Emperor Journal of Applied Scientific Research

ISSN: 2581-964X **© Mayas Publication** **www.mayas.info**

Volume –VI **Issue-I** **March-2024**

Deep learning

Kapilkrishnan A B
Assistant Professor,
Department of Computer Science
E-Mail: kapilkrishnanab@gmail.com

Abstract

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the back propagation 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 shown light on sequential data such as text and speech.

I. INTRODUCTION

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 Smartphone. 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, analyzing particle accelerator data, reconstructing brain circuit, 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.

`

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 labeled 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 labeled 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 weight the gradient vector. The objective function, averaged over all the training examples, can be seen as a kind of hilly landscape in the high-dimensional space of weight values. The negative gradient vector indicates the direction of steepest descent in this landscape, taking it closer to a minimum, where the output error is low on average.

In practice, most practitioners use a procedure called stochastic gradient descent (SGD). This consists of showing the input vector for a few examples, computing the outputs and the errors, computing the average gradient for those examples, and adjusting the weights accordingly. The process is repeated for many small sets of examples from the training set until the average of the objective function stops decreasing. It is called stochastic because each small set of examples gives a noisy estimate of the average gradient over all examples. This simple procedure usually finds a good set of weights surprisingly quickly when compared with far more elaborate optimization techniques. 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.

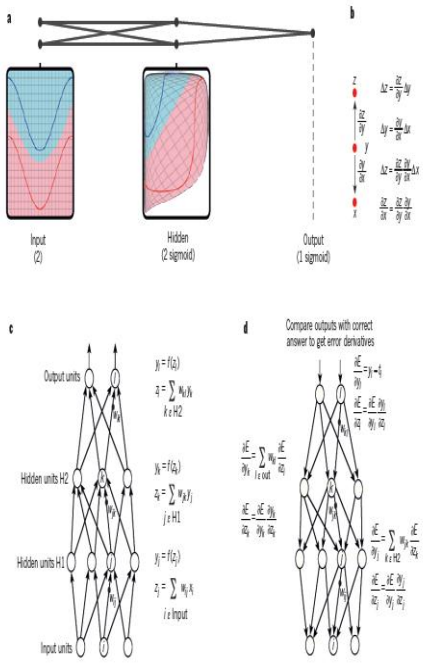

Figure 1 | Multilayer neural networks and backpropagation. a, A multi-layer neural network (shown by the connected dots) can distort the input space to make the classes of data (examples of which are on the red and blue lines) linearly separable. Note how a regular grid (shown on the left) in input space is also transformed (shown in the middle panel) by hidden units. This is an illustrative example with only two input units, two hidden units and one output unit, but the networks used for object

recognition or natural language processing contain tens or hundreds of thousands of units. Reproduced with permission from C
Olah **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 ∂y/∂x and ∂z/∂x. 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 throughich 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 $f(.)$ 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 hyberbolic 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 $f(z)$.

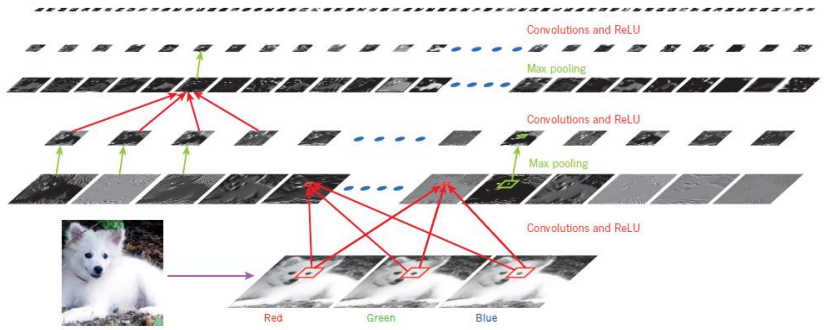

Figure 2 | Inside a convolutional network The outputs (not the filters) of each layer (horizontally) of a typical convolutional network architecture applied to the image of a Samoyed dog (bottom left; and RGB (red, green, blue) inputs, bottom right). Each rectangular image is a feature map

Back propagation 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 back propagation procedure. The idea that this could be done, and that it worked, was discovered independently by several different groups during the 1970s and 1980s24–27.

The back propagation procedure to compute the gradient of an objective function with respect to the weights of a multilayer stack of modules is nothing 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 more than a practical application n 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 back propagation 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 feed forward neural network architectures (Fig. 1), which learn to map a fixed-size input (for example, an image) to a fixed-size output. 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.

In the late 1990s, neural nets and back propagation 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 combinatorial large number of saddle points where the gradient is zero, and the surface curves up in most dimensions and curves down in the remainder

Convolutional neural networks

ConvNets are designed to process data that come in the form of multiple arrays, for example a color image composed of three 2D arrays containing pixel intensities in the three color channels. Many data modalities are in the form of multiple arrays: 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). Neighboring 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 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. Back propagating 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, 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 explain half of the variance of random sets of 160 neurons in the monkeys infer temporal cortex. 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 back propagation.

A primitive 1D ConvNet called a time-delay neural net was used for the recognition of phonemes and simple words. There have been numerous applications of convolutional networks going back to the early 1990s, starting with time-delay neural networks for speech recognition and document reading.

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.

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 labeled data was relatively abundant, such as traffic sign recognition53, the segmentation of biological images54 particularly for connectomics, and the detection of faces, text, pedestrians and human bodies in natural images.A major recent practical success of ConvNets is face recognition importantly, images can be labeled at the pixel level, which will have applications in technology, including autonomous mobile robots and self-driving cars.Companies such as Mobil eye 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 recognition.

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 representation. 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) Second, composing layers of representation in a deep net brings the potential for another exponential advantage(exponential in the depth).

The hidden layers of a multilayer neural network learn to represent the network's inputs in a way that makes it easy to predict the target outputs. This is nicely demonstrated by training a multilayer neural network to predict the next word in a sequence from a local context of earlier words. 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.

Recurrent Neural Networks

When back propagation 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. 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 it becomes clear how we can apply back propagation to train RNNs.

RNNs are very powerful dynamic systems, but training them has proved to be problematic because the back propagated gradients either grow or shrink at each time step, so over many time steps they typically explode or vanish thanks to advances in their architecture and ways of training them RNNs have been found to be very good at predicting the next character in the text or the next word in a sequence 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. 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.

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.

III. REFERENCE

1. 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)

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

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

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

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

6. Hinton, G. *et al.* Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine* 29, 82–97 (2012)

7. 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)

8. 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)

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