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 technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users’ interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, constructing a pattern-recognition or 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 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.

Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition and speech recognition, it has beaten other machine-learning techniques at predicting the activity of potential drug molecules, analysing particle accelerator data, reconstructing brain circuits, and predicting the effects of mutations in non-coding DNA on gene expression and disease. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding, particularly topic classification, sentiment analysis, question answering and language translation.

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 parameters to reduce this error. These adjustable parameters, often called weights, are real numbers that can be seen as ‘knobs’ that define the input–output function of the machine. 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.

To properly adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would increase or decrease if the weight were increased by a tiny amount. The weight vector is then adjusted in the opposite direction to the gradient vector.

The objective function, averaged over all the training examples, can

---

1 Facebook AI Research, 770 Broadway, New York, New York 10003 USA. 2 New York University, 715 Broadway, New York, New York 10003, USA. 3 Department of Computer Science and Operations Research, Université de Montréal, Pavillon André-Aisenstadt, PO Box 6128 Centre-Ville STN Montréal, Quebec H3C 3J7, Canada. 4 Google, 1600 Amphitheatre Parkway, Mountain View, California 94043, USA. 5 Department of Computer Science, University of Toronto, 6 King’s College Road, Toronto, Ontario M5S 3G4, Canada.
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\(^\text{14}\). 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 input is classified as belonging to a particular category.

Since the 1960s we have known that linear classifiers can only carve their input space into very simple regions, namely half-spaces separated by a hyperplane\(^\text{19}\). But problems such as image and speech recognition require the input–output function to be insensitive to irrelevant variations of the input, such as variations in position, orientation or illumination of an object, or variations in the pitch or accent of speech, while being very sensitive to particular minute variations (for example, the difference between a white wolf and a breed of wolf-like white dog called a Samoyed). At the pixel level, images of two Samoyeds in different poses and in different environments may be very different from each other, whereas two images of a Samoyed and a wolf in the same position and on similar backgrounds may be very similar to each other. A linear classifier, or any other ‘shallow’ classifier operating on

![Figure 1](https://example.com/fig1.png)

**Figure 1 | Multilayer neural networks and backpropagation.** a, A multilayer 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 \(y\) on \(x\), and that of \(y\) on \(z\)) are composed. A small change \(\Delta x\) in \(x\) gets transformed first into a small change \(\Delta y\) in \(y\) by getting multiplied by \(\partial y/\partial x\) (that is, the definition of partial derivative). Similarly, the change \(\Delta y\) creates a change \(\Delta z\) in \(z\). Substituting one equation into the other gives the chain rule of derivatives — how \(\Delta z\) gets turned into \(\Delta z\) through multiplication by the product of \(\partial y/\partial x\) and \(\partial x/\partial z\). It also works when \(x\), \(y\) and \(z\) are vectors (and the derivatives are Jacobian matrices). c, The equations used for computing the forward pass in a neural net with two hidden layers and one output layer, each constituting a module through which one can backpropagate gradients. At each layer, we first compute the total input \(z\) to each unit, which is a weighted sum of the outputs of the units in the layer below. Then a non-linear function \(f(\cdot)\) is applied to \(z\) to get the output of the unit. For simplicity, we have omitted bias terms. The non-linear functions used in neural networks include the rectified linear unit (ReLU) \(f(z) = \max(0, z)\), commonly used in recent years, as well as the more conventional sigmoids, such as the hyperbolic tangent, \(f(z) = (\exp(z) − \exp(−z))/(\exp(z) + \exp(−z))\) and logistic function logistic, \(f(z) = 1/(1 + \exp(−z))\). d, The equations used for computing the backward pass. At each hidden layer we compute the error derivative with 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 \(y_i − t_i\), if the cost function for unit \(i\) is \(0.5(y_i − t_i)^2\), where \(t_i\) is the target value. Once the \(\partial E/\partial z_i\) is known, the error-derivative for the weight \(w_{ij}\) on the connection from unit \(j\) in the layer below is just \(y_j \partial E/\partial z_i\).
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, but generic features such as those arising with the Gaussian kernel do not allow the learner to generalize well far from the training examples. 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 background, pose, lighting and surrounding objects.

Backpropagation to train multilayer architectures

From the earliest days of pattern recognition, the aim of researchers has been to replace hand-engineered features with trainable multilayer networks, but despite its simplicity, the solution was not widely understood until the mid 1980s. As it turns out, multilayer architectures can be trained by simple stochastic gradient descent. As long as the modules are relatively smooth functions of their inputs and of their internal weights, one can compute gradients using the backpropagation procedure. The idea that this could be done, and that it worked, was discovered independently by several different groups during the 1970s and 1980s.

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.

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. 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).

In the late 1990s, neural nets and backpropagation were largely forsaken by the machine-learning community and ignored by the computer-vision and speech-recognition communities. It was widely thought that learning useful, multistage, feature extractors with little prior knowledge was infeasible. In particular, it was commonly thought that simple gradient descent would get trapped in poor local minima — weight configurations for which no small change would reduce the average error.

In practice, poor local minima are rarely a problem with large networks. Regardless of the initial conditions, the system nearly always reaches solutions of very similar 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
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 31–34) 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 33–35. This worked remarkably well for recognizing handwritten digits or for detecting pedestrians, especially when the amount of labelled data was very limited 36.

The first major application of this pre-training approach was in speech recognition, and it was made possible by the advent of fast graphics processing units (GPUs) that were convenient to program 37 and allowed researchers to train networks 10 or 20 times faster. In 2009, the approach was used to map short temporal windows of coefficients extracted from a sound wave to a set of probabilities for the various fragments of speech that might be represented by the frame in the centre of the window. It achieved record-breaking results on a standard speech recognition benchmark that used a small vocabulary 38 and was quickly developed to give record-breaking results on a large vocabulary task 39. By 2012, versions of the deep net from 2009 were being developed by many of the major speech groups 40 and were already 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.

The architecture of a typical ConvNet (Fig. 2) is structured as a series of stages. The first few stages are composed of two types of layers: convolutional layers and pooling layers. Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the previous layer through a set of weights called a filter bank. The result of this local weighted sum is then passed through a non-linearity such as a ReLU. All units in a feature map share the same filter bank. Different feature maps in a layer use different filter banks. The reason for this architecture is twofold. First, in array data such as images, local groups of values are often highly correlated, forming distinctive local motifs that are easily detected. Second, the local statistics of images and other signals are invariant to location. In 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.

Although the role of the convolutional layer is to detect local conjunctions of features from the previous layer, the role of the pooling layer is to merge semantically similar features into one. Because the relative positions of the features forming a motif can vary somewhat, reliably detecting the motif can be done by coarse-graining the position of each feature. A typical pooling unit computes the maximum of a local patch of units in one feature map (or in a few feature maps). Neighbouring pooling units take input from patches that are shifted by more than one row or column, thereby reducing the dimension of the representation and creating an invariance to small shifts and distortions. Two or three stages of convolution, non-linearity and 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.

There have been numerous applications of convolutional networks going back to the early 1990s, starting with time-delay neural networks for speech recognition 49 and document reading 50. The document reading system used a ConvNet trained jointly with a end-to-end supervised-learning algorithm such as backpropagation. The document reading system used a ConvNet trained jointly with a probabilistic model that implemented language constraints. By the late 1990s this system was reading over 10% of all the cheques in the United States. A number of ConvNet-based optical character recognition and handwriting recognition systems were later deployed by Microsoft 51. ConvNets were also experimented with in the early 1990s for object detection in natural images, including faces and hands 52,53, and for face recognition 54.

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 55, the segmentation of biological images 56 particularly for connectomics 57, and the detection of faces, text, pedestrians and human bodies in natural images 58,59. A major recent practical success of ConvNets is face recognition 60.

Importantly, images can be labelled at the pixel level, which will have applications in technology, including autonomous mobile robots and
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\textsuperscript{14} and speech recognition\textsuperscript{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\textsuperscript{1}. This success came from the efficient use of GPUs, ReLUs, a new regularization technique called dropout\textsuperscript{62}, and techniques to generate more training examples by deforming the existing ones. This success has brought about a revolution in computer vision; ConvNets are now the dominant approach for almost all recognition and detection tasks\textsuperscript{4,58,59,63–65} and approach human performance on some tasks. A recent stunning demonstration combines ConvNets and recurrent net modules for the generation of image captions (Fig. 3).

Recent ConvNet architectures have 10 to 20 layers of ReLUs, hundreds of millions of weights, and billions of connections between units. Whereas training such large networks could have taken weeks only two years ago, progress in hardware, software and algorithm parallelization have reduced training times to a few hours.

The performance of ConvNet-based vision systems has caused most major technology companies, including Google, Facebook, Microsoft, IBM, Yahoo!, Twitter and Adobe, as well as a quickly growing number of start-ups to initiate research and development projects and to deploy ConvNet-based image understanding products and services.

ConvNets are easily amenable to efficient hardware implementations in chips or field-programmable gate arrays\textsuperscript{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\textsuperscript{21}. Both of these advantages arise from the power of composition and depend on the underlying data-generating distribution having an appropriate componential structure\textsuperscript{40}. First, learning distributed representations enable generalization to new combinations of the values of learned features beyond those seen during training (for example, 2\textsuperscript{n} combinations are possible with n binary features)\textsuperscript{68,69}. Second, composing layers of representation in a deep net brings the potential for another exponential advantage\textsuperscript{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
The issue of representation lies at the heart of the debate between the logic-inspired and the neural-network-inspired paradigms for cognition. In the logic-inspired paradigm, 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 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.\textsuperscript{4,14,17,72–76}

Before the introduction of neural language models\textsuperscript{21}, 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 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. When we consider the outputs of the hidden units at different discrete time steps as if they were the outputs of different neurons in a deep multilayer network (Fig. 5, right), it becomes clear how we can apply backpropagation to train RNNs.

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.\textsuperscript{77,78}

Thanks to advances in their architecture\textsuperscript{79,80} and ways of training them\textsuperscript{81,82}, RNNs have been found to be very good at predicting the next character in the text\textsuperscript{83} or the next word in a sequence\textsuperscript{84}, but they can also be used for more complex tasks. For example, after reading an English sentence one word at a time, an English ‘encoder’ network can be trained so that the final state vector of its hidden units is a good representation of the thought expressed by the sentence. This thought vector can then be used as the initial hidden state of (or as extra input to) a jointly trained French ‘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\textsuperscript{14,17,72,76}. 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.

![Graph showing distribution of words](image-url)

**Figure 4** | Visualizing the learned word vectors. On the left is an illustration of word representations learned for modelling language, non-linearly projected to 2D for visualization using the t-SNE algorithm. On the right is a 2D representation of phrases learned by an English-to-French encoder–decoder recurrent neural network. One can observe that semantically similar words or sequences of words are mapped to nearby representations. The distributed representations of words are obtained by using backpropagation to jointly learn a representation for each word and a function that predicts a target quantity such as the next word in a sequence (for language modelling) or a whole sequence of translated words (for machine translation).
reasoning and symbol manipulation. Neural Turing machines can be thought of as a kind of associative memory that the RNN can choose to read from or write to. Although their main purpose is to learn long-term dependencies, recurrent neural network can map an input sequence with elements \( x_t \) into a sequence in which each symbol is accompanied by a real value that indicates its priority in the list. Memory networks can be trained to keep track of the state of the world in a setting similar to a text adventure game and after reading a story, they can answer questions that require complex inference. In one test example, the network is shown a 15-sentence version of the *The Lord of the Rings* and correctly answers questions such as “where is Frodo now?”

### The future of deep learning

Unsupervised learning had a catalytic effect in reviving interest in deep learning, but has since been overshadowed by the successes of purely 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 at classification tasks and produce impressive results in learning to play many different video games.

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.

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 recognition for a long time, new paradigms are needed to replace rule-based manipulation of symbolic expressions by operations on large vectors.

Received 25 February; accepted 1 May 2015.


This paper showed state-of-the-art machine translation results 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.


This paper showed that supervised training of very deep neural networks is much faster if the hidden layers are composed of ReLU.


This paper introduced a novel and effective way of training very deep neural networks by pre-training one hidden layer at a time using the unsupervised learning procedure for restricted Boltzmann machines.


This report demonstrated that the unsupervised pre-training methodology introduced in ref. 32 significantly improves performance on test data and generalizes the method to other unsupervised representation-learning techniques, such as auto-encoders.

This paper introduced LSTM recurrent networks, which have become a crucial ingredient in recent advances with recurrent networks because they are good at learning long-range dependencies.


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


Acknowledgements The authors would like to thank the Natural Sciences and Engineering Research Council of Canada, the Canadian Institute For Advanced Research (CIFAR), the National Science Foundation and Office of Naval Research for support. Y.L. and Y.B. are CIFAR fellows.

Author Information Reprints and permissions information is available at www.nature.com/reprints. The authors declare no competing financial interests. Readers are welcome to comment on the online version of this paper at go.nature.com/7cjbaa. Correspondence should be addressed to Y.L. (yann@cs.nyu.edu).