Final Project report of the module Python for Engineering Data Analysis - from Machine Learning to Visualization

‘Convolutional Autoencoders vs principal component analysis (PCA) as feature extraction methods for face recognition ’

Supervised by M.Sc. Felix Mayr
Associate Professorship of Simulation of
Nanosystems for Energy Conversion

Submitted by Malek Ben Alaya
Student ID: 03712504

Submitted on Munich, 21.03.2021

Table of contents:

1 - Introduction :

Face recognition requires the classification of input data of high dimensionality such as videos or images. In order to make the process computationally more affordable, methods for dimensionality reduction exist. Their objective is to find a proper projection method that maps data from high feature space to low feature space.

In this work I will be comparing the results of two Unsupervised Learning methods used for dimensionality reduction, namely, Principal Component Analysis (PCA) and Autoencoders. Both methods are applied to the same Face Database to encode the input images. The encoded output would then serve as the input attributes for the final recognizing system.

2 - Methods:

PCA is a linear method used for feature extraction. Because it only incorporates linear transformations, PCA is simple and relatively computationally fast. In contrast, Autoencoders have the ability to include nonlinear transformations. This makes Autoencoders computationally more expensive but also a more powerful feature extraction method.

In this project, images are to be encoded, which is why a so-called convolutional Autoencoder (CAE) is recommended.

2.1 - Convolutional Autoencoder for feature extraction:

CAE's rely on the convolution operator, which allows filtering an input signal in order to extract some part of its content. In their traditional formulation, Autoencoders do not take into account the fact that a signal can be considered as a sum of other signals. Convolutional Autoencoders, instead, exploit this observation by the use of the convolution operator. They learn to encode the input in a set of simple signals and then try to reconstruct the input from these signals.[1] A convolutional Autoencoder uses different Convolutional and MaxPooling (Downsampling) layers for the encoding part. In the decoding part, it uses different Deconvolutional and Unpooling (UpSampling) layers to expand the hidden reduced state to the original dimensions [2]. Usually, the ReLU activation function is used for the hidden layers. Additionally, each convolutional layer is followed by a MaxPooling layer and each Unpooling layer is followed by a deconvolutional layer. The structure of an example CAE is shown in Fig. 1.

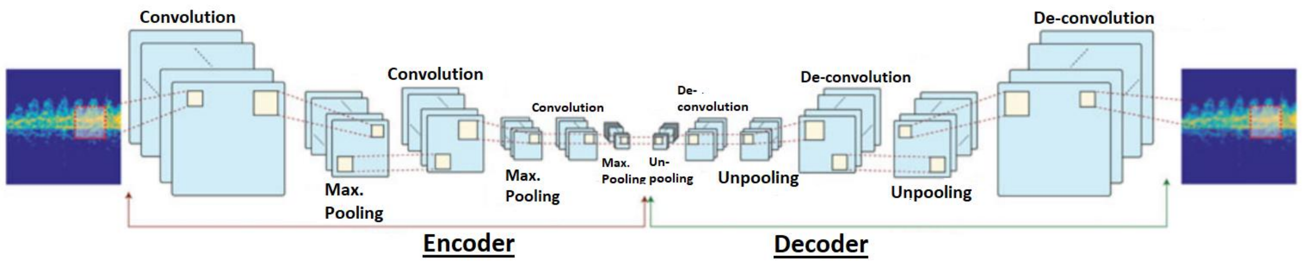


Fig. 1: Structure of an example CAE.

3 - Numerical Experiments:

3.1 - Database:

The Database used in this work is a subset of "the Extended Yale Face Database B" [3]. It consists of 2432 grayscale images of 38 human subjects under the same pose in 64 illumination conditions. The images are manually aligned, cropped, and then re-sized to 168x192 Pixel images [4]. Fig. 2 shows a set of images for 2 different persons taken from the Database.

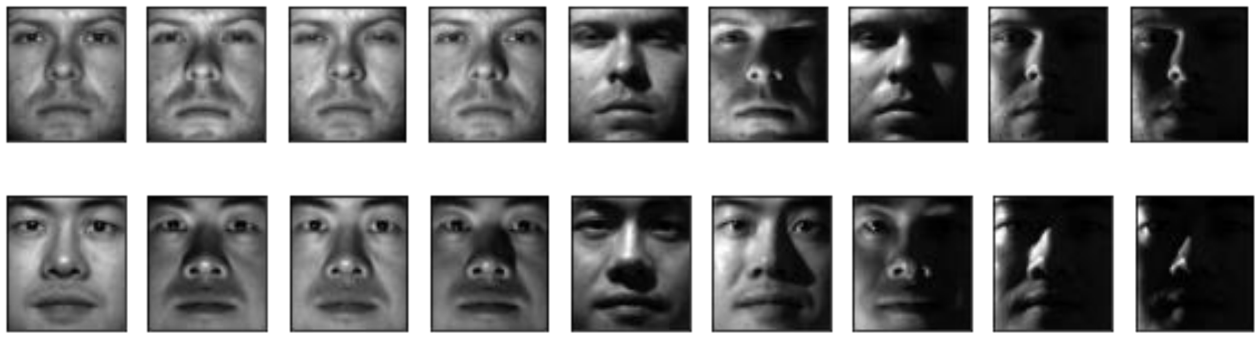


Fig. 2: Exemplary 9 representatives of faces for 2 different persons under different lighting conditions taken from the Extended Yale Face Database B.

3.2 - Implementation and Results:

In this work, neural networks are implemented using Keras[6], which uses Tensorflow as its backend [7]. PCA is computed using Scikit-learn [8].

Each images of the 38 individuals is loaded in a 3-dimensional array of shape (192,168,1), where 192 and 168 represent the height and the width of each image, respectively. These images are then randomly split into a training set and a test set. The training/test set ratio is: 0.75/0.25.

3.2.1 - PCA Implementation:

For the case of PCA each image initially loaded in a 3-D array has to be vectorized, i.e, stored in a vector consisting of 192*168=32256 attributes, where 32256 represents the number of pixels for every image. The number of components for PCA is chosen to be 300. After performing PCA on the training and test set we obtain new sets with reduced dimensionality, where each image is transformed to a vector of 300 attributes. This corresponds to a dimensionality reduction of: [1-(300/32256)]x100=99.07%.

In the next step we want to perform classification on the transformed test set. To achieve that, we first train a multilayer neural network on the transformed training set. The neural network used for classification consists of an input layer of size 300, followed by 2 fully connected layers of size 1000 and 'ReLu' activation function, which are followed by a dropout layer with rate=0.5. The dropout layer serves as a regularization technique used to prevent the neural network from overfitting. This is done by disabling randomly chosen neurons and their connections. The dropped neurons stay inactive during the feedforward and backpropagation training phases, thus forcing the network to learn different nonlinear combinations of features on each epoch.[5] The output layer is a ‘softmax’ layer of size 38, which corresponds to the number of individuals to be classified. This network architecture has a total number of trainable parameters of 1,340,038. The Optimizer used in Backprobapagation is stochastic gradient descent with momentum. The learning rate is alpha=0.01 and the Momentum is m=0.9. The loss function is 'sparse_categorical_crossentropy' which is less time-consuming compared to ‘categorical_crossentropy’. The network is trained on 40 epochs with a minibatch size of 20 and early stopping. The validation accuracy and validation loss are computed by setting aside 20% of training samples as the validation set, and then evaluating the model after each epoch using the validation set. Early stopping stops the training of the neural network if there is no decrease in the validation loss for a certain number of consecutive epochs N=10.

3.2.2 - Autoencoder Model:

The Architecture of the Autoencoder used for feature extraction is shown in Fig. 3.

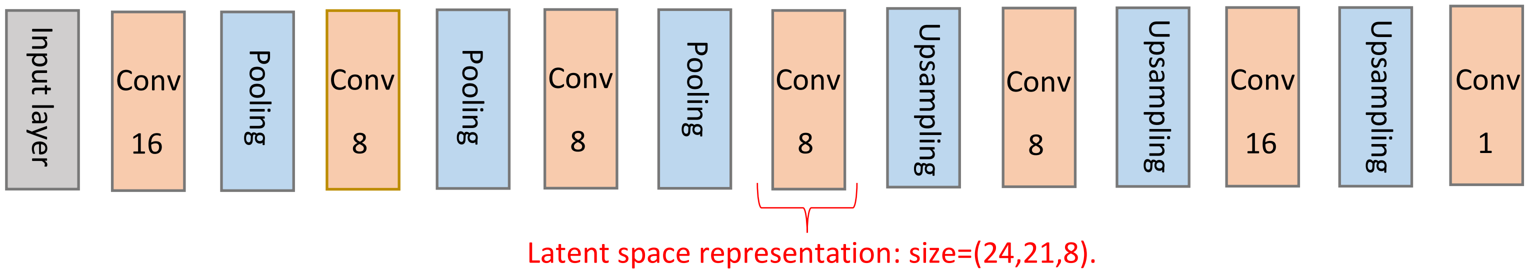


Fig. 3: Structure of the CAE.

The input layer has a size of (192,168,1), which corresponds to the size of the input image. Each convolution layer has a ‘Relu’ activation function, a filter size of 3*3 and same padding. The number written inside the box of each convolution layer corresponds to the number of channels of the layer. It is in decreasing order for the Encoding part and in increasing order for the decoding part. Each Maxpooling layer uses (2,2) strides to halve the image size and each Upsampling layer doubles the size of the image. The output layer corresponds to the reconstructed image. It has a ‘sigmoid’ activation function and has a size of (192,168,1), which corresponds to the size of the original image. The latent space representation layer has a size of (24,21,8), which corresponds to 4032 features. This corresponds to a dimensionality reduction of: [1-(4032/32256)]x100=87.5%. Because the network doesn’t use fully connected layers, it has a low number of trainable parameters. In fact, the CAE 4,385 trainable parameters..

Unlike PCA, the training of the CAE does not require the images to be vectorized. They are stored in 3-D arrays of size (192,168,1). Furthermore, because neural networks are sensitive to values of high magnitude, the training and test data are normalized. The CAE is trained with the ADAM optimizer, a minibatch size of 20 and validation split ratio of 0.3. The number of epochs is set to 10. For the classification part, we use the vectorized output of the latent space representation, where each image is encoded through a vector of 4032 attributes. We use the same classifier used for PCA, with the only difference being the input layer size being 4032 instead of 300.

3.2.3 - Results:

Results of the reconstruction of an image from the testing set by the CAE are shown in Fig. 4. Fig. 4 (a) shows the original image of an individual and Fig. 4 (b) shows the image reconstructed by the CAE. With as little parameters as 4,385 we can consider that the CAE yields acceptable results at reconstructing images it has never seen before. The top 6 eigenfaces computed by PCA and corresponding to the highest variance are shown in Fig. 5.

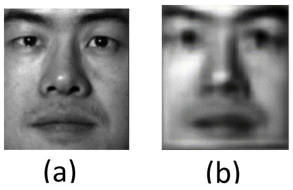


Fig. 4: original image and the reconstructed image by the CAE.

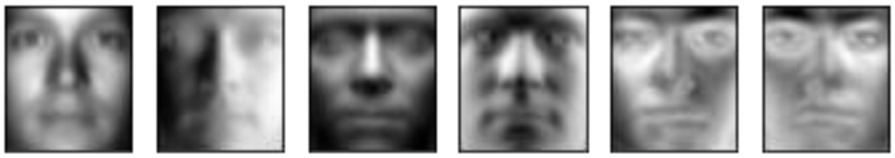
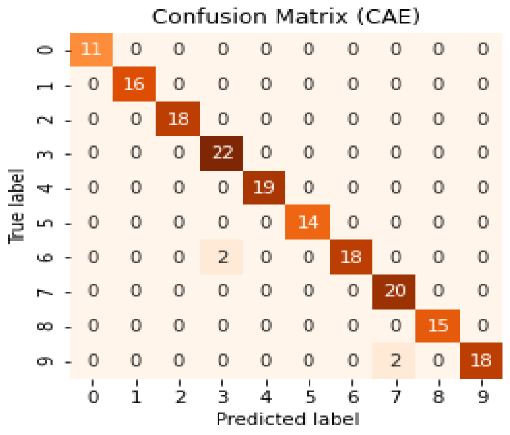


Fig. 5: eigenfaces computed by PCA corresponding to the highest variance.

Because training a neural network for classification yields different results at every run, the neural networks used for the classification of the 38 individuals after performing PCA or CAE as a feature extraction method were trained 10 times. The statistical results in the form of the mean testing accuracy and standard deviation cases are presented in Fig. 6. The results show that both methods yield similar results with CAE being slightly better.

	CAE	PCA
Mean testing accuracy	95.94 %	95.45 %
Standard deviation	0.6 %	0.5 %

Fig. 6: statistical results of PCA and CAE in terms of mean accuracy and standard deviation of the classification of the individuals on the test dataset.



	0	1	2	3	4	5	6	7	8	9
0	11	0	0	0	0	0	0	0	0	0
1	0	16	0	0	0	0	0	0	0	0
2	0	0	18	0	0	0	0	0	0	0
3	0	0	0	22	0	0	0	0	0	0
4	0	0	0	0	19	0	0	0	0	0
5	0	0	0	0	0	14	0	0	0	0
6	0	0	0	2	0	0	18	0	0	0
7	0	0	0	0	0	0	0	20	0	0
8	0	0	0	0	0	0	0	0	15	0
9	0	0	0	0	0	0	0	2	0	18

Fig. 7 : The confusion matrix of 10 classes in face recognition at application of a convolutional autoencoder as a feature extraction method.

Fig. 7 presents the confusion matrix for one run of the CAE solution in recognition of the first 10 individuals. It shows only scarce-off diagonal elements which are different from zero.

4 - Conclusion:

In this work we have applied PCA and CAE as feature extraction methods on 'the Extended Yale Face Database B' for face recognition. As a classifier we used a neural network with a softmax output layer. Results showed similar testing accuracy results at the range of 95% for both methods with a slight advantage of CAE. Another thing to take in consideration is the better dimensionality reduction of PCA (99%) compared to CAE (87.5%).

5 - References:

[1] P. Galeone's , https://pgaleone.eu/neural-networks/2016/11/24/convolutional-autoencoders/

[2] Sridhar, A. and Suman, K.A. Beginning Anomaly Detection Using Python-Based Deep learning, Apress, 2019.

[3] Georghiades, A.S. and Belhumeur, P.N. and Kriegman, D.J, From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose, IEEE Trans. Pattern Anal. Mach. Intelligence, vol 23 ,pp. 643-660, 2001.

[4] K.C. Lee and J. Ho and D. Kriegman, Acquiring Linear Subspaces for Face Recognition under Variable Lighting, IEEE Trans. Pattern Anal. Mach. Intelligence, vol 27, pp. 684-698, 2005.

[5] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov Dropout: a simple way to prevent neural networks from overfitting J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, 2014.

[6] F. Chollet Keras, 2015. [Online]. Available: https://github.com/fchollet/ keras.

[7] M. Abadi Tensorflow: Large-scale machine learning on heterogeneous distributed systems arXiv preprint arXiv:1603.04467, 2016.

[8] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.