Low-loss data compression using deep learning framework with attention-based autoencoder

S. Sriram, P. Chitra* and V. Vijay Sankar

Thiagarajar College of Engineering,
Madurai, Tamilnadu, India
Email: sriramrsk111@gmail.com
Email: pccse@tce.edu
Email: vvs221199@gmail.com
*Corresponding author

S. Abirami

School of Computer Science and Engineering,
Vellore Institute of Technology,
Chennai, Tamil Nadu, India
Email: abirami.2t@gmail.com

S.J. Rethina Durai

Thiagarajar College of Engineering,
Madurai, Tamilnadu, India
Email: rethinadurai@gmail.com

Abstract: With rapid development of media, data compression plays a vital role in efficient data storage and transmission. Deep learning can help the research objective of compression by exploring its technical avenues to overcome the challenges faced by the traditional Windows archivers. The proposed work initially investigates multi-layer autoencoder models, which achieve higher compression rates than traditional Windows archivers but suffer from reconstruction loss. To address this, an attention layer is proposed in the autoencoder to reduce the difference between the encoder and decoder latent representation of an input along with the difference between the original input and reconstructed output. The proposed attention-based autoencoder is extensively evaluated on the atmospheric and oceanic data obtained from the Centre for Development of Advanced Computing (CDAC). The results show that the proposed model performs better with around 89.7% improved compression rate than traditional Windows archiver and 25% reduced reconstruction loss than multi-layer autoencoder.

Keywords: deep learning; multi-layer autoencoder; compression ratio; attention; reconstruction loss.

Reference to this paper should be made as follows: Sriram, S., Chitra, P., Sankar, V.V., Abirami, S. and Durai, S.J.R. (2023) 'Low-loss data compression using deep learning framework with attention-based autoencoder', *Int. J. Computational Science and Engineering*, Vol. 26, No. 1, pp.90–100.

Biographical notes: S. Sriram completed his Bachelor's in Computer Science and Engineering in the Thiagarajar College of Engineering, India. His current research interests include deep learning and natural language processing.

P. Chitra is a Professor in the Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, India. Her areas of interest include cloud computing, multi-core architectures and machine learning.

V. Vijay Sankar completed his Bachelor's in Computer Science and Engineering in the Thiagarajar College of Engineering, India. His current research interests include deep learning and data mining.

S. Abirami is currently pursuing her PhD in Anna University, India. Her research interests include spatiotemporal modelling, machine learning and deep learning algorithms.

S.J. Rethina Durai completed his Bachelor's in Computer Science and Engineering in the Thiagarajar College of Engineering, India. His current research interests include blockchain and machine learning.

1 Introduction

Compression has the power to reduce the data size to even half the original size or a significantly higher percentage of its original size (Yildirim et al., 2018). While information is transferred it can be transmitted in a compressed format. Compressed files occupy less storage space when compared to uncompressed files, meaning a significant decrease in expenses for storage. A compressed file also requires less time for transmission and in the process, it also results in the consumption of less network bandwidth. Newer forms of media, changing hardware create a need for new compression algorithms which are more flexible than the existing ones. Data compression is extensively used in the field of medicine (Yildirim et al., 2018), electrical engineering (Ahmeda and Abo-Zahhad, 2001; Chen et al., 2019), nuclear energy (Cherezov et al., 2020), internet of things (Park et al., 2018), etc. The above-mentioned fields demand that the data used for storage and transmission should be in a compressed format so that the application pipeline difficulties are reduced.

Compression ratio (CR) is the ratio of the size of compressed data to its size in the uncompressed form. Statistical compression algorithms, have very less CR as they do not differentiate between the different kinds of data used for compression (Ziv and Lempel, 2006). The underlying meaning behind the data is not considered by traditional compression algorithms. All types of data are dealt as same, regardless of the nature and format of the data. Data like images (Ameen Suhail and Sankar, 2020; Chen et al., 2017; Cheng et al., 2020; Li et al., 2017; Yang et al., 2020; Zeng et al., 2017; Zhang et al., 2016), sensor data (Nuha et al., 2019), graphical readings (Wang et al., 2020b), instrumental data (Huang et al., 2019), etc. can be compressed more efficiently if the characteristics of data are taken into account. A key challenge in compression is deciding which of the benefits of the particular algorithm to use. Either lossless algorithms, which can compress data without loss but are limited by the amount of compression that can be done, or lossy compression algorithms, which have a higher CR but suffer data loss (Chen et al., 2019). Real-time streaming applications such as environmental monitoring, flood management and so on, requires compression algorithms to have better CR with minimal loss (Wang et al., 2019).

Deep learning algorithms are known to capture trends and features in data that cannot be captured by the human brain (Yildirim et al., 2018). Deep learning algorithms find a wide range of applications (Abirami and Chitra, 2021, 2020; Cao et al., 2021; Chang et al., 2021; Das and Chand, 2021; Jiang et al., 2021; Sun et al., 2021; Varghese and Thampi, 2021; Ying et al., 2021; Zhu et al., 2020). When deep-learning and machine-learning models are employed for the purpose of analysing the trends in the data to compress it, the CR that can be attained post-compression is impressive and even better than the CR achieved by traditional (Ilkhechi et al., 2020) and transformation-based compression algorithms (Ahmeda and Abo-Zahhad, 2001; Al-Nashash, 1995; Wang et al., 2020a). Autoencoders, a well-known deep learning algorithm, are efficient feature extractors and can be employed to identify key patterns in datasets by iterative reconstruction of input data (Ameen Suhail and Sankar, 2020; Chen et al., 2019, 2017; Cheng et al., 2020; Huang et al., 2019; Kim et al., 2020; Ma et al., 2018; Romero et al., 2017; Yang et al., 2020, 2019; Yildirim et al., 2018; Zeng et al., 2017; Zhang et al., 2016).

The proposed work attempts to compress data by considering the characteristics of data using deep learning algorithms. But the trade-off here is the loss of data. In applications where the loss of data is permitted, deep-learning algorithms outperform traditional compression algorithms (Chen et al., 2019; Ilkhechi et al., 2020). However, the demand for reducing the reconstruction loss is critical while dealing with sensitive medical images or environmental data on disasters. The dataset for this study is obtained from the Centre for Development of Advanced Computing (CDAC), a self-governing scientific organisation in India, monitored by the Ministry of Electronics and Information Technology. The dataset consisted of several files representing atmospheric and data in the '.nc' (NetCDF) format. Compression techniques such as 'WinZip' and 'WinRAR' produced a CR of about 0.24 for the considered data (discussed in Section 5). The lower CR obtained by the traditional approaches is not optimal for streaming applications. The proposed work develops an attention-based autoencoder to overcome this challenge faced by the traditional compression algorithms, by obtaining high CR with minimal loss for the dataset considered.

The rest of the paper is organised as follows. Section 2 provides the background and literature study for the compression techniques using autoencoders for varied purposes in different configurations of encoder-decoder architecture. Section 3 proposes various data analysis methods for the dataset under consideration. Section 4 formulates the research problem mathematically. Section 5 introduces the proposed autoencoder framework for data compression. Section 6 introduces case study to compare the performance of proposed models with existing baseline benchmarks algorithm with respect to rate of compression and decompression power and time taken for compression and decompression. Section 7 concludes the paper.

2 Related works

Many research works have been carried over in the field of compression of data using statistical and machine learning algorithms. In the case of compression, the algorithm can either be lossy or lossless. Lossless algorithms retain the original data without any data loss, but the CR achieved would be inferior to that of lossless algorithms (Chen et al., 2019). Statistical lossless algorithms like Huffman (2006) encoding are known to compress the data without any loss. Huffman (2006) propose an optimum method of coding an ensemble of messages consisting of a finite number of members. A minimum-redundancy code is one that is constructed in such a way that the average number of

coding digits per message is minimised. Another popular algorithm LZ77 suggested by Ziv and Lempel (2006) uses lossless methods like window sliding and a look ahead buffer for lossless compression of data.

Lossy methodologies like transformation-based algorithms (Ahmeda and Abo-Zahhad, 2001; Al-Nashash, 1995; Wang et al., 2020a) encode the given data into a newer domain of reduced dimensions. The inverse operation of the former is carried out in order to retrieve the original data back. Al-Nashash et al. (1995) suggest an approach where ECG signals are modelled as a dynamic Fourier series. Fourier coefficients are continuously estimated using either a fast Fourier transform algorithm or the adaptive least mean square algorithm. Ahmeda and Abo-Zahhad (2001) proposed a hybrid transformation algorithm using wavelet transformation and achieved a CR of 20 to 1 is achieved with a percentage root-mean-square difference (PRD) less than 4%. Wang et al. (2020a) suggest a PCA approach to compression which consists of effectively transforming high-dimensional feature space to low-dimensional feature space. The results of this study convey the fact that PCA-based approaches perform better than transformation-based algorithms like wavelet decomposition in terms of better reconstruction results and better compression performance.

Later on, with the rise in computational power, machine learning algorithms became popular and were studied extensively for the purpose of data compression. Park et al. (2018) suggested an approach to use regression for the representation of vectors and the divide-and-conquer method for cropping the chunks of data at each range and merging them using Euclidean distance. Cosine similarity was to determine the accuracy of the model performance. Linda Senigagliesi et al. (2020) proposed data dimension reduction algorithms such as principal component analysis (PCA) and t-distributed stochastic neighbour embedding (t-SNE) for reducing the dimensions of features which helps in proper fast management of samples on IoT devices. PCA has provided the best trade-off on comparing with the complexity and accuracy for which the computational burden is very much reduced. Cherezov et al. (2020) used PCA compression on multi-physics reactor data for reducing the storage and transfer difficulties. They decomposed data into strong and weak components where strong parts were approximated using PCA and a sparse matrix was used to store weak parts in the memory. Chowdhury et al. (2021) used multivariate normal – autoregressive integrated moving average for compression and decompression. Cross-correlation of variables was studied and analysed for reducing the dimension of the data. This resulted in reducing the network resource requirements and bandwidth utilisation.

Deep learning approaches were suggested for lossy compression of data wherein a particular amount of loss was accepted as part of a trade-off between data loss and CR. Yildirim et al. (2018) used deep convolutional autoencoders compression of ECG signals. Signals were reduced to a lower dimension using 27 layers of encoders and decoders. The compression rate was 32.35 which was experimented on 4,800 ECG signal fragments from 48 unique patients. Romero et al. (2017) used quantum autoencoders for compression of data of quantum states. Classical algorithms were used to train the parameters and the trained model was then applied to the ground states of the Hubbard model and Hamiltonians. Huang et al. (2019) proposed a deep-stacked autoencoder (SAE) for compression and classification of electric load data. The accuracy of the fine-tuned SAE using a softmax classifier was 91.63% on one of the datasets. Nuha et al. (2019) used deep neural networks with extreme learning machine for seismic data compression. It consists of a deep asymmetric autoencoder where both linear and nonlinear activation functions are employed. This model achieved a normalised mean squared error of 0.00128 with a CR of 10:1.

Li et al. (2017) used an optimised convolutional networks for content-weighted image compression. This produced a better result than JPEG and JPEG 2000 with sharp edges and rich textures. Cheng et al. (2020) used convolutional autoencoder for compression of energy compaction-based images. They adopted sampling operations, mathematical analysis, and bit allocation method to achieve a high CR. Wang et al. (2020b) compressed panorama map by employing a densely connected convolutional network block-based autoencoder. They designed a new cubic projection-based block partition scheme, weighted loss function and greedy block-wise training method. This method saves 79.69% bit rates compared with JPEG. Ameen Suhail and Sankar (2020) adopted compression and encryption combining autoencoder and chaotic logistic map. Noise attack and occlusion attack analysis were carried out for testing the recovering accuracy of original images from cipher images. This model compresses the size of the input images from 65,536 pixels to 100 pixels.

Yang et al. (2020) performed variable rate deep image compression with a modulated autoencoder. The proposed method achieved the same rate-distortion performance as independent models with significantly fewer parameters. Kim et al. (2020) proposed lossy image compression using deep learning via asymmetric autoencoder and pruning. Their model reduces 28.4% and 39.0% of the weight parameters and floating point operations (FLOPs), respectively. Yang et al. (2019) used a feedback recurrent autoencoder for compression of online sequential data with temporal dependency. They achieved lower variable bitrate using an entropy coder. The model achieved the best rate-distortion of around 1.5 Kbps. Chen et al. (2017) used convolutional autoencoder neural network for medical image analysis. Eight layers were used which consists of a convolutional layer, pooling layer, fully connected layer and a classifier which produces an accuracy of 0.97%.

Zeng et al. (2017) adopted a coupled deep autoencoder for representing both low and high-resolution images. The model produces the highest peak signal to noise ratio (PSNR) 0.27 dB higher than that of the second-best approach. Zhang et al. (2016) used deep neural networks for

compression and classification of halftone images based on sparse autoencoder. They proposed an effective patch extraction method by measuring the mean and variance of the halftone images. The proposed method achieves an average correct classification rate (ACCR) of over 99.44%. Ma et al. (2018) used a deep coupling autoencoder (DCAE) for fault diagnosis with multi-modal sensory data. They developed data fusion strategies to effectively handle multi-modal sensory signals. The DCAE model produces a classification rate of 94.3%.

All deep learning-based approaches discussed above including the autoencoders were able to obtain high CR than traditional algorithms However, improvement in terms of reconstruction loss is still a challenge. Environmental data compression is critical in environmental monitoring applications that require a high CR with minimal reconstruction loss. The contributions of the paper are as follows:

1 Determines the relationship between various parameters in the dataset and validates the suitability of deep learning algorithms for its compression.

2 Utilises deep learning architecture to obtain improved CR when compressing environmental data with no inter-parameter relationships.

3 Proposes an autoencoder stacked with attention layer that prevents the input data spatial transitions during the compression and decompression to minimise the reconstruction loss.

4 Performs extensive experimentation with model architecture to overcome the performance degradation issues faced by machine learning models and deep learning models.

3 Exploratory data analysis

In this study, before feeding the data to the proposed algorithm, it is subjected to analysis for identifying the main characteristics and relations by using certain statistical and visualisation techniques. The inferences obtained are discussed below.

3.1 Data overview

The dataset under consideration was provided by CDAC, a self-governing scientific organisation in India, monitored by the Ministry of Electronics and Information Technology. The dataset had multiple files of atmospheric data such as temperature, rain, sea level pressure, recorded across the globe through 180 latitudes and 360 longitudes. The dataset accounted for a total size of 211.56 MB. The type of data that the dataset comprised were numeric values. Data belonging to adjacent cells were closer to each other in terms of numerical value. The dataset files are temperature in Celsius [TC], sea level pressure [SLP], dew point temperature [TD], wind speed [WS], temperature at 2 metres from ground [T2], dew point temperature at 2 metres from ground [TD2]. The values were represented as numerical values. The dataset files TC, TD and WS comprise of 360 columns and 30,408 rows while SLP, TD2 and T2 comprise of 360 columns and 3,801 rows respectively.

3.2 Data pre-processing

The provided dataset was of '.nc' (NetCDF) format. For exploratory data analysis and further processing through the algorithm, the dataset had to be represented in a format suitable for processing through deep learning libraries. Hence, the different files were converted from '.nc' into '.csv' files using the Panoply tool. Comma separated values files are easier to integrate into data analysis tools and storage media regardless of system configuration.

3.3 Pearson correlation analysis

In deep learning, exploratory data analysis is frequently required prior to the algorithm's application. In this study, correlation of parameters across different files in the considered dataset is analysed using Pearson coefficient (Zou et al., 2003). Pearson coefficient which ranges between –1 and +1 is a measure of relation or correlation between two variables x and y. A correlation coefficient of 0 indicates no correlation whereas values above and below 0 indicate a positive and negative correlation respectively.

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 (y_i - \bar{y})^2}} \tag{1}$$

where r is the correlation coefficient, x_i is the values of x-variable in a sample, $\bar{x}$ is the mean of values of x-variable, y_i is the values of y-variable in a sample, $\bar{y}$ is the mean of values of y-variable.

Figure 1 depicts the Pearson correlation coefficient values between the different parameter files TC, SLP, TD, TD2, T2, and WS, of the considered dataset. From Figure 1 it can be inferred that there is no correlation between the parameter files TC, SLP, TD, TD2, T2, and WS. Zou et al. (2003) suggest that Pearson coefficient can be used to investigate the association of two variables. If two variables are said to be 'highly correlated', the prediction of one variable using the other is possible by generating a hypothesis while considering the fact that the two variables are correlated. Based on this study, from the plot it was inferred that linear modelling and prediction of one type of atmospheric data using multiple other atmospheric data would not yield satisfactory results. If such a correlation existed among the different variables, linear modelling such as regression would have been possible. Since the variables do not exhibit such favourable characteristics, deep learning algorithms like autoencoders are suitable for complex modelling of feature extraction.

Figure 1 Pearson correlation for the dataset (see online version for colours)

4 Problem formulation

The objective of the proposed work is to identify the nonlinear mapping functions F_E and F_D that helps to compress and decompress an input data X respectively as shown below,

$$\hat{X} = F_D\left(F_E(X)\right) \tag{2}$$

where $\hat{X}$ is the reconstructed data. The proposed attention-based autoencoder aims to increase the CR ($F_E(X)$ / X) preserving very negligible difference ($X - \hat{X}$) between the actual data and reconstructed data.

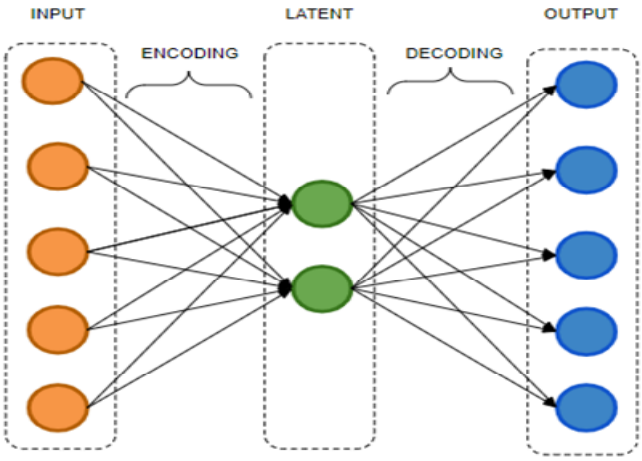
5 Model formulation and configuration

5.1 Autoencoders

Autoencoders (Ameen Suhail and Sankar, 2020; Chen et al., 2019, 2017; Cheng et al., 2020; Huang et al., 2019; Kim et al., 2020; Ma et al., 2018; Romero et al., 2017; Yang et al., 2020, 2019; Yildirim et al., 2018; Zeng et al., 2017; Zhang et al., 2016) are unsupervised deep learning models that set the target values to be equal to the inputs, so during training, the loss would indirectly indicate reconstruction error. Autoencoders can reduce the dimensions of the inputs and then represent the input data in a smaller form. To obtain the original data back, the encoded component, i.e., latent representation should be passed through the decoding end of the autoencoder. The general thumb rule in deep learning is to use a set of features X in order to predict a feature Y. The gravity of the rule can be shifted such that the set of features X can be used to predict the same input. This property was used in employing autoencoders for compression.

The input data enclosed by the first bounding box of Figure 2 represents the input data. After propagation of input through the various layers of the autoencoder architecture, a latent or compressed representation of the input would be obtained at the end of the encoding phase, represented by the neurons enclosed by the second bounding box of Figure 2. It can be inferred that the latent representation is of reduced dimensions than the input data. On the application of weights of decoder through multiple layers and activations, the decompressed representation can be obtained. The reconstructed output would be the end-result of the decoding phase which is enclosed by the third bounding box of Figure 2. The reconstructed output would be of the same dimensions as the input data that was propagated initially through the encoder.

Figure 2 Components of autoencoders (see online version for colours)



5.2 Attention mechanism

The proposed attention mechanism forces the eccentric difference between the latent representation of an input at the encoder and its corresponding representation in the decoder to be minimum so that all common factors of variation in the dataset are well captured. This attention is added as an additional weightage to the conventional loss function of a multi-layer autoencoder. The clusters in the dataset are identified using k-means clustering and the Euclidean distance between any input in the encoder and decoder from its cluster head is estimated as Z_e and Z_d. When the input is passed from one layer to the next in the encoder, the base mathematical process that occurs is mapping the input vector of higher dimensions to lower dimensions. By introducing an attention mechanism that focuses on linear alignment of encoded and decoded data in latent representations, the model can be penalised with an additional factor whenever reconstruction error is calculated. Additional calculation is made to the loss function where the cluster distance of each and every point in the original input form and the points in latent representation is added. If points that are supposed to be closer to each other in the initial form are spaced out from each other in the latent representation during the model construction, it may lead to a higher error rate during the reconstruction phase. Hence this property is added to the conventional loss calculation metric. Now the model will able to easily align and try to reconstruct the data during decompression. Alignment is one of the key performance indicators since alignment score will give how well the

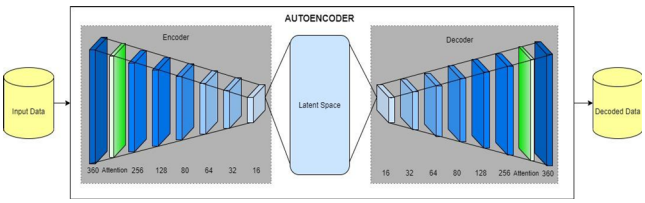
input encoded model can match the decoded value at current level. This helps to reduce the reconstruction loss during decompression.

5.3 Proposed architecture

The dataset files TC, SLP, TD, TD2, T2 and WS consisted of 360 columns, each column corresponding to reading from different latitudes that were recorded for the same longitude. The number of columns in each of the data parameters was found to be a constant – 360, hence columnar reduction would be the right approach instead of a row-wise compression approach. Input layer of the model was designed with 360 neurons in such a way that each value of a row in the dataset is fed into one of the 360 neurons while also maintaining that every neuron receives precisely one value. In each subsequent layer of the encoder, the number of neurons was reduced when compared to the previous layer. The general rule used for the encoding phase was $n(i) < n(i-1)$, where $n(i)$ corresponds to the number of neurons in i^{th} layer of encoder part of autoencoder architecture and $n(i-1)$ corresponds to the number of neurons in the layer immediately preceding the i^{th} layer of the encoder.

This principle forces the autoencoder model to learn the general trends and latent characteristic of data. On the other end of the autoencoder, the decoding phase is characterised by the property where the number of neurons in each subsequent layer increases. A symmetric architecture is followed while designing the encoder and decoder. Figure 3 depicts the proposed architecture of autoencoder. The numerical values under each layer correspond to the number of neurons used in that particular layer.

Figure 3 Architecture of autoencoder with attention (see online version for colours)



The mathematical functions of the autoencoder can be split into 2: the encoder and the decoder (Chen et al., 2019). The transformation equations of the decoder are the transpose of the encoder. The quality of the reconstruction is judged based on the similarity between the input given to the encoder and the output produced by the decoder. Let us consider an input vector fed to the input layer as X_i. The loss function of the proposed model is defined as

$$Loss = \frac{1}{n}\sum_{i=1}^{n}\left|X_i - \hat{X}_i\right| + \frac{1}{n}\sum_{i=1}^{n}\left|Z_e - Z_d\right| \tag{3}$$

where X_i is the input data point, $\hat{X}_i$ is the reconstructed data point, n is the number of total data points, Z_e is the distance between the data point and cluster centre in latent space of encoder, Z_d is the distance between the data point and cluster center in latent space of decoder. The difference between Z_e and Z_d in the loss function, inhibits the spatial transition of the data points during compression and decompression thereby reducing the reconstruction loss. From equation (3), the functions F_E, F_D are the parameters of the model estimated using Adam optimiser techniques such that,

$$F_E^*, F_D^* = \arg\min_{F_E, F_D} Loss \tag{4}$$

Algorithm for the proposed method is shown in Figure 4, which explains the stepwise training of the proposed attention-based autoencoder for data compression.

Figure 4 Training of proposed autoencoder with attention (see online version for colours)

Algorithm: Proposed Autoencoder

Require: Input Data

1: Read Input data
2: Initialize the input layer of autoencoder
3: epoch = 800
4: iter = 0
5: **while** (iter < epoch)
6: **for** given batch of input data X **do**
7: **Propagate** input data through encoding layers to obtain encoded vector P
8: **Apply** the weights of decoder layers on latent vector P
9: **Compute** current reconstruction error $\frac{1}{n}\sum_{i=1}^{n}|X_i - \hat{X}_i| + \frac{1}{n}\sum_{i=1}^{n}|Z_e - Z_d|$
10: **if** (current reconstruction error not improved for 10 iter) **then**
11: iter = epoch
12: **break**
13: **end if**
14: **if** (current reconstruction error < reconstruction) **then**
15: reconstruction error = current reconstruction error
16: **end if**
17: **Adjust** model weight parameters
18: **end for**
19: **end while**

5.4 Model configuration

The total number of layers in the proposed architecture is 14. There are seven layers in the encoder and seven layers in the decoder. The number of neurons in the first layer of the encoder, i.e., input layer is 360 neurons, which matches the dimension of the input vector. In the second layer, the number of neurons is reduced to 256 and this is the first part of the bottleneck. After this the bottleneck is narrowed even further by using 128, 80, 64, 32 and 16 neurons in each of the subsequent layers of the encoder. The architecture of the decoder is a symmetric inverse of the encoder's architecture. The number of neurons in the first layer of the decoder is 16, which matches the dimension of the input that the decoder receives from the encoder. Since decoder is concerned with reconstructing the output vector, the number of neurons in each of the subsequent layers of decoders are increased from 16 to 32, then 64, 80, 128, 256 and then finally to 360 neurons in the output layer.

Before feeding the output of a layer to its successor, the output vector is passed through a LeakyReLU activation function. LeakyReLU was preferred over ReLU to avoid the

'dying ReLU' problem. ReLU is represented as $f(x) = \max(0, x)$ and ReLU considers zero as the value when the inputs are lesser than zero. During the backpropagation, zero valued gradients boil down to zero and convergence to a good local minimum does not occur.

5.5 Platform and packages

The processes of training, testing, comparison of various deep learning architectures was carried out using a computer of 16 GB RAM, Intel Core i7-9750H 2.60 GHz processor with a 64-bit operating system. Keras was used for the technical implementation of neural networks. Pandas, Numpy Python packages were used for loading and manipulating datasets. Seaborn, Matplotlib Python packages were used for exploratory data analysis and for generating graphical representations. Panoply tool was used for converting the '.nc' format data into '.csv' files. The comparison baseline would be to achieve a higher CR than that of compression algorithms used in windows operating systems.

6 Implementation results and discussion

6.1 Performance metrics

6.1.1 Mean absolute error

Mean absolute error (MAE) (Chen et al., 2019) is the measured average difference between the predicted values and actual value. MAE was used synonymously to the reconstruction error depicting loss before compression and after decompression.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \tag{5}$$

where, 〖 Y〗_i is the input data point, Yˆ_i is the reconstructed data point, n is number of total data points.

From an interpretation standpoint, MAE is best suited. Mean Squared Error or Root Mean Squared Error do not describe the average error and have other implications that are more difficult to understand.

6.1.2 Compression ratio

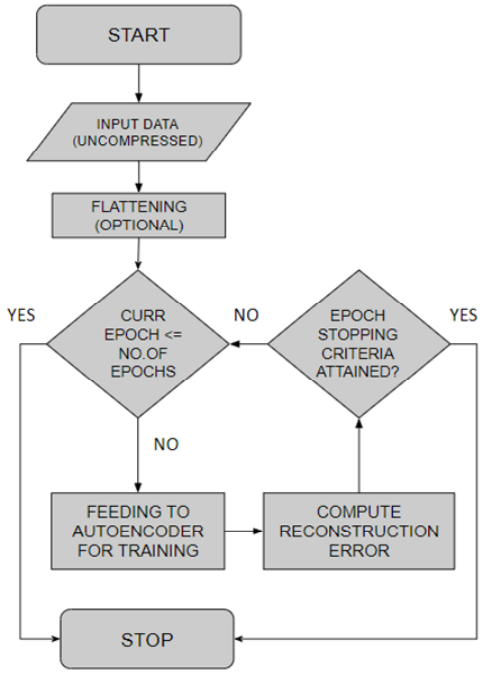
Data CR (Ilkhechi et al., 2020) is the relative reduction in the size of data produced by a data compression algorithm. Smaller the value, better the compression performance it is expressed as the division of compressed size by the uncompressed size.

$$CR = \frac{size\ of\ compressed\ file}{size\ of\ uncompressed\ file} \tag{6}$$

6.2 Autoencoder training process

Input X in the form of a multi-dimensional vector was passed into the network via the input layer. It was encoded into a vector Z of lower dimensions by a mapping function. Z is considered as the encoded or latent representation of input X. This is termed as the compressed version. Z was then decoded into the output vector Y, which was of the same dimensions as X, aimed to replicate the input X. Reconstruction error $|X - Y|$ with the added attention was computed. The goal was to minimise reconstruction error and this would be the objective of the training process of the model. The error was back propagated for weight updation. These were the proceedings that comprised one epoch. Iterative training throughout multiple epochs was necessary for the complete learning process of the algorithm. Figure 5 depicts the training process for the proposed autoencoder.

Figure 5 Autoencoder training pipeline



6.3 Train-test fissure and validation using unseen data

For the purpose of training, the given dataset was split in an 80:10:10 ratio (Yildirim et al., 2018) for training, validation and testing respectively. For a given atmospheric or oceanic parameter, readings from multiple days were present in individual files. Validation was performed by utilising the data records from the files corresponding to days that were not used for training. This method of training the model with unseen data aids in generalisation capabilities and adaptability of the autoencoder algorithm.

6.4 The compression size – reconstruction error trade-off

The various trail architectures for the file 'tc.csv' which had temperature values of the sea, is tabulated in Table 1. The architecture of the decoder was designed to replicate the encoder in reverse. The original size of the file before compression was 67.2 megabytes. Few sample values from

the dataset are –33.25, –46.15, –62.89, and 4.96. A reconstruction error of 0.5 would imply that a value, say 20 in the reconstructed file (compressed and again decompressed) could have been between 19.5 and 20.5 in the original input file.

Table 1 Performance comparison

Model number	*Number of neurons of subsequent encoding layers of different model architectures*	*Reconstruction error – mean absolute error*	*Compressed file size MB – megabytes*
Model 1	360-256-128-64	Loss: 0.5225; val_loss: 0.5629	19.83 MB
Model 2	360-256-180-128-64	Loss: 0.5461; val_loss: 0.5843	20.19 MB
Model 3	360-256-128-64-32	Loss: 0.6573; val_loss: 0.7494	10.56 MB
Model 4	360-256-128-64-32-16	Loss: 0.7846; val_loss: 0.7957	5.02 MB
Model 5	360-256-128-80-64-32-16	Loss: 0.6536; val_loss: 0.6991	5.01 MB

In Table 1, higher compression of a file, by reducing the number of neurons in the final output layer of the encoding phase of the autoencoder. An individualistic yet interesting property was observed wherein, the further the file was compressed, higher was the loss. Reduction in the number of neurons is performed by stepwise reduction of neurons in multiple layers and also in some cases by direct replacement of a higher number of neurons with lower number of neurons without introduction of additional layers.

Among the models tested in Table 1, model 5 performed the best. The improvement of model 5 over other was achieved via usage of additional data for training, experimentation with hyperparameters such as learning rate and batch size. The conventional batch sizes of powers of 2 such as 32, 64 and 256 did not yield the best results. Due to presence of 181 latitudes (from –90 till +90), batch sizes of powers of 181 were tried and a batch size of 905 yielded the best result, which is used in model 5. Also, Table 1 infers that model 5 – an autoencoder-based compression algorithm is able to get improved performance than traditional compression algorithms, which failed to compress the file size below 11 megabytes.

6.5 *Comparison with statistical and deep learning benchmarks*

In order to validate the performance of the proposed attention-based autoencoder, their evaluation results are compared with the baseline approaches. Among the traditional methods, WinRAR and WinZip are the baselines considered for this study. Whereas, among the deep learning algorithms, multi-layer Autoencoder is the baseline considered for comparing the compression performance of the proposed algorithm. The research outcome criteria are divided into three aspects: CR, time taken for compression and decompression, and reconstruction loss. On evaluation, it is observed that the size of the compressed file obtained using the multi-layer autoencoder and the proposed autoencoder with attention are the same. Hence the CR obtained by both are the same. Furthermore, it is observed that the proposed autoencoder with attention outperforms the multi-layer autoencoder in terms of significantly reduced reconstruction loss.

The raw numerical compression size obtained by the compression mechanisms such as WinRAR, WinZip, multi-layer autoencoder and proposed autoencoder with attention is illustrated in Figure 6. The variables under 'name of the parameter' refer to different files in the dataset considered. They are 'tc' – temperature in Celsius, '*slp*' – sea level pressure, 'td' – dew point temperature, 'ws' – wind speed, 't2' – temperature at 2 metres above ground, 'td2' – dew point temperature at 2 metres above ground. Figure 6 shows that the compression performance of the multi-layer autoencoder and the proposed autoencoder are the same in terms of CR, with both achieving an average of 40% better CR than the traditional windows achievers. Figure 6 also demonstrates that autoencoder's pattern capturing property is more appropriate for the 'TD' file, for which it achieves the highest CR of around 0.18. This establishes the efficient nature of the autoencoder model over 'WinRAR' and 'WinZip' compressions in terms of the extent to which a file can be compressed.

Figure 6 Numerical bar representation of size of compressed file by traditional and proposed methods (see online version for colours)

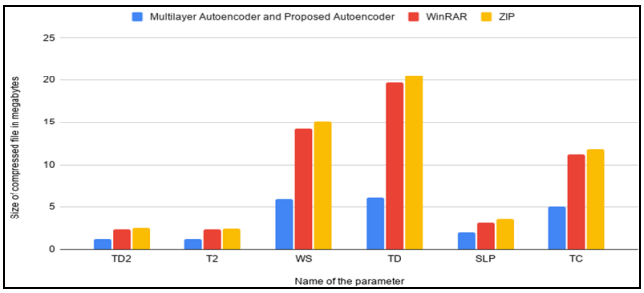


Figure 7 depicts the performance comparison of competitors during compression process. Comparison of the percentage value of each algorithm's compressed size with respect to the other algorithms in the graph as a whole aid in the analysis of one-to-one and part-wise performance comparison. The algorithm that produced the least result in terms of compression size was taken as the base reference, i.e., one whole part (100%). In the above case, 'WinZip' produced the worst results while the trained autoencoder model produced the best result. In terms of percentages, autoencoders were able to compress the files to approximately 20% more than the compression achieved by 'WinZip' in every case. The overall CR for the proposed autoencoder model, WinRAR and WinZip were 0.103,

0.247 and 0.260 respectively. Improved CR of autoencoders is due to its ability to capture layers of abstraction in data during compression.

Figure 7 Comparison of compression performance in terms of percentage (see online version for colours)

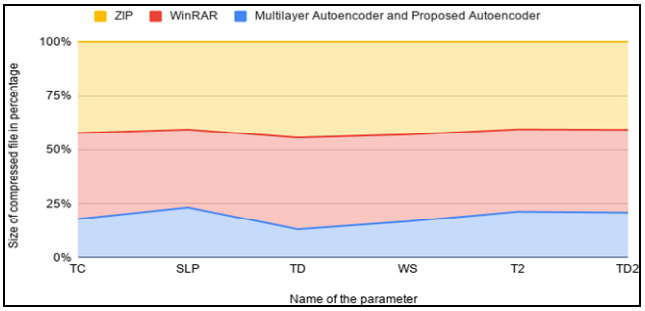


Table 2 Performance comparison

Name of parameter	*Sample values in the dataset*	*Loss obtained by multi-layer autoencoder*	*Loss obtained by using proposed multi-layer autoencoder with custom attention*
TC	–33.25, –46.15, –62.89	0.6536	0.4401
SLP	992.78, 998.76, 1,009.60	1.2494	0.8034
TD	–35.53, –32.55, –31.95	2.7521	1.9923
WS	37.2, 31.3, 32.6	3.1015	2.5012
T2	–44.06, –43.86, –40.86	1.2861	0.7577
TD2	–44, –42.2, –41.3	1.6909	0.9234

The improved CR of a multi-layer autoencoder is only significant when the reconstruction loss is reduced. The proposed attention in the autoencoder aids in this endeavour. The reconstruction loss, in terms of MAE between the original file and the decompressed file is obtained for all files in the dataset. The reconstruction loss by both multi-layer autoencoder and the proposed autoencoder with attention is summarised in the Table 2. It can be inferred from Table 2 that the proposed autoencoder model achieves on par CR as that of multi-layer autoencoder, but overpowers the multi-layer autoencoder in terms of the reconstruction error. On an average the proposed autoencoder with attention obtains 34% reduced reconstruction loss than the multi-layer autoencoder. This illustrates the significance of the proposed attention layer in autoencoder. The improved CR of the proposed autoencoder model with attention than WinRAR and WinZip and its enhanced performance towards reducing the reconstruction loss than the multi-layer encoder demonstrates its efficiency towards environmental data compression.

After validating the proposed autoencoder's performance in terms of CR and reconstruction loss, it is finally evaluated in terms of the time required for the processes of original file compression and latent-state file decompression. The compression and decompression time of the proposed autoencoder and baseline for various files in the dataset in the dataset is noted and illustrated in Figures 8 and 9. Figure 8 shows the time taken for compression and Figure 9 shows the time taken for decompression of the different parameter files. All tests were carried out under standard conditions using a computer of 16 GB RAM, Intel Core i7-9750H 2.60 GHz processor with a 64-bit operating system.

Figure 8 Comparison of time taken for compression by traditional and proposed methods (see online version for colours)

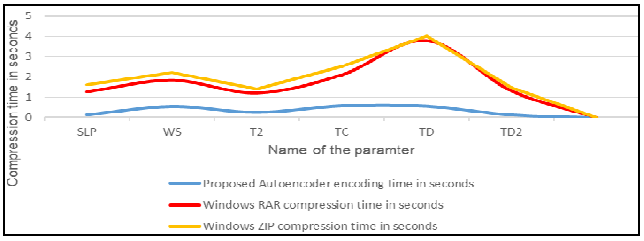
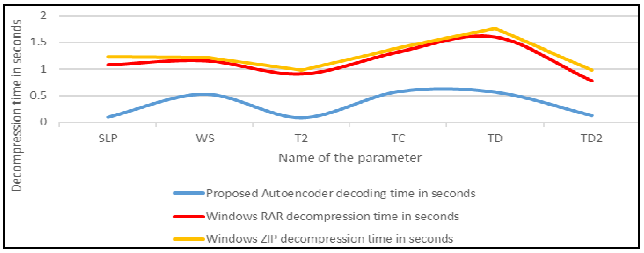


Figure 9 Comparison of time taken for decompression by traditional and proposed methods (see online version for colours)



Figures 8 and 9 infers that the proposed autoencoder model outperforms WinRAR and WinZip compression with respect to the time taken for compression and decompression. This is due to the fact that the general process in autoencoder consists of applying the parametric weights from the trained model file, whereas traditional compression algorithms include more complex computations.

According to the results of the above-mentioned evaluations, the proposed autoencoder with attention exhibits efficient compression performance with improved CR, reduced time for compression and decompression, and minimal reconstruction loss. This makes it more appropriate for real-world applications like automatic weather stations, IoT powered weather forecasting, and pro-active natural disaster response systems. These applications necessitate that data collected from various sources, such as weather sensors and atmospheric sensors, be compressed as much as possible. The proposed autoencoder with attention, which outperforms traditional archivers, would be critical in such use-cases demanding less transmission time and network bandwidth.

7 Conclusions and future work

Data compression is required for data storage in smaller sizes, while further reducing bandwidth consumption during transmission. Deep learning can achieve effective data compression by iteratively propagating reconstruction error between original and reconstructed data through the network. In terms of compression limit, autoencoders outperform traditional algorithms. However, it results in significant data loss. To overcome the challenge of reducing reconstruction loss, the proposed work employs an attention mechanism in the multi-layer autoencoder. The attention layer reduces reconstruction loss by preventing data points from moving away from a spatial reference during compression and decompression. This study evaluated the compression performance of the proposed attention-based autoencoder with the traditional windows archiver for compressing the atmospheric and oceanic data obtained from the CDAC. The overall CR of autoencoder model, WinRAR and WinZip were 0.103, 0.247 and 0.260 respectively and from this it can be inferred that the autoencoder model outperforms WinRAR and WinZip. Also, the reduced reconstruction loss proves the proficiency of the proposed model in reducing the variance of the input points in the latent or compressed representation. Therefore, the proposed attention-based autoencoder is better suited than the existing codecs. However, the proposed methodology's performance against datasets devoid of statistical/numerical patterns and datasets with greater scarcity is dubious. In the future, variational deep learning models that could well suit such challenges will be investigated for data compression.

References

Abirami, S. and Chitra, P. (2020) 'Chapter fourteen – energy-efficient edge based real-time healthcare support system', *Advances in Computers*, Elsevier, Vol. 117. No. 1, pp.339–368 [online] https://doi.org/https://doi.org/10.1016/bs.adcom.2019.09.007.

Abirami, S. and Chitra, P. (2021) 'Regional air quality forecasting using spatiotemporal deep learning', *Journal of Cleaner Production*, Vol. 283, p.125341 [online] https://doi.org/10.1016/j.jclepro.2020.125341.

Ahmeda, S.M. and Abo-Zahhad, M. (2001) 'A new hybrid algorithm for ECG signal compression based on the wavelet transformation of the linearly predicted error', *Medical Engineering & Physics*, Vol. 23, pp.117–126 [online] https://doi.org/10.1016/S1350-4533(01)00026-1.

Al-Nashash, H.A.M. (1995) 'A dynamic fourier series for the compression of ECG using FFT and adaptive coefficient estimation', *Medical Engineering & Physics*, Vol. 17, pp.197–203 [online] https://doi.org/10.1016/1350-4533(95)95710-R.

Ameen Suhail, K.M. and Sankar, S. (2020) 'Image compression and encryption combining autoencoder and chaotic logistic map', *Iranian Journal of Science and Technology, Transactions A: Science*, Vol. 44, pp.1091–1100 [online] https://doi.org/10.1007/s40995-020-00905-4.

Cao, Z., Zhou, Y., Yang, A. and Peng, S. (2021) 'Deep transfer learning mechanism for fine-grained cross-domain sentiment classification', *Connection Science*, Vol. 33, pp.911–928 [online] https://doi.org/10.1080/09540091.2021.1912711.

Chang, F., Ge, L., Li, S., Wu, K. and Wang, Y. (2021) 'Self-adaptive spatial-temporal network based on heterogeneous data for air quality prediction', *Connection Science*, Vol. 33, pp.427–446 [online] https://doi.org/10.1080/09540091.2020.1841095.

Chen, H., Wang, S., Wu, L. and Wang, J. (2019) 'A novel smart meter data compression method via stacked convolutional sparse auto-encoder', *International Journal of Electrical Power & Energy Systems*, Vol. 118 [online] https://doi.org/10.1016/j.ijepes.2019.105761.

Chen, M., Shi, X., Zhang, Y., Wu, D. and Guizani, M. (2017) 'Deep features learning for medical image analysis with convolutional autoencoder neural network', *IEEE Transactions on Big Data*, Vol. 1 [online] https://doi.org/10.1109/TBDATA.2017.2717439.

Cheng, Z., Sun, H., Takeuchi, M. and Katto, J. (2020) 'Energy compaction-based image compression using convolutional autoencoder', *IEEE Transactions on Multimedia*, Vol. 22, pp.860–873 [online] https://doi.org/10.1109/TMM.2019.2938345.

Cherezov, A., Jang, J. and Lee, D. (2020) 'A PCA compression method for reactor core transient multiphysics simulation', *Progress in Nuclear Energy*, Vol. 128, p.103441 [online] https://doi.org/10.1016/j.pnucene.2020.103441.

Chowdhury, M.R., Tripathi, S. and De, S. (2021) 'Adaptive multivariate data compression in smart metering internet of things', *IEEE Transactions on Industrial Informatics*, Vol. 17, pp.1287–1297 [online] https://doi.org/10.1109/TII.2020.2981382.

Das, P. and Chand, S. (2021) 'Extracting road maps from high-resolution satellite imagery using refined DSE-LinkNet', *Connection Science*, Vol. 33, pp.278–295 [online] https://doi.org/10.1080/09540091.2020.1807466.

Huang, X., Hu, T., Ye, C., Xu, G., Wang, X. and Chen, L. (2019) 'Electric load data compression and classification based on deep stacked auto-encoders', *Energies* [online] https://doi.org/10.3390/en12040653.

Huffman, D.A. (2006) 'A method for the construction of minimum-redundancy codes', *Resonance*, Vol. 11, pp.91–99 [online] https://doi.org/10.1007/BF02837279.

Ilkhechi, A., Crotty, A., Galakatos, A., Mao, Y., Fan, G., Shi, X. and Cetintemel, U. (2020) 'DeepSqueeze: deep semantic compression for tabular data', in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, SIGMOD'20*, Association for Computing Machinery, New York, NY, USA, pp.1733–1746 [online] https://doi.org/10.1145/3318464.3389734.

Jiang, D., Qu, H., Zhao, J., Zhao, J. and Hsieh, M Y. (2021) 'Aggregating multi-scale contextual features from multiple stages for semantic image segmentation', *Connection Science*, Vol. 33, pp.605–622 [online] https://doi.org/10.1080/09540091.2020.1862059.

Kim, J., Choi, J., Chang, J. and Lee, J. (2020) 'Efficient deep learning-based lossy image compression via asymmetric autoencoder and pruning', in *ICASSP 2020 – 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp.2063–2067 [online] https://doi.org/10.1109/ICASSP40776.2020.9053102.

Li, M., Zuo, W., Gu, S., Zhao, D. and Zhang, D. (2017) *Learning Convolutional Networks for Content-Weighted Image Compression*, CoRR abs/1703.1.

Ma, M., Sun, C. and Chen, X. (2018) 'Deep coupling autoencoder for fault diagnosis with multimodal sensory data', *IEEE Transactions on Industrial Informatics*, Vol. 14, pp.1137–1145 [online] https://doi.org/10.1109/TII.2018.2793246.

Nuha, H., Balghonaim, A., Liu, B., Mohandes, M., Deriche, M. and Fekri, F. (2019) 'Deep neural networks with extreme learning machine for seismic data compression', *Arabian Journal for Science and Engineering*, Vol. 45 [online] https://doi.org/10.1007/s13369-019-03942-3.

Park, J., Park, H. and Choi, Y. (2018) 'Data compression and prediction using machine learning for industrial IoT', in *2018 International Conference on Information Networking (ICOIN)*, pp.818–820 [online] https://doi.org/10.1109/ICOIN.2018.8343232.

Romero, J., Olson, J.P. and Aspuru-Guzik, A. (2017) 'Quantum autoencoders for efficient compression of quantum data', *Quantum Science and Technology*, Vol. 2, p.45001 [online] https://doi.org/10.1088/2058-9565/aa8072.

Senigagliesi, L., Baldi, M. and Gambi, E. (2020) 'Physical layer authentication techniques based on machine learning with data compression', in *2020 IEEE Conference on Communications and Network Security (CNS)*, IEEE, pp.1–6 [online] https://doi.org/10.1109/CNS48642.2020.9162280.

Sun, X., Wang, Q., Zhang, X., Xu, C. and Zhang, W. (2021) 'Deep blur detection network with boundary-aware multi-scale features', *Connection Science*, pp.1–19 [online] https://doi.org/10.1080/09540091.2021.1933906.

Varghese, E.B. and Thampi, S.M. (2021) 'Towards the cognitive and psychological perspectives of crowd behaviour: a vision-based analysis', *Connection Science*, Vol. 33, No. 2, pp.380–405 [online] https://doi.org/10.1080/09540091.2020.1772723.

Wang, K., Zhang, M., Zhang, S. and Xu, Z. (2020a) 'A PQ data compression algorithm based on wavelet domain principal component analysis', in *2020 Asia Energy and Electrical Engineering Symposium (AEEES)*, pp.347–350 [online] https://doi.org/10.1109/AEEES48850.2020.9121347.

Wang, S., Wang, H., Xiang, S. and Yu, L. (2020b) 'Densely connected convolutional network block based autoencoder for panorama map compression', *Signal Processing: Image Communication*, Vol. 80, p.115678 [online] https://doi.org/10.1016/j.image.2019.115678.

Wang, W., Feng, C., Zhang, B. and Gao, H. (2019) 'Environmental monitoring based on fog computing paradigm and internet of things', *IEEE Access*, Vol. 7, pp.127154–127165 [online] https://doi.org/10.1109/ACCESS.2019.2939017.

Yang, F., Herranz, L., Weijer, J.V.d., Guitián, J.A.I., López, A.M. and Mozerov, M.G. (2020) 'Variable rate deep image compression with modulated autoencoder', *IEEE Signal Processing Letters*, Vol. 27, pp.331–335 [online] https://doi.org/10.1109/LSP.2020.2970539.

Yang, Y., Sautière, G., Ryu, J.J. and Cohen, T.S. (2019) *Feedback Recurrent AutoEncoder*, CoRR abs/1911.0.

Yildirim, O., Tan, R.S. and Acharya, U.R. (2018) 'An efficient compression of ECG signals using deep convolutional autoencoders', *Cognitive Systems Research*, Vol. 52, pp.198–211 [online] https://doi.org/10.1016/j.cogsys.2018.07.004.

Ying, L., Nan, Z.Q., Ping, W.F., Kiang, C.T., Pang, L.K., Chang, Z.H., Lu, C., Jun, L.G. and Nam, L. (2021) 'Adaptive weights learning in CNN feature fusion for crime scene investigation image classification', *Connection Science*, Vol. 33, pp.719–734 [online] https://doi.org/10.1080/09540091.2021.1875987.

Zeng, K., Yu, J., Wang, R., Li, C. and Tao, D. (2017) 'Coupled deep autoencoder for single image super-resolution', *IEEE Transactions on Cybernetics*, Vol. 47, pp.27–37 [online] https://doi.org/10.1109/TCYB.2015.2501373.

Zhang, Y., Zhang, E. and Chen, W. (2016) 'Deep neural network for halftone image classification based on sparse auto-encoder', *Engineering Applications of Artificial Intelligence*, Vol. 50, pp.245–255 [online] https://doi.org/10.1016/j.engappai.2016.01.032.

Zhu, X., Zuo, J. and Ren, H. (2020) 'A modified deep neural network enables identification of foliage under complex background', *Connection Science*, Vol. 32, pp.1–15 [online] https://doi.org/10.1080/09540091.2019.1609420.

Ziv, J. and Lempel, A. (2006) 'A universal algorithm for sequential data compression', *IEEE Trans. Inf. Theor.*, Vol. 23, pp.337–343 [online] https://doi.org/10.1109/TIT.1977.1055714.

Zou, K.H., Tuncali, K. and Silverman, S.G. (2003) 'Correlation and simple linear regression', *Radiology*, Vol. 227, pp.617–628 [online] https://doi.org/10.1148/radiol.2273011499.