is not yet published in a peer-reviewed journal.

The aforementioned work elucidates various techniques, as illustrated in Figure 3, encompassing linear-scale quantization, dataset blocking, error-bounded data compression, as well as offline and online compression strategies. Our research leverages the Hurricane Isabel dataset to implement some of these techniques and perform a performance evaluation of our results.

3.3 Enhancements of Baler

With inspiration from the techniques used and developed in reference [17], we implemented the following features into Baler and its source code.

3.3.1 Blocked Training, Compression and Decompression

As described in the reference paper [17], optimal model performance is attained when all frames are processed within a single batch, which facilitates the discernment of non-linear patterns within the dataset. Nonetheless, datasets of considerable size, such as SLAC's Exafel (151x151x151) and Hurricane Isabel (90x500x500), pose a significant challenge for efficient handling by the model in the absence of a highly capable GPU. To overcome this challenge, the report proposes a solution involving the partitioning of datasets into user-configurable blocks, enabling batch processing. This approach, while introducing a discernible algorithmic deceleration due to the additional computational workload associated with interconnecting these blocks, represents a pivotal strategy to counteract overfitting and ultimately enhance the quality of results in online testing scenarios. Within Baler, The old and new shape relationship of the blocked dataset is depicted in the below equation 1 -

$$\text{Frames} \cdot X \cdot Y = (\text{Frames} \cdot N \cdot M) \cdot \left(\frac{X}{N}\right) \cdot \left(\frac{Y}{M}\right) \tag{1}$$

3.3.2 Generalized Divisive Normalization (GDN)

Covariate shift presents a challenge in neural networks, manifesting as a discrepancy in the input data distribution between the training and testing phases. This discrepancy results in a decline in network performance. Batch Normalization emerges as an effective strategy to mitigate this issue. It achieves this by normalizing input data within mini-batches, thereby stabilizing the training dynamics, enhancing convergence, reducing over-fitting, and facilitating the utilization of higher learning rates. Consequently, it effectively alleviates the impacts of covariate shift and enhances the model's generalization capabilities.

In this report, we opt for a novel approach, employing the Generalized Divisive Normalization Activation Function [21] instead of Batch Normalization with Activation Function. This novel activation function is specially tailored to parametrically transform data, rendering it particularly well-suited for the task of Gaussianizing image data. The GDN Activation Functions essentially generate a probability density model based on the outputs of Convolutional Layers. The differentiability of these models contributes to expedited model convergence, as substantiated by the observed loss plot trends. In our implementation in Baler, we utilize Jorge Pessoa's[22] PyTorch implementation of the GDN Function, inspired by the original paper's [21] implementation in TensorFlow.

3.3.3 Slized WAE (SZ) Regularizer

Inspired by reference paper [17], we apply a Sliced-Wasserstein Autoencoders (SZ AE), a technique introduced by Kolouri et al [23]. The primary objective of SZ AE is to minimize the sliced-Wasserstein distance between the reconstructed output and the original data. The Wasserstein distance quantifies the amount of effort required to transform one probability distribution into another. In our report, we employ Generalized Density Normalization (GDN) functions, which facilitate the transformation of images into Gaussian density models. Subsequently, we utilize the Sliced Wasserstein distance to guide our model's convergence towards the desired target Gaussian distribution.

In Baler, we make use of an open-source implementation of SWAE [24] that adheres to the methodology outlined in the original paper. Within this framework, we incorporate the Sliced Wasserstein distance as an L1 regularizer in conjunction with the Mean Squared Error (MSE) loss. This strategic combination has been chosen to enhance the convergence of our model.

3.3.4 Error Bounded Compression

Incorporating error-bounded compression into the Baler framework, as detailed in the reference paper[17], we introduce an innovative approach that functions independently of the training paradigm. This compression technique enables users to specify their desired error tolerance as a percentage. Throughout the compression process, the model initially decompresses the data and subsequently assesses the disparities between the decompressed output and the original output. Whenever discrepancies surpass the predetermined error threshold, the system records these variations, along with their positional information (batch ID, X, Y) and their values represented as 8-bit floating-point numbers. Given that the indices are exclusively integers, we apply a conventional lossless compression method, such as "numpy savez", to minimize their storage footprint. The precision of the delta values is not critical, as we employ an 8-bit floating-point representation prior to applying lossless compression.

3.3.5 3D Convolutional Neural Networks

In this work, we introduce a 3D Convolutional Neural Network with a multi-layer architecture (1x4x8x16x32x64) in Baler. Our primary focus is to ensure the compatibility of the existing codebase with both dense 2D CNN and memory-intensive 3D CNN. Notably, 3D CNNs are renowned for their high memory usage. We have successfully conducted a proof-of-concept study using a Computational Fluid Dynamics (CFD) dataset, with opportunities for enhancement through hyper-parameter optimization. When applying the compression and decompression techniques to the Exafel and Hurricane datasets, which are 3 to 10 times larger than the CFD dataset, we encountered memory-related challenges. However, we were able to successfully train the datasets on NVIDA A100 with 16GB RAM, which exposed a current drawback in our system pertaining to the lack of optimization in the compression and decompression of Baler processes on GPUs.

3.3.6 Evaluation of Offline and Online Performance

Traditional compression tools like gzip and SZ work by installing a piece of software which is able to compress and decompress various different datasets using the exact same software. Machine learning based compression methods on the other hand fundamentally is different in the respect that they require more auxilliary files to perform compression and decompression. Auxilliary files are files required other than the compressed data itself, in our case these files are the autoencoder model itself, including architecture and weights. These files can also include normalization factors, label headers, list of deltas. The presence of auxilliary files opens up for two different modes of compression: Offline and online compression.

In offline compression, the data set is already taken in whole, and a model is trained to just compress this one dataset. This means that the training can be performed without validation and overtraining is encouraged as the model will only ever be used to reconstruct this one datset. The problem with offline compression is that the model is specific to this dataset and should be included in calculating the compressed size of the data. Large model sizes is our main blocker right now, and is the reason why we pursue online compression.

Contrary to offline compression, in online compression multiple different datasets are used to derive a model which will be used to compress future similar datsets. This means that the model is not specific to each dataset and therefore is included in the size of the tool itself, not the size of the compressed file. This model of compression makes Baler more competetive and also opens up the field of real-time compression and bandwidth compression implementations.

This report presents the studies we conducted evaluating Baler's performance in both offline and online compression for various configurations and datasets.

4 Results

To gauge the efficacy of contemporary and historical methodologies and draw comparisons with the conclusions presented in the referenced paper [17], our study will center on the publicly accessible Hurricane Isabel dataset. As we scrutinize the outcomes, our initial phase will entail an assessment of the top-performing model's performance through the execution of offline training, employing the unaltered image as a benchmark.

In addition to intrinsic Baler analysis, we'll conduct a comparative analysis between Baler's models and the contemporary state-of-the-art SZ3[3][25][26] lossy compressor. SZ3 is an innovative modular compression framework that has been specially crafted to address the challenges associated with data volume reduction in scientific simulations. This framework is designed to accommodate the diverse characteristics inherent in various datasets and to cater to the compression quality and performance preferences of users. One of the key features of SZ3 is its abstraction for prediction-based compression, which facilitates the customization of compression modules based on the specific attributes of the data and user requirements. SZ3's effectiveness has been empirically established through the development of tailored compression pipelines for specific datasets like GAMESS and APS data. For the context of this research, we will utilize the default configuration of SZ3 without any custom modifications.

4.1 Offline Reconstructions and Observations

Offline model training involved training on the Hurricane dataset's 20th hour, with the hurricane positioned in the center of the data. We tested different model architectures, all underwent 2000 epochs of training on an NVIDIA A100 GPU using 50x50 blocks. Compression ratio adjustments were made to ensure comparability with other models.

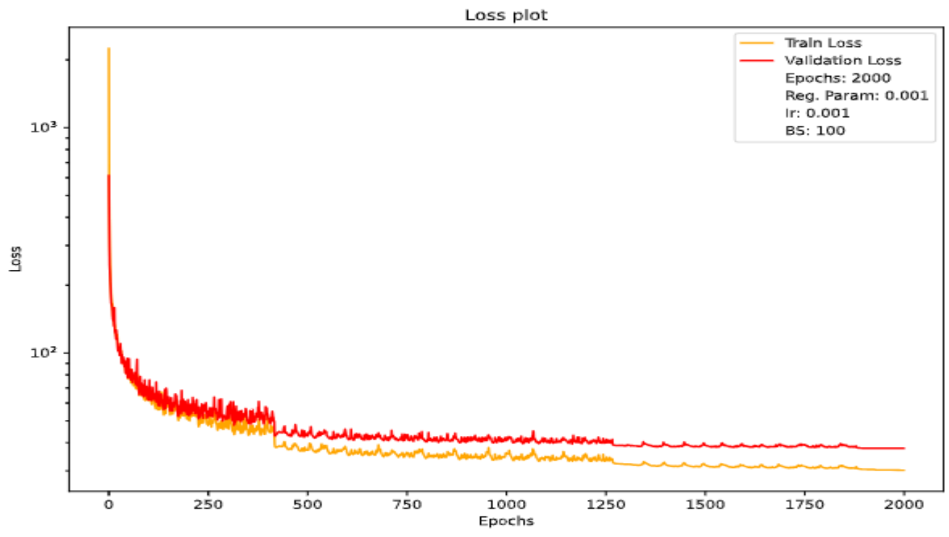
Reconstruction after decompression is shown in Figure 5 to reveal superior performance of the Dense Network over the Convolutional Network, yielding accurate hurricane data reconstructions. Additionally, Table 1 shows that SZ3 compression achieves a form factor similar to the Dense Network (excluding the model), demonstrating competitive compression capabilities.

The training duration of a Dense Network concludes in a mere 20 minutes, in stark contrast to the protracted 7-hour training period required by a Convolutional Network. This stark contrast underscores the substantial efficiency advantage of the Dense Network architecture. Moreover, as seen in Figure 4 it is evident that the Dense Network manifests closely aligned training and validation loss curves, denoting its capacity for robust training. Conversely, the Convolutional Network demonstrates discrepancy between training and validation loss, signifying the presence of potential overfitting issues or challenges pertaining to generalization.

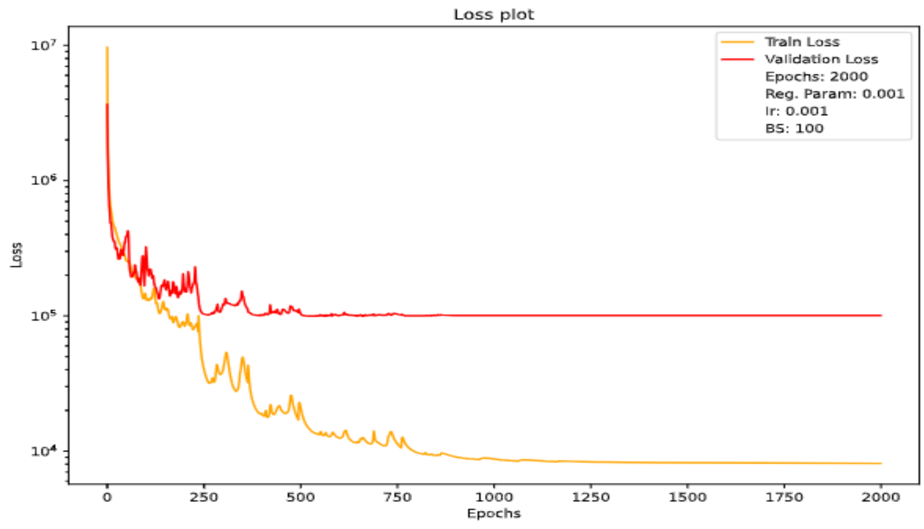
In summary, this report highlights the competitive reconstruction quality and training efficiency of the Dense Network, while SZ3 compression offer superior results (Table 2). The Convolutional Network will require further optimization to address overfitting concerns.

Model Name	Block Size	Compressed File Size	Model Size	Decompressed File Size	Execu-tion/Training Time	Compres-sion Ratio
SZ3	-	0.9 MB	-	85.8 MB	~ 1 min	~ 99
Dense	10x50x50	0.9 MB	4.03MB	85.8 MB	~ 20 mins	~ 100
2D CNN w GDN+SZ	10x50x50	44.8MB	1.09GB	85.8 MB	~ 7 hour	~ 2

Table 1: Observation Metrics for Offline Training of Hurricane Dataset



(a) Dense Network



(b) Convolutional Network

Figure 4: Train and Validation Loss Plots for Dense (a) and 2D convolutional (b) networks for 2000 epochs and same hyper-parameters

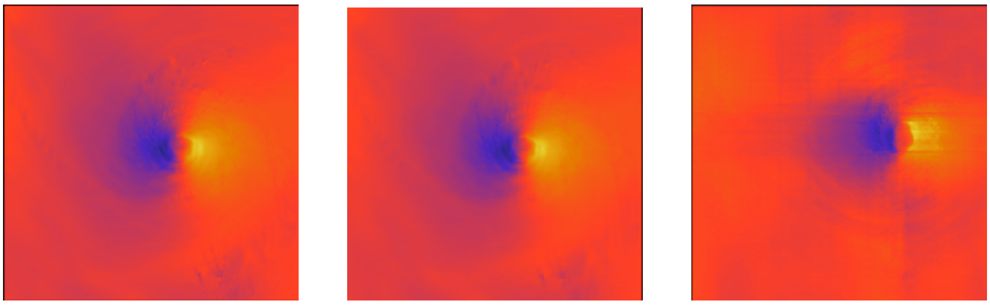

(a) Original (b) Dense (c) 2D CNN w GDN+SZ

Figure 5: Offline Reconstruction for Hour 20th of the Hurricane

4.2 Online Reconstruction and PSNR

In order to evaluate online compression, we initially trained each model using modified data that involved concatenating data from both the first and the final hour (0th and 48th) of the hurricane dataset whilst randomly transposing, flipping, and rotating the. This was done to ensure that the network was trained on multiple different locations and shapes of the hurricane. Online compression was implemented by training 2000 epochs on the concatenated and transformed 0th and 48th hour dataset, and then using this pre-trained model to compress, decompress, and evaluate data form the 20th hour. At the 20th hour the hurricane is at the center of the frame.

Our analysis commences with an evaluation of Peak Signal-to-Noise Ratio (PSNR)[27]. PSNR quantifies the ratio between the maximum potential power of a signal and the power of disruptive noise that impacts the accuracy of its representation, defined in (2). Where D represents the original dataset, D' the reconstructed data after compression and decompression, and MSE the mean squared error of the two.

$$PSNR = 20 \cdot \log(D) - 10 \cdot MSE(D, D') \tag{2}$$

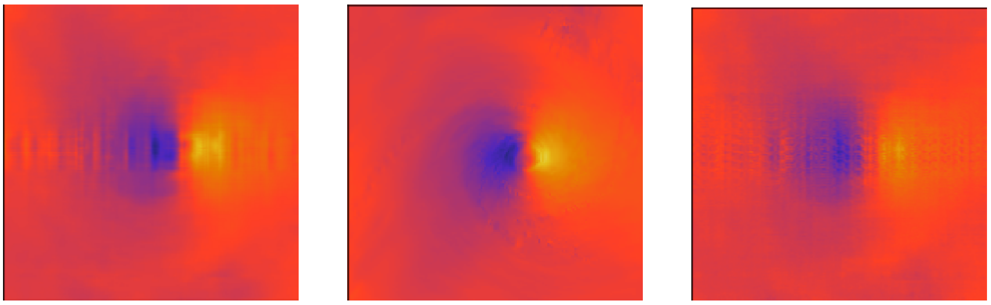
The results in Table 2 clearly indicate that SZ3 outperforms both Baler's Dense network and the Convolutional Network in this context. Based on the reconstructed images, it is evident that SZ accurately reproduces the original image, whereas Dense and Convolutional networks encounter challenges in achieving a faithful reconstruction.

In Table 2, it is evident that the Dense Network achieves results that are remarkably similar to those of the SZ, even though the compression ratio of the convolutional network is considerably lower than that of the Dense network. Analysis of the Online Image reconstructions reveals that the Dense network correctly captures the overall image but lacks smoothness, whereas the Conv Net has a vague understanding of the image but struggles significantly in the reconstruction process. Additionally, figure 6 illustrates the distribution of relative differences ($D - D'/D$), demonstrating that SZ performs notably better with its reconstruction, while both Dense and Convolutional networks exhibit significant variability.

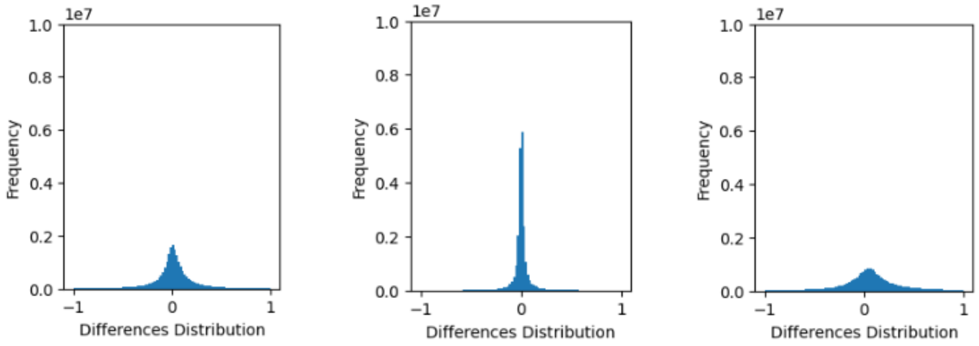
One approach to improving distribution accuracy involves the utilization of Error Bounded techniques, even though this may result in a reduced compression ratio. The substantial gap between the Training and Validation performance of Convolutional networks, as illustrated in Figure 5, indicates a significant overfitting issue. By fine-tuning hyperparameters, we can increase the importance of the SZ regularizer to mitigate model overfitting.

Model Name	Offline PSNR	Online PSNR
SZ3	132.78	132.78
Dense Network	108.8	89.32
2D CNN w GDN+SZ	45.62	45.23

Table 2: PNSR Ratio



(a) Dense (b) SZ (c) Conv2D



(d) Dense (e) SZ (f) Conv2D

Figure 6: (a-c) Online Reconstructions for 20th Hour of Hurricane. (d-f) Differences distribution from original plotted as histrograms for each model

5 Conclusion

5.1 Conclusion

Given that Baler's Dense network is relatively new, unrefined, and straightforward, it achieves reconstructions that are remarkably similar to SZ. Where SZ is a well regarded off-the-shelf lossy compression tool. By employing methods such as error-bounded models and quantization, Dense networks can potentially in the future produce competitive reconstructions.

In contrast, 2D Convolution Networks encounter challenges with this dataset. At lower compression ratios, they excel in image reconstruction, but as the compression ratio increases, they tend to lose substantial image information, making reconstruction difficult. It appears that achieving block-by-block reconstruction is more challenging than pixel-by-pixel reconstruction. Nevertheless, there exist various hyperparameter tuning techniques and regularization methods that may help approach the reconstruction performance of the Dense model and SZ.

5.2 Future Scope

In our upcoming research efforts, we plan to build upon the current study by integrating ideas from the 'Error-Controlled Quantization' paper[28]. Furthermore, we intend to harness advanced distributed computing technologies such as SPARK to enhance the efficiency of compression and decompression processes. This will enable concurrent processing for a multitude of files within the Baler framework. It's important to acknowledge that Baler currently faces several challenges, including discrepancies between actual compression and user-specified compression ratios, suboptimal code optimization for GPUs, and the utilization of error-bounded methods in dense networks. We are dedicated to addressing and resolving these issues to enhance the overall performance of Baler.

6 My Contributions

A detailed description of the improvements I implemented in the Baler compression tool can be found in Section 3.3. All my Baler GitHub Issues can be searched from link[3].

- Major Issues and Pull Requests
  - **Issue #270 - Running 2D CFD Data with 3D Conv Net**
    In this pull request, I enhance Baler with a 3D CNN by introducing memory-adaptive flags for flexible execution on either local or high-performance computing (HPC)
    https://github.com/baler-collaboration/baler/pull/284
  - **Issue #285 - Error Bounded AutoEncoder Support**
    Using this PR, I introduce a framework to preserve deltas exceeding error-bound requirements during compression and subsequently reintegrate them into the decompressed output to improve reconstruction fidelity with the original image.
    https://github.com/baler-collaboration/baler/pull/288
  - **Issue #299 - Hurricane Isabel Dataset**
    I incorporate the Hurricane Isabel dataset[14] with its visualization and analysis sub-routines in this PR.
    https://github.com/baler-collaboration/baler/pull/300
  - **Issue #305 - Adding Blocking of Datasets | GDN Activation Function**
    In this significant pull request, I introduce the capability to block datasets according to user-configurable settings and incorporate GDN Activation Function[22]
    https://github.com/baler-collaboration/baler/pull/306
  - **Issue #309 - Slized WAE (SWAE) Loss Function**
    In this pull request, I integrate the Sliced WAE[24] after evaluating it with the Baler Dense and Convolutional Network. I also enhance code modularity, enabling users to select MSE or SWAE loss functions from the configuration.
    https://github.com/baler-collaboration/baler/pull/308
- Minor Bug Fixes
  - Issue #242 - torch can't load models trained on GPUs on CPU only machines
  - Issue #264 - Decoder CUDA Device Issue
  - Issue #266 - Decoder Fails for large files
  - Issue #272 - Error when running tutorial decompression step
  - Issue #261 - compression ratio bug
  - Issue #295 - Global flag for error-bound compression
  - Issue #307 - Data dimensionality not preserved

[3]https://github.com/baler-collaboration/baler/issues/assigned/singh96aman

References

[1] M. Gaillard, *Cern data centre passes the 200-petabyte milestone*, https://home.cern/news/news/computing/cern-data-centre-passes-200-petabyte-milestone (2017), [Accessed 5-11-2023]

[2] P. Calafiura, J. Catmore, D. Costanzo, A. Di Girolamo, Tech. rep., CERN, Geneva (2020), http://cds.cern.ch/record/2729668

[3] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A.M. Gok, J. Tian, J. Deng, J.C. Calhoun, D. Tao et al., IEEE Transactions on Big Data **9**, 485 (2023)

[4] K. Sayood, *Introduction to Data Compression, Third Edition*, The Morgan Kaufmann Series in Multimedia Information and Systems (Morgan Kaufmann, 500 Sansome Street, Suite 400, San Francisco, CA 94111, 2006)

[5] *Lossless Compression - an overview | ScienceDirect Topics — sciencedirect.com*, https://www.sciencedirect.com/topics/computer-science/lossless-compression#:~:text=Typically%2C%20depending%20on%20the%20image,no%20loss%20in%20visual%20fidelity, [Accessed 31-10-2023]

[6] CERN, *dataexplosion*, https://information-technology.web.cern.ch/sites/default/files/CERNDataCentre_KeyInformation_Nov2021V1.pdf, [Accessed 31-10-2023]

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

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

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

[10] *Deep Autoencoders for ATLAS Data Compression - George Dialektakis - Google Summer of Code 2021 Project — zenodo.org*, https://zenodo.org/records/5482611#.Y3Yysy2l3Jz, [Accessed 31-10-2023]

[11] *Google Summer of Code — summerofcode.withgoogle.com*, https://summerofcode.withgoogle.com/programs/2023/projects/Nks9akq7, [Accessed 31-10-2023]

[12] *Index — cfd.web.cern.ch*, https://cfd.web.cern.ch/, [Accessed 31-10-2023]

[13] *ExaFEL - Exascale Computing Project — exascaleproject.org*, https://www.exascaleproject.org/research-project/exafel/, [Accessed 31-10-2023]

[14] hurricaneisabel, *hurricaneisabel*, https://www.earthsystemgrid.org/dataset/isabeldata.html, [Accessed 31-10-2023]

[15] D.T.J. Jankun-Kelly, *IEEE Visualization 2004 Contest: Data Set — vis.computer.org*, http://vis.computer.org/vis2004contest/data.html, [Accessed 31-10-2023]

[16] *arxiv.org*, https://arxiv.org/pdf/2305.02283.pdf, [Accessed 31-10-2023]

[17] *Exploring Autoencoder-based Error-bounded Compression for Scientific Data — arxiv.org*, https://arxiv.org/abs/2105.11730, [Accessed 31-10-2023]

[18] now publishers, *now publishers - An Introduction to Variational Autoencoders — nowpublishers.com*, https://www.nowpublishers.com/article/Details/MAL-056, [Accessed 31-10-2023]

[19] *Autoencoders, Unsupervised Learning, and Deep Architectures — proceedings.mlr.press*, https://proceedings.mlr.press/v27/baldi12a.html, [Accessed 31-10-2023]

[20] *From Variational to Deterministic Autoencoders — arxiv.org*, https://arxiv.org/abs/1903.12436, [Accessed 31-10-2023]

[21] *Density Modeling of Images using a Generalized Normalization Transformation — arxiv.org*, https://arxiv.org/abs/1511.06281, [Accessed 31-10-2023]

[22] *GitHub - jorge-pessoa/pytorch-gdn: PyTorch implementation of the Generalized divisive normalization non-linearity layer — github.com*, https://github.com/jorge-pessoa/pytorch-gdn, [Accessed 31-10-2023]

[23] *Sliced-Wasserstein Autoencoder: An Embarrassingly Simple Generative Model — arxiv.org*, https://arxiv.org/abs/1804.01947, [Accessed 31-10-2023]

[24] *PyTorch-VAE/models/swae.py at master · AntixK/PyTorch-VAE — github.com*, https://github.com/AntixK/PyTorch-VAE/blob/master/models/swae.py, [Accessed 31-10-2023]

[25] K. Zhao, S. Di, M. Dmitriev, T.L.D. Tonellot, Z. Chen, F. Cappello, *Optimizing Error-Bounded Lossy Compression for Scientific Data by Dynamic Spline Interpolation*, in *2021 IEEE 37th International Conference on Data Engineering (ICDE)* (2021), pp. 1643–1654

[26] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, F. Cappello, *Error-Controlled Lossy Compression Optimized for High Compression Ratios of Scientific Datasets*, in *2018 IEEE International Conference on Big Data (Big Data)* (2018), pp. 438–447

[27] *Peak signal-to-noise ratio - Wikipedia — en.wikipedia.org*, https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio, [Accessed 05-11-2023]

[28] linearscalequant, *linearscalequant*, https://arxiv.org/abs/1706.03791, [Accessed 31-10-2023]