

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

Image Compression Using Autoencoder Neural Networks

Nancy Ashica A¹, Sathia Priya R², Vignesh K³

U.G. Student, Department of Computer Science and Engineering, Loyola-ICAM College of Engineering and
Technology, Chennai, Tamil Nadu, India¹

Assistant Professor, Department of Computer Science and Engineering, Loyola-ICAM College of Engineering. and
Technology, Chennai, Tamil Nadu, India²

U.G. Student, Department of Computer Science and Engineering, Loyola-ICAM College of Engineering and
Technology, Chennai, Tamil Nadu, India³

ABSTRACT: Compression involves processing data (image) to reduce its size so that it occupies less space. There are two types of image compression; lossy and lossless. In lossless compression, one can retrieve the original image data, while in lossy compression one cannot. Image compression is very crucial in order to reduce the size of disk space used as well as reduce the amount of internet bandwidth used while loading images. A compression autoencoder usually has three parts Encoder, Code and Decoder. The autoencoder neural network is implemented with the help of these three models. The reconstructed image is obtained and the performance assessment is done on both the original and the reconstructed images.

KEYWORDS: Autoencoder, Image Compression, Feature extraction, Latent Space Representation, Neural Networks.

I. INTRODUCTION

We know the problem of sending large files over a network requires more bandwidth for transferring the data which in turn makes the transmission also slower. The Compressing images work on mechanisms such as Discrete Cosine Transform (DCT), chroma subsampling and fractal compression. The problems associated with these methods include real-valued outputs, posterization and time consumption. Compression involves reducing the color space, chroma subsampling and fractal compression. An autoencoder is a type of artificial neural network used to learn efficient data coding in an unsupervised manner. It is a linear model capable of performing image compression. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal “noise”. Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input, hence its name. Fig. 1 shows the basic block diagram of our proposed system. An autoencoder consists of three components namely encoder, decoder, and latent space representation. The encoder and decoder models are used to enact the working of the autoencoder model. Paper is organized as follows. Section II describes the literature survey that our team has carried out on related journals which made us understand the process and the system in a better manner. Section III shows the algorithms that we have used to implement the proposed system. Section IV presents the implementation of the proposal by using relevant technologies. Section V shows the experimental results and the analysis that has been carried out using the results obtained. Finally, Section VI presents the conclusion.

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

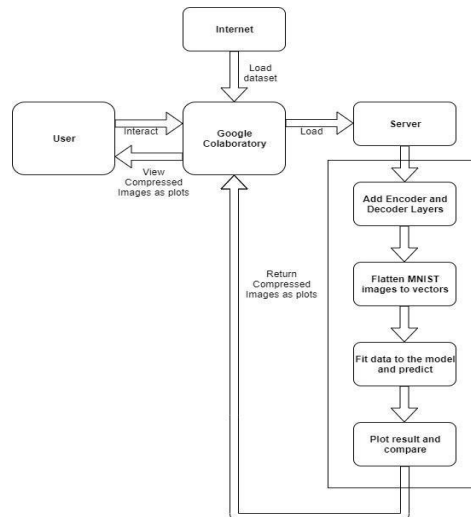


Fig. 1. Basic block diagram of Proposed System

II. RELATED WORKS

1. Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang (2018) presented a novel adversarial graph framework for graph data which encodes the topological structure and node content in a graph to a compact representation, on which a decoder is trained to reconstruct the graph structure. A graph convolutional autoencoder algorithm is proposed where the discriminator is updated with its stochastic gradient.

2. Daiki Kimura (2018) presented a generative model using autoencoders that is used for compressing a large number of samples involved in training the neural network model. Three different types are conducted in this study namely: cart-pole game, an Atari game which is implemented in the arcade learning environment, and an interactive game on an actual robot in the real environment.

3. Jinghui Chenm, Saket Sathe, Charu Aggarwal and Deepak Turaga (2016) presented RandNet, which stands for Randomized Neural Network for Outlier Detection is designed by varying the connectivity structure of autoencoders and combining with an adaptive sampling method to decrease the noise and to optimize the process of outlier detection. Randomized connection dropping is done to convert fully connected autoencoder layers to randomly connected autoencoder layers.

4. Lei Le, Andrew Patterson and Martha White (2018) presented a supervised linear autoencoder that is designed such that it predicts targets and inputs (reconstruction) and includes reconstruction error which proves uniform stability. A comparison between normal and autoencoder neural networks is done based on the number of hidden units and activation functions. The authors empirically demonstrated that adding reconstruction error never harms performance.

These references helped us to gain more insights about autoencoder neural networks and image compression and made us gain a clear idea about the algorithms, loss functions, and optimizers required to build the system.

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

III. ALGORITHM

A) AutoEncoding algorithm

The autoencoding algorithm is what makes an autoencoder neural network different from others. Autoencoding is a data compression algorithm which makes a feature extraction from the input and stores it into the middle layer as code (or) latent space representation. The latent space representation is then reconstructed into the output of the neural network.

B) Feature Extraction

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these data sets is a large number of variables that require a lot of computing resources to process.

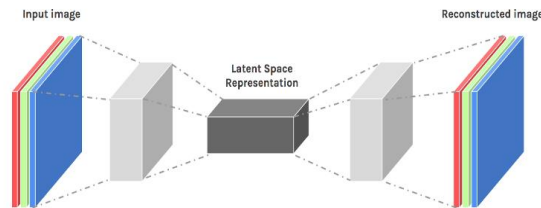


Fig. 2. Architecture of an Autoencoder

IV. IMPLEMENTATION

A) Preparation of Dataset

The dataset we used is MNIST (Modified National Institute of Standards and Technology) handwritten Digits dataset. The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. The MNIST handwritten digits dataset is widely used in the field of image processing and it is available in the keras framework.



Fig. 3. Screenshot of MNIST Dataset

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

B) Building necessary models

Three models namely autoencoder, encoder and decoder are created. The models are built using Keras framework. All the three models are Dense models.

Dense implements the operation: **Output = activation (dot (input, kernel) + bias)**, where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer.

Activation functions are really important for an Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable. They introduce non-linear properties to our Network. Their main purpose is to convert an input signal of a node in an ANN to an output signal. That output signal now is used as an input in the next layer in the stack. In the context of artificial neural networks, the rectifier is an activation function defined as the positive part of its argument:

$$f(x) = x^+ = \max(0, x),$$

Where x is the input to a neuron. This is also known as a ramp function.

A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve. A common example of a sigmoid function is the logistic function shown in the first figure and defined by the formula:

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}.$$

The layers for the three models are designed in such a way that it supports the architecture of an autoencoder neural network.

Layer	Model	Dimensions	Input
input_img	autoencoder, encoder	784 (28x28)	-
encoded	encoder, decoder	196 (code size)	input_img
decoded	autoencoder, decoder	784 (28x28)	encoded

Table.1. Specification of layers with input and output

C) Flattening and normalizing data

The flattening and normalization of data is the data preprocessing phase where the data from the MNIST dataset is made analysis-ready. Data flattening refers to the process of converting the shape of the input image (2D pixel array) to one-dimensional vectors. The purpose of data flattening is to make the data feed able into the neural network. The process consists of reshaping the input images (28x28) to a vector by mentioning the dimension of the vector (784), which will be the first layer of the neural network. Data normalization refers to the process of converting the range of the values of the one-dimensional vectors to a range that will enable easy processing when fed into the neural network.

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

The range of the pixel values will be from 0 to 255. These values are mapped onto a range of 0 to 1 which can enable effective computation and propagation of values through the neurons of the autoencoder neural network.

D) Fit and predict models

Compiling a model is used to define the optimizer, loss and metrics corresponding to the model. The autoencoder model is compiled to define the loss function, optimizer and other metrics. A loss function or cost function is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event. Machines learn by means of a loss function. It's a method of evaluating how well specific algorithms model the given data. The loss function we use is "binary_crossentropy". Binary Cross Entropy loss functions are used in binary classifications. The formula for binary_crossentropy can be represented as,

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

Where, y is the label (1 for true and 0 for false) and p(y) is the predicted probability of the point being green for all the samples.

Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses. Optimization algorithms or strategies are responsible for reducing the losses and to provide the most accurate results possible.

The compiled model is now trained by fitting the MNIST data into the autoencoder model by passing the following parameters such as training data as MNIST, 60 epochs, 300 images per batch, shuffling enabled and a validation set as MNIST images. The model predicts function generates output predictions for the input samples, processing the samples in batches. It will return a NumPy array of predictions. It generates class probability predictions for the input samples batch by batch. It also returns a numpy array of probability predictions. For each epoch, metrics such as time taken for execution, model loss and step loss are recorded.



Fig.4. Cross-Entropy Loss for a model

E) Plot result and compare

E.1) Plotting Results

Two subplots are created. One for plotting original images and another one for plotting reconstructed images. 20 images among the images in the dataset are picked and are plotted along with their corresponding reconstructed images.

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

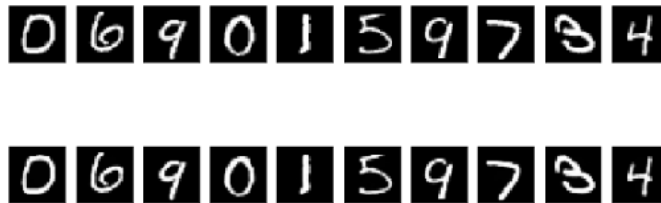


Fig.5. Final Implementation of Original and Reconstructed Images

The clarity of reconstructed images will vary according to the size of the intermediate latent space representation. The code layer will act as the storage that is used to store the most important features out of the image. Varying the optimizer also influences the reconstruction phase. This is because the way the optimization occurs during the reconstruction affects the features to move more towards the goal direction, eventually attaining something near to the global solution (original images). The effect of variation of code size and optimizers is tabulated as follows:





Code Size	Optimizer	Reconstructed Images
32	adadelta	
32	adam	
50	adam	
196	adam	

Table.2.Comparisson of clarity of the reconstructed images

E.2) Performance Comparison Model

In order to perform a statistical comparison between the original and reconstructed images, a sequential model is built and is evaluated against both original and reconstructed images. The comparison is done by the steps such as loading data, reshaping data, normalizing data, one hot encoding of data, building the model and evaluating the model. The comparison model is built using metrics such as sequential type of model, 2D Convolutional layer, MaxPooling2D as a pooling layer and 0.2 dropout factor. History module from keras.callbacks used to plot accuracy. Keys () which is a function present in keras. Callbacks module is used to get values of evaluated models. For the evaluated model, 2 different plots are generated namely loss and accuracy. For each plot, values are plotted for 2 different phases namely training and validation phases.

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

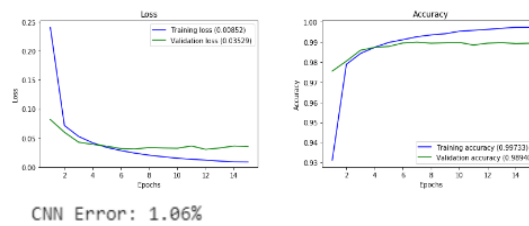


Fig.6. Loss and accuracy for the original images in the Convolutional Neural Network

V. RESULTS AND PERFORMANCE ANALYSIS

The original images are loaded from the dataset. The reconstructed images are generated by varying the code size, loss function, epochs and optimizer. These are the factors that influence the clarity of the reconstructed images. Firstly, it can be seen that the variation of the code size of the latent space representation directly affects the clarity of the image. This statement can be interpreted in such a way that the feature extraction process would require an optimal amount of code size to store its features, so that the reconstruction process would result in better image clarity. Secondly, it is evident that the loss function affects the clarity of the image. This is because, the more efficient the loss function gets near to its goal state, the better the model extracts the features of the image. The loss function, 'binary_crossentropy' serves the purpose of the proposed system. Thirdly, we can say that the effect of optimizer in the clarity of the reconstructed images is fair enough, as the better optimizer will help the model to recover from its loss by doing better back propagation and going to the more efficient parents, thereby minimizing the number of steps to go to the global optimal solution. We used 'adam' optimizer to obtain the best results for our proposed system.

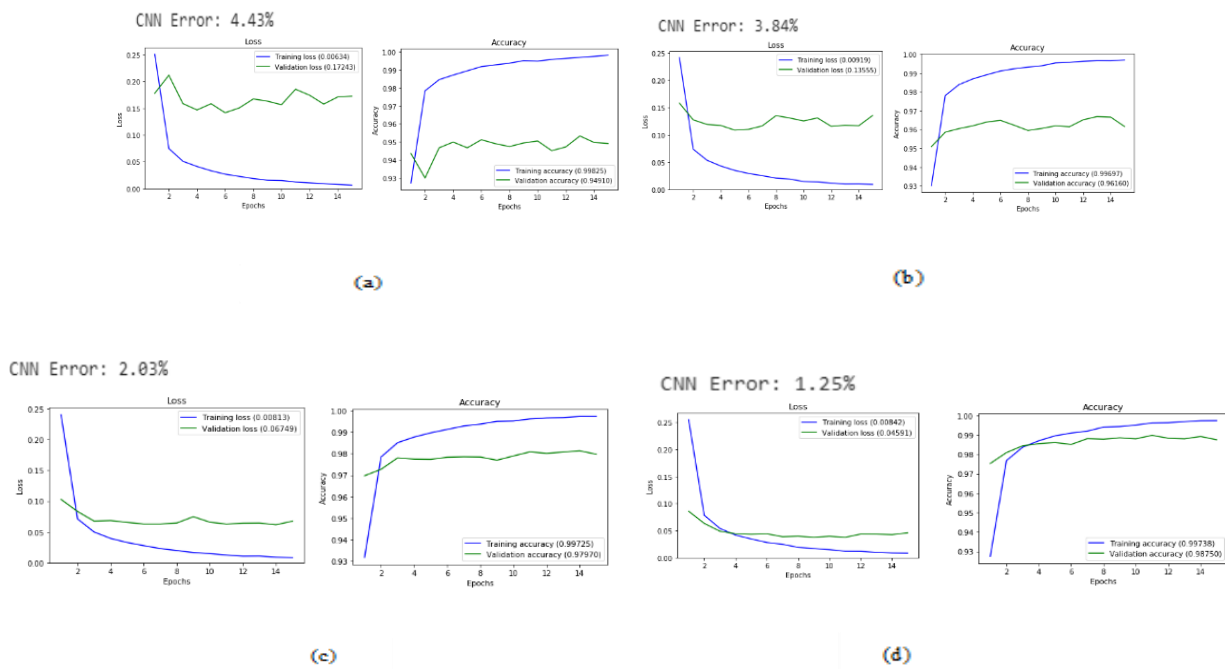


Fig.7. Loss and accuracy for reconstructed images with varying code size and optimizer types - (a) Code Size: 32, Optimizer: Adadelata (b) Code Size: 32, Optimizer: Adam (c) Code Size: 50, Optimizer: Adam (d) Code Size: 196, Optimizer: Adam

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

VI. CONCLUSION

This idea proposes an automated method for performing image compression using autoencoder neural networks. It also helps in finding the optimal parameters that lead to a good network structure. The network is trained with MNIST handwritten digits dataset. The data pre-processing phase is done and is fed into the neural network. The training and prediction is done, and the original and reconstructed images are plotted and compared. Separate convolution neural network is set up for performing statistical comparison between the original and reconstructed images. The metrics such as loss and accuracy are computed and compared.

ACKNOWLEDGEMENT

We thank faculties from the Department of Computer Science and Engineering, Loyola-ICAM College of Engineering and Technology for their valuable support and guidance.

REFERENCES

- [1] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang. "Adversarially Regularized Graph Autoencoder for Graph Embedding." *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. IJCAI*, 2018.
- [2] Daiki Kimura. "DAQN: Deep Auto-encoder and Q-Network" *IBM Research AI. arXiv*, 2018.
- [3] Jinghui Chenm, Saket Sathe, Charu Aggarwal and Deepak Turaga. "Outlier Detection with Autoencoder Ensembles." *University of Virginia.. IEEE*, 2016.
- [4] Lei Le, Andrew Patterson and Martha White. "Supervised autoencoders: Improving generalization performance with unsupervised regularizers." *32nd Conference on Neural Information Processing Systems. NeurIPS*, 2018.
- [5] Pierre Baldi. "Autoencoders, Unsupervised Learning, and Deep Architectures." *Workshop on Unsupervised and Transfer Learning. JMLR Workshop*, 2012.
- [6] JE.Laxmi Lydia Gogineni, Hima Bindu, Pasam Prudhvi Kiran, and Kollati Vijaya. "Artificial Way of Characterizing unsupervised Data using Auto-Encoders with Deep Learning Cluster Analysis." *International Journal of Recent Technology and Engineering. IJRTE*, 2019.
- [7] Sascha Lange and Martin Riedmiller. "Deep Auto-Encoder Neural Networks in Reinforcement Learning." *University of Freiburg, IEEE*, 2008.
- [8] Ji Feng and Zhi-Hua Zhou. "AutoEncoder by Forest." *The Thirty-Second AAAI Conference on Artificial Intelligence. AAAI*, 2018.