Autoencoders And It's Application Based on Neural Networks

ABHINANDHU A
Dept of Computer Science Engineering (AI)
Amrita School of Engineering ,
Amrita Vishwa Vidyapeetham,
Amritapuri ,
Kollam,Kerala,India
amenu4aie20002@am.students.amrita.edu

MADDALA H S M KRISHNA KARTHIK
Dept of Computer Science Engineering (AI)
Amrita School of Engineering ,
Amrita Vishwa Vidyapeetham,
Amritapuri ,
Kollam,Kerala,India
amenu4aie20046@am.students.amrita.edu

MARASANI JAYASURYA
Dept of Computer Science Engineering (AI)
Amrita School of Engineering ,
Amrita Vishwa Vidyapeetham,
Amritapuri ,
Kollam,Kerala,India
amenu4aie20048@am.students.amrita.edu

***Abstract*—Nowadays noisy data is still one of the most common machine learning problems for data scientists. Artificial Neural Networks are a set of algorithms that tries to recognize the patterns, relationships, and information from the data through the process which is inspired by and works like the human brain/biology. An autoencoder is a type of artificial neural network used to learn data encodings in an unsupervised manner. An autoencoder is one of techniques that enable computer systems to solve data related problems more efficiently. Many variants of autoencoder have been proposed by different researchers and have been successfully applied in many fields, such as computer vision, speech recognition and natural language processing.**

***Index Terms*—encoder,decoder,regularization,latent representation**

I. Introduction

Neural networks are a method of artificial intelligence that educates computers to process information in a way propelled by the human brain. It is a sort of machine learning called deep learning that employs interconnected hubs or neurons in a layered structure that are similar to the human brain. It creates a versatile framework that permits computers to memorise from their failures and make continuous enhancements.
An autoencoder comes under the category of Artificial Neural Network which is used to encode the input data and represent the encoded data in a bottleneck and decodes the data that is compressed using the encoder. In mathematical point of view it can be defined as:
In simple case, given one single hidden layer.The encoder takes the input X Rn and the function can be defined as h = g(WX+b).The decoder takes the input from the bottleneck and the function can be defined as x' = f(W'h+c) where h is encoder function, x' is decoder function , g is activation function like sigmoid function or relu function, b and c are biases, W and W' are weight matrices. Now after that these autoencoders are trained to minimise the reconstruction errors also known as loss. Mathematically can be defined as: loss = min $\||\|x' - x\||\|2$

.

The role of autoencoders in unsupervised machine learning and Neural Networks is very crucial. An Autoencoder takes the higher dimensional data as input and reduces it to lower dimensional data and learns how to represent that higher dimensional data in the form of lower dimensional data. That means an autoencoder typically does the job of PCA in a better and efficient manner. The idea of Autoencoders was introduced in the 1980s by Hinton and his group.

II. Architecture of Autoencoder

Autoencoder is made up of three primary parts. They are encoder ,bottle neck or the hidden layer and the decoder.The encoder is made up of a sequence of layers with fewer nodes that eventually reduce to a bottleneck representation.In bottleneck representation the input data is reduced but most of the characteristics of the input is maintained.The decoding part is a mirror copy of the encoding , except that the number of nodes in each layer grows, and the result is virtually identical.An autoencoder model that has been fine-tuned should be able to reconstitute the same input that was supplied in the first layer.

III. Types of Autoencoders

A. Under Complete Autoencoder:

This is the fundamental or simplest autoencoder where the number of nodes present in the hidden layer is less than the input layer. These are unsupervised learners because they do not take labels and the output it produces is the same as the input. This Autoencoder is used in dimensionality reduction, which is very efficient when it is compared with PCA because PCA can not form a lower dimensional hyperplane for non linear relationships. So Undercomplete Autoencoder has the advantage over PCA because it can learn non-linear manifolds also known as manifold learning. If we remove all non-linear activation functions and add only linear activation functions then it just looks similar to PCA. The loss function used in this autoencoder is known as reconstruction loss where it can be calculated using input data and the output data. There are many loss functions like L1 Norm, L2 Norm and MSE(Mean Square Error). Since the loss function does not possess the regularisation term the only way to stop this autoencoder from memorising the input data is to regularise the size of the bottleneck.

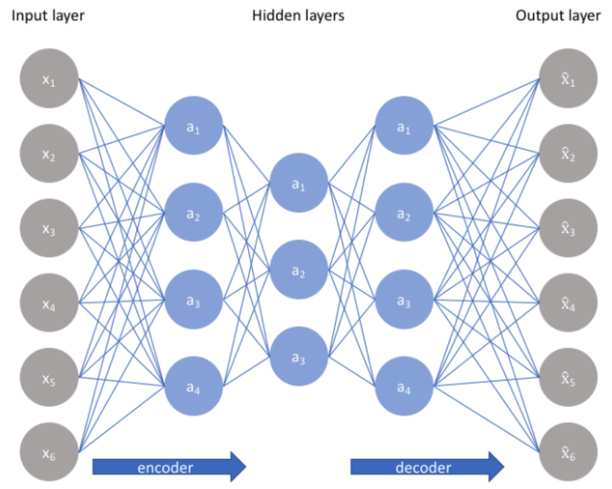

Fig. 1. Example of Under complete autoencoder

B. *Regularized Autoencoder:*

If hidden layer dimension is greater than or equal to the input layer dimension then a problem of memorising the input occurs which causes the failure of the autoencoder. These types of autoencoders can also be called overcomplete autoencoders. In these cases even a linear encoder and linear decoder can learn to copy the input to the output without learning anything useful about the data distribution. So in order to overcome this problem a regularization is added to the loss function which restricts the copying of input directly into the bottleneck or hidden layer. Since it is using a regularization term these autoencoders are known as Regularized Autoencoders. There are three types of regularized autoencoders. They are:

a.Sparse Autoencoders:

Generally regularization applies to weights of the network but in this type of autoencoders regularization is applied to penalise the activation functions within a layer. When we penalise the activation functions, it results in different outputs because different inputs activate different nodes in the network. In Fig-2 we can observe there are some nodes that are not activated in hidden layers. So This results in usage of each attribute for each hidden layer whereas Under Complete Autoencoder uses the whole network in order to compress all the features or attributes of the input. Here the network can easily extract the data but it can not memorise the whole input. This allows us to choose sparsity constraints which consider latent state regularization. There are two types of sparsity constraints. They are:

i. L1 Regularization:

Penalises the activation a layer h for observation i can be scaled by tuning the parameter .

$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i \left|a_i^{(h)}\right|$$

ii. KL-Divergence:

Uses cross entropy between average activation and desired activation.

$$\mathcal{L}(x, \hat{x}) + \sum_j KL\left(\rho \| \hat{\rho}_j\right)$$

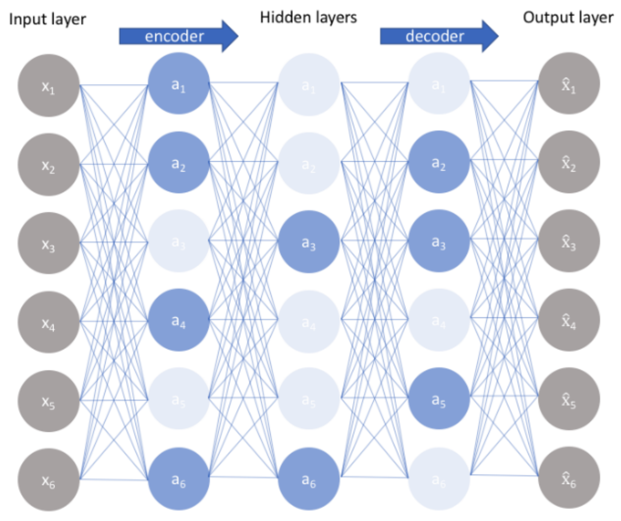

Fig. 2. Example of Sparse autoencoder

b.Denoising Autoencoders:

This type of autoencoders removes noise from the input and gives the output that is noise free data. In this type of autoencoder the model does not simply develop a mapping that memorises the input because the target data is not the same with the input. Here the input data is added with noises like gaussian noise, salt pepper noise etc which makes the input into a corrupt data. Now this corrupted data is mapped with hidden layers of the network and the model decodes the bottleneck to give output which has the minimum loss of data. The loss functions that can be used here are L1 or L2 regularization functions.

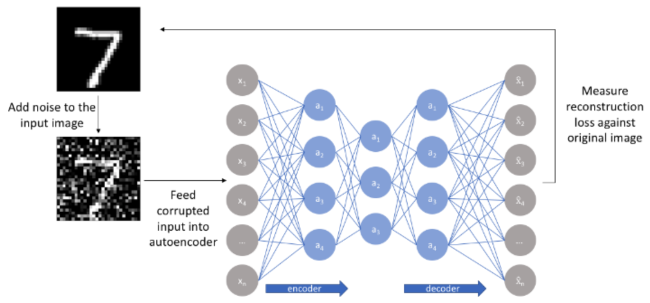

Fig. 3. Example of denoising autoencoder

c.Contractive Autoencoders:

In this type of autoencoders instead of penalising the activation functions, a different approach is followed. A new term is added to loss function inorder to regularise the loss function so that the model cannot copy the input data and and resconstucts the output data from it. Here, for the loss function frobenius norm of the jacobian matrix of the encoder activations with respect to the input is added in order to regularise the model.

C. *Variational Autoencoders:*

We must ensure that the latent space of our autoencoder is regular enough before we can use the decoder for generative purposes. Introducing explicit regularisation throughout the training phase is one viable technique for achieving such regularity. Variational autoencoders are autoencoders whose learning is regularised to prevent overfitting and guarantee that the hidden space has appropriate properties that allow for generative processes.

A variational autoencoder, like a regular autoencoder, is an architecture that combines an encoder as well as a decoder and therefore is trained to reduce the error function between both the encrypted information and the original data. However, we modify the encoding-decoding procedure somewhat to incorporate additional regularisation of something like the latent space: rather than encoding an input as a specific point, instead encode it as a distribution throughout the latent space. After that, the model is trained as follows:

1. The input is first encoded as a latent space distribution.

2. Then, from that distribution, a point from the embedding layer is sampled.

3.Third, the received point is decoded, allowing the reconstruction error to be calculated.

4.Finally, the network propagates the reconstruction mistake backwards.

IV. DATASET

Fashion-MNIST is a dataset that includes a 60,000-item training set and a 10,000-item test set. Each sample is a 28x28 grayscale image with a label from one of ten categories. There are a lot of handwritten digits in the original MNIST dataset. This dataset is adored by the AI/ML/Data Science community, who use it as a benchmark to test their algorithms. Researchers claim that they use the MNIST dataset in all of their studies, and if it doesn't work, it won't work for other datasets.

Each image has a height of 28 pixels and a width of 28 pixels, for a total of 784 pixels. Each pixel has a single pixel-value that indicates its lightness or darkness, with larger numbers signifying darker pixels. This pixel value is an integer ranging from 0 to 255. There are 785 columns in both the training and test data sets. The class labels make up the first column, which represents the article of clothing. The pixel values of the corresponding image are contained in the remaining columns.

The ultimate focus of this dataset is to build an autoencoder network that can perform autoencoder tasks such as image reconstruction and denoising. The dataset is taken from the Kaggle website.

V. APPLICATIONS OF AUTOENCODERS

There are various types of applications in autoencoders. They are as follow.

A. *Dimensionality Reduction*

Autoencoders build a system to convert the data's inherent structure into a lower-dimensional representation that is more efficient. It accomplishes this by minimising the reconstruction error through decoding and encoding strategies. The use of incomplete autoencoders for dimensionality reduction is common. These can also be used as pre-processing phase for dimensionality reduction since they can provide timely and efficient dimensionality reductions without sacrificing a lot of data. Furthermore, although PCA can only conduct linear dimensionality transformations, undercomplete autoencoders could accomplish group of non-dimensionality reductions.

B. *Image Denoising*

Real-world raw information is frequently noisy, and training a successful supervised model necessitates clean, noise-free data. To denoise the data, autoencoders can be utilised. Denoising autoencoders, such as the denoising autoencoder, can be used to produce efficient and precise image denoising. One of the most common applications is image denoising, in which autoencoders attempt to reconstruct a noiseless image from a chaotic input image.

Autoencoders, unlike traditional denoising methods, do not look for noise; instead, they learn a representation of the noisy input and retrieve the image from it. After that, the

representation is decompressed to provide a noise-free image. Denoising autoencoders can thus denoise complicated images that would otherwise be impossible to denoise using existing approaches.
The noisy input image is sent into autoencoder as input, and the noiseless result is reconstructed from the original estimate output by minimising the reconstruction loss (noiseless). The autoencoder weights can then be used to remove the noise the raw image once they've been trained.

C. Anomaly Detection

Anomaly detection can also be done with incomplete autoencoders.Take, for example, an autoencoder that was trained on a single dataset P. The autoencoder is expected to offer a minimal reconstruction loss for each image chosen for the training examples and to recreate the image.The autoencoder, on the other hand, cannot recreate any image that is not in the training sample since the latent characteristics are not tailored for the particular picture that the network has never seen.As a consequence, the outlier image has a large reconstruction loss and may be easily spotted as an anomaly using an appropriate threshold.
Another practical implementation of an autoencoder system is anomaly detection. A fraud detection model or any heavily imbalanced supervised job can be detected using an anomaly detection model. The objective is to educate autoencoders using only one class of data (majority class). This allows the network to rebuild the input with minimal or no reconstruction loss. When a sample of data from a different target class is fed into the autoencoder network, the reconstruction loss is much higher. A threshold value for reconstruction loss (abnormality score) can be set, with values higher than that indicating an anomaly.

D. Image Compression

An autoencoder network can also be used to compress images. The encoder network can receive the unfiltered input image and produce a compressed quantity of encoded data. By utilising a decoder network to rebuild the picture from the compressed encoding, the autoencoder hidden layers can be learned. In most cases, autoencoders aren't very good at compressing data; instead, simple compression algorithms perform effectively.
It is impossible for such an autoencoder to outperform fundamental algorithms like JPEG in image compression, and by being solely specific for a single sort of image, we can disprove this notion. As a result, autoencoders' data-specific trait makes compression of hard data impracticable. They can only be used with the information on which they have been trained, therefore generalisation necessitates a large amount of data.

VI. RESULTS

Here, some of the applications like image reconstruction and image denoising were taken and applied respective autoencoders and generated the output. For image reconstruction, we used a 3 layer denser under complete autoencoder and reconstructed the images. We can see that in the Fig-4 the output images are almost close to the input images.For image denoising, we used convolution 2d layer and used max pooling also done for each layer that model gives us the denoising autoencoder. Initially, salt and pepper noise is added to the original images and using the denoising autoencoder model generates the output back from the bottleneck. In the Fig-5 we can see the original image, and denoised images. The denoised images are pretty close to the original images.

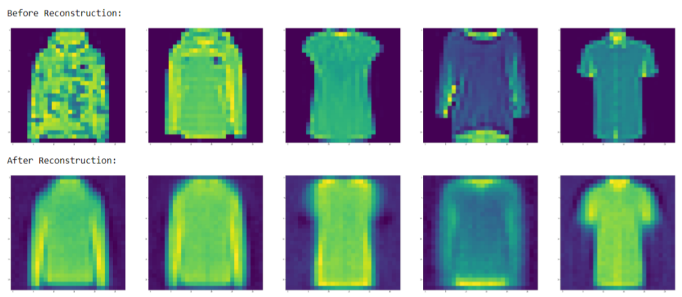

Fig. 4. Input and generated output for Image Reconstruction

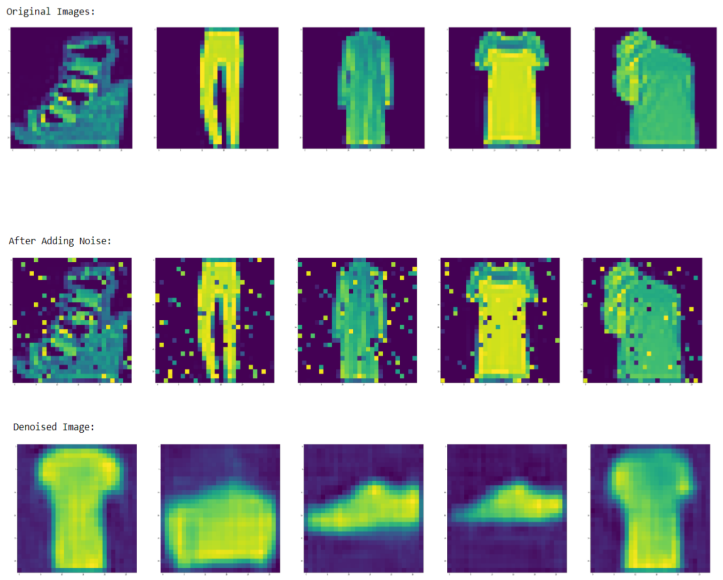

Fig. 5. Input image ,noise added image and generated output for Image denoising

VII. CONCLUSION

Autoencoders are a type of neural network architecture that has a wide range of applications in computer vision, natural language processing, and other fields. This paper provides a brief overview of autoencoders and their various types and applications. In this paper, an undercomplete autoencoder is used to implement an image reconstruction application, and a denoising autoencoder is used to implement an image denoising application. Autoencoders remain a good alternative for a range of situations, despite the fact that other classes of models are now employed for representation learning.

References

[1] J. Zhai, S. Zhang, J. Chen and Q. He, ”Autoencoder and Its Various Variants,” 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2018, pp. 415-419, doi: 10.1109/SMC.2018.00080.

[2] Yuan, F.-N Zhang, L. Shi, J.-T Xia, X. Li, G.. (2019). Theories and Applications of Auto-Encoder Neural Networks: A Literature Survey. Jisuanji Xuebao/Chinese Journal of Computers. 42. 203-230. 10.11897/SP.J.1016.2019.00203.

[3] Z. Chen, C. K. Yeo, B. S. Lee and C. T. Lau, ”Autoencoder-based network anomaly detection,” 2018 Wireless Telecommunications Symposium (WTS), 2018, pp. 1-5, doi: 10.1109/WTS.2018.8363930.

[4] C. -H. Chang, ”Managing Credit Card Fraud Risk by Autoencoders : (ICPAI2020),” 2020 International Conference on Pervasive Artificial Intelligence (ICPAI), 2020, pp. 118-122, doi: 10.1109/ICPAI51961.2020.00029.

[5] Pinaya, Walter Hugo Lopez, Sandra Vieira, Rafael Garcia-Dias, and Andrea Mechelli. ”Autoencoders.” In Machine learning, pp. 193-208. Academic Press, 2020.

[6] Doersch, Carl. ”Tutorial on variational autoencoders.” arXiv preprint arXiv:1606.05908 (2016).

[7] Baldi, Pierre. ”Autoencoders, unsupervised learning, and deep architectures.” In Proceedings of ICML workshop on unsupervised and transfer learning, pp. 37-49. JMLR Workshop and Conference Proceedings, 2012.