Title: Autoencoder

URL: https://en.wikipedia.org/wiki/Autoencoder

PageID: 6836612

Categories: Category:Dimension reduction, Category:Neural network architectures, Category:Unsupervised learning

---

Supervised learning

Unsupervised learning

Semi-supervised learning

Self-supervised learning

Reinforcement learning

Meta-learning

Online learning

Batch learning

Curriculum learning

Rule-based learning

Neuro-symbolic AI

Neuromorphic engineering

Quantum machine learning

Classification

Generative modeling

Regression

Clustering

Dimensionality reduction

Density estimation

Anomaly detection

Data cleaning

AutoML

Association rules

Semantic analysis

Structured prediction

Feature engineering

Feature learning

Learning to rank

Grammar induction

Ontology learning

Multimodal learning

Apprenticeship learning

Decision trees

Ensembles Bagging Boosting Random forest

Bagging

Boosting

Random forest

k -NN

Linear regression

Naive Bayes

Artificial neural networks

Logistic regression

Perceptron

Relevance vector machine (RVM)

Support vector machine (SVM)

BIRCH

CURE

Hierarchical

k -means

Fuzzy

Expectation–maximization (EM)

DBSCAN

OPTICS

Mean shift

Factor analysis

CCA

ICA

LDA

NMF

PCA

PGD

t-SNE

SDL

Graphical models Bayes net Conditional random field Hidden Markov

Bayes net

Conditional random field

Hidden Markov

RANSAC

k -NN

Local outlier factor

Isolation forest

Autoencoder

Deep learning

Feedforward neural network

Recurrent neural network LSTM GRU ESN reservoir computing

LSTM

GRU

ESN

reservoir computing

Boltzmann machine Restricted

Restricted

GAN

Diffusion model

SOM

Convolutional neural network U-Net LeNet AlexNet DeepDream

U-Net

LeNet

AlexNet

DeepDream

Neural field Neural radiance field Physics-informed neural networks

Neural radiance field

Physics-informed neural networks

Transformer Vision

Vision

Mamba

Spiking neural network

Memtransistor

Electrochemical RAM (ECRAM)

Q-learning

Policy gradient

SARSA

Temporal difference (TD)

Multi-agent Self-play

Self-play

Active learning

Crowdsourcing

Human-in-the-loop

Mechanistic interpretability

RLHF

Coefficient of determination

Confusion matrix

Learning curve

ROC curve

Kernel machines

Bias–variance tradeoff

Computational learning theory

Empirical risk minimization

Occam learning

PAC learning

Statistical learning

VC theory

Topological deep learning

AAAI

ECML PKDD

NeurIPS

ICML

ICLR

IJCAI

ML

JMLR

Glossary of artificial intelligence

List of datasets for machine-learning research List of datasets in computer vision and image processing

List of datasets in computer vision and image processing

Outline of machine learning

v

t

e

An autoencoder is a type of artificial neural network used to learn efficient codings of unlabeled data (unsupervised learning). An autoencoder learns two functions: an encoding function that transforms the input data, and a decoding function that recreates the input data from the encoded representation. The autoencoder learns an efficient representation (encoding) for a set of data, typically for dimensionality reduction , to generate lower-dimensional embeddings for subsequent use by other machine learning algorithms. [1]

Variants exist which aim to make the learned representations assume useful properties. [2] Examples are regularized autoencoders (sparse , denoising and contractive autoencoders), which are effective in learning representations for subsequent classification tasks, [3] and variational autoencoders , which can be used as generative models . [4] Autoencoders are applied to many

problems, including facial recognition , [5] feature detection , [6] anomaly detection , and learning the meaning of words . [7] [8] In terms of data synthesis , autoencoders can also be used to randomly generate new data that is similar to the input (training) data. [6]

Mathematical principles

Definition

An autoencoder is defined by the following components:

Two sets: the space of decoded messages X {\displaystyle {\mathcal {X}}} ; the space of encoded messages Z {\displaystyle {\mathcal {Z}}} . Typically X {\displaystyle {\mathcal {X}}} and Z {\displaystyle {\mathcal {Z}}} are Euclidean spaces , that is, X = R m , Z = R n {\displaystyle {\mathcal {X}}=\mathbb {R} ^{m},{\mathcal {Z}}=\mathbb {R} ^{n}} with m > n . {\displaystyle m>n.}

Two parametrized families of functions: the encoder family E φ : X → Z {\displaystyle E_{\phi }:{\mathcal {X}}\rightarrow {\mathcal {Z}}} , parametrized by φ {\displaystyle \phi } ; the decoder family D θ : Z → X {\displaystyle D_{\theta }:{\mathcal {Z}}\rightarrow {\mathcal {X}}} , parametrized by θ {\displaystyle \theta } .

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

Usually, both the encoder and the decoder are defined as multilayer perceptrons (MLPs). For example, a one-layer-MLP encoder E φ {\displaystyle E_{\phi }} is:

where σ {\displaystyle \sigma } is an element-wise activation function , W {\displaystyle W} is a "weight" matrix, and b {\displaystyle b} is a "bias" vector.

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 μ r e f {\displaystyle \mu _{ref}} over X {\displaystyle {\mathcal {X}}} , and a "reconstruction quality" function d : X × X → [0 , ∞] {\displaystyle d:{\mathcal {X}}\times {\mathcal {X}}\to [0,\infty]} , such that d (x , x ′) {\displaystyle d(x,x')} measures how much x ′ {\displaystyle x'} differs from x {\displaystyle x} .

With those, we can define the loss function for the autoencoder as L (θ , φ) := E x ~ μ r e f [d (x , D θ (E φ (x)))] {\displaystyle L(\theta ,\phi):=\mathbb {\mathbb {E} } _{x\sim \mu _{ref}}[d(x,D_{\theta }(E_{\phi }(x)))]} The optimal autoencoder for the given task (μ r e f , d) {\displaystyle (\mu _{ref},d)} is then arg ■ min θ , φ L (θ , φ) {\displaystyle \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 , . . . , x N } ⊂ X {\displaystyle \{x_{1},...,x_{N}\}\subset {\mathcal {X}}} , so that μ r e f = 1 N ∑ i = 1 N δ x i {\displaystyle \mu _{ref}={\frac {1}{N}}\sum _{i=1}^{N}\delta _{x_{i}}}

where δ x i {\displaystyle \delta _{x_{i}}} is the Dirac measure , the quality function is just L2 loss: d (x , x ′) = ■ x – x ′ ■ 2 2 {\displaystyle d(x,x')=\|x-x'\|_{2}^{2}} , and ■ · ■ 2 {\displaystyle \|\cdot \|_{2}} is the Euclidean norm . Then the problem of searching for the optimal autoencoder is just a least-squares optimization: min θ , φ L (θ , φ) , where L (θ , φ) = 1 N ∑ i = 1 N ■ x i – D θ (E φ (x i)) ■ 2 2 {\displaystyle \min _{\theta ,\phi }L(\theta ,\phi),\qquad {\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 {\displaystyle d} .

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

Such an autoencoder is called undercomplete . It can be interpreted as compressing the message, or reducing its dimensionality . [9] [10]

At the limit of an ideal undercomplete autoencoder, every possible code z {\displaystyle z} in the code space is used to encode a message x {\displaystyle x} that really appears in the distribution μ r e f {\displaystyle \mu _{ref}} , and the decoder is also perfect: D θ (E φ (x)) = x {\displaystyle 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 {\displaystyle z} and obtaining D θ (z) {\displaystyle D_{\theta }(z)} , which is a message that really appears in the distribution μ r e f {\displaystyle \mu _{ref}} .

If the code space Z {\displaystyle {\mathcal {Z}}} has dimension larger than (overcomplete), or equal to, the message space X {\displaystyle {\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 . [11]

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]

Variations

Variational autoencoder (VAE)

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

Given an input dataset x {\displaystyle x} characterized by an unknown probability function P (x) {\displaystyle P(x)} and a multivariate latent encoding vector z {\displaystyle z} , the objective is to model the data as a distribution p θ (x) {\displaystyle p_{\theta }(x)} , with θ {\displaystyle \theta } defined as the set of the network parameters so that p θ (x) = ∫ z p θ (x , z) d z {\displaystyle p_{\theta }(x)=\int _{z}p_{\theta }(x,z)dz} .

Sparse autoencoder (SAE)

Inspired by the sparse coding hypothesis in neuroscience, sparse autoencoders (SAE) are variants of autoencoders, such that the codes E φ (x) {\displaystyle E_{\phi }(x)} for messages tend to be sparse codes , that is, E φ (x) {\displaystyle E_{\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. [12] Encouraging sparsity improves performance on classification tasks. [13]

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 . [13]

The k-sparse autoencoder inserts the following "k-sparse function" in the latent layer of a standard autoencoder: f k (x 1 , . . . , x n) = (x 1 b 1 , . . . , x n b n) {\displaystyle f_{k}(x_{1},...,x_{n})=(x_{1}b_{1},...,x_{n}b_{n})} where b i = 1 {\displaystyle b_{i}=1} if | x i | {\displaystyle |x_{i}|} ranks in the top k, and 0 otherwise.

Backpropagating through f k {\displaystyle f_{k}} is simple: set gradient to 0 for b i = 0 {\displaystyle b_{i}=0} entries, and keep gradient for b i = 1 {\displaystyle b_{i}=1} entries. This is essentially a generalized ReLU function. [13]

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 θ , φ L (θ , φ) + λ L sparse (θ , φ) {\displaystyle \min _{\theta ,\phi }L(\theta ,\phi)+\lambda L_{\text{sparse}}(\theta ,\phi)} where λ > 0 {\displaystyle \lambda >0} measures how much sparsity we want to enforce. [14]

Let the autoencoder architecture have K {\displaystyle K} layers. To define a sparsity regularization loss, we need a "desired" sparsity ρ ^ k {\displaystyle {\hat {\rho }}_{k}} for each layer, a weight w k {\displaystyle w_{k}} for how much to enforce each sparsity, and a function s : [0 , 1] × [0 , 1] → [0 , ∞] {\displaystyle s:[0,1]\times [0,1]\to [0,\infty]} to measure how much two sparsities differ.

For each input x {\displaystyle x} , let the actual sparsity of activation in each layer k {\displaystyle k} be ρ k (x) = 1 n ∑ i = 1 n a k , i (x) {\displaystyle \rho _{k}(x)={\frac {1}{n}}\sum _{i=1}^{n}a_{k,i}(x)} where a k , i (x) {\displaystyle a_{k,i}(x)} is the activation in the i {\displaystyle i} -th neuron of the k {\displaystyle k} -th layer upon input x {\displaystyle x} .

The sparsity loss upon input x {\displaystyle x} for one layer is s (ρ ^ k , ρ k (x)) {\displaystyle 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 sparse (θ , φ) = E x ~ μ X [∑ k ∈ 1 : K w k s (ρ ^ k , ρ k (x))] {\displaystyle L_{\text{sparse}}(\theta ,\phi)=\mathbb {\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 {\displaystyle s} is either the Kullback-Leibler (KL) divergence , as [13] [14] [15] [16]

or the L1 loss, as s (ρ , ρ ^) = | ρ − ρ ^ | {\displaystyle s(\rho ,{\hat {\rho }})=|\rho -{\hat {\rho }}|} , or the L2 loss, as s (ρ , ρ ^) = | ρ − ρ ^ | 2 {\displaystyle 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 define the sparsity regularization loss as L sparse (θ , φ) = E x ~ μ X [∑ k ∈ 1 : K w k ■ h k ■] {\displaystyle L_{\text{sparse}}(\theta ,\phi)=\mathbb {\mathbb {E} } _{x\sim \mu _{X}}\left[\sum _{k\in 1:K}w_{k}\|h_{k}\|\right]} where h k {\displaystyle h_{k}} is the activation vector in the k {\displaystyle k} -th layer of the autoencoder. The norm ■ · ■ {\displaystyle \|\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, originally called a "robust autoassociative network" by Mark A. Kramer, [17] is trained by intentionally corrupting the inputs of a standard autoencoder during training. A noise process is defined by a probability distribution μ T {\displaystyle \mu _{T}} over functions T : X → X {\displaystyle T:{\mathcal {X}}\to {\mathcal {X}}} . That is, the function T {\displaystyle T} takes a message x ∈ X {\displaystyle x\in {\mathcal {X}}} , and corrupts it to a noisy version T (x) {\displaystyle T(x)} . The function T {\displaystyle T} is selected randomly, with a probability distribution μ T {\displaystyle \mu _{T}} .

Given a task (μ ref , d) {\displaystyle (\mu _{\text{ref}},d)} , the problem of training a DAE is the optimization problem: min θ , φ L (θ , φ) = E x ~ μ X , T ~ μ T [d (x , (D θ ■ E φ ■ T) (x))] {\displaystyle \min _{\theta ,\phi }L(\theta ,\phi)=\mathbb {\mathbb {E} } _{x\sim \mu _{X},T\sim \mu _{T}}[d(x,(D_{\theta }\circ E_{\phi }\circ T)(x))]} 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 {\displaystyle 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 (CAE) adds the contractive regularization loss to the standard autoencoder loss: min θ , φ L (θ , φ) + λ L cont (θ , φ) {\displaystyle \min _{\theta ,\phi }L(\theta ,\phi)+\lambda L_{\text{cont}}(\theta ,\phi)} where λ > 0 {\displaystyle \lambda >0} measures how much contractive-ness we want to enforce. The contractive regularization loss itself is defined as the expected square of Frobenius norm of the Jacobian matrix of the encoder activations with respect to the input: L cont (θ , φ) = E x ∼ μ r e f ■ ∇ x E φ (x) ■ F 2 {\displaystyle L_{\text{cont}}(\theta ,\phi)=\mathbb {E} _{x\sim \mu _{ref}}\|\nabla _{x}E_{\phi }(x)\|_{F}^{2}} To understand what L cont {\displaystyle L_{\text{cont}}} measures, note the fact ■ E φ (x + δ x) − E φ (x) ■ 2 ≤ ■ ∇ x E φ (x) ■ F ■ δ x ■ 2 {\displaystyle \|E_{\phi }(x+\delta x)-E_{\phi }(x)\|_{2}\leq \|\nabla _{x}E_{\phi }(x)\|_{F}\|\delta x\|_{2}} for any message x ∈ X {\displaystyle x\in {\mathcal {X}}} , and small variation δ x {\displaystyle \delta x} in it. Thus, if ■ ∇ x E φ (x) ■ F 2 {\displaystyle \|\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.

Minimum description length autoencoder (MDL-AE)

A minimum description length autoencoder (MDL-AE) is an advanced variation of the traditional autoencoder, which leverages principles from information theory, specifically the Minimum Description Length (MDL) principle . The MDL principle posits that the best model for a dataset is the one that provides the shortest combined encoding of the model and the data. In the context of autoencoders , this principle is applied to ensure that the learned representation is not only compact but also interpretable and efficient for reconstruction.

The MDL-AE seeks to minimize the total description length of the data, which includes the size of the latent representation (code length) and the error in reconstructing the original data. The objective can be expressed as L code + L error {\displaystyle L_{\text{code}}+L_{\text{error}}} , where L code {\displaystyle L_{\text{code}}} represents the length of the compressed latent representation and L error {\displaystyle L_{\text{error}}} denotes the reconstruction error. [18]

Concrete autoencoder (CAE)

The concrete autoencoder is designed for discrete feature selection. [19] 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.

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.

Depth can exponentially decrease the amount of training data needed to learn some functions.

Experimentally, deep autoencoders yield better compression compared to shallow or linear autoencoders. [10]

Training

Geoffrey Hinton developed the deep belief network technique for training many-layered deep autoencoders. His method involves treating each neighboring set of two layers as a restricted

Boltzmann machine so that pretraining approximates a good solution, then using backpropagation to fine-tune the results. [10]

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. [20] A 2015 study showed that joint training learns better data models along with more representative features for classification as compared to the layerwise method. [20] However, their experiments showed that the success of joint training depends heavily on the regularization strategies adopted. [20] [21]

History

(Oja, 1982) [22] noted that PCA is equivalent to a neural network with one hidden layer with identity activation function. In the language of autoencoding, the input-to-hidden module is the encoder, and the hidden-to-output module is the decoder. Subsequently, in (Baldi and Hornik, 1989) [23] and (Kramer, 1991) [9] generalized PCA to autoencoders, a technique which they termed "nonlinear PCA".

Immediately after the resurgence of neural networks in the 1980s, it was suggested in 1986 [24] that a neural network be put in "auto-association mode". This was then implemented in (Harrison, 1987) [25] and (Elman, Zipser, 1988) [26] for speech and in (Cottrell, Munro, Zipser, 1987) [27] for images. [28] In (Hinton, Salakhutdinov, 2006), [29] deep belief networks were developed. These train a pair restricted Boltzmann machines as encoder-decoder pairs, then train another pair on the latent representation of the first pair, and so on. [30]

The first applications of AE date to early 1990s. [2] [31] [18] Their most traditional application was dimensionality reduction or feature learning , but the concept became widely used for learning generative models of data. [32] [33] Some of the most powerful AIs in the 2010s involved autoencoder modules as a component of larger AI systems, such as VAE in Stable Diffusion , discrete VAE in Transformer-based image generators like DALL-E 1 , etc.

During the early days, when the terminology was uncertain, the autoencoder has also been called identity mapping, [23] [9] auto-associating, [34] self-supervised backpropagation , [9] or Diabolo network. [35] [11]

Applications

The two main applications of autoencoders are dimensionality reduction and information retrieval (or associative memory), [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, [10] 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] [10]

Reducing 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. [37]

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). [28] [38] The weights of an autoencoder with a single hidden layer of size p {\displaystyle p} (where p {\displaystyle p} is less than the size of the input) span the same vector subspace as the one spanned by the first p {\displaystyle 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 . [39]

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. [10]

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. [37] 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 . [17] [40] [41] [42] [43] [44] 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. [42] Reconstruction error (the error between the original data and its low dimensional reconstruction) is used as an anomaly score to detect anomalies. [42] Typically, this means that on a validation set the empirical distribution of reconstruction errors is recorded and then (e.g.) the empirical 95-percentile x p {\displaystyle x_{p}} is taken as threshold t := x p {\displaystyle t:=x_{p}} to flag anomalous data points: loss (x , reconstruction (x)) > t ■ anomaly {\displaystyle {\text{loss}}(x,{\text{reconstruction}}(x))>t\implies {\text{anomaly}}} . Since the threshold is an empirical quantile estimate, there is an inherent difficulty with "correctly" setting this threshold:

In many cases the distribution of the empirical quantile is asymptotically a normal distribution empirical p-quantile ~ N (μ = p , σ 2 = p (1 − p) n f (x p) 2) , {\displaystyle {\text{empirical p-quantile}}\sim {\mathcal {N}}\left(\mu =p,\sigma ^{2}={\frac {p(1-p)}{nf(x_{p})^{2}}}\right),} with f (x p) {\displaystyle f(x_{p})} the probability density at the quantile. This means that the variance grows if an extreme quantile is considered (because f (x p) {\displaystyle f(x_{p})} is small there). This means that there is a, potentially, a big uncertainty in what is the right choice for the threshold since it is estimated from a validation set.

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. [45] [46] Intuitively, this can be understood by considering those one layer auto encoders which are related to PCA - also in this case there can be perfect rein reconstructions for points which are far away from the data region but which lie on a principal component axis.

It is best to analyze if the anomalies which are flagged by the auto encoder are true anomalies. In this sense all the metrics in Evaluation of binary classifiers can be considered. The fundamental challenge which comes with the unsupervised (self-supervised) learning setting is, that labels for rare events do not exist (in which case the labels first have to be gathered and the data set will be imbalanced) or anomaly indicating labels are very rare, introducing larger confidence intervals for these performance estimates.

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 . [47] [48]

Another useful application of autoencoders in image preprocessing is image denoising . [49] [50] [51]

Autoencoders found use in more demanding contexts such as medical imaging where they have been used for image denoising [52] as well as super-resolution . [53] [54] In image-assisted diagnosis, experiments have applied autoencoders for breast cancer detection [55] and for modelling the relation between the cognitive decline of Alzheimer's disease and the latent features of an autoencoder trained with MRI . [56]

Drug discovery

In 2019 molecules generated with variational autoencoders were validated experimentally in mice. [57] [58]

Popularity prediction

Recently, a stacked autoencoder framework produced promising results in predicting popularity of social media posts, [59] 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). [60] [61] 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. [62] Machine translation is rarely still done with autoencoders, due to the availability of more effective transformer networks.

Communication Systems

Autoencoders in communication systems are important because they help in encoding data into a more resilient representation for channel impairments, which is crucial for transmitting information while minimizing errors. In Addition, AE-based systems can optimize end-to-end communication performance. This approach can solve the several limitations of designing communication systems such as the inherent difficulty in accurately modeling the complex behavior of real-world channels. [63]

See also

Representation learning

Singular value decomposition

Sparse dictionary learning

Deep learning

Further reading

Bank, Dor; Koenigstein, Noam; Giryes, Raja (2023). "Autoencoders". Machine Learning for Data Science Handbook . Cham: Springer International Publishing. doi : 10.1007/978-3-031-24628-9_16 . ISBN 978-3-031-24627-2 .

Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "14. Autoencoders" . Deep learning . Adaptive computation and machine learning. Cambridge, Mass: The MIT press. ISBN 978-0-262-03561-3 .

References

v

t

e

History timeline

timeline

Companies

Projects

Parameter Hyperparameter

Hyperparameter

Loss functions

Regression Bias–variance tradeoff Double descent Overfitting

Bias–variance tradeoff

Double descent

Overfitting

Clustering

Gradient descent SGD Quasi-Newton method Conjugate gradient method

SGD

Quasi-Newton method

Conjugate gradient method

Backpropagation

Attention

Convolution

Normalization Batchnorm

Batchnorm

Activation Softmax Sigmoid Rectifier

Softmax

Sigmoid

Rectifier

Gating

Weight initialization

Regularization

Datasets Augmentation

Augmentation

Prompt engineering

Reinforcement learning Q-learning SARSA Imitation Policy gradient

Q-learning

SARSA

Imitation

Policy gradient

Diffusion

Latent diffusion model

Autoregression

Adversary

RAG

Uncanny valley

RLHF

Self-supervised learning

Reflection

Recursive self-improvement

Hallucination

Word embedding

Vibe coding

Machine learning In-context learning

In-context learning

Artificial neural network Deep learning

Deep learning

Language model Large language model NMT

Large language model

NMT

Reasoning language model

Model Context Protocol

Intelligent agent

Artificial human companion

Humanity's Last Exam

Artificial general intelligence (AGI)

AlexNet

WaveNet

Human image synthesis

HWR

OCR

Computer vision

Speech synthesis 15.ai ElevenLabs

15.ai

ElevenLabs

Speech recognition Whisper

Whisper

Facial recognition

AlphaFold

Text-to-image models Aurora DALL-E Firefly Flux Ideogram Imagen Midjourney Recraft Stable Diffusion

Aurora

DALL-E

Firefly

Flux

Ideogram

Imagen

Midjourney

Recraft

Stable Diffusion

Text-to-video models Dream Machine Runway Gen Hailuo AI Kling Sora Veo

Dream Machine

Runway Gen

Hailuo AI

Kling

Sora

Veo

Music generation Riffusion Suno AI Udio

Riffusion

Suno AI

Udio

Word2vec

Seq2seq

GloVe

BERT

T5

Llama

Chinchilla AI

PaLM

GPT 1 2 3 J ChatGPT 4 4o o1 o3 4.5 4.1 o4-mini 5

1

2

3

J

ChatGPT

4

4o

o1

o3

4.5

4.1

o4-mini

5

Claude

Gemini Gemini (language model) Gemma

Gemini (language model)

Gemma

Grok

LaMDA

BLOOM

DBRX

Project Debater

IBM Watson

IBM Watsonx

Granite

PanGu-Σ

DeepSeek

Qwen

AlphaGo

AlphaZero

OpenAI Five

Self-driving car

MuZero

Action selection AutoGPT

AutoGPT

Robot control

Alan Turing

Warren Sturgis McCulloch

Walter Pitts

John von Neumann

Claude Shannon

Shun'ichi Amari

Kunihiko Fukushima

Takeo Kanade

Marvin Minsky

John McCarthy

Nathaniel Rochester

Allen Newell
Cliff Shaw
Herbert A. Simon
Oliver Selfridge
Frank Rosenblatt
Bernard Widrow
Joseph Weizenbaum
Seymour Papert
Seppo Linnainmaa
Paul Werbos
Geoffrey Hinton
John Hopfield
Jürgen Schmidhuber
Yann LeCun
Yoshua Bengio
Lotfi A. Zadeh
Stephen Grossberg
Alex Graves
James Goodnight
Andrew Ng
Fei-Fei Li
Alex Krizhevsky
Ilya Sutskever
Oriol Vinyals
Quoc V. Le
Ian Goodfellow
Demis Hassabis
David Silver
Andrej Karpathy
Ashish Vaswani
Noam Shazeer
Aidan Gomez
John Schulman
Mustafa Suleyman
Jan Leike
Daniel Kokotajlo
François Chollet
Neural Turing machine

Differentiable neural computer

Transformer Vision transformer (ViT)

Vision transformer (ViT)

Recurrent neural network (RNN)

Long short-term memory (LSTM)

Gated recurrent unit (GRU)

Echo state network

Multilayer perceptron (MLP)

Convolutional neural network (CNN)

Residual neural network (RNN)

Highway network

Mamba

Autoencoder

Variational autoencoder (VAE)

Generative adversarial network (GAN)

Graph neural network (GNN)

Category

v

t

e

Acoustic quieting

Distortion

Noise cancellation

Noise control

Noise measurement

Noise power

Noise reduction

Noise temperature

Phase distortion

Audio

Buildings

Electronics

Environment

Government regulation

Human health

Images

Radio

Rooms

Ships
Sound masking
Transportation
Video
Additive white Gaussian noise (AWGN)
Atmospheric noise
Background noise
Brownian noise
Burst noise
Cosmic noise
Flicker noise
Gaussian noise
Grey noise
Infrasound
Jitter
Johnson–Nyquist noise (thermal noise)
Pink noise
Quantization error (or q. noise)
Shot noise
White noise
Coherent noise Value noise Gradient noise Worley noise
Value noise
Gradient noise
Worley noise
Channel noise level
Circuit noise level
Effective input noise temperature
Equivalent noise resistance
Equivalent pulse code modulation noise
Impulse noise (audio)
Noise figure
Noise floor
Noise shaping
Noise spectral density
Noise, vibration, and harshness (NVH)
Phase noise
Pseudorandom noise
Statistical noise

Carrier-to-noise ratio (C / N)
Carrier-to-receiver noise density (C / kT)
dBrnC
E b /N 0 (energy per bit to noise density)
E s /N 0 (energy per symbol to noise density)
Modulation error ratio (MER)
Signal, noise and distortion (SINAD)
Signal-to-interference ratio (S / I)
Signal-to-noise ratio (S / N , SNR)
Signal-to-noise ratio (imaging)
Signal-to-interference-plus-noise ratio (SINR)
Signal-to-quantization-noise ratio (SQNR)
Contrast-to-noise ratio (CNR)
List of noise topics
Acoustics
Colors of noise
Interference (communication)
Noise generator
Spectrum analyzer
Thermal radiation
Low-pass filter
Median filter
Total variation denoising
Wavelet denoising
Gaussian blur
Anisotropic diffusion
Bilateral filter
Non-local means
Block-matching and 3D filtering (BM3D)
Shrinkage Fields
Denoising autoencoder (DAE)
Deep Image Prior