Dimensionality Reduction: PCA vs Autoencoders

Ojasa Chitre
Department of Computer Engineering
Sardar Patel Institute of Technology
Mumbai, India
ojasa.chitre@spit.ac.in

Arka Haldi
Department of Computer Engineering
Sardar Patel Institute of Technology
Mumbai, India
arka.haldi@spit.ac.in

***Abstract*—Dimensionality reduction has become an essential part of data processing. Overfitting and long computational times can result from having too many dimensions. There are numerous known methods for reducing dimensionality. In this paper we are discussing Principal Component Analysis(PCA) and Auto Encoders(AE) and comparing their ability to perform Dimensionality Reduction. PCA applies linear transformation and AE uses non-linear tranformation. In this paper, we look at how reducing dimensions of image data to various number of dimensions using affects the result with these two methods.**

***Index Terms*—Principal Component Analysis, Auto Encoders, Dimensionality Reduction, Linear transformation, Non-linear transformation**

I. Introduction

Data grows by the day. More and more advanced and large datasets are being formed. The dimensions of a data are the number of features that are used to describe that data. Some of the risks of a larger data are:

1) **Overfitting:** Larger data can lead to overfitting of a model. Overfitting refers to the scenalrio where a model is too specific to its training data and does not generalise well for new incoming data. This can lead to poorer performance of the model.
2) **More similarity in data:** With lesser data there is a higher chance that there is a proper distinction between the data-points, leading to better distinction between different classes of data. Since the hyperdimensional space remains the same more distinct data points are packed in the same space which means there will be lesser distance/distinction between the different classes in the data.
3) **Larger Compute time:** There is a simple correlation between the size of data and the amount of time needed to compute it. Each and every algorithm's compute time is dependent on the size of data that is being given to it. But the world of technology requires speed of operation. Hence, it becomes imperative to try and reduce the amount of data present.

To counter this Dimensionality problem, the technique of Dimensionality reduction is used. It basically emplys various methods to reduce the number of features used in a dataset in order to reduce the feature space. In theory we can do this by Feature Selection or Feature Transformation. In this tutorial we will be focussing on Feature Transformation, specifically on PCA and Autoencoders. Feature Transformation entails projecting high dimensional features to a lower dimensional level.

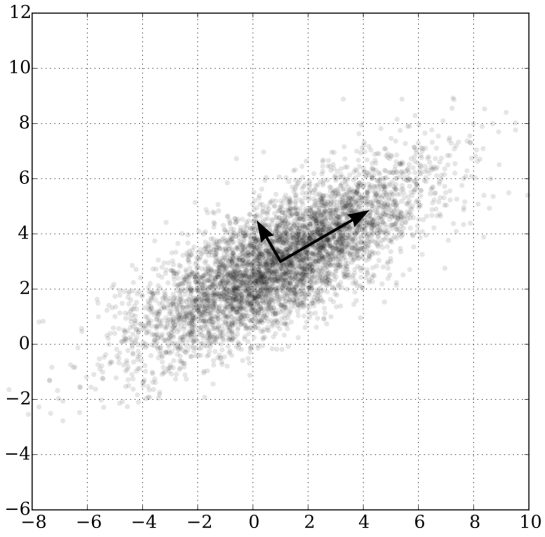

Fig. 1. Principal Component Analysis

Principal Component Analysis(PCA) is the technique of projecting data onto a direction with higher variance. Mathmatically the directions that have high variance are orthogonal to one another meaning they have nearly 0 correlation. PCA is a linear transformation of data. In more mathematical terms it is a line/plane that is a least square approximation of a the data with maximum variance. Autoencoders are artificial neural network that essentially compresses the data. They reduce dimensions by stacking multiple non-linear layers. The data is reduced to a lower dimensional latent space.

These two methods help in reducing the dimensions of the dataset, while preserving the properties of the full dataset. We will see two capabilities of PCA and AutoEncoders, firstly it's the ability to reduce the dimensions so as to preserve the properties of the original features and still give considerable accuracy on similar tasks; and secondly it's the ability to be able to reconstruct the original features of the data, and hence

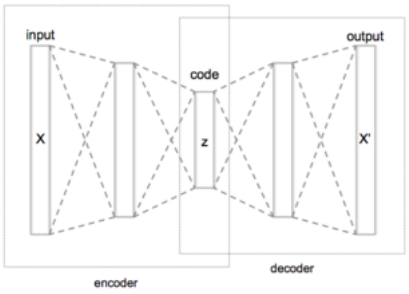

Fig. 2. AutoEncoder Structure

it's robustness in representation.

The following section is a description of all the different papers referred for the purpose of this research. It includes the methodologies, the dataset and the evaluation parameters used in the research papers. The section after that elaborates about a potential solution taking inspiration from the research papers.

II. Dataset Used

MNIST database: The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset. It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.The task is to classify a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9, inclusively. It is a widely used and deeply understood dataset and, for the most part, is "solved." Top-performing models are deep learning convolutional neural networks that achieve a classification accuracy of above 99%, with an error rate between 0.4 %and 0.2% on the hold out test dataset.

III. Literature Survey

There have been research papers that establish the effectiveness of autoencoders by comparing with PCA and other methods, and even papers showing the capability of advanced autoencoders as Variational AutoEncoder performing better than both PCA and autoencoders. In fact, in current research, Variational autoencoders are used for generating datasets [1], as they are able to learn the non-linear reduced features of sound, images and other forms of unstructured data. The most cutting edge work done in this space is feature transfer by using a different Decoder on an image's compressed format to get results like DeepFakes, that can transfer the features of one face to another, and can generate real-looking fake videos [4].

The Feature Representation Ability of Variational AutoEncoder Variational autoencoders play an important role in image feature extraction, text generation, and text compression. This paper investigates the representation ability and stability of variational autoencoders for image features from the standpoint of feature expression. The representation ability of the variational autoencoder is evaluated and compared using the image classification task to traditional methods of dimensionality reduction; principal components analysis, autoencoder. [1]

To perform handwriting recognition, this paper used a high performance autoencoder and Principal Component Analysis (PCA). The non-linear approach improves results for this type of data. The implementation entails the deployment of a neural network, as well as the use of an autoencoder and PCA to perform data compression and classification. [2]

This paper proposes a Relation Autoencoder model that takes both data features and their relationships into account. They also make it compatible with other major autoencoder models, such as Sparse Autoencoder, Denoising Autoencoder, and Variational Autoencoder. These autoencoder models are tested on various datasets, and the experimental results show that considering data relationships can generate more robust features with lower construction loss and then lower error rate in further classification when compared to other autoencoder variants. [3]

"Deepfake" is one of the most recent deep learning-powered applications to emerge. Deepfake algorithms can generate fake images and videos that are indistinguishable from real ones. This paper is a survey of algorithms used to create deep fakes and methods proposed in the literature to detect deepfakes to date. Extensive discussions are provided on the challenges, research trends, and directions associated with deepfake technologies and the technological components that enable them to function, such as AutoEncoders, CNNs, and GANs. [4]

Considering PCA and autoencoders use as feature generation and selection, this paper has compared these methods for face recognition. The non-linearity introduced by autoencoders is beneficial for this purpose. The numerical results obtained on a multiclass base of faces demonstrated the superiority of the autoencoding principle, particularly when the number of recognised classes is very large. [5]

IV. Methodology

In order to compare Autoencoders to PCA we have applied logistic regression on mnist dataset.

A. Data Preprocessing

The images are typically of the shape (x,y,z). Normally we look at data that is of shape (x,y) where y is the number of dimensions for x data points. Hence, we have reshaped the data to be of the form (x,y) where y incorporates the initial y and z. In our train data the initial shape was (60000, 28, 28). We reshaped it to (60000, 784). We can now use this data for our PCA vs Autoencoder comparison.

B. Base Model

In any type of comparison we need a base or control calue that we can compare our experiment to. In our case our base model is one that takes in the data with dimensions 784. Here we train our Logistic Regression on this complete data and

then apply calculate the score of the model. In case of the base model we get an accuracy of 92.58%. Since more data means more details we postulate that the accuracy will be best for the data with most dimensions. We would also say that with increasing set of dimensions the accuracy will increase.

C. Autoencoder

Experimental setup of AutoEncoders, had 2 parts of evaluation each for the cases of feature reduction to sizes 32, 64, 128. The first part of evaluation is using the reduced features obtained by the encoder of AutoEncoder, and applying Logistic Regression to predict it's class. The second part of the evaluation is using the reduced features and reconstructing back the original data, to check the robustness of feature reduction and data representation. This is repeated for each of the cases of 32, 64, 128 autoencoders discussed below.

The initial naive configuration of AutoEncoder used for feature reduction to size 32 was as shown in the figure 3:

Layer (type)	Output Shape	Param #
input_9 (InputLayer)	[(None, 784)]	0
dense_40 (Dense)	(None, 32)	25120
dense_41 (Dense)	(None, 784)	25872

Total params: 50,992
Trainable params: 50,992
Non-trainable params: 0

Fig. 3. AutoEncoder 32 naive

This configuration for AutoEncoder is to show the ineffectiveness of insufficient hidden layers in autoencoder, with just the bottleneck layer in middle of input and output layer. This is further discussed in the results section along with the PCA results.

The next configuration is a deeper AutoEncoder used for feature reduction to size 32 was as shown in the figure 4:

Layer (type)	Output Shape	Param #
input_10 (InputLayer)	[(None, 784)]	0
dense_42 (Dense)	(None, 128)	100480
dense_43 (Dense)	(None, 64)	8256
dense_44 (Dense)	(None, 32)	2080
dense_45 (Dense)	(None, 64)	2112
dense_46 (Dense)	(None, 128)	8320
dense_47 (Dense)	(None, 784)	101136

Total params: 222,384
Trainable params: 222,384
Non-trainable params: 0

Fig. 4. AutoEncoder 32

The next configuration is an AutoEncoder used for feature reduction to size 64 was as shown in the figure 5:

The next configuration is an AutoEncoder used for feature reduction to size 128 was as shown in the figure 6:

Layer (type)	Output Shape	Param #
input_11 (InputLayer)	[(None, 784)]	0
dense_48 (Dense)	(None, 256)	200960
dense_49 (Dense)	(None, 128)	32896
dense_50 (Dense)	(None, 64)	8256
dense_51 (Dense)	(None, 128)	8320
dense_52 (Dense)	(None, 256)	33024
dense_53 (Dense)	(None, 784)	201488

Total params: 484,944
Trainable params: 484,944
Non-trainable params: 0

Fig. 5. AutoEncoder 64

Layer (type)	Output Shape	Param #
input_12 (InputLayer)	[(None, 784)]	0
dense_54 (Dense)	(None, 512)	401920
dense_55 (Dense)	(None, 216)	110808
dense_56 (Dense)	(None, 128)	27776
dense_57 (Dense)	(None, 216)	27864
dense_58 (Dense)	(None, 512)	111104
dense_59 (Dense)	(None, 784)	402192

Total params: 1,081,664
Trainable params: 1,081,664
Non-trainable params: 0

Fig. 6. AutoEncoder 128

D. Principal Component Analysis

PCA employs linear transformation onto orthogonal axes. Hence unlike AE where we have 1 layer AE which is essentially linear and multi layer ones that are non-linear, here we only have linear transformations.

The experimental setup of Principal Component Analysis, had 2 parts each for the cases of feature reduction to sizes 32, 64, 128. The first part of evaluation is using the reduced features obtained by the using PCA, and applying Logistic Regression to predict it's class. The second part of the evaluation is using the reduced features and reconstructing back the original data, to check the robustness of feature reduction and data representation.

V. Results

In this section, we first discuss the exact results and then a higher level analysis of the accuracy and reconstruction losses to compare the performance of the PCA and AutoEncoder.

- We used the 32 dimensions data to perform logistic regression and image reconstruction using naive AutoEncoder implementation. In case of the Logistic regression model we got an accuracy of 72.8%. In Fig. 7 we can see the reconstruction using AE with output features as 32. It gave a RMSE loss of 0.06444269158650837.
- We used the 32 dimensions data to perform logistic regression and image reconstruction. In case of the Lo-

Fig. 7. AE

gistic regression model we got an accuracy of 89.84%. In Fig. 8 we can see the reconstruction using AE with output features as 32. It gave a RMSE loss of 0.04336553782862171.

Fig. 8. AE 32

- We used the 64 dimensions data to perform logistic regression and image reconstruction. In case of the Logistic regression model we got an accuracy of 91.54%. In Fig. 9 we can see the reconstruction using AE with output features as 64. It gave a RMSE loss of 0.03780648082324084.

Fig. 9. AE 64

- We used the 128 dimensions data to perform logistic regression and image reconstruction. In case of the Logistic regression model we got an accuracy of 91.95%. In Fig. 10 we can see the reconstruction using AE with output features as 128. It gave a RMSE loss of 0.03780648082324084.

Fig. 10. AE 128

- We used the 32 dimensions data to perform logistic regression and image reconstruction. In case of the Logistic regression model we got an accuracy of 90.06%. In Fig. 11 we can see the reconstruction using PCA with output features as 32. It gave a RMSE loss of 0.10379093122522375.
- We used the 64 dimensions data to perform logistic regression and image reconstruction. In case of the Logistic regression model we got an accuracy of 91.7%.

Fig. 11. PCA 32

In Fig. 12 we can see the reconstruction using PCA with output features as 32. It gave a RMSE loss of 0.07823686092512176.

Fig. 12. PCA 64

- We used the 128 dimensions data to perform logistic regression and image reconstruction. In case of the Logistic regression model we got an accuracy of 92.17%. In Fig. 13 we can see the reconstruction using PCA with output features as 32. It gave a RMSE loss of 0.05481270512260481.

Fig. 13. PCA 128

The final results for the accuracy of prediction, in all of the experiments held, are as shown in figure 14. We can see that the accuracy/precision/recall/f1 score of the Logistic Regressor for all the features is the maximum, as expected since it involves all the features/ pixels.

Looking at the results of the PCA and AE, we observe that:

- The results are very close, and AE performs as good as PCA in predicting the classes; in each of the cases of 32, 64 or 128 layers/features that it is reduced to.
- Individual entries are indicated with a black dot, a so-called bullet.
- As expected, we see that the naive implementation of AutoEncoder gives us the least accuracy since it has linear activations as far as the values of the pixels are concerned; since it takes the value between 0 and 1.
- Autoencoders gives an almost identical/better accuracy/precision/recall and f1 score, with PCA performing slightly better than the AutoEncoders, but their scores show that they have a very high accuracy with very less misclassification rate.
- Also, we note that only using the first 2 Principal components in PCA, all the metrics gave a very poor performance, close to 44%. We noticed that on increasing the number of Principal Components to 32 the performance metrics go up to 90%, which means an improvement of over 46%.

On analysis of the Reconstruction loss, we find that in every case AutoEncoder performs much better than the PCA coun-

Sno.	Method	Accuracy Score	Recall Score	Precision Score	F1 Score
1	784 (all)features	0.9258	0.9258	0.9258	0.9258
2	32 features using AutoEncoders	0.7046	0.7046	0.7046	0.7046
3	32 features using Deeper AutoEncoders	0.9011	0.9011	0.9011	0.9011
4	64 features using Deeper AutoEncoder	0.9128	0.9128	0.9128	0.9128
5	128 features using Deeper AutoEncoder	0.9205	0.9205	0.9205	0.9205
6	2 features using PCA	0.4462	0.4462	0.4462	0.4462
6	32 features using PCA	0.9008	0.9008	0.9008	0.9008
7	64 features using PCA	0.9172	0.9172	0.9172	0.9171999999999999
8	128 features using PCA	0.9229	0.9229	0.9229	0.9229

Fig. 14. Comparsion of Performance using different metrics

terpart. We see that there's atmost a difference of 0.4 between the performances of the reconstruction. This shows that the autoencoders are able to understand the non-linear patterns in the data and also reconstruct it with significantly lesser loss, showing it's robustness to reduce features of the data. As the number of features to reduce to; were increased, their accuracy went up and approached the accuracy of applying logistic regression on all the 784 features. But at the same time, we noticed a significant reduction in the reconstruction RMSE loss using AutoEncoders, than PCA.

Sno.	Method	RMSE loss
1	32 features using AutoEncoders	0.31155957453395033
2	32 features using Deeper AutoEncoders	0.06444269158650837
3	64 features using Deeper AutoEncoder	0.04336553782862171
4	128 features using Deeper AutoEncoder	0.03780648082324084
5	32 features using PCA	0.10379093122522375
6	64 features using PCA	0.07823686092512176
7	128 features using PCA	0.05481270512260481

Fig. 15. RMSE loss

VI. Conclusion and Future Scope

Our work compared the feature reduction and reconstruction capacity to demonstrate the flexibility and applicability of autoencoders in feature reduction/representation. We also confirmed our hypothesis that PCA can effectively capture the latent features of the data, just like AutoEncoders, and thus compared their results on various sizes of reduced dimensions, to note the similarities and differences in the results of PCA and AutoEncoders in feature reduction, and image compression. This particular setup used the MNIST dataset to draw it's conclusions about the performance of autoencoders. In the future, we can use different datasets and test the reduced features for robustness by not just using the representation to only be able to predict/represent the target class, but other functions that the reduced data is expected to represent. Since we can use the data in only one application, the scope of measurement of flexibility and robustness of AutoEncoder vs PCA is limited, and is the future scope of this project.

Also the AutoEncoders have been found to be very flexible, and is applicable to many different domains, the latest of which is adoption in deepfake technology [4]. There are much powerful autoencoders present in literature like Variational AutoEncoders [1] & Relation Autoencoders [3], and this paper successfully demonstrates that even the most basic implementation of autoencoders can compete well against established methods of feature reduction like PCA, and provide more robustness and generalisation, with the benefit of non-linear representations from the dataset.

References

[1] C. Dong, T. Xue and C. Wang, "The Feature Representation Ability of Variational AutoEncoder," 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), 2018, pp. 680-684, doi: 10.1109/DSC.2018.00108.

[2] J. Almotiri, K. Elleithy and A. Elleithy, "Comparison of autoencoder and Principal Component Analysis followed by neural network for e-learning using handwritten recognition," 2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 2017, pp. 1-5, doi: 10.1109/LISAT.2017.8001963.

[3] Meng, Q., Catchpoole, D., Skillicom, D., & Kennedy, P. (2017). "Relational autoencoder for feature extraction." 2017 International Joint Conference on Neural Networks (IJCNN).

[4] Nguyen, Thanh Thi, Quoc Viet Hung Nguyen, Cuong M. Nguyen, Dung Nguyen, Duc Thanh Nguyen, and Saeid Nahavandi. "Deep learning for deepfakes creation and detection: A survey" (2019).

[5] K. Siwek and S. Osowski, "Autoencoder versus PCA in face recognition," 2017 18th International Conference on Computational Problems of Electrical Engineering (CPEE), 2017, pp. 1-4, doi: 10.1109/CPEE.2017.8093043.

[6] Ladjal, S., Newson, A., & Pham, C.H.. (2019). "A PCA-like Autoencoder" 2019 Computer Vision and Pattern Recognition arXiv ,doi: https://doi.org/10.48550/arXiv.1904.01277

[7] Almotiri, Jasem & Elleithy, Khaled Elleithy, Abdelrahman. (2017). Comparison of autoencoder and Principal Component Analysis followed by neural network for e-learning using handwritten recognition. 1-5. 10.1109/LISAT.2017.8001963.

[8] Lin Y, Lee C, Chen C. 2022. "Robustness of autoencoders for establishing psychometric properties based on small sample sizes: results from a Monte Carlo simulation study and a sports fan curiosity study." PeerJ Computer Science 8:e782 https://doi.org/10.7717/peerj-cs.782

[9] Rolinek, M., Zietlow, D., Martius, G.. (2018). "Variational Autoencoders Pursue PCA Directions (by Accident)." 2018 Machine Learning arXiv ,doi: https://doi.org/10.48550/arXiv.1812.06775

[10] Q. Fournier and D. Aloise, "Empirical Comparison between Autoencoders and Traditional Dimensionality Reduction Methods," 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), 2019, pp. 211-214, doi: 10.1109/AIKE.2019.00044.

[11] K. Siwek and S. Osowski, "Autoencoder versus PCA in face recognition," 2017 18th International Conference on Computational Problems of Electrical Engineering (CPEE), 2017, pp. 1-4, doi: 10.1109/CPEE.2017.8093043.