# A Showcase of the Use of Autoencoders in Feature Learning Applications

David Charte[1(✉)], Francisco Charte[2], María J. del Jesus[2], and Francisco Herrera[1]

[1] Andalusian Research Institute in Data Science and Computational Intelligence, Department of Computer Science and Artificial Intelligence, Universidad de Granada, Periodista Daniel Saucedo Aranda, s/n, 18071 Granada, Spain
fdavidcl@ugr.es, herrera@decsai.ugr.es

[2] Andalusian Research Institute in Data Science and Computational Intelligence, Computer Science Department, Universidad de Jaén, Campus Las Lagunillas, s/n, 23071 Jaén, Spain
{fcharte,mjjesus}@ujaen.es

**Abstract.** Autoencoders are techniques for data representation learning based on artificial neural networks. Differently to other feature learning methods which may be focused on finding specific transformations of the feature space, they can be adapted to fulfill many purposes, such as data visualization, denoising, anomaly detection and semantic hashing.

This work presents these applications and provides details on how autoencoders can perform them, including code samples making use of an R package with an easy-to-use interface for autoencoder design and training, `ruta`. Along the way, the explanations on how each learning task has been achieved are provided with the aim to help the reader design their own autoencoders for these or other objectives.

**Keywords:** Autoencoders · Deep learning · Feature learning

## 1 Introduction

Autoencoders (AEs) [8] are versatile unsupervised learning methods. Also known as autoassociators, since their first uses their purpose has usually been to transform the input variable space into a more useful one, either one which is more compact, or whose structure is simpler or more convenient. In the last decade, deep learning methods have been gaining use since computing capabilities and optimization methods have improved. As a consequence, the topic of representation learning [2] has attained more interest, and AEs with it.

Nowadays, many variants of AEs [4] have been developed with several applications in mind. These cover from simple dimensionality reduction to instance

D. Charte is supported by the Spanish Ministry of Science, Innovation and Universities under the FPU National Program (Ref. FPU17/04069). This work is supported by the Spanish National Research Projects TIN2015-68854-R and TIN2017-89517-P.


J. M. Ferrández Vicente et al. (Eds.): IWINAC 2019, LNCS 11487, pp. 412–421, 2019.
https://doi.org/10.1007/978-3-030-19651-6_40

generation, data enhancement and hashing. Furthermore, they are not limited to problems with a standard (input, label) structure, but AE models have been defined for many nonstandard learning problems such as multi-view learning or multi-label learning [3,16,17].

In the following sections, AEs as a feature learning tool are described and several different applications are detailed and illustrated with examples. Each example comprises several lines of code which tackle a given task and some graphical output. The complete code samples are available at https://github.com/ari-dasci/autoencoder-showcase.

This document is structured as follows. Section 2 introduces the inner workings of AEs. Section 3.1 shows a simple visualization task, followed by an example on image denoising in Sect. 3.2. Afterwards, Sect. 3.3 describes how to detect anomalous samples with AEs and Sect. 3.4 a way to perform semantic hashing. Last, Sect. 3.5 mentions other possible applications of AEs, and Sect. 4 draws some conclusions.

## 2 Fundamentals of Autoencoders

AEs are artificial neural networks designed to find an alternative representation of data with some desirable properties. They are generally unsupervised techniques, that is, AEs are usually intended to complete their task without any class information. They are comprised of two distinct components: the encoder and the decoder. Both have interesting outputs: the first provides the encodings for each input instance, and the second obtains a reconstruction of the original instance. The basic objective for an AE consists in finding the parameters that allow to retrieve the most faithful reconstructions.

Figure 1 shows the architecture of a simple AE, where input data is feeded into the leftmost layer and the obtained output must match it with as little error as possible. This can be interpreted as a parametrized mapping $f_\theta$ which is fitted to the data by minimizing a loss function $\mathcal{J}(\theta) = \sum_x L(x, f_\theta(x))$.

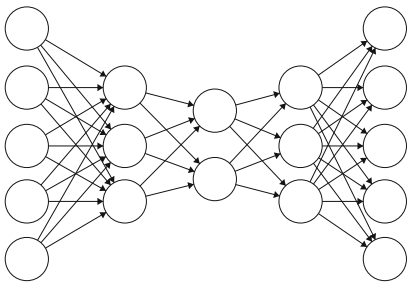

**Fig. 1.** Typical organization of neurons inside an autoencoder, in this case, for a 2-variable encoding.

A common application of these models is transforming the variable space into a lower-dimensional one, while keeping enough information about samples to accurately recover their values in the original variable space. In this case, AEs can be seen as a nonlinear generalization of principal components analysis [1]. However, they can show much more potential when combined with certain restrictions or modifications.

### 2.1 Encoded Space Structure

AEs can be altered in order for the encoded variable space to have some desirable structure, or to verify some properties. Following are some viable structures that may be applied to an AE. Among other possibilities, the encoding space can:

- be sparse, i.e. few neurons have a high value for each instance (sparse AE)
- resist distortions present in input data (denoising AE, robust AE)
- preserve local behavior and relative distances from the input space (contractive AE)
- follow a given distribution (adversarial AE)

### 2.2 Available Software

Any well known deep learning framework may serve as base for the construction of AEs, but very few tools facilitate the design process by abstracting common traits from usual AEs. One of them is R package `ruta` [5], which provides a variety of AE variants, allowing for easy definition, training and usage of these models. The package is available on CRAN and thus can be installed with:

```
install.packages("ruta")
```

The upcoming sections will make use of this software and other libraries in order to provide compact code listings which achieve the different tasks. Other available software packages for the purposes of autoencoder training are: `autoencoder` (R), `h2o` (multiplatform) and `yadlt` (Python).

## 3 Examples of Use

The following sections contain four complete examples where an AE is used to complete a learning task. Each example is comprised of a description of the tackled problem, some explanation on how an AE solves or can help solve it, a code snippet which trains an AE adequate for the task and some visual results and associated analysis.

### 3.1 Visualization of High-Dimensional Data

AEs can be arranged so as to produce a two-dimensional or three-dimensional code. This way, encoded instances can be directly represented in a graph. If samples are labeled with classes or target values, this can be used to provide a visual intuition on how these labels are organized.

In the following example, R packages `ruta`, `scatterplot3d` and `colorspace` are loaded. This allows to train an AE which compresses data to a three-dimensional space, extract codifications for the desired dataset and lastly plot them with colors according to their class. The chosen AE has several hidden layers in order to allow it to learn short encodings; the middle layer uses sigmoid activation which limits the codes to the $[0, 1]$ interval.

```
network <- input() + dense(12, activation = "relu") + dense(3,
  activation = "sigmoid") + dense(12, activation = "relu") + output("linear")
model <- autoencoder_sparse(network) %>% train(x_train, epochs = 40)
codes <- model %>% encode(x_train)
scatterplot3d(codes, color = rainbow_hcl(7)[class], (1:7)[class])
```

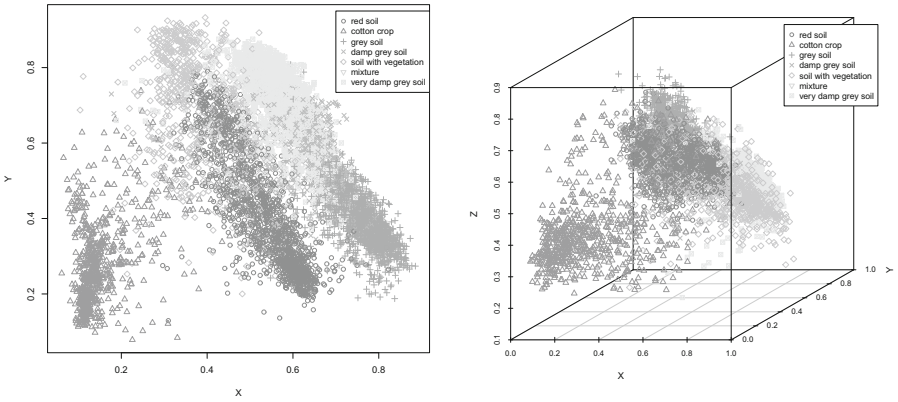


**Fig. 2.** Two and three-dimensional projections of the 36-variable dataset Statlog.

Figure 2 shows the result of running this code on the 36-dimensional dataset Statlog (Landsat Satellite) [6], where each instance represents a small patch of land and is labeled with the type of land associated to the central pixel. Although the graph shows some fairly separated instance clusters, it is important to remark that the AE receives no class information and thus it can be assumed that the input features also present some degree of class separation.

### 3.2 Image Denoising

Since AEs are not only trained to transform the variable space but also to reconstruct the original one, a specific strategy can be used so that the reconstruction

process eliminates distortions or noise [18]. AEs that are trained this way are usually called denoising AEs.

For the purposes of image denoising, a convolutional AE can be used. Bidimensional convolutional layers [7] help models take advantage of the structure of images to reduce the number of parameters. The architecture of this AE does not need to reduce the dimension of the input data in the encoding layer. Instead, it is possible to increase the dimension in order for it to be able to discern useful information from noise.

In this case, a very simple architecture has been used: one convolutional layer with an upsampling operation, another one with a max-pooling operation (to decrease the dimension) and an output convolutional layer with as many filters as channels in the data. The last layer uses a sigmoid activation because the inputs are normalized to the $[0, 1]$ interval. This AE has been trained and tested with the CIFAR10 dataset included in Keras.

```
network <- input() + conv(16, 3, upsampling=2, activation="relu") + conv(16,
    3, max_pooling=2, activation="relu") + conv(3, 3, activation="sigmoid")
model <- autoencoder_denoising(network, loss = "binary_crossentropy",
  noise_type = "gaussian", sd = .05) %>%
  train(x_train, epochs = 30, batch_size = 500, optimizer = "adam")
noisy <- noise_gaussian(sd = .05) %>% apply_filter(x_test)
recovered <- model %>% reconstruct(noisy)
```

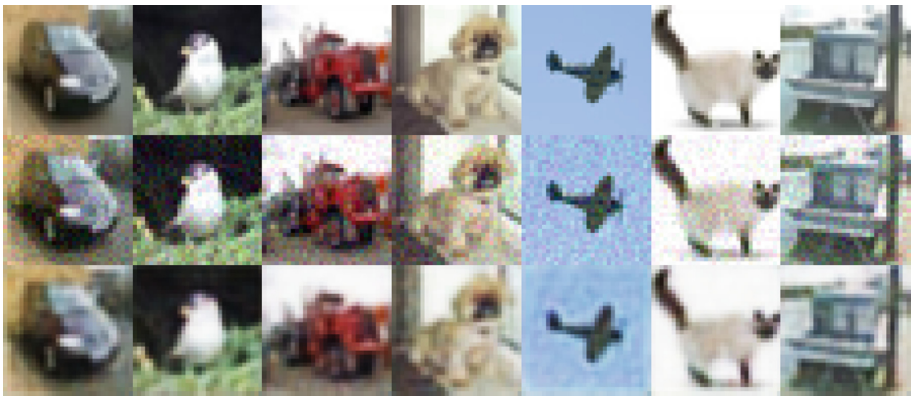

**Fig. 3.** Image denoising. First row shows original test samples, second row displays the noisy images feeded to the AE, and third row shows reconstructed images.

The listing above also shows a way of introducing noise to the test subset and reconstructing it by means of the trained model. Figure 3 shows a sample of the obtained results drawn by means of the `grid` package: each test image, previously unseen by the AE, is joined by its noisy version, which is feeded to the trained model, and the reconstructed version, obtained at the output of the AE.

### 3.3 Anomaly Detection

The encoded variable space generated by an AE usually omits information that is common to most or all instances, since the objective of an AE is to produce precise reconstructions while retaining only necessary information. Any traits present in all samples can be easily learned and recovered by the decoder, so that they do not need to be encoded. This means that, in the case that a new instance is very different from those of the training set, its reconstruction by the AE will predictably lose information and produce a high error. In these scenarios, AEs can serve as anomaly detectors [12,13]. This is especially useful when treating time series that may have abnormal regions, since detecting single examples based just on their reconstruction can be more challenging.

The following code trains a very simple denoising AE with a 16-variable encoding, and then measures the individual error made for each instance. The reason a denoising AE has been chosen in this case is that it can indirectly find lower dimensional manifolds in the data, and attempt to push instances to those when decoding. Thus, anomalous samples far from the manifold should stand out more when measuring the error.

```
model <- autoencoder_denoising(input() + dense(16, "sigmoid") + output()) %>%
  train(x_train, epochs = 50, batch_size = 32)
errors <- rowMeans((model %>% reconstruct(x_test) - x_test) ** 2)
```

The AE above has been trained with a synthetic multi-valued time series based on samples from a solution to a Lorenz system, following the experimentation in [13]. An anomalous region has been artificially introduced within the test subset. The reconstruction error has been measured for each test instance, and the results are shown in the plot in Fig. 4: anomalies present much higher errors in general, which can be used to detect that region.

### 3.4 Semantic Hashing

Semantic hashing [8] is a task consisting on finding short binary codes for data points so that codes with small Hamming distance correspond to similar samples, and those separated by a high Hamming distance belong to very different points.

An AE can produce encodings in the range $[0, 1]$ when the encoding layer has a sigmoid activation function. This can be joined with a high Gaussian noise at the input of that layer in order to strongly polarize its outputs [8]. The results are very close to binary, and a thresholding function can be applied to obtain exact integers.

In the following example, the network defined includes a custom Keras layer which introduces this Gaussian noise and is otherwise very standard. A basic AE is trained using this network, afterwards it encodes test instances, and the `hash` function applies a threshold to each encoding to find a binary hash.

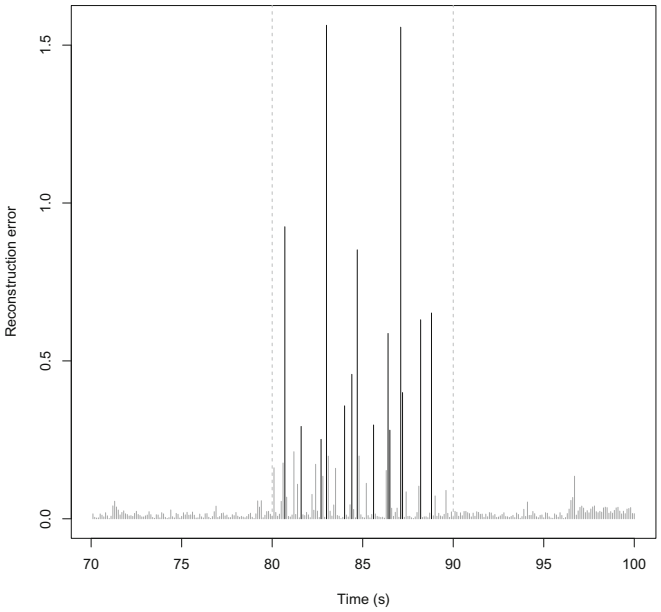


**Fig. 4.** Reconstruction error of a test subset of a synthetic time series. Artificially generated anomalous data is placed between the dashed lines. High reconstruction errors (above the mean error plus its standard deviation) are marked in black.

```
net <- input() + dense(256) + layer_keras("gaussian_noise", stddev = 16) +
  dense(10, activation = "sigmoid") + dense(256) + output("sigmoid")
model <- autoencoder(net, "binary_crossentropy") %>%
  train(x_train, epochs = 50)
hash <- function(model, x, threshold = 0.5) {
  t(encode(model, x) %>% apply(1, function(r) as.integer(r > threshold)))
}
hashes <- model %>% hash(x_test)
```

In order to illustrate the effectiveness of this AE, we have trained it with the training subset of the IMDB dataset available in Keras, composed of 25 000 movie reviews, from which just the 1 000 most frequent words have been used (i.e. the training subset had 1 000 variables and 25 000 samples). Then, we use the obtained model to produce hashes for each test instance.

Since the objective of semantic hashing is to obtain similar binary hashes for similar inputs, we measure the distance among instances and associate it to that among their hashes. The dataset used contains text documents, so the cosine distance is a good measure for pairs of instances in this case. Hashes will be considered distant according to their Hamming distance. Figure 5 shows that, the more different two hashes are, more distance is between the corresponding instances.

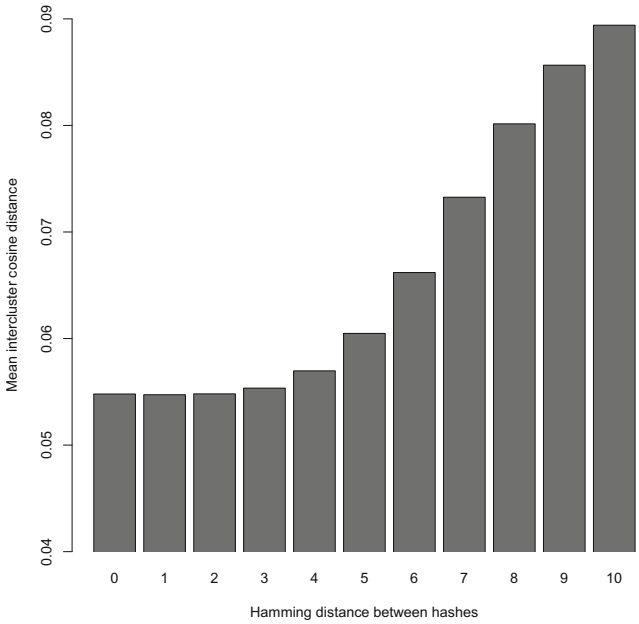


**Fig. 5.** Measure of average cosine distance among instances whose hash encodings differ in any Hamming distance.

### 3.5 Other Applications

The applications described in the examples above are only a sample of the possible purposes an AE can be used for. Another common objective for AEs is to improve classification performance [20,21]. As with image denoising, AEs can serve as enhancers for other kinds of data; particularly speech [11] and electrocardiogram signals [19].

Combinations of AEs can merge information from several sources, such as two and three-dimensional human poses [9] or pairs of image and text and other multi-view data [16]. Other AE-based models have been developed in order to improve tag recommendation [15], prediction of movement from static images [14], and translation involving sentence reordering [10].

## 4 Conclusions

Feature learning is a crucial task which can determine the performance of a machine learning model. In this work, AEs have been described as an adaptable basis for many different tasks which involve finding alternative representations of data.

Four example applications have been described in detail and solved using AEs: data visualization in two and three dimensions, denoising of images, detec-

tion of abnormal patterns in time series and semantic hashing for text documents. The sample experiments have been performed with well known datasets and using common features in AEs readily available in a published software package, `ruta`.

The main objective of this work has been to gather the necessary knowledge and ideas to guide the reader into designing their own experiments based on AE models when facing feature learning tasks.

## References


1. Baldi, P., Hornik, K.: Neural networks and principal component analysis: learning from examples without local minima. Neural Netw. **2**(1), 53–58 (1989). https://doi.org/10.1016/0893-6080(89)90014-2
2. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013). https://doi.org/10.1109/TPAMI.2013.50
3. Charte, D., Charte, F., García, S., Herrera, F.: A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations. Prog. Artif. Intell. **8**, 1–14 (2019). https://doi.org/10.1007/s13748-018-00167-7
4. Charte, D., Charte, F., García, S., del Jesus, M.J., Herrera, F.: A practical tutorial on autoencoders for nonlinear feature fusion: taxonomy, models, software and guidelines. Inf. Fusion **44**, 78–96 (2018). https://doi.org/10.1016/j.inffus.2017.12.007
5. Charte, D., Herrera, F., Charte, F.: Ruta: implementations of neural autoencoders in R. Knowl.-Based Syst. (in press)
6. Dheeru, D., Karra Taniskidou, E.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
7. Goodfellow, I., Bengio, Y., Courville, A.: Convolutional Networks. In: Deep Learning, pp. 326–366. MIT Press (2016). http://www.deeplearningbook.org
8. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006). https://doi.org/10.1126/science.1127647
9. Hong, C., Yu, J., Wan, J., Tao, D., Wang, M.: Multimodal deep autoencoder for human pose recovery. IEEE Trans. Image Process. **24**(12), 5659–5670 (2015). https://doi.org/10.1109/TIP.2015.2487860
10. Li, P., Liu, Y., Sun, M.: Recursive autoencoders for ITG-based translation. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 567–577 (2013)
11. Lu, X., Tsao, Y., Matsuda, S., Hori, C.: Speech enhancement based on deep denoising autoencoder. In: Interspeech, pp. 436–440 (2013)
12. Park, S., Kim, M., Lee, S.: Anomaly detection for http using convolutional autoencoders. IEEE Access **6**, 70884–70901 (2018). https://doi.org/10.1109/ACCESS.2018.2881003
13. Sakurada, M., Yairi, T.: Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, pp. 4–11. ACM (2014). https://doi.org/10.1145/2689746.2689747

14. Walker, J., Doersch, C., Gupta, A., Hebert, M.: An uncertain future: forecasting from static images using variational autoencoders. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 835–851. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_51
15. Wang, H., Shi, X., Yeung, D.Y.: Relational stacked denoising autoencoder for tag recommendation. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
16. Wang, X., Peng, D., Hu, P., Sang, Y.: Adversarial correlated autoencoder for unsupervised multi-view representation learning. Knowl.-Based Syst. (2019). https://doi.org/10.1016/j.knosys.2019.01.017
17. Wicker, J., Tyukin, A., Kramer, S.: A nonlinear label compression and transformation method for multi-label classification using autoencoders. In: Bailey, J., Khan, L., Washio, T., Dobbie, G., Huang, J.Z., Wang, R. (eds.) PAKDD 2016. LNCS (LNAI), vol. 9651, pp. 328–340. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31753-3_27
18. Xie, J., Xu, L., Chen, E.: Image denoising and inpainting with deep neural networks. In: Advances in Neural Information Processing Systems, pp. 341–349 (2012)
19. Xiong, P., Wang, H., Liu, M., Zhou, S., Hou, Z., Liu, X.: ECG signal enhancement based on improved denoising auto-encoder. Eng. Appl. Artif. Intell. **52**, 194–202 (2016). https://doi.org/10.1016/j.engappai.2016.02.015
20. Xu, J., et al.: Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images. IEEE Trans. Med. Imaging **35**(1), 119–130 (2016). https://doi.org/10.1109/tmi.2015.2458702
21. Xu, W., Sun, H., Deng, C., Tan, Y.: Variational autoencoder for semi-supervised text classification. In: AAAI Conference on Artificial Intelligence (2017)