Published: 27 May 2015

Deep learning

Yann LeCun, Yoshua Bengio & Geoffrey Hinton

Nature **521**, 436–444 (2015)

Abstract

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input.

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. For classification tasks, higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations. An image, for example, comes in the form of an array of pixel values, and the learned features in the first layer of representation typically represent the presence or absence of edges at particular orientations and locations in the image. The second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations in the edge positions. The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key expression and disease. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding[14], particularly topic classification, sentiment analysis, question answering[15] and language translation[16,17].

We think that deep learning will have many more successes in the near future because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data. New learning algorithms and architectures that are currently being developed for deep neural networks will only accelerate this progress.

Supervised learning

The most common form of machine learning, deep or not, is supervised learning. Imagine that we want to build a system that can classify images as containing, say, a house, a car, a person or a pet. We first collect a large data set of images of houses, cars, people and pets, each labelled with its category. During training, the machine is shown an image and produces an output in the form of a vector of scores, one for each category. We want the desired category to have the highest score of all categories, but this is unlikely to happen before training. We compute an objective function that measures the error (or distance) between the output scores and the desired pattern of scores. The machine then modifies its internal adjustable

The objective function, averaged over all the training examples, can be seen as a kind of hilly landscape in the high-dimensional space of weight values. 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.

In practice, most practitioners use a procedure called stochastic gradient descent (SGD). This consists of showing the input vector for a few examples, computing the outputs and the errors, computing the average gradient for those examples, and adjusting the weights accordingly. The process is repeated for many small sets of examples from the training set until the average of the objective function stops decreasing. It is called stochastic because each small set of examples gives a noisy estimate of the average gradient over all examples. This simple procedure usually finds a good set of weights surprisingly quickly when compared with far more elaborate optimization techniques[18]. After training, the performance of the system is measured on a different set of examples called a test set. This serves to test the generalization ability of the machine — its ability to produce sensible answers on new inputs that it has never seen during training.

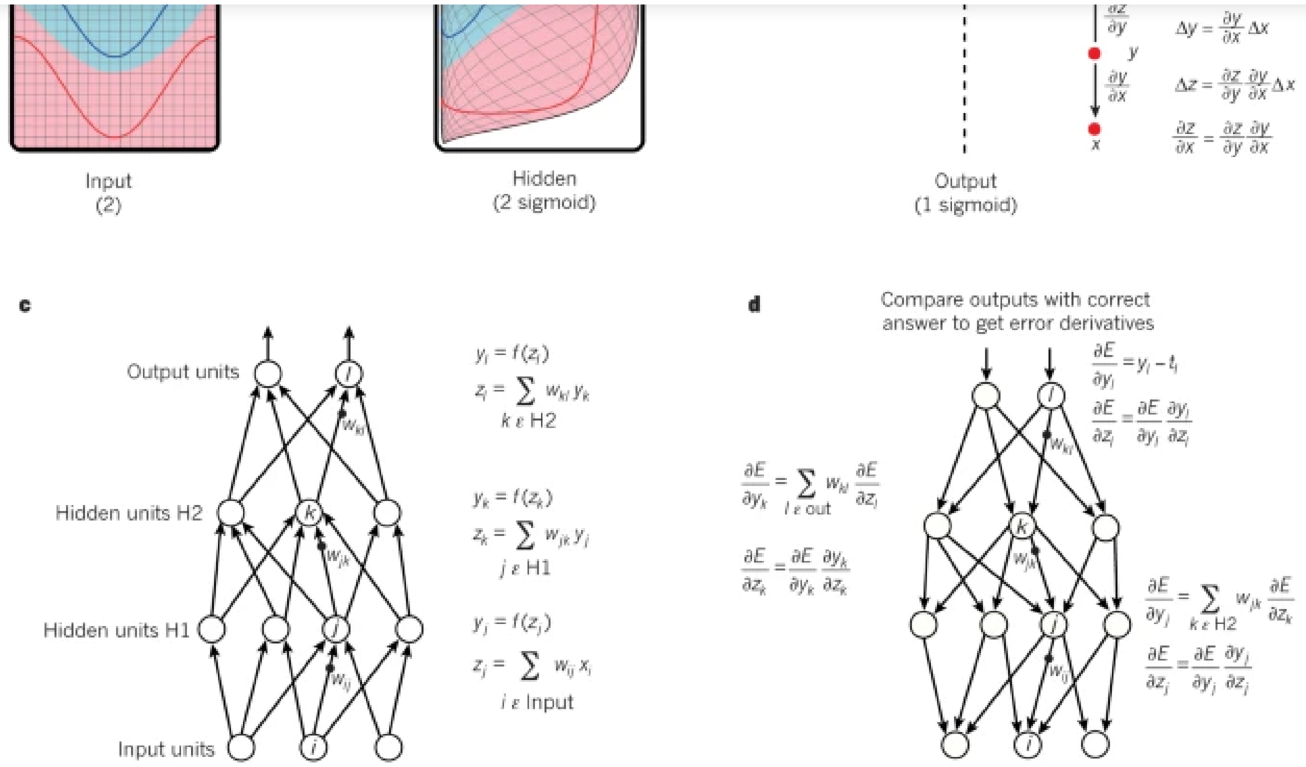
Many of the current practical applications of machine learning use linear classifiers on top of hand-engineered features. A two-class linear classifier computes a weighted sum of the feature vector components. If the weighted sum is above a threshold, the

the same position and on similar backgrounds may be very similar to each other. A linear classifier, or any other 'shallow' classifier operating on raw pixels could not possibly distinguish the latter two, while putting the former two in the same category. This is why shallow classifiers require a good feature extractor that solves the selectivity–invariance dilemma — one that produces representations that are selective to the aspects of the image that are important for discrimination, but that are invariant to irrelevant aspects such as the pose of the animal. To make classifiers more powerful, one can use generic non-linear features, as with kernel methods[20], but generic features such as those arising with the Gaussian kernel do not allow the learner to generalize well far from the training examples[21]. The conventional option is to hand design good feature extractors, which requires a considerable amount of engineering skill and domain expertise. But this can all be avoided if good features can be learned automatically using a general-purpose learning procedure. This is the key advantage of deep learning.

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 — distinguishing Samoyeds from white wolves — and insensitive to large irrelevant variations such as the that this could be done, and that it worked, was discovered independently by several different groups during the 1970s and 1980s[24,25,26,27].

The backpropagation procedure to compute the gradient of an objective function with respect to the weights of a multilayer stack of modules is nothing more than a practical application of the chain rule for derivatives. 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.

Figure 1: Multilayer neural networks and backpropagation.



a, A multi-layer neural network (shown by the connected dots) can distort the input space to make the classes of data (examples of which are on the red and blue lines) linearly separable. Note how a regular grid (shown on the left) in input space is also transformed (shown in the middle panel) by hidden units. This is an illustrative example with only two input units, two hidden units and one output unit, but the networks used for object recognition or natural language processing contain tens or hundreds of thousands of units. Reproduced with permission from C. Olah (http://colah.github.io/). **b**, The chain rule of derivatives tells us how two small effects (that of a small change of x on y, and that of y on z) are composed. A small change Δx in x gets transformed first into a small change Δy in y by getting multiplied by $\partial y/\partial x$ (that is, the definition of partial derivative). Similarly, the respect to the output of each unit, which is a weighted sum of the error derivatives with respect to the total inputs to the units in the layer above. We then convert the error derivative with respect to the output into the error derivative with respect to the input by multiplying it by the gradient of $f(z)$. At the output layer, the error derivative with respect to the output of a unit is computed by differentiating the cost function. This gives $y_l - t_l$ if the cost function for unit l is $0.5(y_l - t_l)^2$, where t_l is the target value. Once the $\partial E/\partial z_k$ is known, the error-derivative for the weight w_{jk} on the connection from unit j in the layer below is just $y_j\, \partial E/\partial z_k$.

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-training[28]. 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).

quality. Recent theoretical and empirical results strongly suggest that local minima are not a serious issue in general. Instead, the landscape is packed with a combinatorially large number of saddle points where the gradient is zero, and the surface curves up in most dimensions and curves down in the remainder[29,30]. The analysis seems to show that saddle points with only a few downward curving directions are present in very large numbers, but almost all of them have very similar values of the objective function. Hence, it does not much matter which of these saddle points the algorithm gets stuck at.

Interest in deep feedforward networks was revived around 2006 (refs 31,32,33,34) by a group of researchers brought together by the Canadian Institute for Advanced Research (CIFAR). The researchers introduced unsupervised learning procedures that could create layers of feature detectors without requiring labelled data. The objective in learning each layer of feature detectors was to be able to reconstruct or model the activities of feature detectors (or raw inputs) in the layer below. By 'pre-training' several layers of progressively more complex feature detectors using this reconstruction objective, the weights of a deep network could be initialized to sensible values. A final layer of output units could then be added to the top of the network and the whole deep system could be fine-tuned using standard backpropagation[33,34,35]. This worked remarkably well for recognizing handwritten digits or for detecting pedestrians, especially when the amount of labelled data was very limited[36].

being deployed in Android phones. For smaller data sets, unsupervised pre-training helps to prevent overfitting[40], leading to significantly better generalization when the number of labelled examples is small, or in a transfer setting where we have lots of examples for some 'source' tasks but very few for some 'target' tasks. Once deep learning had been rehabilitated, it turned out that the pre-training stage was only needed for small data sets.

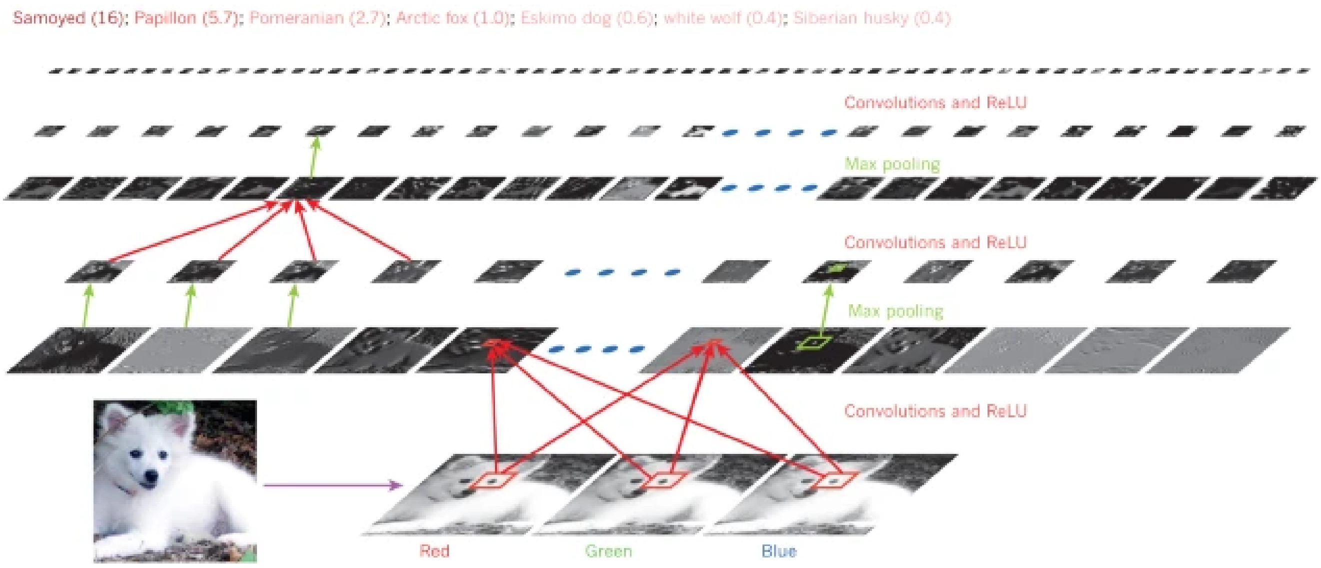
There was, however, one particular type of deep, feedforward network that was much easier to train and generalized much better than networks with full connectivity between adjacent layers. This was the convolutional neural network (ConvNet)[41,42]. It achieved many practical successes during the period when neural networks were out of favour and it has recently been widely adopted by the computer-vision community.

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. Many data modalities are in the form of multiple arrays: 1D for signals and sequences, including language; 2D for images or audio spectrograms; and 3D for video or volumetric images. There are four key ideas behind ConvNets that take advantage of the properties of natural signals: local connections, shared weights, pooling and the use of many layers.

other words, if a motif can appear in one part of the image, it could appear anywhere, hence the idea of units at different locations sharing the same weights and detecting the same pattern in different parts of the array. Mathematically, the filtering operation performed by a feature map is a discrete convolution, hence the name.

Figure 2: Inside a convolutional network.



The outputs (not the filters) of each layer (horizontally) of a typical convolutional network architecture applied to the image of a Samoyed dog (bottom left; and RGB (red, green, blue) inputs, bottom right). Each rectangular image is a feature map corresponding to the output for one of the learned features, detected at each of the image positions. Information flows bottom up, with lower-level features acting as oriented edge detectors, and a score is computed for each image class in output. ReLU, rectified linear unit.

pooling are stacked, followed by more convolutional and fully-connected layers. Backpropagating gradients through a ConvNet is as simple as through a regular deep network, allowing all the weights in all the filter banks to be trained.

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. In images, local combinations of edges form motifs, motifs assemble into parts, and parts form objects. Similar hierarchies exist in speech and text from sounds to phones, phonemes, syllables, words and sentences. The pooling allows representations to vary very little when elements in the previous layer vary in position and appearance.

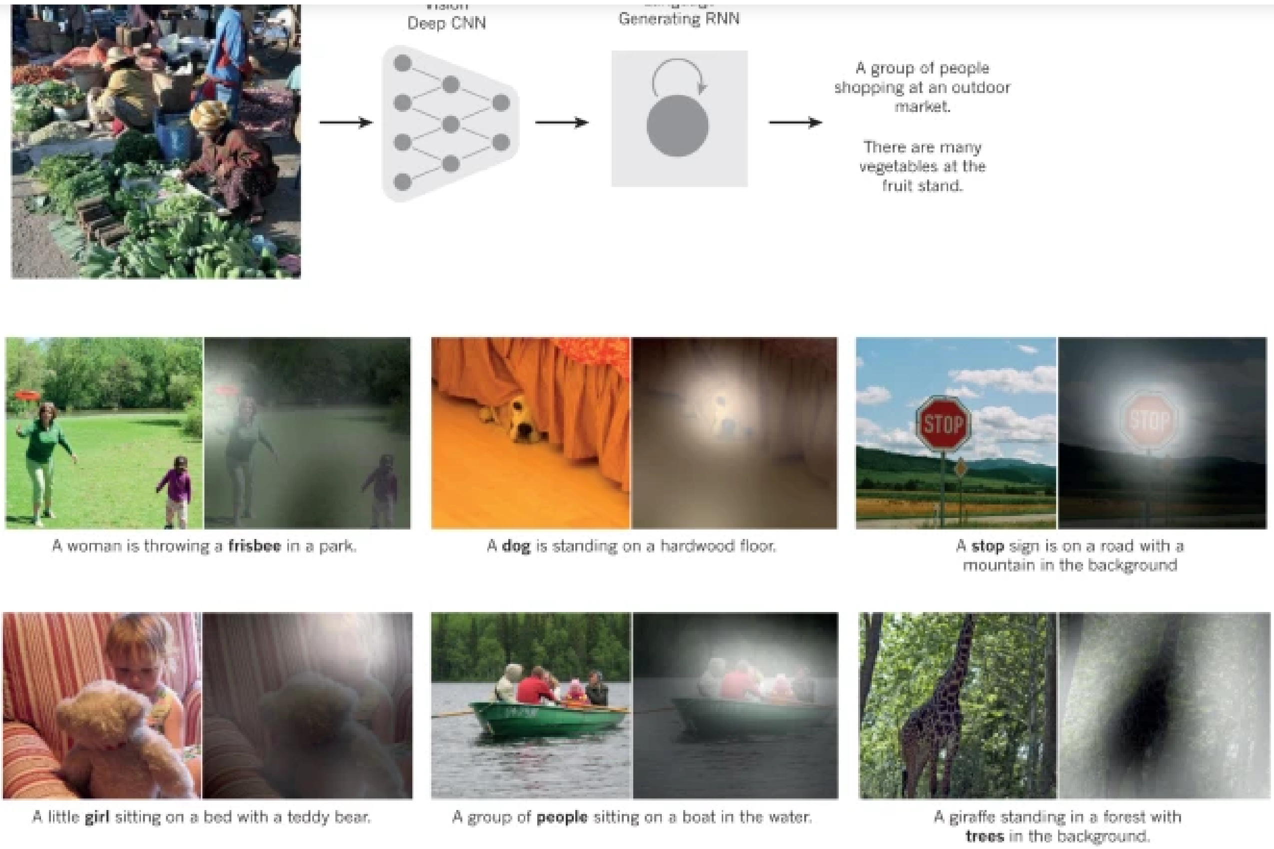
The convolutional and pooling layers in ConvNets are directly inspired by the classic notions of simple cells and complex cells in visual neuroscience[43], and the overall architecture is reminiscent of the LGN–V1–V2–V4–IT hierarchy in the visual cortex ventral pathway[44]. When ConvNet models and monkeys are shown the same picture, the activations of high-level units in the ConvNet explains half of the variance of random sets of 160 neurons in the monkey's inferotemporal cortex[45]. ConvNets have their roots in the neocognitron[46], the architecture of which was somewhat similar, but did not have an end-to-end supervised-learning algorithm such as backpropagation. A primitive 1D ConvNet called a time-delay neural net was used for the recognition of phonemes and simple words[47,48].

Image understanding with deep convolutional networks

Since the early 2000s, ConvNets have been applied with great success to the detection, segmentation and recognition of objects and regions in images. These were all tasks in which labelled data was relatively abundant, such as traffic sign recognition[53], the segmentation of biological images[54] particularly for connectomics[55], and the detection of faces, text, pedestrians and human bodies in natural images[36,50,51,56,57,58]. A major recent practical success of ConvNets is face recognition[59].

Importantly, images can be labelled at the pixel level, which will have applications in technology, including autonomous mobile robots and self-driving cars[60,61]. Companies such as Mobileye and NVIDIA are using such ConvNet-based methods in their upcoming vision systems for cars. Other applications gaining importance involve natural language understanding[14] and speech recognition[7].

Despite these successes, ConvNets were largely forsaken by the mainstream computer-vision and machine-learning communities until the ImageNet competition in 2012. When deep convolutional networks were applied to a data set of about a million images from the web that contained 1,000 different classes, they achieved spectacular results, almost halving the error rates of the best competing approaches[1].



Captions generated by a recurrent neural network (RNN) taking, as extra input, the representation extracted by a deep convolution neural network (CNN) from a test image, with the RNN trained to 'translate' high-level representations of images into captions (top). Reproduced with permission from ref. 102. When the RNN is given the ability to focus its attention on a different location in the input image (middle and bottom; the lighter patches were given more attention) as it generates each word (bold), we found[86] that it exploits this to achieve better 'translation' of images into captions.

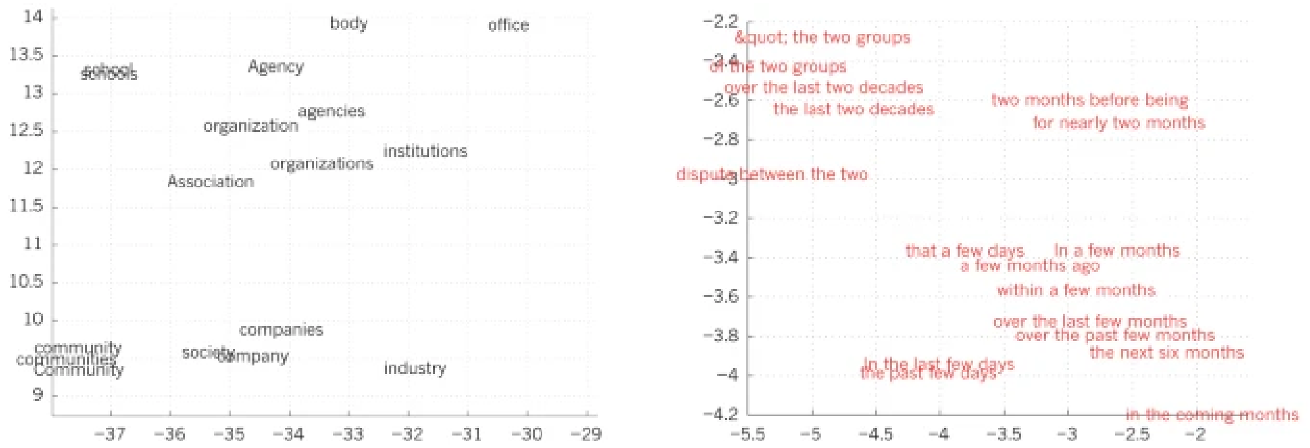
ConvNets are easily amenable to efficient hardware implementations in chips or field-programmable gate arrays[66,67]. A number of companies such as NVIDIA, Mobileye, Intel, Qualcomm and Samsung are developing ConvNet chips to enable real-time vision applications in smartphones, cameras, robots and self-driving cars.

Distributed representations and language processing

Deep-learning theory shows that deep nets have two different exponential advantages over classic learning algorithms that do not use distributed representations[21]. Both of these advantages arise from the power of composition and depend on the underlying data-generating distribution having an appropriate componential structure[40]. First, learning distributed representations enable generalization to new combinations of the values of learned features beyond those seen during training (for example, 2^n combinations are possible with n binary features)[68,69]. Second, composing layers of representation in a deep net brings the potential for another exponential advantage[70] (exponential in the depth).

The hidden layers of a multilayer neural network learn to represent the network's inputs in a way that makes it easy to predict the target outputs. This is nicely demonstrated by training a multilayer neural network to predict the next word in a sequence from a local context of earlier words[71]. Each word in the context is multiple 'micro-rules'. Learning word vectors turned out to also work very well when the word sequences come from a large corpus of real text and the individual micro-rules are unreliable[71]. When trained to predict the next word in a news story, for example, the learned word vectors for Tuesday and Wednesday are very similar, as are the word vectors for Sweden and Norway. Such representations are called distributed representations because their elements (the features) are not mutually exclusive and their many configurations correspond to the variations seen in the observed data. These word vectors are composed of learned features that were not determined ahead of time by experts, but automatically discovered by the neural network. Vector representations of words learned from text are now very widely used in natural language applications[14,17,72,73,74,75,76].

Figure 4: Visualizing the learned word vectors.

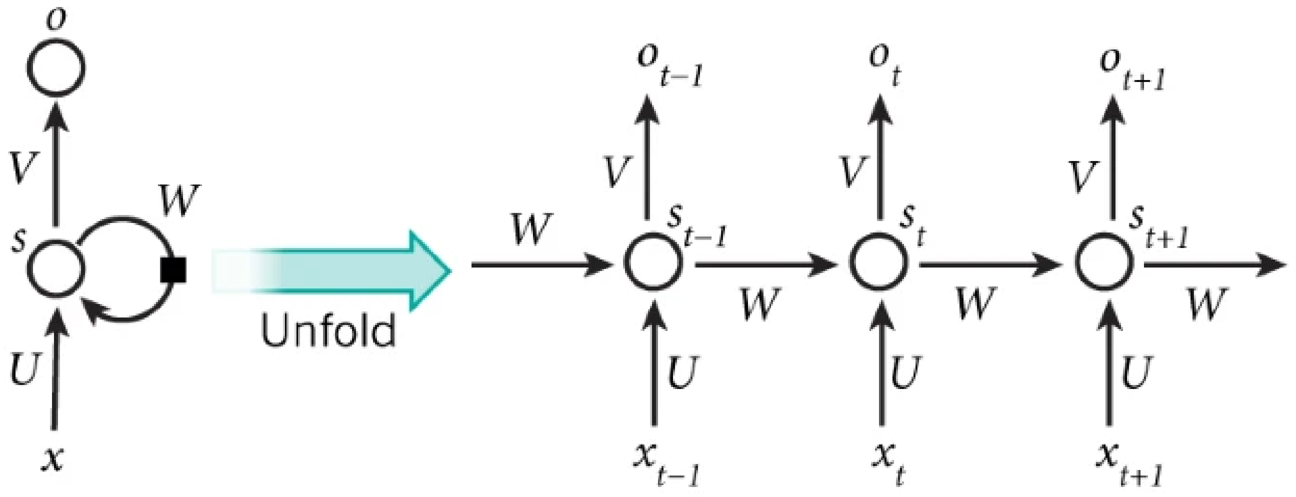


On the left is an illustration of word representations learned for modelling language, nonparadigm, an instance of a symbol is something for which the only property is that it is either identical or non-identical to other symbol instances. It has no internal structure that is relevant to its use; and to reason with symbols, they must be bound to the variables in judiciously chosen rules of inference. By contrast, neural networks just use big activity vectors, big weight matrices and scalar non-linearities to perform the type of fast 'intuitive' inference that underpins effortless commonsense reasoning.

Before the introduction of neural language models[71], the standard approach to statistical modelling of language did not exploit distributed representations: it was based on counting frequencies of occurrences of short symbol sequences of length up to N (called N-grams). The number of possible N-grams is on the order of V^N, where V is the vocabulary size, so taking into account a context of more than a handful of words would require very large training corpora. N-grams treat each word as an atomic unit, so they cannot generalize across semantically related sequences of words, whereas neural language models can because they associate each word with a vector of real valued features, and semantically related words end up close to each other in that vector space (Fig. 4).

Recurrent neural networks

When backpropagation was first introduced, its most exciting use was for training



The artificial neurons (for example, hidden units grouped under node *s* with values s_t at time *t*) get inputs from other neurons at previous time steps (this is represented with the black square, representing a delay of one time step, on the left). In this way, a recurrent neural network can map an input sequence with elements x_t into an output sequence with elements o_t, with each o_t depending on all the previous $x_{t'}$ (for $t' \leq t$). The same parameters (matrices *U*,*V*,*W*) are used at each time step. Many other architectures are possible, including a variant in which the network can generate a sequence of outputs (for example, words), each of which is used as inputs for the next time step. The backpropagation algorithm (Fig. 1) can be directly applied to the computational graph of the unfolded network on the right, to compute the derivative of a total error (for example, the log-probability of generating the right sequence of outputs) with respect to all the states s_t and all the parameters.

'decoder' network, which outputs a probability distribution for the first word of the French translation. If a particular first word is chosen from this distribution and provided as input to the decoder network it will then output a probability distribution for the second word of the translation and so on until a full stop is chosen[17,72,76]. Overall, this process generates sequences of French words according to a probability distribution that depends on the English sentence. This rather naive way of performing machine translation has quickly become competitive with the state-of-the-art, and this raises serious doubts about whether understanding a sentence requires anything like the internal symbolic expressions that are manipulated by using inference rules. It is more compatible with the view that everyday reasoning involves many simultaneous analogies that each contribute plausibility to a conclusion[84,85].

Instead of translating the meaning of a French sentence into an English sentence, one can learn to 'translate' the meaning of an image into an English sentence (Fig. 3). The encoder here is a deep ConvNet that converts the pixels into an activity vector in its last hidden layer. The decoder is an RNN similar to the ones used for machine translation and neural language modelling. There has been a surge of interest in such systems recently (see examples mentioned in ref. 86).

RNNs, once unfolded in time (Fig. 5), can be seen as very deep feedforward networks clear the content of the memory.

LSTM networks have subsequently proved to be more effective than conventional RNNs, especially when they have several layers for each time step[87], enabling an entire speech recognition system that goes all the way from acoustics to the sequence of characters in the transcription. LSTM networks or related forms of gated units are also currently used for the encoder and decoder networks that perform so well at machine translation[17,72,76].

Over the past year, several authors have made different proposals to augment RNNs with a memory module. Proposals include the Neural Turing Machine in which the network is augmented by a 'tape-like' memory that the RNN can choose to read from or write to[88], and memory networks, in which a regular network is augmented by a kind of associative memory[89]. Memory networks have yielded excellent performance on standard question-answering benchmarks. The memory is used to remember the story about which the network is later asked to answer questions.

Beyond simple memorization, neural Turing machines and memory networks are being used for tasks that would normally require reasoning and symbol manipulation. Neural Turing machines can be taught 'algorithms'. Among other things, they can learn to output a sorted list of symbols when their input consists of an unsorted sequence in which each symbol is accompanied by a real value that indicates its priority in the list[88]. Memory networks can be trained to keep track of the supervised learning. Although we have not focused on it in this Review, we expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object.

Human vision is an active process that sequentially samples the optic array in an intelligent, task-specific way using a small, high-resolution fovea with a large, low-resolution surround. We expect much of the future progress in vision to come from systems that are trained end-to-end and combine ConvNets with RNNs that use reinforcement learning to decide where to look. Systems combining deep learning and reinforcement learning are in their infancy, but they already outperform passive vision systems[99] at classification tasks and produce impressive results in learning to play many different video games[100].

Natural language understanding is another area in which deep learning is poised to make a large impact over the next few years. We expect systems that use RNNs to understand sentences or whole documents will become much better when they learn strategies for selectively attending to one part at a time[76,86].

Ultimately, major progress in artificial intelligence will come about through systems that combine representation learning with complex reasoning. Although deep learning and simple reasoning have been used for speech and handwriting

community.

2 Farabet, C., Couprie, C., Najman, L. & LeCun, Y. Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1915–1929 (2013).

3 Tompson, J., Jain, A., LeCun, Y. & Bregler, C. Joint training of a convolutional network and a graphical model for human pose estimation. In *Proc. Advances in Neural Information Processing Systems 27* 1799–1807 (2014).

4 Szegedy, C. et al. Going deeper with convolutions. Preprint at http://arxiv.org/abs/1409.4842 (2014).

5 Mikolov, T., Deoras, A., Povey, D., Burget, L. & Cernocky, J. Strategies for training large scale neural network language models. In *Proc. Automatic Speech Recognition and Understanding* 196–201 (2011).

6 Hinton, G. et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine* **29**, 82–97 (2012). **This joint paper from the major speech recognition laboratories, summarizing the breakthrough achieved with deep learning on the task of phonetic classification for automatic speech recognition, was the first major industrial**

9 Ciodaro, T., Deva, D., de Seixas, J. & Damazio, D. Online particle detection with neural networks based on topological calorimetry information. *J. Phys. Conf. Series* **368**, 012030 (2012).

10 Kaggle. Higgs boson machine learning challenge. *Kaggle* https://www.kaggle.com/c/higgs-boson (2014).

11 Helmstaedter, M. et al. Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature* **500**, 168–174 (2013).

12 Leung, M. K., Xiong, H. Y., Lee, L. J. & Frey, B. J. Deep learning of the tissue-regulated splicing code. *Bioinformatics* **30**, i121–i129 (2014).

13 Xiong, H. Y. et al. The human splicing code reveals new insights into the genetic determinants of disease. *Science* **347**, 6218 (2015).

14 Collobert, R., et al. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **12**, 2493–2537 (2011).

15 Bordes, A., Chopra, S. & Weston, J. Question answering with subgraph

with the architecture introduced in ref. 72, with a recurrent network trained to read a sentence in one language, produce a semantic representation of its meaning, and generate a translation in another language.

18 Bottou, L. & Bousquet, O. The tradeoffs of large scale learning. In *Proc. Advances in Neural Information Processing Systems 20* 161–168 (2007).

19 Duda, R. O. & Hart, P. E. *Pattern Classification and Scene Analysis* (Wiley, 1973).

20 Schölkopf, B. & Smola, A. *Learning with Kernels* (MIT Press, 2002).

21 Bengio, Y., Delalleau, O. & Le Roux, N. The curse of highly variable functions for local kernel machines. In *Proc. Advances in Neural Information Processing Systems 18* 107–114 (2005).

22 Selfridge, O. G. Pandemonium: a paradigm for learning in mechanisation of thought processes. In *Proc. Symposium on Mechanisation of Thought Processes* 513–526 (1958).

Connaissance et des Neurosciences [in French] 599–604 (1985).

27 Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).

28 Glorot, X., Bordes, A. & Bengio. Y. Deep sparse rectifier neural networks. In *Proc. 14th International Conference on Artificial Intelligence and Statistics* 315–323 (2011). **This paper showed that supervised training of very deep neural networks is much faster if the hidden layers are composed of ReLU.**

29 Dauphin, Y. et al. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proc. Advances in Neural Information Processing Systems 27* 2933–2941 (2014).

30 Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B. & LeCun, Y. The loss surface of multilayer networks. In *Proc. Conference on AI and Statistics* http://arxiv.org/abs/1412.0233 (2014).

31 Hinton, G. E. What kind of graphical model is the brain? In *Proc. 19th International Joint Conference on Artificial intelligence* 1765–1775 (2005).

test data and generalizes the method to other unsupervised representation-learning techniques, such as auto-encoders.**

34 Ranzato, M., Poultney, C., Chopra, S. & LeCun, Y. Efficient learning of sparse representations with an energy-based model. In *Proc. Advances in Neural Information Processing Systems 19* 1137–1144 (2006).

35 Hinton, G. E. & Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science* **313**, 504–507 (2006).

36 Sermanet, P., Kavukcuoglu, K., Chintala, S. & LeCun, Y. Pedestrian detection with unsupervised multi-stage feature learning. In *Proc. International Conference on Computer Vision and Pattern Recognition* http://arxiv.org/abs/1212.0142 (2013).

37 Raina, R., Madhavan, A. & Ng, A. Y. Large-scale deep unsupervised learning using graphics processors. In *Proc. 26th Annual International Conference on Machine Learning* 873–880 (2009).

38 Mohamed, A.-R., Dahl, G. E. & Hinton, G. Acoustic modeling using deep belief networks. *IEEE Trans. Audio Speech Lang. Process.* **20**, 14–22 (2012).

the first paper on convolutional networks trained by backpropagation for **the task of classifying low-resolution images of handwritten digits.**

42 LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998). **This overview paper on the principles of end-to-end training of modular systems such as deep neural networks using gradient-based optimization showed how neural networks (and in particular convolutional nets) can be combined with search or inference mechanisms to model complex outputs that are interdependent, such as sequences of characters associated with the content of a document.**

43 Hubel, D. H. & Wiesel, T. N. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *J. Physiol.* **160**, 106–154 (1962).

44 Felleman, D. J. & Essen, D. C. V. Distributed hierarchical processing in the primate cerebral cortex. *Cereb. Cortex* **1**, 1–47 (1991).

45 Cadieu, C. F. et al. Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS Comp. Biol.* **10**, e1003963 (2014).

46 Fukushima, K. & Miyake, S. Neocognitron: a new algorithm for pattern

recognition. In *Proc. EuroSpeech 89* 537–540 (1989).

49 Simard, D., Steinkraus, P. Y. & Platt, J. C. Best practices for convolutional neural networks. In *Proc. Document Analysis and Recognition* 958–963 (2003).

50 Vaillant, R., Monrocq, C. & LeCun, Y. Original approach for the localisation of objects in images. In *Proc. Vision, Image, and Signal Processing* **141**, 245–250 (1994).

51 Nowlan, S. & Platt, J. in *Neural Information Processing Systems* 901–908 (1995).

52 Lawrence, S., Giles, C. L., Tsoi, A. C. & Back, A. D. Face recognition: a convolutional neural-network approach. *IEEE Trans. Neural Networks* **8**, 98–113 (1997).

53 Ciresan, D., Meier, U. Masci, J. & Schmidhuber, J. Multi-column deep neural network for traffic sign classification. *Neural Networks* **32**, 333–338 (2012).

54 Ning, F. et al. Toward automatic phenotyping of developing embryos from videos. *IEEE Trans. Image Process.* **14**, 1360–1371 (2005).

58 Tompson, J., Goroshin, R. R., Jain, A., LeCun, Y. Y. & Bregler, C. C. Efficient object localization using convolutional networks. In *Proc. Conference on Computer Vision and Pattern Recognition* http://arxiv.org/abs/1411.4280 (2014).

59 Taigman, Y., Yang, M., Ranzato, M. & Wolf, L. Deepface: closing the gap to human-level performance in face verification. In *Proc. Conference on Computer Vision and Pattern Recognition* 1701–1708 (2014).

60 Hadsell, R. et al. Learning long-range vision for autonomous off-road driving. *J. Field Robot.* **26**, 120–144 (2009).

61 Farabet, C., Couprie, C., Najman, L. & LeCun, Y. Scene parsing with multiscale feature learning, purity trees, and optimal covers. In *Proc. International Conference on Machine Learning* http://arxiv.org/abs/1202.2160 (2012).

62 Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Machine Learning Res.* **15**, 1929–1958 (2014).

http://arxiv.org/abs/1409.1556 (2014).

66 Boser, B., Sackinger, E., Bromley, J., LeCun, Y. & Jackel, L. An analog neural network processor with programmable topology. *J. Solid State Circuits* **26**, 2017–2025 (1991).

67 Farabet, C. et al. Large-scale FPGA-based convolutional networks. In *Scaling up Machine Learning: Parallel and Distributed Approaches* (eds Bekkerman, R., Bilenko, M. & Langford, J.) 399–419 (Cambridge Univ. Press, 2011).

68 Bengio, Y. *Learning Deep Architectures for AI* (Now, 2009).

69 Montufar, G. & Morton, J. When does a mixture of products contain a product of mixtures? *J. Discrete Math.* **29**, 321–347 (2014).

70 Montufar, G. F., Pascanu, R., Cho, K. & Bengio, Y. On the number of linear regions of deep neural networks. In *Proc. Advances in Neural Information Processing Systems 27* 2924–2932 (2014).

71 Bengio, Y., Ducharme, R. & Vincent, P. A neural probabilistic language model. In

74 Socher, R., Lin, C. C-Y., Manning, C. & Ng, A. Y. Parsing natural scenes and natural language with recursive neural networks. In *Proc. International Conference on Machine Learning* 129–136 (2011).

75 Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. Distributed representations of words and phrases and their compositionality. In *Proc. Advances in Neural Information Processing Systems 26* 3111–3119 (2013).

76 Bahdanau, D., Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate. In *Proc. International Conference on Learning Representations* http://arxiv.org/abs/1409.0473 (2015).

77 Hochreiter, S. Untersuchungen zu dynamischen neuronalen Netzen [in German] Diploma thesis, T.U. Münich (1991).

78 Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks* **5**, 157–166 (1994).

79 Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**,

82 Pascanu, R., Mikolov, T. & Bengio, Y. On the difficulty of training recurrent neural networks. In *Proc. 30th International Conference on Machine Learning* 1310–1318 (2013).

83 Sutskever, I., Martens, J. & Hinton, G. E. Generating text with recurrent neural networks. In *Proc. 28th International Conference on Machine Learning* 1017–1024 (2011).

84 Lakoff, G. & Johnson, M. *Metaphors We Live By* (Univ. Chicago Press, 2008).

85 Rogers, T. T. & McClelland, J. L. *Semantic Cognition: A Parallel Distributed Processing Approach* (MIT Press, 2004).

86 Xu, K. et al. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. International Conference on Learning Representations* http://arxiv.org/abs/1502.03044 (2015).

87 Graves, A., Mohamed, A.-R. & Hinton, G. Speech recognition with deep recurrent neural networks. In *Proc. International Conference on Acoustics, Speech and Signal Processing* 6645–6649 (2013).

(2015).

91 Hinton, G. E., Dayan, P., Frey, B. J. & Neal, R. M. The wake-sleep algorithm for unsupervised neural networks. *Science* **268**, 1558–1161 (1995).

92 Salakhutdinov, R. & Hinton, G. Deep Boltzmann machines. In *Proc. International Conference on Artificial Intelligence and Statistics* 448–455 (2009).

93 Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proc. 25th International Conference on Machine Learning* 1096–1103 (2008).

94 Kavukcuoglu, K. et al. Learning convolutional feature hierarchies for visual recognition. In *Proc. Advances in Neural Information Processing Systems 23* 1090–1098 (2010).

95 Gregor, K. & LeCun, Y. Learning fast approximations of sparse coding. In *Proc. International Conference on Machine Learning* 399–406 (2010).

96 Ranzato, M., Mnih, V., Susskind, J. M. & Hinton, G. E. Modeling natural images

99 Ba, J., Mnih, V. & Kavukcuoglu, K. Multiple object recognition with visual attention. In *Proc. International Conference on Learning Representations* http://arxiv.org/abs/1412.7755 (2014).

100 Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).

101 Bottou, L. From machine learning to machine reasoning. *Mach. Learn.* **94**, 133–149 (2014).

102 Vinyals, O., Toshev, A., Bengio, S. & Erhan, D. Show and tell: a neural image caption generator. In *Proc. International Conference on Machine Learning* http://arxiv.org/abs/1502.03044 (2014).

103 van der Maaten, L. & Hinton, G. E. Visualizing data using t-SNE. *J. Mach. Learn.Research* **9**, 2579–2605 (2008).

Acknowledgements

The authors would like to thank the Natural Sciences and Engineering Research

Download PDF

STN, Montreal, H3C 3J7, Quebec, Canada
Yoshua Bengio

Google, 1600 Amphitheatre Parkway, Mountain View, 94043, California, USA
Geoffrey Hinton

Department of Computer Science, University of Toronto, 6 King's College Road, Toronto, M5S 3G4, Ontario, Canada
Geoffrey Hinton

Corresponding author

Correspondence to Yann LeCun.

Ethics declarations

Competing interests

The authors declare no competing financial interests.

Additional information

Reprints and permissions information is available at www.nature.com/reprints.

Rights and permissions

Download PDF

DOI

https://doi.org/10.1038/nature14539

Share this article

Anyone you share the following link with will be able to read this content:

Get shareable link

Provided by the Springer Nature SharedIt content-sharing initiative

Subjects [Computer science](#) • [Mathematics and computing](#)

This article is cited by

[A deep learning model for the classification of atrial fibrillation in critically ill patients](#)

Brian Chen, David M. Maslove ... Stephanie Sibley

Intensive Care Medicine Experimental (2023)

[Prospects and applications of on-chip lasers](#)

Zhican Zhou, Xiangpeng Ou ... John E. Bowers

eLight (2023)

Waqas W. Ahmed, Mohamed Farhat ... Ying Wu

Communications Physics (2023)

Nature (*Nature*) | ISSN 1476-4687 (online) | ISSN 0028-0836 (print)