# Autoencoder

An **autoencoder** is a type of artificial neural network used to learn efficient codings of unlabeled data (unsupervised learning).[1] The encoding is validated and refined by attempting to regenerate the input from the encoding. The autoencoder learns a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore insignificant data ("noise").

Variants exist, aiming to force the learned representations to assume useful properties.[2] Examples are regularized autoencoders (*Sparse*, *Denoising* and *Contractive*), which are effective in learning representations for subsequent classification tasks,[3] and *Variational* autoencoders, with applications as generative models.[4] Autoencoders are applied to many problems, including facial recognition,[5] feature detection,[6] anomaly detection and acquiring the meaning of words.[7][8] Autoencoders are also generative models which can randomly generate new data that is similar to the input data (training data).[6]

## Contents



## Mathematical principles

## Definition

An autoencoder is defined by the following components:

> Two sets: the space of decoded messages $\mathcal{X}$; the space of encoded messages $\mathcal{Z}$. Almost always, both $\mathcal{X}$ and $\mathcal{Z}$ are Euclidean spaces, that is, $\mathcal{X} = \mathbb{R}^m, \mathcal{Z} = \mathbb{R}^n$ for some $m, n$.

> Two parametrized families of functions: the encoder family $E_\phi : \mathcal{X} \to \mathcal{Z}$, parametrized by $\phi$; the decoder family $D_\theta : \mathcal{Z} \to \mathcal{X}$, parametrized by $\theta$.

For any $x \in \mathcal{X}$, we usually write $z = E_\phi(x)$, and refer to it as the code, the latent variable, latent representation, latent vector, etc. Conversely, for any $z \in \mathcal{Z}$, we usually write $x' = D_\theta(z)$, and refer to it as the (decoded) message.

Usually, both the encoder and the decoder are defined as multilayer perceptrons. For example, a one-layer-MLP encoder $E_\phi$ is:

$$E_\phi(\mathbf{x}) = \sigma(Wx + b)$$

where $\sigma$ is an element-wise activation function such as a sigmoid function or a rectified linear unit, $W$ is a matrix called "weight", and $b$ is a vector called "bias".

## Training an autoencoder

An autoencoder, by itself, is simply a tuple of two functions. To judge its *quality*, we need a *task*. A task is defined by a reference probability distribution $\mu_{ref}$ over $\mathcal{X}$, and a "reconstruction quality" function $d : \mathcal{X} \times \mathcal{X} \to [0, \infty]$, such that $d(x, x')$ measures how much $x'$ differs from $x$.

With those, we can define the loss function for the autoencoder as

$$L(\theta, \phi) := \mathbb{E}_{x \sim \mu_{ref}}[d(x, D_\theta(E_\phi(x)))]$$

The *optimal* autoencoder for the given task $(\mu_{ref}, d)$ is then $\arg\min_{\theta,\phi} L(\theta, \phi)$. The search for the optimal autoencoder can be accomplished by any mathematical optimization technique, but usually by gradient descent. This search process is referred to as "training the autoencoder".

In most situations, the reference distribution is just the empirical distribution given by a dataset $\{x_1, \ldots, x_N\} \subset \mathcal{X}$, so that

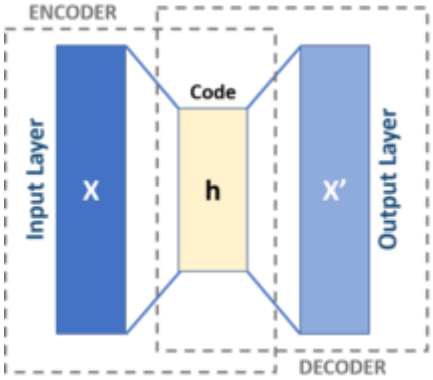
$$\mu_{ref} = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_i}$$

and the quality function is just L2 loss: $d(x, x') = \|x - x'\|_2^2$. Then the problem of searching for the optimal autoencoder is just a least-squares optimization:

$$\min_{\theta,\phi} L(\theta, \phi), \text{where } L(\theta, \phi) = \frac{1}{N} \sum_{i=1}^{N} \|x_i - D_\theta(E_\phi(x_i))\|_2^2$$

### Interpretation

An autoencoder has two main parts: an encoder that maps the message to a code, and a decoder that reconstructs the message from the code. An optimal autoencoder would perform as close to perfect reconstruction as possible, with "close to perfect" defined by the reconstruction quality function $d$.



Schema of a basic Autoencoder

The simplest way to perform the copying task perfectly would be to duplicate the signal. To suppress this behavior, the code space $\mathcal{Z}$ usually has fewer dimensions than the message space $\mathcal{X}$.

Such an autoencoder is called *undercomplete*. It can be interpreted as compressing the message, or reducing its dimensionality.[9]

At the limit of an ideal undercomplete autoencoder, every possible code $z$ in the code space is used to encode a message $x$ that really appears in the distribution $\mu_{ref}$, and the decoder is also perfect: $D_\theta(E_\phi(x)) = x$. This ideal autoencoder can then be used to generate messages indistinguishable from real messages, by feeding its decoder arbitrary code $z$ and obtaining $D_\theta(z)$, which is a message that really appears in the distribution $\mu_{ref}$.

If the code space $\mathcal{Z}$ has dimension larger than (*overcomplete*), or equal to, the message space $\mathcal{X}$, or the hidden units are given enough capacity, an autoencoder can learn the identity function and become useless. However, experimental results found that overcomplete autoencoders might still learn useful features.[10]

In the ideal setting, the code dimension and the model capacity could be set on the basis of the complexity of the data distribution to be modeled. A standard way to do so is to add modifications to the basic autoencoder, to be detailed below.[2]

## History

The autoencoder has also been called the autoassociator,[11] or Diabolo network[12].[10] Its first applications date to the 1980s.[2][13][14] Their most traditional application was dimensionality reduction or feature learning, but the concept became widely used for learning generative models of data.[15][16] Some of the most powerful AIs in the 2010s involved autoencoders stacked inside deep neural networks.[17]

## Variations

### Regularized autoencoders

Various techniques exist to prevent autoencoders from learning the identity function and to improve their ability to capture important information and learn richer representations.

### Sparse autoencoder (SAE)

Inspired by the sparse coding hypothesis in neuroscience, sparse autoencoders are variants of autoencoders, such that the codes $D_\phi(x)$ for messages tend to be *sparse codes*, that is, $D_\phi(x)$ is close to zero in most entries. Sparse autoencoders may include more (rather than fewer) hidden units than inputs, but only a small number of the hidden units are allowed to be active at the same time.[17] Encouraging sparsity improves performance on classification tasks.[18]

There are two main ways to enforce sparsity. One way is to simply clamp all but the highest-k activations of the latent code to zero. This is the **k-sparse autoencoder**.[19]

The k-sparse autoencoder inserts the following "k-sparse function" in the latent layer of a standard autoencoder:

$$f_k(x_1, \ldots, x_n) = (x_1 b_1, \ldots, x_n b_n)$$

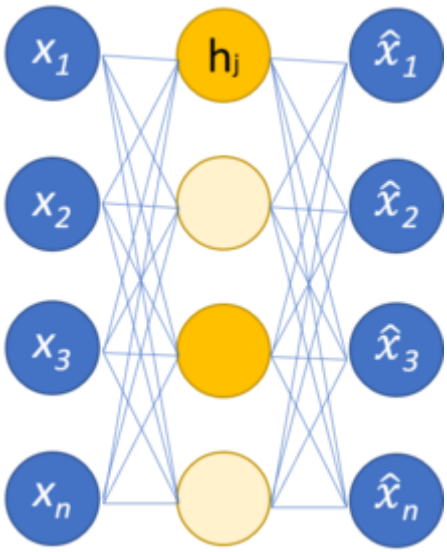
where $b_i = 1$ if $|x_i|$ ranks in the top k, and 0 otherwise.

Backpropagating through $f_k$ is simple: set gradient to 0 for $b_i = 0$ entries, and keep gradient for $b_i = 1$ entries. This is essentially a generalized ReLU function.[19]

The other way is a relaxed version of the k-sparse autoencoder. Instead of forcing sparsity, we add a **sparsity regularization loss**, then optimize for

$$\min_{\theta,\phi} L(\theta, \phi) + \lambda L_{sparsity}(\theta, \phi)$$

where $\lambda > 0$ measures how much sparsity we want to enforce.[20]



Simple schema of a single-layer sparse autoencoder. The hidden nodes in bright yellow are activated, while the light yellow ones are inactive. The activation depends on the input.

Let the autoencoder architecture have $K$ layers. To define a sparsity regularization loss, we need a "desired" sparsity $\hat{\rho}_k$ for each layer, a weight $w_k$ for how much to enforce each sparsity, and a function $s : [0, 1] \times [0, 1] \to [0, \infty]$ to measure much two sparsities differ.

For each input $x$, let the actual sparsity of activation in each layer $k$ be

$$\rho_k(x) = \frac{1}{n} \sum_{i=1}^{n} a_{k,i}(x)$$

where $a_{k,i}(x)$ is the activation in the $i$ -th neuron of the $k$ -th layer upon input $x$.

The sparsity loss upon input $x$ for one layer is $s(\hat{\rho}_k, \rho_k(x))$, and the sparsity regularization loss for the entire autoencoder is the expected weighted sum of sparsity losses:

$$L_{sparsity}(\theta, \phi) = \mathbb{E}_{x \sim \mu_X}\left[\sum_{k \in 1:K} w_k s(\hat{\rho}_k, \rho_k(x))\right]$$

Typically, the function $s$ is either the Kullback-Leibler (KL) divergence, as[18][20][21][22]

$$s(\rho, \hat{\rho}) = KL(\rho || \hat{\rho}) = \rho \log \frac{\rho}{\hat{\rho}} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}}$$

or the L1 loss, as $s(\rho, \hat{\rho}) = |\rho - \hat{\rho}|$, or the L2 loss, as $s(\rho, \hat{\rho}) = |\rho - \hat{\rho}|^2$.

Alternatively, the sparsity regularization loss may be defined without reference to any "desired sparsity", but simply force as much sparsity as possible. In this case, one can sparsity regularization loss as

$$L_{sparsity}(\theta, \phi) = \mathbb{E}_{x \sim \mu_X}\left[\sum_{k \in 1:K} w_k \|h_k\|\right]$$

where $h_k$ is the activation vector in the $k$-th layer of the autoencoder. The norm $\|\cdot\|$ is usually the L1 norm (giving the L1 sparse autoencoder) or the L2 norm (giving the L2 sparse autoencoder).

### Denoising autoencoder (DAE)

Denoising autoencoders (DAE) try to achieve a *good* representation by changing the *reconstruction criterion*.[2][3]

A DAE is defined by adding a noise process to the standard autoencoder. A noise process is defined by a probability distribution $\mu_T$ over functions $T : \mathcal{X} \to \mathcal{X}$. That is, the function $T$ takes a message $x \in \mathcal{X}$, and corrupts it to a noisy version $T(x)$. The function $T$ is selected randomly, with a probability distribution $\mu_T$.

Given a task $(\mu_{ref}, d)$, the problem of training a DAE is the optimization problem:

$$\min_{\theta, \phi} L(\theta, \phi) = \mathbb{E}_{x \sim \mu_X, T \sim \mu_T}\left[d(x, (D_\theta \circ E_\phi \circ T)(x))\right]$$

That is, the optimal DAE should take any noisy message and attempt to recover the original message without noise, thus the name "denoising".

Usually, the noise process $T$ is applied only during training and testing, not during downstream use.

The use of DAE depends on two assumptions:

- There exist representations to the messages that are relatively stable and robust to the type of noise we are likely to encounter;

- The said representations capture structures in the input distribution that are useful for our purposes.[3]

Example noise processes include:

- additive isotropic Gaussian noise,
- masking noise (a fraction of the input is randomly chosen and set to 0)
- salt-and-pepper noise (a fraction of the input is randomly chosen and randomly set to its minimum or maximum value).[3]

### Contractive autoencoder (CAE)

A contractive autoencoder adds the contractive regularization loss to the standard autoencoder loss:

$$\min_{\theta,\phi} L(\theta,\phi) + \lambda L_{contractive}(\theta,\phi)$$

where $\lambda > 0$ measures how much contractive-ness we want to enforce. The contractive regularization loss itself is defined as the expected Frobenius norm of the Jacobian matrix of the encoder activations with respect to the input:

$$L_{contractive}(\theta,\phi) = \mathbb{E}_{x\sim\mu_{ref}} \|\nabla_x E_\phi(x)\|_F^2$$

To understand what $L_{contractive}$ measures, note the fact

$$\|E_\phi(x+\delta x) - E_\phi(x)\|_2 \leq \|\nabla_x E_\phi(x)\|_F \|\delta x\|_2$$

for any message $x \in \mathcal{X}$, and small variation $\delta x$ in it. Thus, if $\|\nabla_x E_\phi(x)\|_F^2$ is small, it means that a small neighborhood of the message maps to a small neighborhood of its code. This is a desired property, as it means small variation in the message leads to small, perhaps even zero, variation in its code, like how two pictures may look the same even if they are not exactly the same.

The DAE can be understood as an infinitesimal limit of CAE: in the limit of small Gaussian input noise, DAEs make the reconstruction function resist small but finite-sized input perturbations, while CAEs make the extracted features resist infinitesimal input perturbations.

### Minimal description length autoencoder

[23]

## Concrete autoencoder

The concrete autoencoder is designed for discrete feature selection.[24] A concrete autoencoder forces the latent space to consist only of a user-specified number of features. The concrete autoencoder uses a continuous relaxation of the categorical distribution to allow gradients to pass through the feature selector layer, which makes it possible to use standard backpropagation to learn an optimal subset of input features that minimize reconstruction loss.

## Variational autoencoder (VAE)

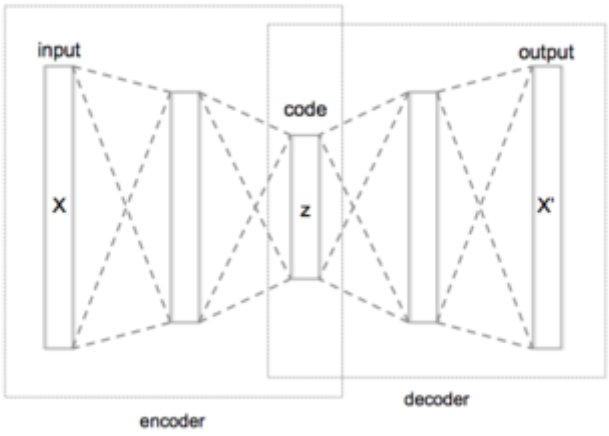
Variational autoencoders (VAEs) belong to the families of variational Bayesian methods. Despite the architectural similarities with basic autoencoders, VAEs are architecture with different goals and with a completely different mathematical formulation. The latent space is in this case composed by a mixture of distributions instead of a fixed vector.

Given an input dataset $x$ characterized by an unknown probability function $P(x)$ and a multivariate latent encoding vector $z$, the objective is to model the data as a distribution $p_\theta(x)$, with $\theta$ defined as the set of the network parameters so that $p_\theta(x) = \int_z p_\theta(x, z)dz$.

# Advantages of depth

Autoencoders are often trained with a single layer encoder and a single layer decoder, but using many-layered (deep) encoders and decoders offers many advantages.[2]

- Depth can exponentially reduce the computational cost of representing some functions.[2]
- Depth can exponentially decrease the amount of training data needed to learn some functions.[2]
- Experimentally, deep autoencoders yield better compression compared to shallow or linear autoencoders.[9]




Schematic structure of an autoencoder with 3 fully connected hidden layers. The code (z, or h for reference in the text) is the most internal layer.

## Training

Geoffrey Hinton developed the deep belief network technique for training many-layered deep autoencoders. His method involves treating each neighbouring set of two layers as a restricted Boltzmann machine so that pretraining approximates a good solution, then using backpropagation to fine-tune the results.[9]

Researchers have debated whether joint training (i.e. training the whole architecture together with a single global reconstruction objective to optimize) would be better for deep auto-encoders.[25] A 2015 study showed that joint training learns better data models along with more representative features for classification as compared to the layerwise method.[25] However, their experiments showed that the success of joint training depends heavily on the regularization strategies adopted.[25][26]

# Applications

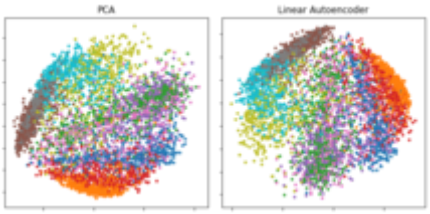
The two main applications of autoencoders are dimensionality reduction and information retrieval,[2] but modern variations have been applied to other tasks.

## Dimensionality reduction

Dimensionality reduction was one of the first deep learning applications.[2]

For Hinton's 2006 study,[9] he pretrained a multi-layer autoencoder with a stack of RBMs and then used their weights to initialize a deep autoencoder with gradually smaller hidden layers until hitting a bottleneck of 30 neurons. The resulting 30 dimensions of the code yielded a smaller reconstruction error compared to the first 30 components of a principal component analysis (PCA), and learned a representation that was qualitatively easier to interpret, clearly separating data clusters.[2][9]

Representing dimensions can improve performance on tasks such as classification.[2] Indeed, the hallmark of dimensionality reduction is to place semantically related examples near each other.[28]

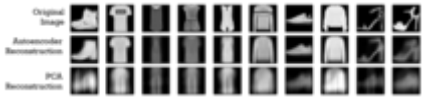


Plot of the first two Principal Components (left) and a two-dimension hidden layer of a Linear Autoencoder (Right) applied to the Fashion MNIST dataset.[27] The two models being both linear learn to span the same subspace. The projection of the data points is indeed identical, apart from rotation of the subspace - to which PCA is invariant.

### Principal component analysis

If linear activations are used, or only a single sigmoid hidden layer, then the optimal solution to an autoencoder is strongly related to principal component analysis (PCA).[29][30] The weights of an autoencoder with a single hidden layer of size $\boldsymbol{p}$ (where $\boldsymbol{p}$ is less than the size of the input) span the same vector subspace as the one spanned by the first $\boldsymbol{p}$ principal components, and the output of the autoencoder is an orthogonal projection onto this subspace. The autoencoder weights are not equal to the principal components, and are generally not orthogonal, yet the principal components may be recovered from them using the singular value decomposition.[31]

However, the potential of autoencoders resides in their non-linearity, allowing the model to learn more powerful generalizations compared to PCA, and to reconstruct the input with significantly lower information loss.[9]


Reconstruction of 28x28pixel images by an Autoencoder with a code size of two (two-units hidden layer) and the reconstruction from the first two Principal Components of PCA. Images come from the Fashion MNIST dataset.[27]

## Information retrieval

Information retrieval benefits particularly from dimensionality reduction in that search can become more efficient in certain kinds of low dimensional spaces. Autoencoders were indeed applied to semantic hashing, proposed by Salakhutdinov and Hinton in 2007.[28] By training the algorithm to produce a low-dimensional binary code, all database entries could be stored in a hash table mapping binary code vectors to entries. This table would then support information retrieval by returning all entries with the same binary code as the query, or slightly less similar entries by flipping some bits from the query encoding.

## Anomaly detection

Another application for autoencoders is anomaly detection.[32] [33][34][35][36] By learning to replicate the most salient features in the training data under some of the constraints described previously, the model is encouraged to learn to precisely reproduce the most frequently observed characteristics. When facing anomalies, the model should worsen its reconstruction performance. In most cases, only data with normal instances are used to train the autoencoder; in others, the frequency of anomalies is small compared to the observation set so that its contribution to the learned representation could be ignored. After training, the

autoencoder will accurately reconstruct "normal" data, while failing to do so with unfamiliar anomalous data.[34] Reconstruction error (the error between the original data and its low dimensional reconstruction) is used as an anomaly score to detect anomalies.[34]

Recent literature has however shown that certain autoencoding models can, counterintuitively, be very good at reconstructing anomalous examples and consequently not able to reliably perform anomaly detection.[37][38]

### Image processing

The characteristics of autoencoders are useful in image processing.

One example can be found in lossy image compression, where autoencoders outperformed other approaches and proved competitive against JPEG 2000.[39][40]

Another useful application of autoencoders in image preprocessing is image denoising.[41][42][43]

Autoencoders found use in more demanding contexts such as medical imaging where they have been used for image denoising[44] as well as super-resolution.[45][46] In image-assisted diagnosis, experiments have applied autoencoders for breast cancer detection[47] and for modelling the relation between the cognitive decline of Alzheimer's disease and the latent features of an autoencoder trained with MRI.[48]

### Drug discovery

In 2019 molecules generated with variational autoencoders were validated experimentally in mice.[49][50]

### Popularity prediction

Recently, a stacked autoencoder framework produced promising results in predicting popularity of social media posts,[51] which is helpful for online advertising strategies.

### Machine translation

Autoencoders have been applied to machine translation, which is usually referred to as neural machine translation (NMT).[52][53] Unlike traditional autoencoders, the output does not match the input - it is in another language. In NMT, texts are treated as sequences to be encoded into the learning procedure, while on the decoder side sequences in the target language(s) are generated. Language-specific autoencoders incorporate further linguistic features into the learning procedure, such as Chinese decomposition features.[54] Machine translation is rarely still done with autoencoders, but rather transformer networks.

## See also

- Representation learning
- Sparse dictionary learning
- Deep learning

## References

1. Kramer, Mark A. (1991). "Nonlinear principal component analysis using autoassociative neural networks" (https://www.researchgate.net/profile/Abir_Alobaid/post/To_learn_a_probability_density_function_by_using_neural_network_can_we_first_estimate_density_using_nonparametric_methods_then_train_the_network/attachment/59d6450279197b80779a031e/AS:451263696510979@1484601057779/download/NL+PCA+by+using+ANN.pdf) (PDF). *AIChE Journal*. **37** (2): 233–243. doi:10.1002/aic.690370209 (https://doi.org/10.1002%2Faic.690370209).
2. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). *Deep Learning* (http://www.deeplearningbook.org). MIT Press. ISBN 978-0262035613.
3. Vincent, Pascal; Larochelle, Hugo (2010). "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". *Journal of Machine Learning Research*. **11**: 3371–3408.
4. Welling, Max; Kingma, Diederik P. (2019). "An Introduction to Variational Autoencoders". *Foundations and Trends in Machine Learning*. **12** (4): 307–392. arXiv:1906.02691 (https://arxiv.org/abs/1906.02691). Bibcode:2019arXiv190602691K (https://ui.adsabs.harvard.edu/abs/2019arXiv190602691K). doi:10.1561/2200000056 (https://doi.org/10.1561%2F2200000056). S2CID 174802445 (https://api.semanticscholar.org/CorpusID:174802445).
5. Hinton GE, Krizhevsky A, Wang SD. Transforming auto-encoders. (http://www.cs.toronto.edu/~fritz/absps/transauto6.pdf) In International Conference on Artificial Neural Networks 2011 Jun 14 (pp. 44-51). Springer, Berlin, Heidelberg.
6. Géron, Aurélien (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Canada: O'Reilly Media, Inc. pp. 739–740.
7. Liou, Cheng-Yuan; Huang, Jau-Chi; Yang, Wen-Chie (2008). "Modeling word perception using the Elman network" (http://ntur.lib.ntu.edu.tw//handle/246246/155195). *Neurocomputing*. **71** (16–18): 3150. doi:10.1016/j.neucom.2008.04.030 (https://doi.org/10.1016%2Fj.neucom.2008.04.030).
8. Liou, Cheng-Yuan; Cheng, Wei-Chen; Liou, Jiun-Wei; Liou, Daw-Ran (2014). "Autoencoder for words". *Neurocomputing*. **139**: 84–96. doi:10.1016/j.neucom.2013.09.055 (https://doi.org/10.1016%2Fj.neucom.2013.09.055).
9. Hinton, G. E.; Salakhutdinov, R.R. (2006-07-28). "Reducing the Dimensionality of Data with Neural Networks". *Science*. **313** (5786): 504–507. Bibcode:2006Sci...313..504H (https://ui.adsabs.harvard.edu/abs/2006Sci...313..504H). doi:10.1126/science.1127647 (https://doi.org/10.1126%2Fscience.1127647). PMID 16873662 (https://pubmed.ncbi.nlm.nih.gov/16873662). S2CID 1658773 (https://api.semanticscholar.org/CorpusID:1658773).
10. Bengio, Y. (2009). "Learning Deep Architectures for AI" (http://www.iro.umontreal.ca/~lisa/pointeurs/TR1312.pdf) (PDF). *Foundations and Trends in Machine Learning*. **2** (8): 1795–7. CiteSeerX 10.1.1.701.9550 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.701.9550). doi:10.1561/2200000006 (https://doi.org/10.1561%2F2200000006). PMID 23946944 (https://pubmed.ncbi.nlm.nih.gov/23946944).
11. Japkowicz, Nathalie; Hanson, Stephen José; Gluck, Mark A. (2000-03-01). "Nonlinear Autoassociation Is Not Equivalent to PCA" (https://doi.org/10.1162/089976600300015691). *Neural Computation*. **12** (3): 531–545. doi:10.1162/089976600300015691 (https://doi.org/10.1162%2F089976600300015691). ISSN 0899-7667 (https://www.worldcat.org/issn/0899-7667). PMID 10769321 (https://pubmed.ncbi.nlm.nih.gov/10769321). S2CID 18490972 (https://api.semanticscholar.org/CorpusID:18490972).
12. Schwenk, Holger; Bengio, Yoshua (1997). "Training Methods for Adaptive Boosting of Neural Networks" (https://proceedings.neurips.cc/paper/1997/hash/9cb67ffb59554ab1dabb65bcb370ddd9-Abstract.html). *Advances in Neural Information Processing Systems*. MIT Press. **10**.

13. Schmidhuber, Jürgen (January 2015). "Deep learning in neural networks: An overview". *Neural Networks*. **61**: 85–117. arXiv:1404.7828 (https://arxiv.org/abs/1404.7828). doi:10.1016/j.neunet.2014.09.003 (https://doi.org/10.1016%2Fj.neunet.2014.09.003). PMID 25462637 (https://pubmed.ncbi.nlm.nih.gov/25462637). S2CID 11715509 (https://api.semanticscholar.org/CorpusID:11715509).
14. Hinton, G. E., & Zemel, R. S. (1994). Autoencoders, minimum description length and Helmholtz free energy. In *Advances in neural information processing systems 6* (pp. 3-10).
15. Diederik P Kingma; Welling, Max (2013). "Auto-Encoding Variational Bayes". arXiv:1312.6114 (https://arxiv.org/abs/1312.6114) [stat.ML (https://arxiv.org/archive/stat.ML)].
16. Generating Faces with Torch, Boesen A., Larsen L. and Sonderby S.K., 2015 torch.ch/blog/2015/11/13/gan.html (http://torch.ch/blog/2015/11/13/gan.html)
17. Domingos, Pedro (2015). "4". *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. Basic Books. "Deeper into the Brain" subsection. ISBN 978-046506192-1.
18. Frey, Brendan; Makhzani, Alireza (2013-12-19). "k-Sparse Autoencoders". arXiv:1312.5663 (https://arxiv.org/abs/1312.5663). Bibcode:2013arXiv1312.5663M (https://ui.adsabs.harvard.edu/abs/2013arXiv1312.5663M).
19. Makhzani, Alireza; Frey, Brendan (2013). "K-Sparse Autoencoders". arXiv:1312.5663 (https://arxiv.org/abs/1312.5663) [cs.LG (https://arxiv.org/archive/cs.LG)].
20. Ng, A. (2011). Sparse autoencoder (https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf). *CS294A Lecture notes*, *72*(2011), 1-19.
21. Nair, Vinod; Hinton, Geoffrey E. (2009). "3D Object Recognition with Deep Belief Nets" (http://dl.acm.org/citation.cfm?id=2984093.2984244). *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*. NIPS'09. USA: Curran Associates Inc.: 1339–1347. ISBN 9781615679119.
22. Zeng, Nianyin; Zhang, Hong; Song, Baoye; Liu, Weibo; Li, Yurong; Dobaie, Abdullah M. (2018-01-17). "Facial expression recognition via learning deep sparse autoencoders". *Neurocomputing*. **273**: 643–649. doi:10.1016/j.neucom.2017.08.043 (https://doi.org/10.1016%2Fj.neucom.2017.08.043). ISSN 0925-2312 (https://www.worldcat.org/issn/0925-2312).
23. Hinton, Geoffrey E; Zemel, Richard (1993). "Autoencoders, Minimum Description Length and Helmholtz Free Energy" (https://proceedings.neurips.cc/paper/1993/hash/9e3cfc48eccf81a0d57663e129aef3cb-Abstract.html). *Advances in Neural Information Processing Systems*. Morgan-Kaufmann. **6**.
24. Abid, Abubakar; Balin, Muhammad Fatih; Zou, James (2019-01-27). "Concrete Autoencoders for Differentiable Feature Selection and Reconstruction". arXiv:1901.09346 (https://arxiv.org/abs/1901.09346) [cs.LG (https://arxiv.org/archive/cs.LG)].
25. Zhou, Yingbo; Arpit, Devansh; Nwogu, Ifeoma; Govindaraju, Venu (2014). "Is Joint Training Better for Deep Auto-Encoders?". arXiv:1405.1380 (https://arxiv.org/abs/1405.1380) [stat.ML (https://arxiv.org/archive/stat.ML)].
26. R. Salakhutdinov and G. E. Hinton, “Deep boltzmann machines,” in AISTATS, 2009, pp. 448–455.
27. "Fashion MNIST" (https://github.com/zalandoresearch/fashion-mnist). *GitHub*. 2019-07-12.
28. Salakhutdinov, Ruslan; Hinton, Geoffrey (2009-07-01). "Semantic hashing" (https://doi.org/10.1016%2Fj.ijar.2008.11.006). *International Journal of Approximate Reasoning*. Special Section on Graphical Models and Information Retrieval. **50** (7): 969–978. doi:10.1016/j.ijar.2008.11.006 (https://doi.org/10.1016%2Fj.ijar.2008.11.006). ISSN 0888-613X (https://www.worldcat.org/issn/0888-613X).

29. Bourlard, H.; Kamp, Y. (1988). "Auto-association by multilayer perceptrons and singular value decomposition" (http://infoscience.epfl.ch/record/82601). *Biological Cybernetics*. **59** (4–5): 291–294. doi:10.1007/BF00332918 (https://doi.org/10.1007%2FBF00332918). PMID 3196773 (https://pubmed.ncbi.nlm.nih.gov/3196773). S2CID 206775335 (https://api.semanticscholar.org/CorpusID:206775335).
30. Chicco, Davide; Sadowski, Peter; Baldi, Pierre (2014). "Deep autoencoder neural networks for gene ontology annotation predictions". *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics - BCB '14* (http://dl.acm.org/citation.cfm?id=2649442). p. 533. doi:10.1145/2649387.2649442 (https://doi.org/10.1145%2F2649387.2649442). hdl:11311/964622 (https://hdl.handle.net/11311%2F964622). ISBN 9781450328944. S2CID 207217210 (https://api.semanticscholar.org/CorpusID:207217210).
31. Plaut, E (2018). "From Principal Subspaces to Principal Components with Linear Autoencoders". arXiv:1804.10253 (https://arxiv.org/abs/1804.10253) [stat.ML (https://arxiv.org/archive/stat.ML)].
32. Morales-Forero, A., & Bassetto, S. (2019, December). Case Study: A Semi-Supervised Methodology for Anomaly Detection and Diagnosis. In *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (p. 4) (pp. 1031-1037). IEEE.
33. Sakurada, M., & Yairi, T. (2014, December). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis* (p. 4). ACM.
34. An, J., & Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, *2*, 1-18.
35. Zhou, C., & Paffenroth, R. C. (2017, August). Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 665-674). ACM.
36. Ribeiro, Manassés; Lazzaretti, André Eugênio; Lopes, Heitor Silvério (2018). "A study of deep convolutional auto-encoders for anomaly detection in videos". *Pattern Recognition Letters*. **105**: 13–22. Bibcode:2018PaReL.105...13R (https://ui.adsabs.harvard.edu/abs/2018PaReL.105...13R). doi:10.1016/j.patrec.2017.07.016 (https://doi.org/10.1016%2Fj.patrec.2017.07.016).
37. Nalisnick, Eric; Matsukawa, Akihiro; Teh, Yee Whye; Gorur, Dilan; Lakshminarayanan, Balaji (2019-02-24). "Do Deep Generative Models Know What They Don't Know?". arXiv:1810.09136 (https://arxiv.org/abs/1810.09136) [stat.ML (https://arxiv.org/archive/stat.ML)].
38. Xiao, Zhisheng; Yan, Qing; Amit, Yali (2020). "Likelihood Regret: An Out-of-Distribution Detection Score For Variational Auto-encoder" (https://proceedings.neurips.cc/paper/2020/hash/eddea82ad2755b24c4e168c5fc2ebd40-Abstract.html). *Advances in Neural Information Processing Systems*. **33**. arXiv:2003.02977 (https://arxiv.org/abs/2003.02977).
39. Theis, Lucas; Shi, Wenzhe; Cunningham, Andrew; Huszár, Ferenc (2017). "Lossy Image Compression with Compressive Autoencoders". arXiv:1703.00395 (https://arxiv.org/abs/1703.00395) [stat.ML (https://arxiv.org/archive/stat.ML)].
40. Balle, J; Laparra, V; Simoncelli, EP (April 2017). "End-to-end optimized image compression". *International Conference on Learning Representations*. arXiv:1611.01704 (https://arxiv.org/abs/1611.01704).
41. Cho, K. (2013, February). Simple sparsification improves sparse denoising autoencoders in denoising highly corrupted images. In *International Conference on Machine Learning* (pp. 432-440).

42. Cho, Kyunghyun (2013). "Boltzmann Machines and Denoising Autoencoders for Image Denoising". arXiv:1301.3468 (https://arxiv.org/abs/1301.3468) [stat.ML (https://arxiv.org/archive/stat.ML)].
43. Buades, A.; Coll, B.; Morel, J. M. (2005). "A Review of Image Denoising Algorithms, with a New One" (https://hal.archives-ouvertes.fr/hal-00271141). *Multiscale Modeling & Simulation*. **4** (2): 490–530. doi:10.1137/040616024 (https://doi.org/10.1137%2F040616024). S2CID 218466166 (https://api.semanticscholar.org/CorpusID:218466166).
44. Gondara, Lovedeep (December 2016). "Medical Image Denoising Using Convolutional Denoising Autoencoders". *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. Barcelona, Spain: IEEE: 241–246. arXiv:1608.04667 (https://arxiv.org/abs/1608.04667). Bibcode:2016arXiv160804667G (https://ui.adsabs.harvard.edu/abs/2016arXiv160804667G). doi:10.1109/ICDMW.2016.0041 (https://doi.org/10.1109%2FICDMW.2016.0041). ISBN 9781509059102. S2CID 14354973 (https://api.semanticscholar.org/CorpusID:14354973).
45. Zeng, Kun; Yu, Jun; Wang, Ruxin; Li, Cuihua; Tao, Dacheng (January 2017). "Coupled Deep Autoencoder for Single Image Super-Resolution". *IEEE Transactions on Cybernetics*. **47** (1): 27–37. doi:10.1109/TCYB.2015.2501373 (https://doi.org/10.1109%2FTCYB.2015.2501373). ISSN 2168-2267 (https://www.worldcat.org/issn/2168-2267). PMID 26625442 (https://pubmed.ncbi.nlm.nih.gov/26625442). S2CID 20787612 (https://api.semanticscholar.org/CorpusID:20787612).
46. Tzu-Hsi, Song; Sanchez, Victor; Hesham, EIDaly; Nasir M., Rajpoot (2017). "Hybrid deep autoencoder with Curvature Gaussian for detection of various types of cells in bone marrow trephine biopsy images". *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*: 1040–1043. doi:10.1109/ISBI.2017.7950694 (https://doi.org/10.1109%2FISBI.2017.7950694). ISBN 978-1-5090-1172-8. S2CID 7433130 (https://api.semanticscholar.org/CorpusID:7433130).
47. Xu, Jun; Xiang, Lei; Liu, Qingshan; Gilmore, Hannah; Wu, Jianzhong; Tang, Jinghai; Madabhushi, Anant (January 2016). "Stacked Sparse Autoencoder (SSAE) for Nuclei Detection on Breast Cancer Histopathology Images" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4729702). *IEEE Transactions on Medical Imaging*. **35** (1): 119–130. doi:10.1109/TMI.2015.2458702 (https://doi.org/10.1109%2FTMI.2015.2458702). PMC 4729702 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4729702). PMID 26208307 (https://pubmed.ncbi.nlm.nih.gov/26208307).
48. Martinez-Murcia, Francisco J.; Ortiz, Andres; Gorriz, Juan M.; Ramirez, Javier; Castillo-Barnes, Diego (2020). "Studying the Manifold Structure of Alzheimer's Disease: A Deep Learning Approach Using Convolutional Autoencoders" (https://doi.org/10.1109%2FJBHI.2019.2914970). *IEEE Journal of Biomedical and Health Informatics*. **24** (1): 17–26. doi:10.1109/JBHI.2019.2914970 (https://doi.org/10.1109%2FJBHI.2019.2914970). PMID 31217131 (https://pubmed.ncbi.nlm.nih.gov/31217131). S2CID 195187846 (https://api.semanticscholar.org/CorpusID:195187846).
49. Zhavoronkov, Alex (2019). "Deep learning enables rapid identification of potent DDR1 kinase inhibitors". *Nature Biotechnology*. **37** (9): 1038–1040. doi:10.1038/s41587-019-0224-x (https://doi.org/10.1038%2Fs41587-019-0224-x). PMID 31477924 (https://pubmed.ncbi.nlm.nih.gov/31477924). S2CID 201716327 (https://api.semanticscholar.org/CorpusID:201716327).
50. Gregory, Barber. "A Molecule Designed By AI Exhibits 'Druglike' Qualities" (https://www.wired.com/story/molecule-designed-ai-exhibits-druglike-qualities/). *Wired*.

51. De, Shaunak; Maity, Abhishek; Goel, Vritti; Shitole, Sanjay; Bhattacharya, Avik (2017). "Predicting the popularity of instagram posts for a lifestyle magazine using deep learning". *2017 2nd IEEE International Conference on Communication Systems, Computing and IT Applications (CSCITA)*. pp. 174–177. doi:10.1109/CSCITA.2017.8066548 (https://doi.org/10.1109%2FCSCITA.2017.8066548). ISBN 978-1-5090-4381-1. S2CID 35350962 (https://api.semanticscholar.org/CorpusID:35350962).
52. Cho, Kyunghyun; Bart van Merrienboer; Bahdanau, Dzmitry; Bengio, Yoshua (2014). "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". arXiv:1409.1259 (https://arxiv.org/abs/1409.1259) [cs.CL (https://arxiv.org/archive/cs.CL)].
53. Sutskever, Ilya; Vinyals, Oriol; Le, Quoc V. (2014). "Sequence to Sequence Learning with Neural Networks". arXiv:1409.3215 (https://arxiv.org/abs/1409.3215) [cs.CL (https://arxiv.org/archive/cs.CL)].
54. Han, Lifeng; Kuang, Shaohui (2018). "Incorporating Chinese Radicals into Neural Machine Translation: Deeper Than Character Level". arXiv:1805.01565 (https://arxiv.org/abs/1805.01565) [cs.CL (https://arxiv.org/archive/cs.CL)].

---