https://www.nature.com/articles/nature14539

* Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned.
* The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure
* very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government.
* In a typical deep-learning system, there may be hundreds of millions of these adjustable weights, and hundreds of millions of labelled examples with which to train the machine
* The negative gradient vector indicates the direction of steepest descent in this landscape, taking it closer to a minimum, where the output error is low on average
* A deep-learning architecture is a multilayer stack of simple modules, all (or most) of which are subject to learning, and many of which compute non-linear input–output mappings. Each module in the stack transforms its input to increase both the selectivity and the invariance of the representation. With multiple non-linear layers, say a depth of 5 to 20, a system can implement extremely intricate functions of its inputs that are simultaneously sensitive to minute details
* The key insight is that the derivative (or gradient) of the objective with respect to the input of a module can be computed by working backwards from the gradient with respect to the output of that module (or the input of the subsequent module) (Fig. 1). The backpropagation equation can be applied repeatedly to propagate gradients through all modules, starting from the output at the top (where the network produces its prediction) all the way to the bottom (where the external input is fed). Once these gradients have been computed, it is straightforward to compute the gradients with respect to the weights of each module
* Many applications of deep learning use feedforward neural network architectures (Fig. 1), which learn to map a fixed-size input (for example, an image) to a fixed-size output (for example, a probability for each of several categories). To go from one layer to the next, a set of units compute a weighted sum of their inputs from the previous layer and pass the result through a non-linear function. At present, the most popular non-linear function is the rectified linear unit (ReLU), which is simply the half-wave rectifier f(z) = max(z, 0). In past decades, neural nets used smoother non-linearities, such as tanh(z) or 1/(1+exp(−z)), but the ReLU typically learns much faster in networks with many layers, allowing training of a deep supervised network without unsupervised pre-training28. Units that are not in the input or output layer are conventionally called hidden units. The hidden layers can be seen as distorting the input in a non-linear way so that categories become linearly separable by the last layer (Fig. 1).
* Convolutional neural networks ConvNets are designed to process data that come in the form of multiple arrays, for example a colour image composed of three 2D arrays containing pixel intensities in the three colour channels
* Deep neural networks exploit the property that many natural signals are compositional hierarchies, in which higher-level features are obtained by composing lower-level ones
* Recurrent neural networks When backpropagation was first introduced, its most exciting use was for training recurrent neural networks (RNNs). For tasks that involve sequential inputs, such as speech and language, it is often better to use RNNs (Fig. 5). RNNs process an input sequence one element at a time, maintaining in their hidden units a ‘state vector’ that implicitly contains information about the history of all the past elements of the sequence.
  + RNNs are very powerful dynamic systems, but training them has proved to be problematic because the backpropagated gradients either grow or shrink at each time step, so over many time steps they typically explode or vanish

https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2020.00004/full

* The basic idea of a neuron model is that an input, **x**, together with a bias, *b* is weighted by, **w**, and then summarized together. The bias, *b*, is a scalar value whereas the input **x** and the weights **w** are vector valued, i.e., **x** ∈ ℝ*n* and **w** ∈ ℝ*n* with *n* ∈ ℕ corresponding to the dimension of the input. Note that the bias term is not always present but is sometimes omitted. The sum of these terms, i.e., *z* = **w***T***x** + *b* forms then the argument of an activation function, ϕ, resulting in the output of the neuron mode
  + y=ϕ(z)=ϕ(wTx+b).
* The activation function, ϕ, (also known as unit function or transfer function) performs a non-linear transformation of *z*.
* In order to build neural networks (NNs), the neurons need to be connected with each other. The simplest architecture of a NN is a feedforward structure.
* The required number to call a Feedforward Neural Network (FFNN) architecture deep is debatable, but architectures with more than two hidden layers are commonly considered as deep
* It can be proven that a Feedforward Neural Network with one hidden layer and a finite number of neurons can approximate any continuous function on a compact subset of ℝ*n* (Hornik, [**1991**](https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2020.00004/full#B65)). This is called the *universal approximation theorem*. The reason for using a FFNN with more than one hidden layer is that the universal approximation theorem does not provide information on how to learn such a network, which turned out to be very difficult. A related issue that contributes to the difficulty of learning such networks is that their width can become exponentially large. Interestingly, the universal approximation theorem can also be proven for FFNN with many hidden layers and a bounded number of hidden neurons (Lu et al., [**2017**](https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2020.00004/full#B90)) for which learning algorithms have been found. Hence, D-FFNNs are used instead of (shallow) FFNNs for practical reasons of learnability.
* Instead of *guessing* the correct family of functions from which ***f*** should be chosen, D-FFNNs learn this function by approximating it via ***ϕ***, which itself is approximated by the *n* hidden layers.
* Typically, in traditional ANNs, each neuron in a layer is connected to all neurons in the next layer, whereas each connection is a parameter in the network. This can result in a very large number of parameters. Instead of using fully connected layers, a CNN uses a local connectivity between neurons, i.e., a neuron is only connected to nearby neurons in the next layer. This can significantly reduce the total number of parameters in the network.

https://arxiv.org/pdf/2206.01640

* Missing and incomplete data are abundant in real-world problems; however, the learning and inference procedures in machine learning (ML) models highly rely on high-quality and complete data. Therefore, it is necessary to develop new methods to deal with data imperfections in rugged environments. Currently, the most popular way to deal with imperfect data is to impute the missing values. However, if we consider the learning and inference procedures in our brain as a role model for ML algorithms, data imputation barely follows the natural principles of incomplete data processing in our brain. This is because the imputation is generally based on using a heuristic for replacing missing values. Our brain does not impute incomplete sensory information but instead uses its incompleteness as a separate source of information for decision making
* Neural networks (NNs) are brain-inspired algorithms that are very popular these days (under the name of deep learning) for learning complex relationships between inputs and target variables.
* they are in principle unable to handle incomplete data with missing values. They mainly rely on matrix operations which cannot operate on not-a-number (NaN) values. Only one NaN in a dataset impairs the forward propagation in a network. There are three solutions to this problem (Garc´ıaLaencina et al., 2010): i) removing samples or features with missing values, ii) imputing the missing values, and iii) modeling the incomplete data. Removing the samples with missing values can be very costly, especially in small-sample size and high-dimensional datasets. For example, data collection in clinical applications is an expensive procedure in terms of time, finance, and patient burden. Moreover, removing even a few samples from small datasets can affect negatively the generalization performance of the final model. Removing informative features with missing values is also compensated with lower model performance. Therefore, filling the information gaps is inevitable

https://watermark02.silverchair.com/kwad107.pdf?token=AQECAHi208BE49Ooan9kkhW\_Ercy7Dm3ZL\_9Cf3qfKAc485ysgAAA28wggNrBgkqhkiG9w0BBwagggNcMIIDWAIBADCCA1EGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMAclzUiLd1au3nq5nAgEQgIIDIkewNyV1gGu3bPtp3L6r1uw4uoMR2bPJD9Mupxpn0kjVkK0GIVysbMnpDflDytzNXrvkimxFwQD5wi2skHgiCEC3Z3Sue-2md0G\_\_DPHyEn3b9ZbMpXs1xQF0UujUJvpFLGgtmreoq5icDAhqK2exifUiQjQSg9gH7v0TdTcFJyfaiFDrjoy7\_XWyL7zTDZIlSUMyCjQTngqgr4744S3Ib66gAFppMO50dbIQYGE9NWNeo4Wy4GI3pMa-OnDJZFEGAoCamXLmIeYXiXAL3QXgiWDW8CskgY7KEJ9eL-632WoosYvavqM0AZ7-UHgWiEkn2sK4Zz7d0Ky27fK7aLjgLCLFuxdtMesl9gWER\_JbXw\_6Belsm\_s0BCyzC-GVjrV4mFJiw8x95uK41WV-sKMwbo24NQlzR8s7eFFPKmSjLld9IvxGmGiDlxIP42jYc9FIImoBphC1akufNpu2pRU1uTfURjLzgCfX8zbLrMz1h\_-v1YPRZORCV8m-G0NA5otSzGcTnAYdO88eLVB9lsbGxcCSzpYav4Bj63LA\_KU7Q5nJL2P51mf-03Xk06zSZeUzMXjOhtwIw6MnBvgu9BjtLDpocC4C-iAfN0l4-C0r\_TygXATk6cZpXP3ucdVvf7AY3uWZGC2ICN67W8Ncs1A4QWywX-gQUqckTNbxbMDxIob7EqYfp0NPaWUTBdY16fIRIkggMbB1bN2LDxVN5LMJfsOiRmCaoGLGBjy\_H\_FYZeFBQRxhdCxbc5sldUEe6SOGWwXdXPriLiNGjh001ZJiI0DAejvk-Giopj4\_WnRQo\_SwKtKrExNYy1jxMAK7WEyQhqn-cn8m8gxv1LvnJ\_wxik5QkwzgfBxjb8NULg0duEohJkZsuXBNOrpNT\_gbz\_6gFie6r6RBvpF0l\_T9zmjg4jzc\_ew7s5IfX4ttVEeZdqzgeLKcnARTFGL5wPEzxsU-Cr-GO3XAPVOdI\_7\_NIrX2L1QIAcRLo89cvkLqlazFdk0Xe5QHdWUhu\_n\_CStF1SIWgF4DK4tZQytEnTvGyk1JdGlW1aYa4B8sMrkMC7jpC2LGo

* Deep learning is a collection of machine learning methods, in which stacked processing layers are used to create abstract representations of data, creating an artificial “neural network” (33, 34). These processing layers form an interconnected path from input features (e.g., age, smoking status, systolic blood pressure, etc.) to the model’s output (e.g., risk of heart disease). Initially inspired by mechanistic theories of brain physiology, each layer consists of processing units called “neurons” or “nodes.
* Neural networks may be considered a more flexible approach to fitting prediction models. Prediction modeling typically involves numerous preprocessing decisions: selecting a subset of the available predictors using prior knowledge or a statistical procedure (e.g., stepwise regression), discretizing continuous predictors, introducing polynomials, or using interaction terms. In deep learning, with enough data (which may vary from hundreds to millions of examples, depending on task complexity and approach), these preprocessing steps become unnecessary because of the flexibility gained by multiple processing layers and nonlinear data transformations, known as activation functions
* The building blocks of neural networks are neurons, which perform 2 simple operations. First, the neuron calculates a weighted sum of the inputs; these weights are randomly chosen at the start of model training and progressively revised to improve model performance (i.e., minimize loss) throughout the learning process. During the second step, the neuron applies a nonlinear mathematical transformation, an activation function, to that weighted sum. While both operations are simple, neurons can be extremely powerful in aggregate. By adding more layers (and more neurons per layer), it is possible to model highly complex functions, including nonlinearities and interactions, without any further specification (Figure 2). The universal approximation theorem (35–37) demonstrates that a sufficiently deep (i.e., many layers) or wide (i.e., many neurons) neural network can approximate any continuous mathematical function
* In fact, a regression model can be expressed as a simple neural network. A network with an input layer, an output layer applying the logistic function, and no hidden layers (or hidden layers of neurons applying linear activation functions) will behave equivalently to logistic regression
* After the weighted sum of the inputs is calculated by the neuron, an activation function applies a mathematical transformation. In theory, both linear and nonlinear transformations may be used, but nonlinear functions are nearly always used to increase the network’s capacity to model nonlinearities in the data. The rectified linear unit (ReLU) is a simple activation function used extensively in deep learning. If the input is positive, it outputs the value of the input. If the output is negative or zero, it outputs zero.
* one could consider modeling choices, such as the use of higher-order terms or interaction terms, to be hyperparameters. In addition to increasing the complexity of the model structure, deep learning also increases the number of hyperparameters: the number of layers, the number of neurons in layers, parameters for regularization, and batch size (i.e., how many examples are shown to the model before updating its weights in training), among others. These hyperparameters define the structure of the neural network and dictate how it will be trained. Values of hyperparameters have a large impact on model performance and should be reported to enhance reproducibility
* The learning rate is among the most important hyperparameters. Unlike linear regression, loss functions cannot be formulaically minimized in deep learning models. Instead, an iterative approach known as gradient descent is commonly used. Gradient descent is a procedure that identifies the direction of steepest decrease in the local loss landscape, like water flowing downhill. In simple terms, calculating the gradient tells us in which direction we should adjust parameter values. The magnitude of the change made is partly determined by the learning rate hyperparameter. The learning rate is critical to the success of the gradient descent algorithm; too large a learning rate can result in a failure to converge, and too small will make the model train slowly and inefficiently
* Feed-forward neural networks (FNN), also known as fully connected neural networks, multilayer perceptrons, or dense neural networks, are the fundamental type of deep learning networks. They consist of 1 or more fully connected layers (Figure 4). In a fully connected layer, each neuron receives the output of all neurons from the previous layer. Based on learned weights, neurons calculate a weighted average of these inputs, apply an activation function, and propagate their output to neurons in the next layer
* Convolutional neural networks (CNNs) are a family of deep learning models designed for images. However, they can also be applied to other data types (e.g., medical records) (39, 40). The networks have 3 core components: convolutional layers, pooling layers, and fully connected layers. These layers can be rearranged into different architectures of varying complexity.
* Recurrent neural networks (RNNs) process data in a sequential fashion, achieving better-than-human ability in tasks such as speech recognition (45
* In its simplest form, a recurrent layer is a fully connected layer that feeds into itself; outputs at one time step become inputs at the next time step (Figure 6).
* A basic RNN unit consists of a single layer and an activation function, and it will process the sequence from left to right. The vector representing the first word, “the,” is propagated through the layer, and an activation function is applied. The output will be concatenated with the vectorized representation of “quick,” passed through the same network, and the output will again be concatenated with the representation of the next word. This process continues as the model iterates through the entire sequence. Notice that the RNN unit remains unchanged; we are simply looping through the inputs
* First, deep learning tends to thrive in settings where outcomes depend on nonlinear combinations of thousands of variables, and hundreds of thousands of examples (or more) are available for training. It has been successful in
* It has shown less benefit in the typical epidemiologic setting, where data sets have hundreds to thousands of examples, and outcomes may be largely explained by tens of variables
* Second, deep learning predictions are difficult to interpret due to their use of nonlinear combinations of variables. In contrast, linear and logistic regression have coefficients that are more readily interpretable.
* Third, deep learning models can have limited generalizability. The ability of deep learning models to capture complex predictive relationships can identify patterns unique to the population or setting that gave rise to the training data. This can increase model performance in a specific population while decreasing generalizability to other popu
* Second, the use of machine learning in clinical settings also has the potential to propagate or increase existing disparities in health care (86). Ensuring fairness requires a holistic approach (87), including consideration of biases in formulation of the machine learning task, training set composition, labeling of observations, nonrandom missingness of data, and real-world use of the models.