

Deep learning

Yann LeCun^{1,2}, Yoshua Bengio³ & Geoffrey Hinton^{4,5}

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^{1–4} and speech recognition^{5–7}, it has beaten other machine-learning techniques at predicting the activity of potential drug molecules⁸, analysing particle accelerator data^{9,10}, reconstructing brain circuits¹¹, and predicting the effects of mutations in non-coding DNA on gene expression and disease^{12,13}. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding¹⁴, particularly topic classification, sentiment analysis, question answering¹⁵ 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 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

¹Facebook AI Research, 770 Broadway, New York, New York 10003 USA. ²New York University, 715 Broadway, New York, New York 10003, USA. ³Department of Computer Science and Operations Research Université de Montréal, Pavillon André-Aisenstadt, PO Box 6128 Centre-Ville STN Montréal, Quebec H3C 3J7, Canada. ⁴Google, 1600 Amphitheatre Parkway, Mountain View, California 94043, USA. ⁵Department 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 techniques¹⁸. 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 hyperplane¹⁹. 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

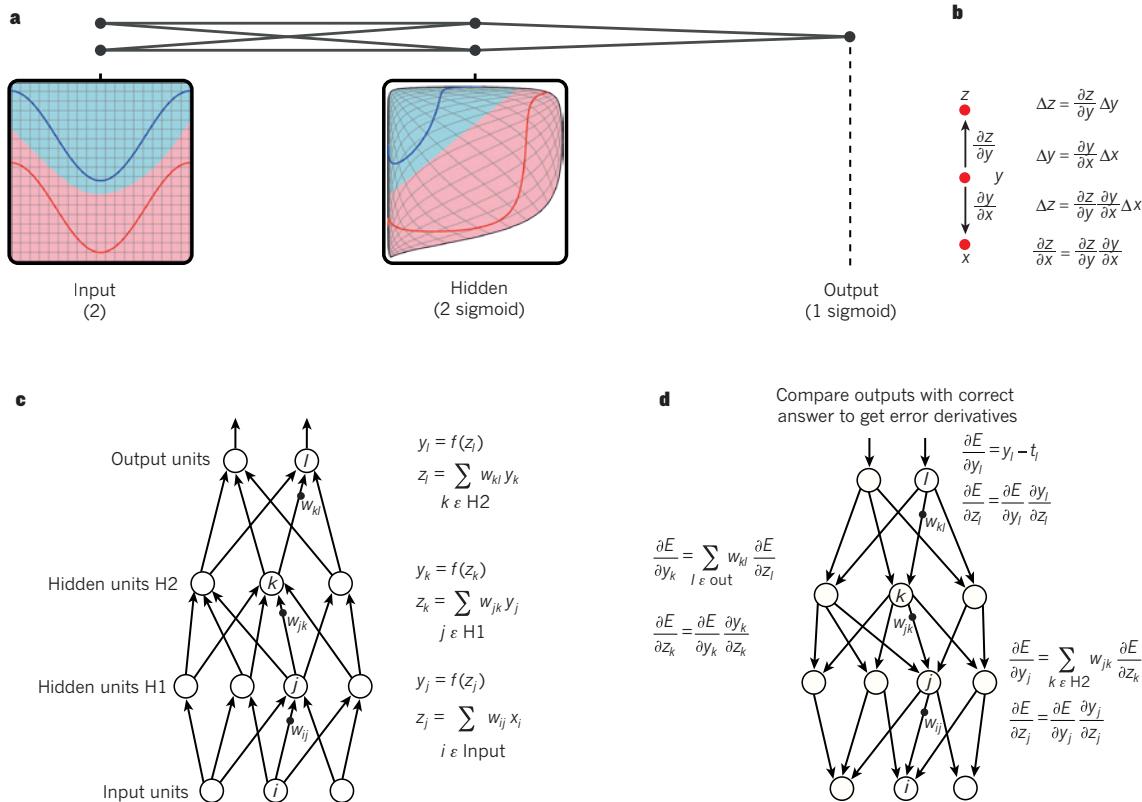


Figure 1 | Multilayer neural networks and backpropagation. **a**, A multilayer neural network (shown by the connected dots) can distort the input space to make the classes of data (examples of which are on the red and blue lines) linearly separable. Note how a regular grid (shown on the left) in input 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 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 change Δy creates a change Δz in z . Substituting one equation into the other gives the chain rule of derivatives — how Δx gets turned into Δz through multiplication by the product of $\partial y / \partial x$ and $\partial z / \partial y$. 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 $y_l - t_l$ if the cost function for unit l is $0.5(y_l - t_l)^2$, where t_l is the target value. Once the $\partial E / \partial z$ is known, the error-derivative for the weight w_{jk} on the connection from unit j in the layer below is just $y_j \partial E / \partial z$.

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic fox (1.0); Eskimo dog (0.6); white wolf (0.4); Siberian husky (0.4)

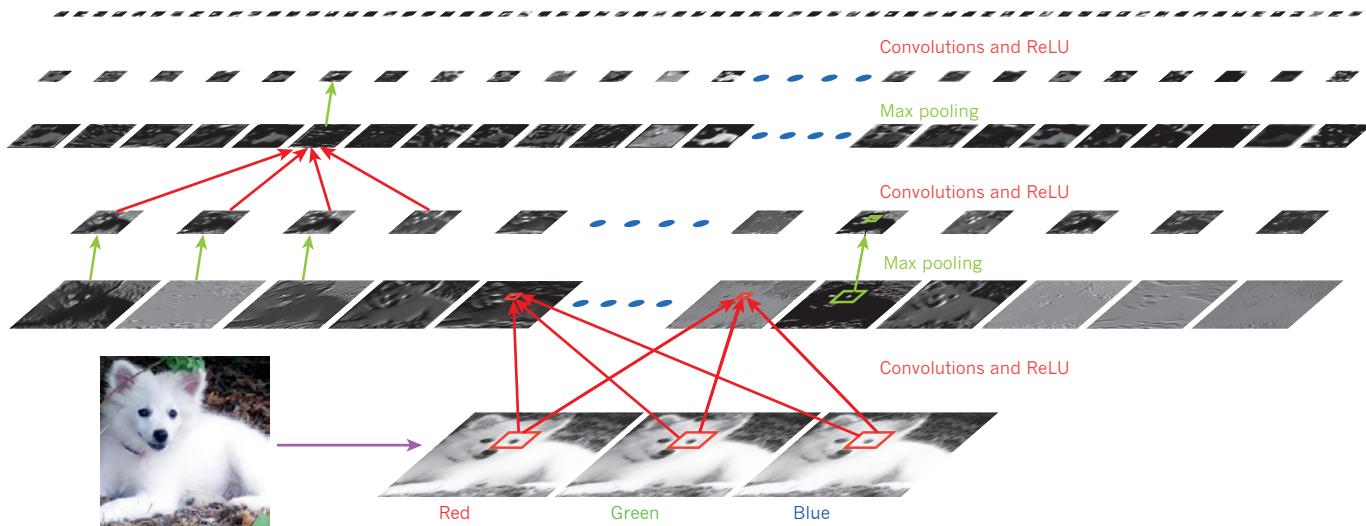


Figure 2 | Inside a convolutional network. The outputs (not the filters) of each layer (horizontally) of a typical convolutional network architecture applied to the image of a Samoyed dog (bottom left; and RGB (red, green, blue) inputs, bottom right). Each rectangular image is a feature map

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 methods²⁰, but generic features such as those arising with the Gaussian kernel do not allow the learner to generalize well far from the training examples²¹. 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^{22,23}, 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^{24–27}.

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

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.

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-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).

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

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 31–34) 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 limited³⁶.

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 program³⁷ 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 vocabulary³⁸ and was quickly developed to give record-breaking results on a large vocabulary task³⁹. By 2012, versions of the deep net from 2009 were being developed by many of the major speech groups⁶ and were already being deployed in Android phones. For smaller data sets, unsupervised pre-training helps to prevent overfitting⁴⁰, 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.

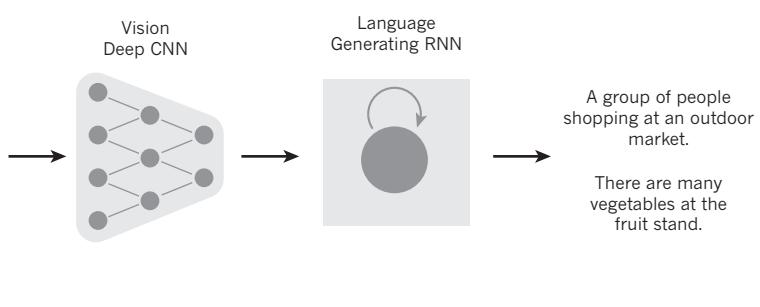
The convolutional and pooling layers in ConvNets are directly inspired by the classic notions of simple cells and complex cells in visual neuroscience⁴³, and the overall architecture is reminiscent of the LGN–V1–V2–V4–IT hierarchy in the visual cortex ventral pathway⁴⁴. When ConvNet models and monkeys 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 cortex⁴⁵. ConvNets have their roots in the neocognitron⁴⁶, 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 recognition⁴⁷ and document reading⁴². 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 Microsoft⁴⁹. ConvNets were also experimented with in the early 1990s for object detection in natural images, including faces and hands^{50,51}, and for face recognition⁵².

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 recognition⁵³, the segmentation of biological images⁵⁴ particularly for connectomics⁵⁵, and the detection of faces, text, pedestrians and human bodies in natural images^{36,50,51,56–58}. A major recent practical success of ConvNets is face recognition⁵⁹.

Importantly, images can be labelled at the pixel level, which will have applications in technology, including autonomous mobile robots and



A woman is throwing a **frisbee** in a park.



A **dog** is standing on a hardwood floor.



A **stop** sign is on a road with a mountain in the background



A little **girl** sitting on a bed with a **teddy bear**.



A group of **people** sitting on a boat in the water.



A **giraffe** standing in a forest with **trees** in the background.

Figure 3 | From image to text. Captions generated by a recurrent neural network (RNN) taking, as extra input, the representation extracted by a deep convolutional neural network (CNN) from a test image, with the RNN trained to ‘translate’ high-level representations of images into captions (top). Reproduced

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 understanding¹⁴ and speech recognition⁷.

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 approaches¹. This success came from the efficient use of GPUs, ReLUs, a new regularization technique called dropout⁶², 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^{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,

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 found⁶⁶ that it exploits this to achieve better ‘translation’ of images into captions.

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^{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 representations²¹. Both of these advantages arise from the power of composition and depend on the underlying data-generating distribution having an appropriate componential structure⁴⁰. 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 advantage⁷⁰ (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 words⁷¹. Each word in the context is presented to the network as a one-of-N vector, that is, one component has a value of 1 and the rest are 0. In the first layer, each word creates a different pattern of activations, or word vectors (Fig. 4). In a language model, the other layers of the network learn to convert the input word vectors into an output word vector for the predicted next word, which can be used to predict the probability for any word in the vocabulary to appear as the next word. The network learns word vectors that contain many active components each of which can be interpreted as a separate feature of the word, as was first demonstrated²⁷ in the context of learning distributed representations for symbols. These semantic features were not explicitly present in the input. They were discovered by the learning procedure as a good way of factorizing the structured relationships between the input and output symbols into 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 unreliable⁷¹. 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^{14,17,72–76}.

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 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 models⁷¹, 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^{77,78}.

Thanks to advances in their architecture^{79,80} and ways of training them^{81,82}, RNNs have been found to be very good at predicting the next character in the text⁸³ or the next word in a sequence⁷⁵, 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^{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

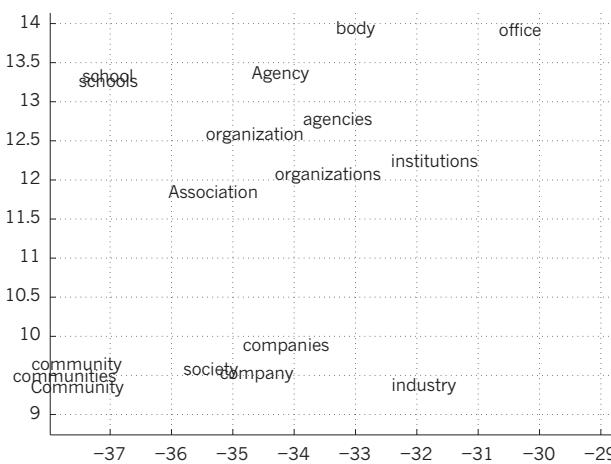


Figure 4 | Visualizing the learned word vectors. On the left is an illustration of word representations learned for modelling language, non-linearly projected to 2D for visualization using the t-SNE algorithm¹⁰³. On the right is a 2D representation of phrases learned by an English-to-French encoder-decoder recurrent neural network⁷⁵. 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)^{18,75}.

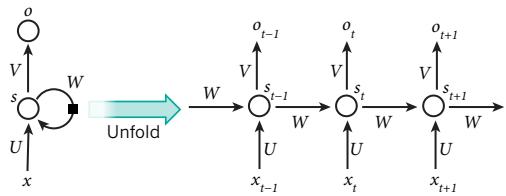


Figure 5 | A recurrent neural network and the unfolding in time of the computation involved in its forward computation. The artificial neurons (for example, hidden units grouped under node s with values s_t at time t) get inputs from other neurons at previous time steps (this is represented with the 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_i (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 , and all the parameters.

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 in which all the layers share the same weights. Although their main purpose is to learn long-term dependencies, theoretical and empirical evidence shows that it is difficult to learn to store information for very long⁷⁸.

To correct for that, one idea is to augment the network with an explicit memory. The first proposal of this kind is the long short-term memory (LSTM) networks that use special hidden units, the natural behaviour of which is to remember inputs for a long time⁷⁹. A special unit called the memory cell acts like an accumulator or a gated leaky neuron: it has a connection to itself at the next time step that has a weight of one, so it copies its own real-valued state and accumulates the external signal, but this self-connection is multiplicatively gated by another unit that learns to decide when to 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 step⁸⁷, 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 to⁸⁸, and memory networks, in which a regular network is augmented by a kind of associative memory⁸⁹. 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 list⁸⁸. 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 inference⁹⁰. 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?”⁸⁹.

The future of deep learning

Unsupervised learning^{91–98} 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 systems⁹⁹ at classification tasks and produce impressive results in learning to play many different video games¹⁰⁰.

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 recognition for a long time, new paradigms are needed to replace rule-based manipulation of symbolic expressions by operations on large vectors¹⁰¹. ■

Received 25 February; accepted 1 May 2015.

- Krizhevsky, A., Sutskever, I. & Hinton, G. ImageNet classification with deep convolutional neural networks. In *Proc. Advances in Neural Information Processing Systems 25* 1090–1098 (2012). **This report was a breakthrough that used convolutional nets to almost halve the error rate for object recognition, and precipitated the rapid adoption of deep learning by the computer vision community.**
- Farabet, C., Couprie, C., Najman, L. & LeCun, Y. Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1915–1929 (2013).
- 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).
- Szegedy, C. et al. Going deeper with convolutions. Preprint at <http://arxiv.org/abs/1409.4842> (2014).
- 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).
- 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 application of deep learning.**
- Sainath, T., Mohamed, A.-R., Kingsbury, B. & Ramabhadran, B. Deep convolutional neural networks for LVCSR. In *Proc. Acoustics, Speech and Signal Processing* 8614–8618 (2013).
- Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E. & Svetnik, V. Deep neural nets as a method for quantitative structure-activity relationships. *J. Chem. Inf. Model.* **55**, 263–274 (2015).
- 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).
- Kaggle. Higgs boson machine learning challenge. Kaggle <https://www.kaggle.com/c/higgs-boson> (2014).
- 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 embeddings. In *Proc. Empirical Methods in Natural Language Processing* <http://arxiv.org/abs/1406.3676v3> (2014).
16. Jean, S., Cho, K., Memisevic, R. & Bengio, Y. On using very large target vocabulary for neural machine translation. In *Proc. ACL-IJCNLP* <http://arxiv.org/abs/1412.2007> (2015).
17. Sutskever, I., Vinyals, O. & Le, Q. V. Sequence to sequence learning with neural networks. In *Proc. Advances in Neural Information Processing Systems* **27** 3104–3112 (2014). **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.**
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).
23. Rosenblatt, F. *The Perceptron — A Perceiving and Recognizing Automaton*. Tech. Rep. 85-460-1 (Cornell Aeronautical Laboratory, 1957).
24. Werbos, P. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard Univ. (1974).
25. Parker, D. B. *Learning Logic Report TR-47* (MIT Press, 1985).
26. LeCun, Y. Une procédure d'apprentissage pour Réseau à seuil assymétrique in *Cognitiva 85: à la Frontière de l'Intelligence Artificielle, des Sciences de la 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).
32. Hinton, G. E., Osindero, S. & Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput.* **18**, 1527–1554 (2006). **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.**
33. Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. Greedy layer-wise training of deep networks. In *Proc. Advances in Neural Information Processing Systems* **19** 153–160 (2006). **This report demonstrated that the unsupervised pre-training method introduced in ref. 32 significantly improves performance on 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).
39. Dahl, G. E., Yu, D., Deng, L. & Acero, A. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Trans. Audio Speech Lang. Process.* **20**, 33–42 (2012).
40. Bengio, Y., Courville, A. & Vincent, P. Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Machine Intell.* **35**, 1798–1828 (2013). **This paper introduced neural language models, which learn to convert a word symbol into a word vector or word embedding composed of learned semantic features in order to predict the next word in a sequence.**
41. LeCun, Y. et al. Handwritten digit recognition with a back-propagation network. In *Proc. Advances in Neural Information Processing Systems* 396–404 (1990). **This is 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. Fellman, 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 tolerant of deformations and shifts in position. *Pattern Recognition* **15**, 455–469 (1982).
47. Waibel, A., Hanazawa, T., Hinton, G. E., Shikano, K. & Lang, K. Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoustics Speech Signal Process.* **37**, 328–339 (1989).
48. Bottou, L., Fogelman-Soulie, F., Blanchet, P. & Lienard, J. Experiments with time delay networks and dynamic time warping for speaker independent isolated digit recognition. In *Proc. EuroSpeech* 89 537–540 (1989).
49. Simard, D., Steinhaus, 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).
55. Turaga, S. C. et al. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Comput.* **22**, 511–538 (2010).
56. Garcia, C. & Delakis, M. Convolutional face finder: a neural architecture for fast and robust face detection. *IEEE Trans. Pattern Anal. Machine Intell.* **26**, 1408–1423 (2004).
57. Osadchy, M., LeCun, Y. & Miller, M. Synergistic face detection and pose estimation with energy-based models. *J. Mach. Learn. Res.* **8**, 1197–1215 (2007).
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., Courville, 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).
63. Sermanet, P. et al. Overfeat: integrated recognition, localization and detection using convolutional networks. In *Proc. International Conference on Learning Representations* <http://arxiv.org/abs/1312.6229> (2014).
64. Girshick, R., Donahue, J., Darrell, T. & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. Conference on Computer Vision and Pattern Recognition* 580–587 (2014).
65. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *Proc. International Conference on Learning Representations* <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 *Proc. Advances in Neural Information Processing Systems* **13** 932–938 (2001). **This paper introduced neural language models, which learn to convert a word symbol into a word vector or word embedding composed of learned semantic features in order to predict the next word in a sequence.**
72. Cho, K. et al. Learning phrase representations using RNN encoder-decoder

- for statistical machine translation. In *Proc. Conference on Empirical Methods in Natural Language Processing* 1724–1734 (2014).
73. Schwenk, H. Continuous space language models. *Computer Speech Lang.* **21**, 492–518 (2007).
 74. Socher, R., Lin, 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**, 1735–1780 (1997).
 - 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.
 80. ElHilli, S. & Bengio, Y. Hierarchical recurrent neural networks for long-term dependencies. In *Proc. Advances in Neural Information Processing Systems* 8 <http://papers.nips.cc/paper/1102-hierarchical-recurrent-neural-networks-for-long-term-dependencies> (1995).
 81. Sutskever, I. *Training Recurrent Neural Networks*. PhD thesis, Univ. Toronto (2012).
 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).
 88. Graves, A., Wayne, G. & Danihelka, I. Neural Turing machines. <http://arxiv.org/abs/1410.5401> (2014).
 89. Weston, J., Chopra, S. & Bordes, A. Memory networks. <http://arxiv.org/abs/1410.3916> (2014).
 90. Weston, J., Bordes, A., Chopra, S. & Mikolov, T. Towards AI-complete question answering: a set of prerequisite toy tasks. <http://arxiv.org/abs/1502.05698> (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 using gated MRFs. *IEEE Trans. Pattern Anal. Machine Intell.* **35**, 2206–2222 (2013).
 97. Bengio, Y., Thibodeau-Laufer, E., Alain, G. & Yosinski, J. Deep generative stochastic networks trainable by backprop. In *Proc. 31st International Conference on Machine Learning* 226–234 (2014).
 98. Kingma, D., Rezende, D., Mohamed, S. & Welling, M. Semi-supervised learning with deep generative models. In *Proc. Advances in Neural Information Processing Systems* 27 3581–3589 (2014).
 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 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).

- title: Deep Learning
- journal: Nature
- year: 2015
- author: Yann LeCun, Yoshua Bengio, and Geoffrey Hinton

Deep Learning

要旨

深層学習は、複数の処理層で構成された計算モデルに、複数の抽象度を持つデータの表現を学習させる手法である。

この手法は、音声認識、視覚物体認識、物体検出などのほか、創薬やゲノミクスなどのさまざまな分野で、最先端の技術を劇的に向上させている。

深層学習は、バックプロパゲーションアルゴリズムを用いて、各層の表現を計算するために使用される機械の内部パラメータを、前層の表現からどのように変更すべきかを示すことで、大規模なデータセットの複雑な構造を発見する。深層畳み込みネットワークは、画像、ビデオ、音声、聴覚の処理に飛躍的な進歩をもたらし、リカレントネットワークはテキストや音声などの連続したデータに光を当てる。

機械学習技術は、現代社会のさまざまな場面で活躍している。ウェブ検索、ソーシャルネットワーク上のコンテンツフィルタリング、eコマースサイトでのレコメンドなど、現代社会のさまざまな場面で機械学習技術が活用されており、カメラやスマートフォンなどのコンシューマ製品にも搭載される場合が増えている。機械学習システムは、画像内の物体を識別したり、音声をテキストに変換したり、ニュース項目や投稿、商品をユーザの興味に合わせてマッチングさせたり、関連性の高い検索結果を選択したりするために使用される。これらの応用事例では、深層学習と呼ばれる技術が利用されることが多くなっている。

従来の機械学習技術では、自然界のデータを生のまま処理するには限界があった。

何十年もの間、パターン認識システムや機械学習システムを構築するためには、慎重なエンジニアリングと、画像の画素値などの生データを、学習サブシステム（多くの場合、分類器）が入力パターンを検出または分類できるような適切な内部表現または特徴ベクトルに変換する特徴抽出器を設計するため、かなりの分野の専門知識が必要であった。

表現学習とは、機械に生データを入力し、検出や分類に必要な表現を自動的に発見することができる一連の手法である。

深層学習法は、複数のレベルの表現を持つ表現学習法であり、あるレベルの表現（生の入力から始まる）を、より高い、やや抽象的なレベルの表現に変換する、単純だが非線形のモジュールを組み合わせることで得られる。このような変換を十分に行なうことで、非常に複雑な機能を学習することができる。分類課題では、表現の上位層は、識別に重要な入力の側面を增幅し、無関係な変動を抑制する。例えば、画像は画素値の配列であり、表現の第1層で学習された特徴は、画像内の特定の方向や位置におけるエッジの有無を表している。第2層では、エッジの位置のわずかな違いに関わらず、エッジの特定の配置を検出することで、モチーフを検出します。第3層では、モチーフをより大きな組み合わせにして、身近な物体のパートに対応させ、後続層ではそのパートの組み合わせとして物体を検出する。深層学習で重要なのは、これらの特微量の層を人間の技術者が設計するのではなく、一般的な方法でデータから学習することである。一般的な学習方法を用いて、データから学習される。

深層学習は、長年にわたって人工知能コミュニティの最善の試みに抵抗してきた問題を解決する上で大きな進歩を遂げている。深層学習は、高次元データの複雑な構造を発見するのに非常に優れていることが判明しており、科学、ビジネス、政府の多くの領域に適用されている。画像認識（1-4）や音声認識（5-7）の記録を塗り替えただけでなく、潜在的な薬物分子の活性予測（8）、粒子加速器データの分析（9, 10）、脳回路の再構築（11）、非コードDNA変異が遺伝子発現や疾患に及ぼす影響の予測（12, 13）などでも、他の機械学習技術を凌駕している。さらに驚くべきことに、深層学習は自然言語理解のさまざまな課題（14）、特にトピック分類、感情分析、質問応答（15）、言語翻訳（16, 17）において極めて有望な結果を出している。

深層学習は、人手をほとんど必要としないため、計算量やデータ量の増加を容易に利用することができ、近い将来、さらに多くの成功を収めることができると考えられる。現在、ディープニューラルネットワークのために開発されている新しい学習アルゴリズムとアーキテクチャは、この進歩をさらに加速させるだろう。

1. 教師あり学習 Supervised learning

機械学習の最も一般的な形態は、ディープかどうかに関わらず、教師あり学習である。例えば、家、車、人、ペットが写っている画像を分類するシステムを作りたいとする。まず、家、車、人、ペットの画像の大規模なデータセットを収集し、それぞれにカテゴリーのラベルを付ける。学習の際には、機械に画像を見せて、各カテゴリーごとに1つのスコアのベクトルという形で出力する。目的のカテゴリーが、全カテゴリーの中で最も高いスコアを持つようにしたいが、学習前にそれが実現する可能性は低い。そこで、出力されたスコアと目的のスコアパターンとの誤差（または距離）を測定する目的関数を計算する。そして、機械はこの誤差を減らすために、内部の調整可能なパラメータを変更する。これらの調整可能なパラメータは、しばしば重みと呼ばれ、機械の入出力機能を定義する「つまみ」と見なすことができる実数である。一般的な深層学習システムでは、これらの調整可能な重みが何億個もあり、機械を訓練するためのラベル付けされた例が何億個もある。

重みベクトルを適切に調整するために、学習アルゴリズムは、各重みについて、その重みをわずかに増加させた場合に誤差がどの程度増加または減少するかを示す勾配ベクトルを計算する。そして、重みベクトルは、勾配ベクトルとは逆方向に調整される。

すべての学習例で平均化された目的関数は、重み値の高次元空間における一種の丘陵地帯と見なすことができる。負の勾配ベクトルは、この風景の中で最も急降下する方向を示しており、出力誤差が平均的に小さくなる最小値に近づく。

実際には、ほとんどの実務者が、確率的勾配降下法（SGD）と呼ばれる手順を使用している。SGDとは、いくつかの例の入力ベクトルを表示し、出力と誤差を計算し、それらの例の平均勾配を計算し、それに応じて重みを調整するというものである。この処理は、目的関数の平均が減少しなくなるまで、学習セットからの多くの小さな例のセットに対して繰り返される。この方法は、小数例のデータセットが全例データセットの平均勾配のノイズの任意の推定値を与えるため、確率的と呼ばれる。この単純な手順は、はるかに精巧な最適化技術と比較すると、通常、驚くほど早く良い重みセットを見つけることができる（18）。学習後、テストデータセットと呼ばれる別の例のデータセットでシステムの性能を測定する。これは、機械の一般化能力、つまり、学習時には見たことのない新しい入力に対して、適切な答えを出す能力を検証するためのものである。

現在、実用化されている機械学習の多くは、手作業で作成した特微量の上に線形分類器を使用している。2クラスの線形分類器は、特徴ベクトル成分の加重和を計算する。この加重和がある閾値以上であれば、入力は特定のカテゴリーに属するものとして分類される。

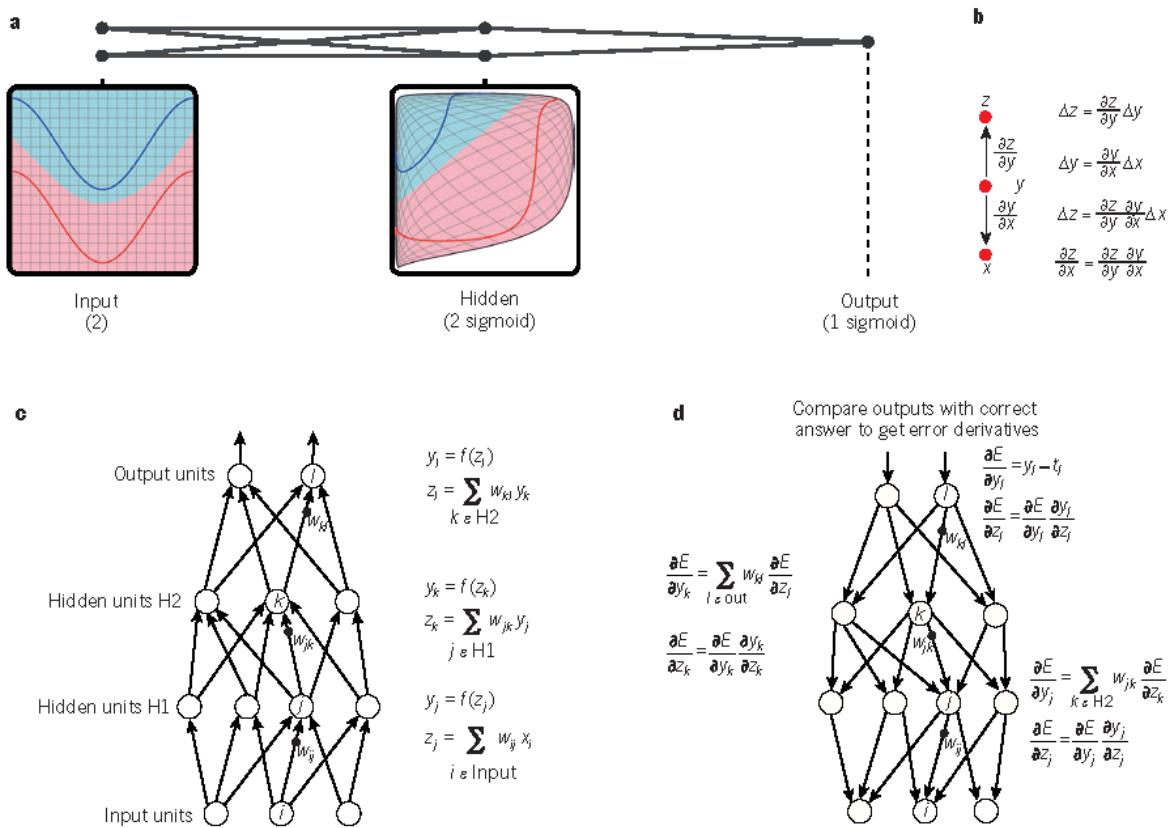


図 1 多層ニューラルネットワークと誤差逆伝播

a 多層構造のニューラルネットワーク(連結されたドットで示す)は、入力空間を歪めて、データのクラス(赤と青の線上にある例)を線形分離可能にすることができる。

入力空間の規則的なグリッド(左図)が、隠れユニットによってどのように変換されるか(中図)に注目。

これは2つの入力ユニット、2つの隠れユニット、1つの出力ユニットを持つ例であるが、物体認識や自然言語処理に用いられるネットワークには、数万から数十万のユニットが含まれている。C. Olah (<http://colah.github.io/>) の許可を得て転載。

b 微分の連鎖則は、2つの小さな効果(x の y に対する微少変化と、 y の z に対する変化)がどのように構成されるかを教えてくれる。

x の微少な変化 Δx は、 $\partial y / \partial x$ をかけられることによって、まず y の小さな変化 Δy に変換される(つまり、偏微分の定義)。同様に、 Δy の変化は z の変化 Δz を生み出す。一方の式を他方の式に代入すると、微分の連鎖則(Δx が $\partial y / \partial x$ と $\partial z / \partial x$ の積の乗算によって Δz に変わる)が得られる。

この法則は x, y, z がベクトルの場合にも有効である(導関数はヤコビ行列となる)。

c 2つの隠れ層と1つの出力層を持つニューラルネットにおいて、勾配を逆伝播するモジュールを構成する順方向バスの計算に使用される方程式。各層では、まず、各ユニットへの全入力 z を計算する。これは、下層のユニットの出力を加重加算したものである。次に、非線形関数 $f(\cdot)$ を z に適用して、ユニットの出力を得る。簡略化のため、バイアス項は省略している。ニューラルネットワークで使われる非線形関数には、近年よく使われる ReLU(整流線形ユニット) $f(z) = \max(0, z)$ のほか、ハイパー・ボリック・タンジェント、 $f(z) = (\exp(z) - \exp(-z)) / (\exp(z) + \exp(-z))$ やロジスティック関数 $f(z) = 1 / (1 + \exp(-z))$ などの一般的なシグモイドがある。

d 逆向バスの計算に使用される方程式。各隠れ層では、各ユニットの出力に関する誤差導関数を計算する。これは、上位層のユニットへの全入力に関する誤差導関数の加重和である。次に、出力に対する誤差導関数に $f(\cdot)$ の勾配を乗じることで、入力に対する誤差導関数に変換する。出力層では、ユニットの出力に関する誤差微分を、コスト関数を微分することで計算する。これにより、ユニット l のコスト関数が $0.5(y_l - t_l)^2$ の場合、 $y_l - t_l$ となり、 t_l は目標値である。 $\partial E / \partial z_k$ がわかれば、下位層のユニット j からの接続の重み w_{jk} の誤差微分は、ちょうど $y_j \partial E / \partial z_k$ となる。

1960年代以降、線形分類器は入力空間を超平面で区切られた半空間という非常に単純な領域にしか切り分けられないことがわかつていて(19)。しかし、画像認識や音声認識のような問題では、入出力関数は、物体の位置や向き、照明の変化、音声ピッチやアクセントの変化などの無関係な入力の変化には鈍感である一方、特定の微細な変化(例えば、白いオオカミとサモエドと呼ばれるオオカミに似た白い犬種の違い)には非常に敏感である必要がある。画素レベルでは、2頭のサモエドが異なるポーズや環境で撮影された画像は互いに大きく異なるが、サモエドとオオカミが同じ姿勢で同じような背景で撮影された2つの画像は互いによく似ていることがある。線形分類器やその他の「浅い」分類器は、前者2つの画像を同じカテゴリに分類する一方で、後者2つの画像を区別することはできない。このため、浅い分類器には、選択性一不变性シレンマを解決する優れた特徴抽出器が必要になる。つまり、識別に重要な画像の側面には選択性があり、動物のポーズなどの無関係な側面には不变性がある表現を生成する必要がある。分類器をより強力にするためには、カーネル法のように汎用の非線形特徴を用いることができる(20)。ガウシアンカーネルで生じるような汎用の特徴では、学習例から遠く離れたところで学習者をうまく一般化することができない(21)。従来手法では、優れた特徴抽出器を手作業で設計していたが、これにはかなりのエンジニアリング技術と専門領域の知識が必要である。しかし、汎用の学習手順を用いて良い特徴を自動的に学習することができれば、このようなことはすべて避けることができる。これが深層学習の最大の利点である。

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic fox (1.0); Eskimo dog (0.6); white wolf (0.4); Siberian husky (0.4)

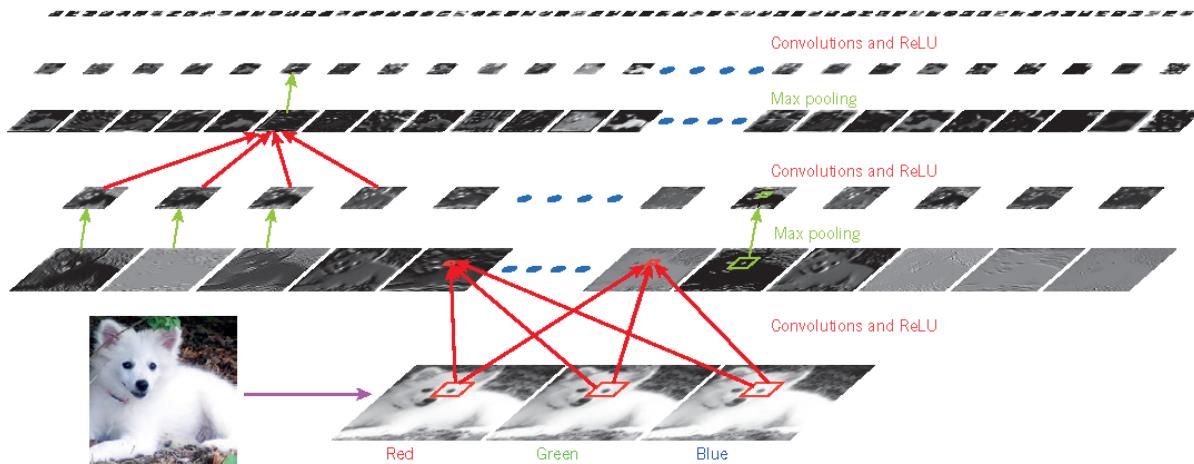


図2: 畳み込みネットワークの内部

典型的な畳み込みネットワークアーキテクチャの各層(水平方向)の出力(フィルタではない)を、サモエド犬の画像(左下:右下はRGB(赤、緑、青)の入力に適用したもの。各矩形の画像は、画像の各位置で検出された、学習された特徴の1つに対する出力に対応する特徴地図である。情報はボトムアップで流れ、下位の特徴は方向性のあるエッジ検出器として機能し、出力の各画像クラスに対してスコアが計算される。ReLU、整流線形ユニット

深層学習アーキテクチャは、単純なモジュールの多層スタックであり、すべて(またはほとんど)のモジュールが学習の対象となり、多くのモジュールが非線形の入出力マッピングを計算します。スタックの各モジュールは、入力を変換して、表現の選択性と不变性の両方を高める。

複数の非線形層、例えば5~20層の深さがあれば、システムは入力に対して非常に複雑な機能を実装することができる。この機能は、サモエドとホワイトウルフを区別するような微細なディテールに敏感であるとともに、背景、ポーズ、照明、周囲の対象などの無関係な大きな変化には影響を受けない。

2. 多層アーキテクチャを訓練するための誤差逆伝播 Backpropagation to train multilayer architectures

パターン認識の黎明期(22, 23)から、研究者の目的は、手作業で作成した特微量を学習可能な多層ネットワークに置き換えることだったが、そのシンプルさにもかかわらず、1980年代半ばまで、この解決策は広く理解されていなかった。

結論から言うと、多層アーキテクチャーは単純な確率的勾配降下法で学習できる。モジュールが入力と内部の重みの比較的滑らかな関数である限り、誤差逆伝播法を用いて勾配を計算することができる。この方法が可能であること、そしてそれが機能することは、1970年代から1980年代にかけて、いくつかの異なるグループによって独自に発見された(24-27)。

多層モジュールの重みに対する目的関数の勾配を計算する誤差逆伝播法は、導関数の連鎖法則を応用したものに他ならない。ここで重要なのは、あるモジュールの入力に対する目的関数の導関数(または勾配)は、そのモジュールの出力(または次のモジュールの入力)に対する勾配から逆算することで計算できるということである(図1)。誤差逆伝播法の方程式を繰り返し適用して、全モジュールに勾配を伝搬させることができる。最上位の出力(ネットワークが予測を生成する場所)から始まり、最下部(外部入力が入力される場所)に至るまで、勾配を伝搬させる。これらの勾配が計算されると、各モジュールの重みに対する勾配を簡単に計算することができる。

深層学習の多くのアプリケーションでは、固定サイズの入力(例えば、画像)を固定サイズの出力(例えば、いくつかのカテゴリーのそれぞれに対する確率)にマッピングすることを学習するフィードフォワードニューラルネットワークアーキテクチャ(図1)が使用されている。

ある層から次の層に移る際には、一連のユニットが、前層からの入力の加重和を計算し、その結果を非線形関数に通す。現在、最もよく使われている非線形関数は、半波整流器 $f(z) = \max(z, 0)$ である整流線形ユニット(ReLU)である。過去数十年のニューラルネットでは、 $\tanh(z)$ や $1/(1 + \exp(-z))$ など、より滑らかな非線形関数が使用されていたが、ReLUは通常、層数の多いネットワークでは学習速度が速く、教師なしの事前学習なしで深い教師付きネットワークの学習が可能である。(28)。入力層や出力層に属さないユニットを從来は隠れユニットと呼んでいた。隠れ層は、入力を非線形に歪ませ、最後の層でカテゴリーが線形に分離できるようにすることができる(図1)。

1990年代後半、ニューラルネットとバックプロパゲーションは、機械学習のコミュニティからは見放され、コンピュータビジョンや音声認識のコミュニティからは無視されていた。少ない予備知識で有用な多段の特徴抽出器を学習することは不可能であると広く考えられていた。特に、単純な勾配降下法では、劣悪なローカルミニマム、つまり、わずかな変化でも平均誤差を減らすことができない重み設定に陥ってしまうと考えられていた。

実際には、大規模なネットワークでは貧弱なローカルミニマムが問題になることはほとんどない。初期条件にかかわらず、システムはほとんど常に非常に似た品質の解に到達する。最近の理論的、経験的な結果は、一般的に局所的な最小値は深刻な問題ではないことを強く示唆している。それどころか、勾配がゼロで、表面がほとんどの次元で上にカーブし、残りの次元で下にカーブするサドルポイントが、組み合わせ的に多数詰め込まれている(29,30)。分析の結果、下向きに曲がる方向がわざかしかないサドルポイントが非常に多く存在するが、ほとんどすべてのサドルポイントで目的関数の値が非常に似通っていることがわかったようだ。したがって、アルゴリズムがこれらのサドルポイントのどれに引っかかるかはあまり重要ではない。

ディープフィードフォワードネットワークへの関心は、2006年頃、カナダ高等研究所(CIFAR)が集めた研究者グループによって復活した(参考文献31~34)。この研究者たちは、ラベル付きデータを必要とせずに特徴検出器の層を作成できる教師なし学習法を導入した。特徴検出器の各層を学習する目的は、下位層にある特徴検出器の活動(または生の入力)を再構成またはモデル化できるようにすることである。この再構築の目的を用いて、徐々に複雑になる特徴検出器の層をいくつか「事前学習」することで、深層ネットワークの重みを適切な値に初期化することができる。その後、出力ユニットの最終層をネットワークの最上部に追加し、標準的なバックプロパゲーションを用いて深層システム全体を微調整することができる(33-35)。これは、手書きの数字を認識したり、歩行者を検出したりする際に、特にラベル付けされたデータの量が非常に限られている場合に、非常にうまく機能した(36)。

この事前学習法を音声認識に初めて適用したのは、プログラムの作成が容易で(37)、ネットワークの学習速度を10倍から20倍に高めることができる高速なGPU(Graphics Processing Unit)の登場であった。2009年には、音波から抽出した係数の短い時間窓を、窓の中心にあるフレームが表すさまざまな音声の断片の確率にマッピングするために、この手法が使用された。このシステムは、少ない語彙を使用する標準的な音声認識ベンチマークで記録的な結果を出し(38)、すぐに大規模な語彙の課題で記録的な結果を出すように開発された(39)。2012年には、2009年に開発されたディープネットのバージョンが、多くの主要な音声認識グループ(6)によって開発され、すでにAndroid携帯電話に搭載されている。データセットが小さい場合、教師なしの事前学習を行うことで、過学習を防ぐことができる(40)。これにより、ラベル付けされた例の数が少ない場合や、「ソース」課題の例はたくさんあるが、「ターゲット」課題の例はほとんどないという転送設定の場合に、一般化が大幅に向上する。深層学習が回復すると、事前学習の段階は小さなデータセットにしか必要ないことがわかった。

しかし、ディープフィードフォワードネットワークの中には、隣接する層間に完全な接続性があるネットワークよりも、はるかに簡単に学習でき、一般化できる特定のタイプがあった。それは、畳み込みニューラルネットワーク(ConvNet)である(41, 42)。ConvNetは、ニューラルネットワークが人気を失っていた時期に多くの実用的な成功を収め、最近ではコンピュータビジョンのコミュニティで広く採用されている。

3. 畳み込みニューラルネットワーク Convolutional neural networks

ConvNets は、複数の配列で構成されるデータを処理するように設計されている。例えば、カラー画像は、3つの色チャンネルのピクセル強度を含む3つの2D配列で構成されている。多くのデータモダリティは、複数の配列の形をしている。言語を含む信号や系列は1次元、画像やオーディオのスペクトログラムは2次元、ビデオやボリューム画像は3次元である。ConvNetsには、自然な信号の特性を利用した4つの重要なアイデアがある。局所的な接続、共有重み、ブーリング、多数の層の使用である。

典型的な ConvNet アーキテクチャ(図2)は、一連のステージとして構成されている。最初の数ステージは、2種類の層で構成されている。畳み込み層とブーリング層である。畳み込み層ユニットは、特徴マップで構成されており、各ユニットは、フィルターバンクと呼ばれる重みセットを介して、前層の特徴マップのローカルバッチに接続されている。この局所的な重み付けの結果は、ReLUなどの非線形性に通される。特徴地図の全ユニットは、同じフィルターバンクを共有する。層内の異なる特徴地図は、異なるフィルターバンクを使用する。このようなアーキテクチャを採用した理由は2つある。まず、画像などの配列データでは、局所的な値の集まりは相関性が高く、局所的に特徴的なモチーフが形成されていることが多い、これを容易に検出することができる。2つ目は、画像などの信号の局所的な統計量は、場所に依存しないということである。つまり、画像のある部分に現れるモチーフは、どこにでも現れる可能性がある。そのため、異なる場所にあるユニットが同じ重みを共有し、配列の異なる部分で同じパターンを検出するというアイデアが生まれた。数学的には、特徴地図が行うフィルタリング操作は離散的な畳み込みであり、それが名前の由来となっている。

畳み込み層の役割は、前層の特徴の局所的な結合を検出することだが、ブーリング層の役割は、意味的に類似した特徴を1つにまとめることがある。モチーフを形成する特徴の相対的位置は多少異なることがあるため、各特徴の位置を粗視化することで、モチーフを確実に検出することができる。一般的なブーリングユニットは、1つの特徴地図(またはいくつかの特徴地図)内のユニットのローカルバッチの最大値を計算する。隣接するブーリングユニットは、1行または1列以上シフトしたバッチから入力を受けることで、表現の次元を下げ、小さなシフトや歪みに対する不变性を作り出す。畳み込み、非線形、ブーリングを2段または3段重ね、さらに畳み込み層と完全連結層を重ねる。ConvNetでの勾配のバックプロパゲーションは、通常のディープネットワークと同様に簡単で、すべてのフィルターバンクのすべての重みを学習することができる。

深層ニューラルネットワークは、自然界の信号の多くが、下位の特徴を組み合わせることで上位の特徴が得られる「構成的階層」であるという性質を利用している。画像では、エッジの局所的な組み合わせがモチーフを形成し、モチーフがパートに集まり、パートがオブジェクトを形成する。音声やテキストにおいても、音から音声、音素、音節、単語、文へと同様の階層が存在する。ブーリングにより、前層の要素の位置や見え方が変わっても、表現はほとんど変わらない。

ConvNet の畳み込み層とブーリング層は、視覚神経科学における単純細胞と複雑細胞という古典的な概念に直接インスピライアされたものであり(43)、全体のアーキテクチャは、視覚野の腹側経路におけるLGN-V1-V2-V4-IT階層を彷彿とさせるものである(44)。

ConvNet モデルとサルに同じ絵を見せると、ConvNet の高レベルユニットの活性化は、サルの下頭側頭葉皮質にある160個のニューロンのランダムセットの分散の半分を説明する(45)。ConvNet のルーツは neocognitron(46)で、アーキテクチャは似ているが、バックプロパゲーションのようなエンドツーエンドの教師付き学習アルゴリズムは持っていないかった。時間遅延ニューラルネットと呼ばれる原始的な1次元のConvNetは、音素と単純な単語の認識に使用された(47,48)。

畳み込みネットワークの応用例は、1990年代初頭に数多くあり、音声認識(47)や文書読解のための時間遅延ニューラルネットワークに始まった(42)。この文書読解システムでは、言語制約を実装した確率モデルと共同で学習した畳み込みネットワークを使用していた。1990年代後半には、このシステムは米国の全小切手の10%以上を読み取っていた。ConvNet をベースにした光学式文字認識や手書き認識のシステムは、後に Microsoft 社によって多数導入された(49)。

また、ConvNet は、顔や手を含む自然画像中の物体検出(50,51)や顔認識(52)のために、1990年代初頭に実験された。

ディープ畳み込みネットワークによる画像理解 Image understanding with deep convolutional networks

2000年代初頭から、ConvNets は、画像中の物体や領域の検出、領域切り分け、認識に適用され、大きな成功を収めてきた。これらは、交通標識の認識(53)、生物学的画像の切り分け(54)、特にコネクティオミクス(55)、自然画像中の顔、字、歩行者、人体の検出(36,50,51,56-58)など、ラベル付きデータが比較的豊富に存在する課題であった。

最近の ConvNets の実用的な成功例としては、顔認識が挙げられる(59)。

重要なのは、画像を画素レベルでラベリングできることであり、これは自律移動ロボットや自動運転車などの技術に応用できるだろう(60,61)。Mobileye 社や NVIDIA 社などの企業は、このような ConvNet ベースの手法を、自動車用の次期ビジョンシステムに採用している。また、自然言語理解(14)や音声認識などのアプリケーションも重要になってきている(7)。

このような成功にもかかわらず、ConvNets はコンピュータビジョンや機械学習の主流から見放されていたが、2012年にImageNetコンテストが開催された。深層畳み込みネットワークを、ウェブから収集した約100万枚の画像と1,000種類のクラスを含むデータセットに適用したところ、競合する最良のアプローチのエラー率をほぼ半減させるという素晴らしい結果が得られた(1)。この成功は、GPU, ReLU, dropout(62)と呼ばれる新しい正則化技術、および既存の学習例を変形させてより多くの学習例を生成する技術を効率的に使用したことによるものである。この成功は、コンピュータビジョンに革命をもたらした。ConvNets は現在、ほとんどすべての認識・検出課題で主流となっており(4,58,59,63-65)、いくつかの課題では人間の性能に近づいている。最近の見事なデモンストレーションでは、ConvNets とリカレントネットモジュールを組み合わせて、画像のキャプションを生成している(図3)。

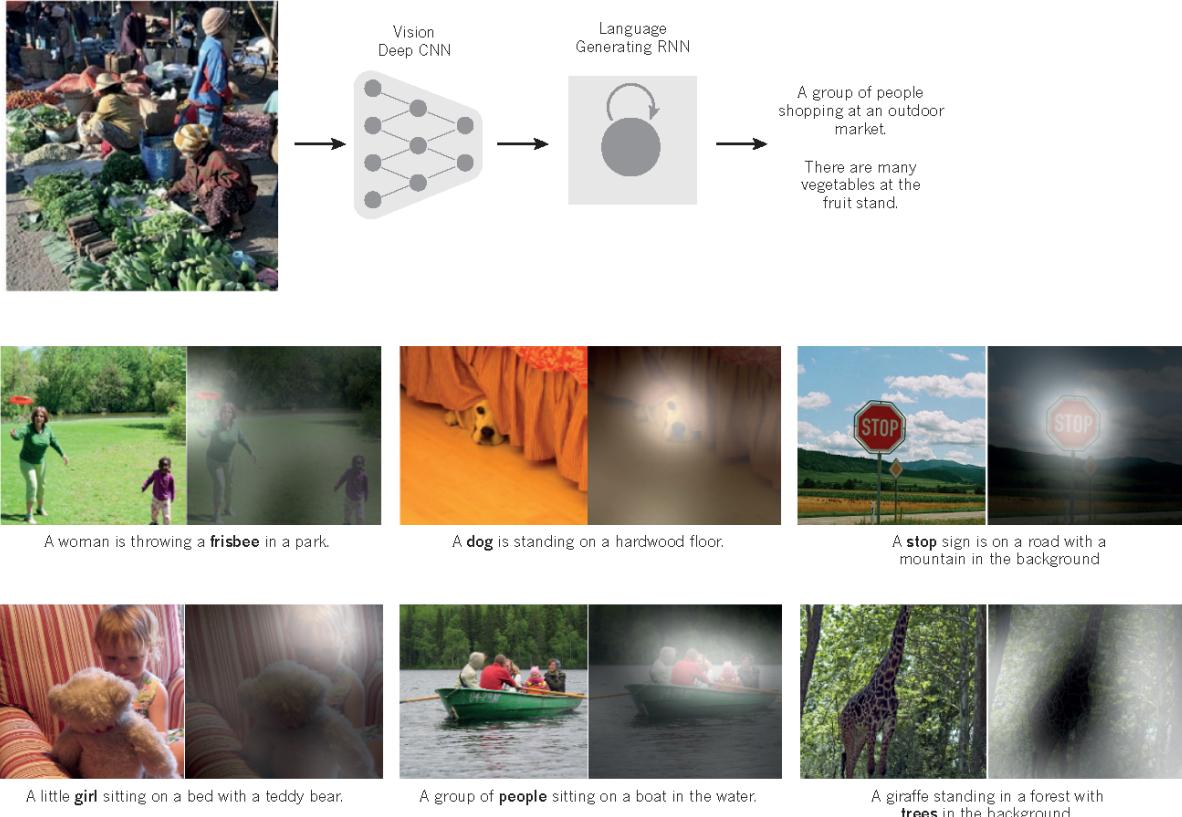


図3: 画像からテキストへ。ディープ畳み込みニューラルネットワーク (CNN) がテスト画像から抽出した表現を追加入力として、リカレントニューラルネットワーク (RNN) が生成したキャプション。参考文献からの許可を得て再掲(102)。RNN に、入力画像の異なる場所に注意を向ける機能を持たせた場合(中段と下段)。RNN が各単語を生成する際に(太字)、入力画像内の異なる場所に注目する能力を与えた場合(中段と下段、明るいバッヂほど注目された)、RNN はこれを利用して画像からキャプションへのより良い「翻訳」を実現することがわかった(86)。

最近の ConvNet アーキテクチャは、10~20 層の ReLU、数億の重み、数十億のユニット間接続を備えている。このような大規模なネットワークの学習は、わずか 2 年前には数週間かかっていたが、ハードウェア、ソフトウェア、アルゴリズムの並列化の進歩により、学習時間は数時間に短縮された。

ConvNet ベースの視覚システムの性能は、Google, Facebook, Microsoft, IBM, Yahoo!, Twitter, Adobeなどのほとんどの主要テクノロジー企業や、急速に増加しているスタートアップ企業が研究開発プロジェクトを開始し、ConvNet ベースの画像理解製品やサービスを展開している。

ConvNet は、チップやフィールド・プログラマブル・ゲート・アレイへの効率的なハードウェア実装に容易に従うことができる(66, 67)。NVIDIA, Mobileye, Intel, Qualcomm, Samsungなどの多くの企業が、スマートフォン、カメラ、ロボット、自動運転車などの実時間視覚アプリケーションを実現するために、ConvNet チップを開発している。

4. 文産業現と言語処理 Distributed representations and language processing

深層学習理論によると、深層ネットワークは、分散表現を使用しない古典的な学習アルゴリズムと比較して、2つの異なる指数的な利点を持っている(21)。これらの利点はいずれも構造力から生じるもので、基礎となるデータ生成分布が適切な構成構造を持つことに依存する(40)。まず、分散表現を学習することで、学習した特徴量の値の、学習時に見られた組み合わせ以外の新たな組み合わせに一般化することができる(例えば、 n 個の二値特徴量で 2^n 通りの組み合わせが可能)(68, 69)。第二に、ディープネットワークで表現の層を構成することで、別の指標的な利点(70)が得られる可能性がある(深さに対して指標的)。

多層ニューラルネットワークの隠れ層は、ネットワークの入力を、ターゲットとなる出力を予測しやすいように表現することを学習する。

このことは、多層ニューラルネットワークを訓練して、以前の単語のローカルな文脈から、連続した次の単語を予測することでよくわかる(71)。文脈中の各単語は、1対 N のベクトルとしてネットワークに提示される。つまり、1つの成分が1の値を持ち、残りは0である。第1層では、各単語が異なるパターンの活性化、すなわち単語ベクトルを作り出す(図4)。言語モデルでは、ネットワークの他の層は、入力された単語ベクトルを、予測された次の単語のための出力単語ベクトルに変換するように学習される。このネットワークは、記号の分散表現を学習する際に初めて実証されたように、それぞれが単語の個別の特徴と解釈できる多くの活性成分を含む単語ベクトルを学習する(27)。これらの意味的特徴は、入力に明示的に存在するものではなかった。これらは、入力シンボルと出力シンボルの間に構造化された関係を複数の「マイクロルール」に因数分解するのに適した方法として、学習手続きによって発見された。単語ベクトルの学習は、単語の配列が実際のテキストの大規模なコバースから来ており、個々のマイクロルールが信頼できない場合にも非常に有効であることが判明した(71)。例えば、ニュース記事中の次の単語を予測するように学習した場合、火曜日と水曜日の学習した単語ベクトルは、スウェーデンとノルウェーの単語ベクトルと同様に酷似する。このような表現は、その要素(特徴)が相互に排他的ではなく、多くの構成が観測されたデータのバリエーションに対応することから、分散表現と呼ばれる。これらの単語ベクトルは、専門家が事前に決めたものではなく、ニューラルネットワークが自動的に発見した学習済みの特徴で構成されている。テキストから学習された単語のベクトル表現は、現在、自然言語のアプリケーションで非常に広く使用されている(14, 17, 72-76)。

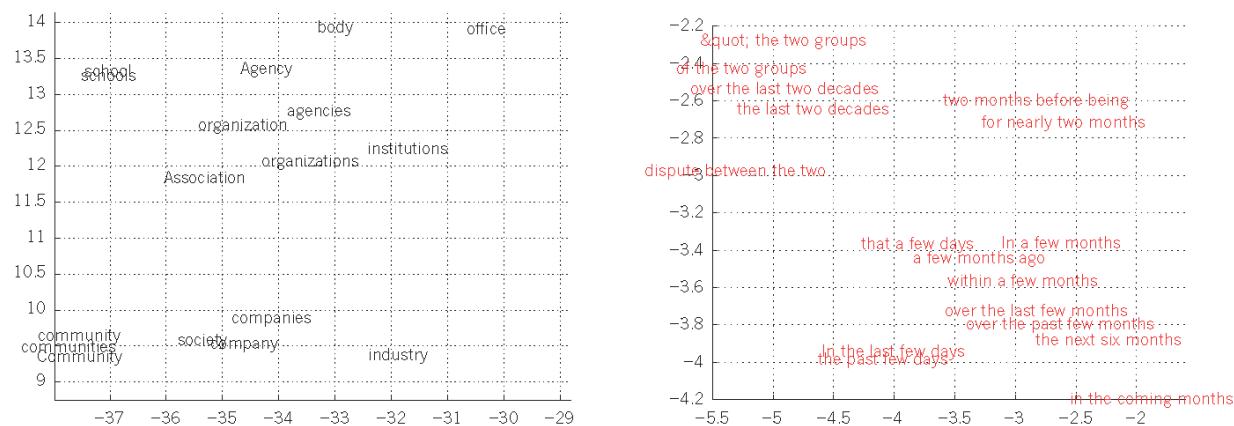


図4 | 学習した単語ベクトルの可視化。左、言語をモデル化するために学習した単語表現を、t-SNE アルゴリズム(1)を用いて非線形に2次元に投影して可視化した図(03)。右、英語からフランス語への符号化器・復号化器リカレントニューラルネットワークによって学習されたフレーズを2次元で表現したもの(75)。意味的に類似した単語や単語の並びが、近くの表現にマッピングされていることがわかる。単語の分散表現は、バックプロパゲーションを用いて、各単語の表現と、一連の単語の次の単語(言語モデルリングの場合)や翻訳された単語系列全体(機械翻訳の場合)などの目標量を予測する閑数を共同で学習することで得られる(18, 75)。

表現の問題は、論理を重視した認知パラダイムと、ニューラルネットワークを重視した認知パラダイムとの間の議論の中心となっている。論理に基づくパラダイムでは、シンボルの実体は、他のシンボルの実体と同一か非同一かのどちらかであるという特性しかないものである。記号を使って推論するためには、慎重に選ばれた推論規則の中で、記号を変数に結びつけなければならない。対照的に、ニューラルネットワークは、大きな活性値ベクトル、大きな重み行列、スカラー非線形性を用いて、楽な常識的推論を支える高速な「直感的」推論を行う。

ニューラル言語モデル(71)が登場する以前、言語の統計的モデルリングの標準的なアプローチは、分散表現を利用していないかった。それは、N 個までの長さの短い記号列(N-gram と呼ばれる)の出現頻度をカウントすることに基づいていた。N-gram の数は V^N (V は語彙数) のオーダーであり、一握りの単語以上の文脈を考慮するには、非常に大きな学習コバースが必要になる。一方、ニューラル言語モデルは、各単語を実数値の特徴量のベクトルと関連付けることで、意味的に関連する単語がそのベクトル空間の中で互いに近くなるため、一般化することができる(図4)。

6. リカレントニューラルネットワーク

バックプロパゲーションが導入された当初は、リカレントニューラルネットワーク(RNN)の学習に使用するのが主流であった。音声や言語など、連続した入力を伴う課題には、RNN を使う方がよい場合が多い(図5)。RNN は、入力系列を1要素ずつ処理し、隠れユニットに、系列の過去の全要素の履歴に関する情報を暗黙のうちに含む「状態ベクトル」を保持する。異なる離散的な時間ステップにおける隠れユニットの出力を、深い多層ネットワークの異なるニューロンの出力であるかのように考えると(図5の右)、RNN の学習にバックプロパゲーションを適用する方法が明らかになる。

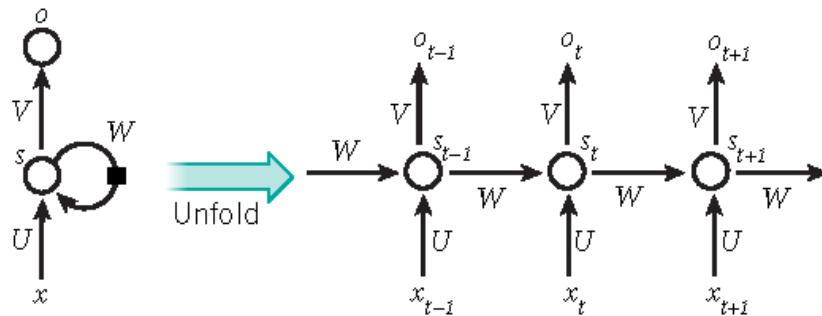


図5 リカレントニューラルネットワークとその前進計算に関わる計算の時間的展開を示したもの

人工ニューロン(例えば、ノード s の下にグループ化された隠れユニットで、時刻 t に値 s_t を持つ)は、前時間ステップで他のニューロンからの入力を得ている(これは、左の1時間ステップの遅延を表す黒い四角で表現されている)。このようにして、リカレントニューラルネットワークは、要素 x_t の入力系列を要素 o_t の出力系列にマッピングすることができ、各 o_t は($t' \leq t$ の場合)すべての前の $x_{t'}$ に依存する。各時間ステップで同じパラメータ(行列 U, V, W)が使用される。ネットワークが一連の出力(例えば単語)を生成し、それぞれが次の時間ステップの入力として使用されるようなバリエーションを含め、他の多くのアーキテクチャが可能である。バックプロパゲーションアルゴリズム(図1)は、右図の展開されたネットワークの計算グラフに直接適用することができ、すべての状態 s_t とすべてのパラメータに関する全誤差(例えば、正しい出力系列スコアを生成する対数確率)の微分を計算することができる。

RNN は非常に強力な動的システムだが、逆伝播された勾配が時間ステップごとに大きくなったり小さくなったりするため、多くの時間ステップで爆発したり消滅したりすることが多く、RNN の学習には問題があることがわかっている(77, 78)。

RNN のアーキテクチャ(79,80)や学習方法(81,82)の進歩により、RNN は、文章中の次の文字(83)や連続する単語(75)を予測するのに非常に適していることがわかっているが、より複雑な課題にも使用できるようになっている。例えば、英語の文章を1単語ずつ読んだ後、英語の「符号化器」ネットワークを訓練して、隠れユニットの最終的な状態ベクトルが、その文章で表現されている思考をうまく表現できるようにすることができる。この思考ベクトルは、共同で学習したフランス語の「復号化器」ネットワークの初期隠れ状態として(あるいは追加入力として)使用することができる。この分布から特定の最初の単語が選択され、復号化器ネットワークの入力として提供されると、翻訳の2番目の単語の確率分布が选出され、終端文字が選択されるまで繰り返される(17,72,76)。この処理では、英語の文に依存した確率分布に従ってフランス語の単語の系列が生成される。このように、かなり素朴な方法で機械翻訳を行うことで、瞬く間に最先端の機械翻訳に対抗できるようになった。このことから、文章を理解するためには、推論ルールを使用して操作される内部の記号表現のようなものが必要なのかどうかについて、重大な疑問が生じる。これは、日常的な推論には多くの類似性が同時に存在し、それぞれが結論の妥当性に寄与するという見解とより相性が良い(84,85)。

フランス語の文章の意味を英語の文章に翻訳する代わりに、画像の意味を英語の文章に「翻訳」することを学習することができる(図3)。ここで符号化器は、深層 ConvNet で、最終隠れ層で画素を活性値ベクトルに変換する。復号化器には、機械翻訳やニューラル言語モデリングに用いられるような RNN を使用している。最近、このようなシステムへの関心が高まっている(文献 86 に記載の例を参照)。

RNN は、時間的に展開すると(図5)、すべての層が同じ重みを共有する、非常に深いフィードフォワードネットワークと見なすことができる。RNN の主な目的は長期的な依存関係を学習することだが、理論的にも経験的にも、情報を非常に長く保存することを学習するのは難しいことがわかっている(78)。

この問題を解決するために、ネットワークに明示的な記憶を持たせることができると考えられる。この種の最初の提案は、特別な隠れユニットを使用する長短期記憶(LSTM)ネットワークで、その自然な動作は入力を長期間記憶することである(79)。メモリセルと呼ばれる特別なユニットは、キューリングレーターやゲートリーニューロンのような働きをする。しかし、この自己接続は、メモリの内容を消去するタイミングを決定することを学習する別のユニットによって乗算的にゲートされる。

その後、LSTM ネットワークは、特に時間ステップごとに複数の層を持つ場合、従来の RNN よりも効果的であることが証明され(87)、音響から転写の文字の並びまで、音声認識システム全体の構築が可能になった。また、機械翻訳で優れた性能を発揮する符号化器と復号化器のネットワークには、現在、LSTM ネットワークやそれに関連する形態のゲートユニットが使用されている(17, 72, 76)。

この1年間、複数の著者が RNN をメモリモジュールで拡張するさまざまな提案を行ってきた。提案には、RNN が読み書きできる「テープのような」メモリでネットワークを拡張する「Neural Turing Machine」(88)や、通常のネットワークを一種の連想メモリで拡張する「メモリネットワーク」(89)などがある。メモリネットワークは、標準的な質問応答のベンチマークで優れた性能を発揮している。記憶は、ネットワークが後に質問に答えるように要求されるストーリーを記憶するために使用される。

単なる記憶にとどまらず、通常は推論や記号の操作が必要な作業にも、ニューラルチューリングマシンやメモリーネットワークが使われている。ニューラルチューリングマシンは、「アルゴリズム」を教えることができる。例えば、各シンボルがリスト内の優先順位を示す実数値を伴うソートされていないシーケンスを入力として、ソートされたシンボルのリストを出力することを学習することができる(88)。メモリネットワークは、テキストアドベンチャーゲームのような設定で世界の状態を把握し、物語を読んだ後、複雑な推論を必要とする質問に答えられるように学習できる(90)。あるテスト例では、ネットワークは「指輪物語」の15章を見せられ、「フロドは今どこにいる?」などの質問に正しく答える(89)。

ディープラーニングの未来

教師なし学習(91-98)は、深層学習への関心を復活させる触媒効果があったが、その後、純粋な教師付き学習の成功の影に隠れてしまった。今回のレビューでは注目していないが、長期的には教師なし学習の重要性が増してくると予想している。人間や動物の学習は、ほとんどが教師なしで行われる。我々は、すべての物体の名前を教えてもらうのではなく、世界を観察することで世界の構造を発見する。

人間の視覚は、小さくて高解像度の焦点と、大きくて低解像度の周囲を使って、知的で課題に特化した方法で視神経配列を順次サンプリングする能動的な処理である。今後の視覚の進歩は、エンドツーエンドで学習され、強化学習を用いてどこを見るべきかを決定する ConvNets と RNN を組み合わせたシステムによってもたらされるものと期待している。深層学習と強化学習を組み合わせたシステムはまだ初期段階にあるが、すでに分類課題では受動的視覚システム(99)を凌駕し、さまざまなかビデオゲームのプレイを学習する際にも素晴らしい結果を出している(100)。

自然言語理解もまた、深層学習が今後数年間で大きな影響を与えることが期待される分野である。RNN を使って文章や文書全体を理解するシステムは、一度に一つの部分だけを選択的に処理する戦略を身につけることで、より優れたものになると期待している(76, 86)。

最終的に、人工知能の大きな進歩は、表現学習と複雑な推論を組み合わせたシステムによってもたらされるだろう。音声認識や手書き認識には、深層学習と単純な推論が古くから用いられてきたが、記号表現のルールベースの操作を大きなベクトルに対する演算に置き換えるには、新しいパラダイムが必要である(101)。