A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines

David Charte
Dept. CS and AI
University of Granada
Granada, España
fdavidcl@ugr.es

Francisco Charte
Dept. CS
University of Jaén
Jaén, España
fcharte@ujaen.es

Salvador García
Dept. CS and AI
University of Granada
Granada, España
salvagl@decsai.ugr.es

María J. del Jesus
Dept. CS
University of Jaén
Jaén, España
mjjesus@ujaen.es

Francisco Herrera
Dept. CS and AI
University of Granada
Granada, España
herrera@decsai.ugr.es

***Abstract*—This is a summary of our article published in Information Fusion [1] to be part of the CAEPIA-18 Key Works.**

***Index Terms*—Autoencoders, Feature fusion, Feature extraction, Representation learning, Deep Learning, Machine Learning**

I. Summary

The performance of most machine learning algorithms depends on the quality of the input data, and in particular on its features. Many techniques exist which combine their information into a new feature set, with the aim of improving learned models. This new set is usually of lower dimension and contains more abstract variables. Procedures which result in new feature sets can be named after several terms: feature engineering, feature learning, representation learning, feature selection, feature extraction and feature fusion. Manifold learning algorithms, especially those based on artificial neural networks (ANNs), fall into the last category.

Autoencoders (AEs) are feedforward ANNs with a symmetric *encoder-decoder* structure (see Fig. 1), where the middle layer represents an encoding of the input data. They are trained to reconstruct their inputs onto their output layer while some restrictions prevent them from copying the data along. AE components can generally be represented as functions: an encoder f and a decoder g which map inputs x to outputs r: $r = g(f(x))$. The desired encoding would be $y = f(x)$.

Typical activation functions such as linear, binary or ReLU are of limited use in AEs, while logistic, hyperbolic tangent and SELU are more common. According to whether the dimension of their encoding is lower or higher than that of the data, AEs can be either *undercomplete* or *overcomplete*, respectively. If they have just one hidden layer they are called *shallow*, and otherwise they are considered *deep*.

A taxonomy of the main variants of AEs has been proposed according to the properties of the inferred model:

This work is supported by the Spanish National Research Projects TIN2015-68454-R and TIN2014-57251-P, and Project BigDaP-TOOLS - Ayudas Fundación BBVA a Equipos de Investigación Científica 2016.

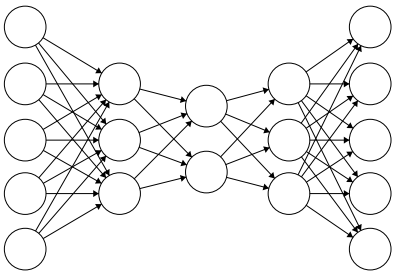

Fig. 1. A possible neural architecture for an autoencoder.

- Lower dimensionality
  - Basic [2]: the basis for most autoencoders, a symmetric feedforward ANN with a distance between the outputs and the inputs as the objective function, trainable with stochastic variants of gradient descent (SGD, AdaGrad, RMSProp, Adam...). Training can be done layer by layer in a stacked fashion.
  - Convolutional [3]: layers in this variant are convolutional, which explicitly consider a 2-dimensional structure for processing images.
  - LSTM-based [4]: models sequential data by placing Long-Short-Term Memory units as encoders and decoders.
- Regularization
  - Sparse [5]: introduces a penalty term for encodings with many activations, thus resulting in a low activation average. The Kullback-Leibler divergence can be used to attract the AE to the desired average activation value.
  - Contractive [6]: achieves local invariance to changes in many directions around the training samples by adding a penalty term based on the Frobenius norm of the Jacobian matrix of the encoder.
- Noise tolerance
  - Denoising [7]: its generated features are less sensitive to noise by training with corrupted versions of

input data.
  - Robust [8]: uses a special loss function, correntropy, in order to build more robust features. Correntropy measures the probability density that two events are equal, and is less affected by outliers than other distance metrics.

- Generative model
  - Variational [9]: applies a variational Bayesian approach to encoding, assuming a latent, unobserved random variable generates the observations. It tries to approximate the distribution of the latent variable given the data.
  - Adversarial [10]: brings adversarial learning to AEs, simultaneously training a discriminator and a generator (the encoder), each competing with the other.

AEs are also the inspiration for other, more complex, neural structures. For example, autoencoder trees which involve decision trees, dual-autoencoders which learn two latent representations, and recursive AEs which introduce more pieces of input as the model deepens.

AEs can be related to the wide range of existing feature fusion methods: in the linear case, they are equivalent to PCA when linear activations and mean squared error loss function are used. As a consequence, they can be seen as generalizations of PCA which allow nonlinearities and other objective functions. They are more easily applicable than Kernel PCA since in that case the choice of kernel highly alters the behavior of the model. Other nonlinear approaches such as Isomap and Locally Linear Embedding can be compared to the contractive AE, as they attempt to preserve the local structure of the data. Restricted Boltzmann Machines are alternatives to AEs for greedy layer-wise initialization, but their models are hard to simulate than those of AEs.

The usual purpose of AEs is performing feature fusion in order to improve classification and regression performance, and to facilitate other unsupervised tasks which can be hard in high-dimensional scenarios, such as clustering. Most applications of AEs can be summarized in the following categories:

- Classification: reducing or transforming the training data in order to achieve better performance in a classifier.
- Data compression: training AEs for specific types of data to learn efficient compressions.
- Detection of abnormal patterns: identification of discordant in- stances by analyzing generated encodings.
- Hashing: summarizing input data onto a binary vector for faster search.
- Visualization: projecting data onto 2 or 3 dimensions with an AE for graphical representation.
- Other purposes: reconstruction of deteriorated images, noise reduction in automatic speech recognition, curation of biological databases, tagging digital resources.

In [1] we propose some guidelines which can help when designing AEs for different tasks. The following is an overview of these:

- Architecture: a starting point is choosing the desired length of the encoding and the type of units for the neural structure.
- Activations and loss function: sigmoid-like functions generally work well in the encoding layer. Place linear or ReLU at the output when using mean squared error, or logistic activation when using cross-entropy error.
- Regularizations: can be used according to the desired properties in the encoding. Weight decay can prevent overfitting, sparsity can be useful in many scenarios and contraction can find a lower-dimensional manifold.

The main software libraries and frameworks which allow for the construction and training of AEs are Tensorflow, Caffe, Torch, MXNet, and Keras. Software pieces which specifically implement AEs are packages Autoencoder and SAENET of the CRAN repository, H2O and yadlt for Python.

A case study is provided in [1] with the well known MNIST handwritten digits dataset, covering several adjustable parameters such as the encoding length, the optimization algorithm, activation functions and some of the predominant AE variants.

References

[1] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78 – 96, 2018.

[2] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4, pp. 291–294, 1988.

[3] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Artificial Neural Networks and Machine Learning – ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I*, T. Honkela, W. Duch, M. Girolami, and S. Kaski, Eds. Springer Berlin Heidelberg, 2011, pp. 52–59.

[4] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using LSTMs," in *International Conference on Machine Learning*, 2015, pp. 843–852.

[5] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[6] S. Rifai, Y. Bengio, P. Vincent, and Y. N. Dauphin, "A generative process for sampling contractive auto-encoders," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ser. ICML'12. Omnipress, 2012, pp. 1811–1818.

[7] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. ACM, 2008, pp. 1096–1103.

[8] Y. Qi, Y. Wang, X. Zheng, and Z. Wu, "Robust feature learning by stacked autoencoder with maximum correntropy criterion," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6716–6720.

[9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[10] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.