# 天氣與人工智慧 II

## *Ch.1 What is deep learning?*

周哲維
david10188@gmail.com

# *AI? Machine Learning? Deep Learning?*

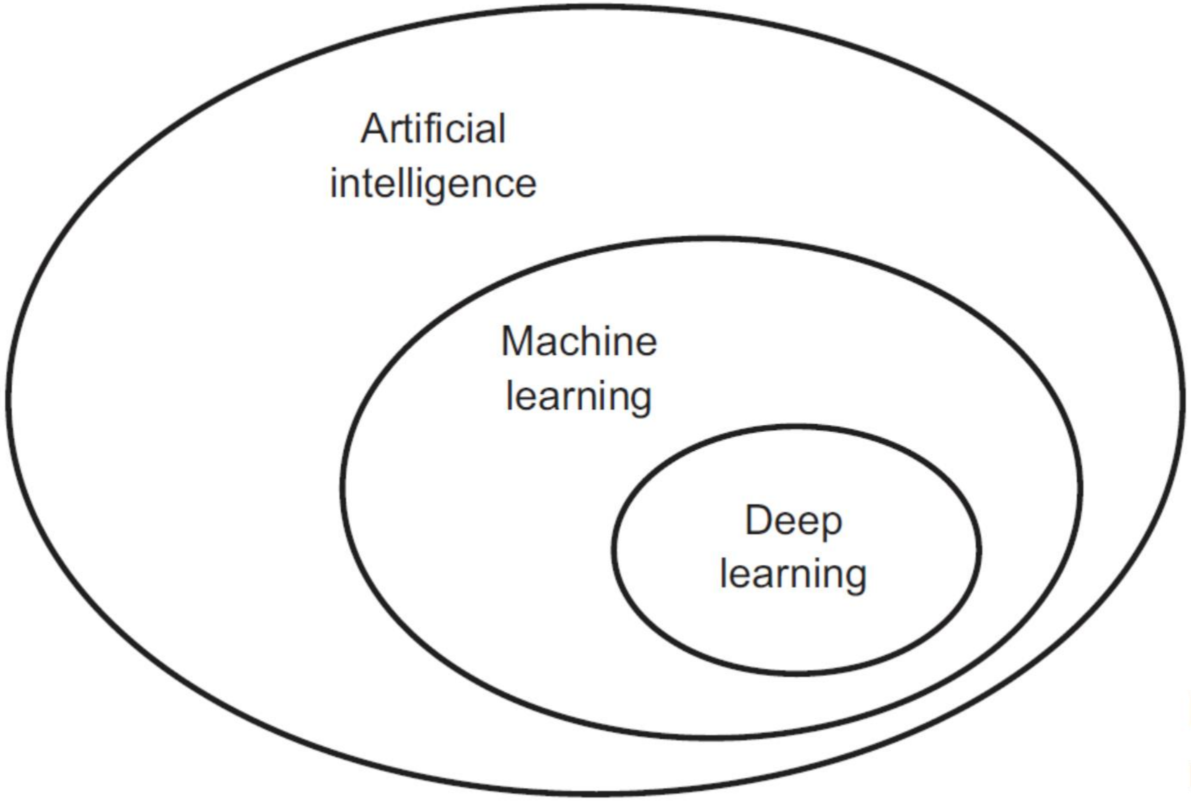


**Figure 1.1 Artificial intelligence, machine learning, and deep learning**

# ***Artificial Intelligence***

- Artificial intelligence was born in the 1950s, when a handful of pioneers from the nascent field of computer science started asking whether computers could be made to "think"—a question whose ramifications we're still exploring today.
- AI is a general field that encompasses machine learning and deep learning, but that also includes many more approaches that don't involve any learning.
- Early chess programs, for instance, only involved hardcoded rules crafted by programmers, and didn't qualify as machine learning.
- For a fairly long time, many experts believed that human-level artificial intelligence could be achieved by having programmers handcraft a sufficiently large set of explicit rules for manipulating knowledge. This approach is known as *symbolic AI*.

# ***Artificial Intelligence***

- Although symbolic AI proved suitable to solve well-defined, logical problems, such as playing chess, it turned out to be intractable to figure out explicit rules for solving more complex, fuzzy problems, such as image classification, speech recognition, and language translation.
- A new approach arose to take symbolic AI’s place: *machine learning*.

# *Machine Learning*

- Machine learning arises from this question:
  - Could a computer go beyond “what we know how to order it to perform” and learn on its own how to perform a specified task?
  - Could a computer surprise us?
  - Rather than programmers crafting data-processing rules by hand, could a computer automatically learn these rules by looking at data?
- This question opens the door to a new programming paradigm. In classical programming, the paradigm of symbolic AI, humans input rules (a program) and data to be processed according to these rules, and out come answers.

# *Machine Learning*

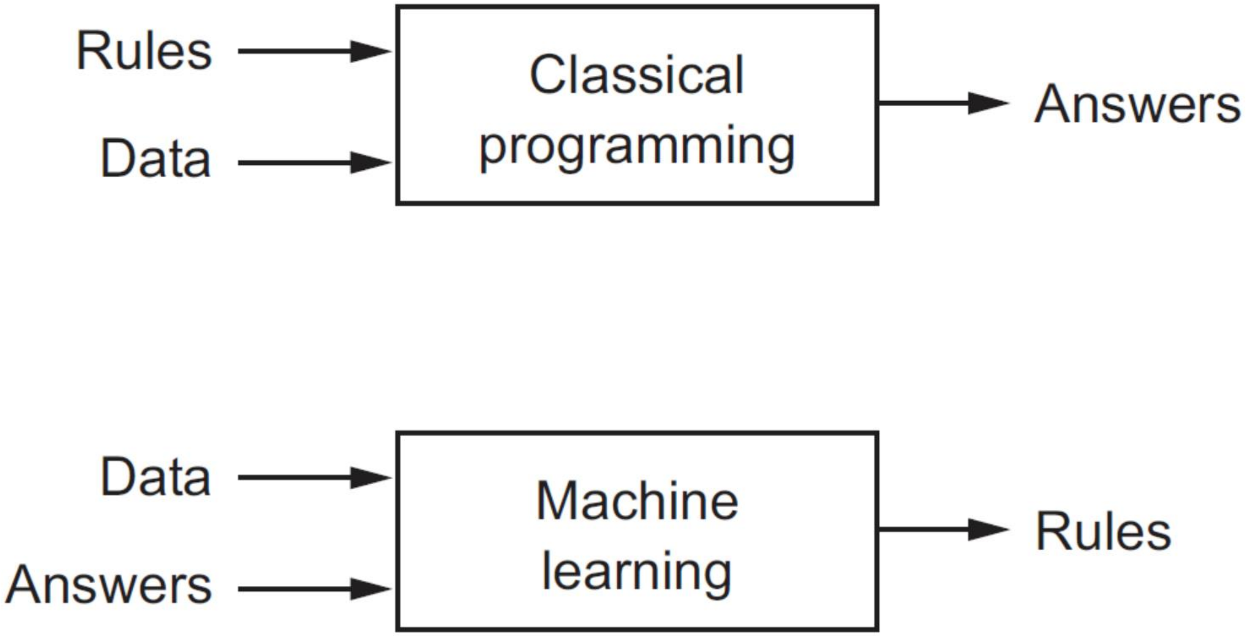


**Figure 1.2 Machine learning: a new programming paradigm**

# ***Machine Learning***

- A machine-learning system is *trained* rather than explicitly programmed.
- It’s presented with many examples relevant to a task, and it finds statistical structure in these examples that eventually allows the system to come up with rules for automating the task.
- For instance, if you wished to automate the task of tagging your vacation pictures, you could present a machine-learning system with many examples of pictures already tagged by humans, and the system would learn statistical rules for associating specific pictures to specific tags.

# ***Machine Learning***

- Machine learning is tightly related to mathematical statistics, but it differs from statistics in several important ways.
- Unlike statistics, machine learning tends to deal with large, complex datasets (such as a dataset of millions of images, each consisting of tens of thousands of pixels) for which classical statistical analysis such as Bayesian analysis would be impractical.

# *Learning representations from data*

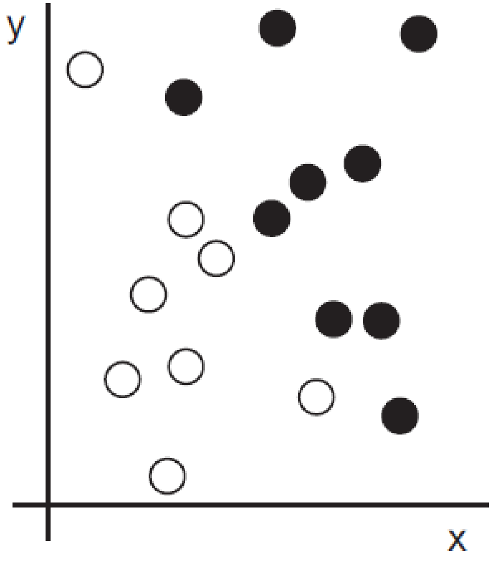
- To define *deep learning* and understand the difference between deep learning and other machine-learning approaches, first we need some idea of what machine learning algorithms *do*.
- To do machine learning, we need three things:
  - *Input data points*
  - *Examples of the expected output*
  - *A way to measure whether the algorithm is doing a good job*
- A machine-learning model transforms its input data into meaningful outputs, a process that is "learned" from exposure to known examples of inputs and outputs.

# ***Learning representations from data***

- The central problem in machine learning and deep learning is to *meaningfully transform data*: in other words, to learn useful *representations* of the input data at hand.
- Before we go any further: what's a representation? At its core, it's a different way to look at data—to *represent* or *encode* data.
- For instance, a color image can be encoded in the RGB format (red-green-blue) or in the HSV format (hue-saturation-value): these are two different representations of the same data.

# *Learning representations from data*

- Machine-learning models are all about finding appropriate representations for their input data—transformations of the data that make it more amenable to the task at hand, such as a classification task.



- The inputs are the coordinates of our points.
- The expected outputs are the colors of our points.
- A way to measure whether our algorithm is doing a good job could be, for instance, the percentage of points that are being correctly classified.

**Figure 1.3**
**Some sample data**

# *Learning representations from data*

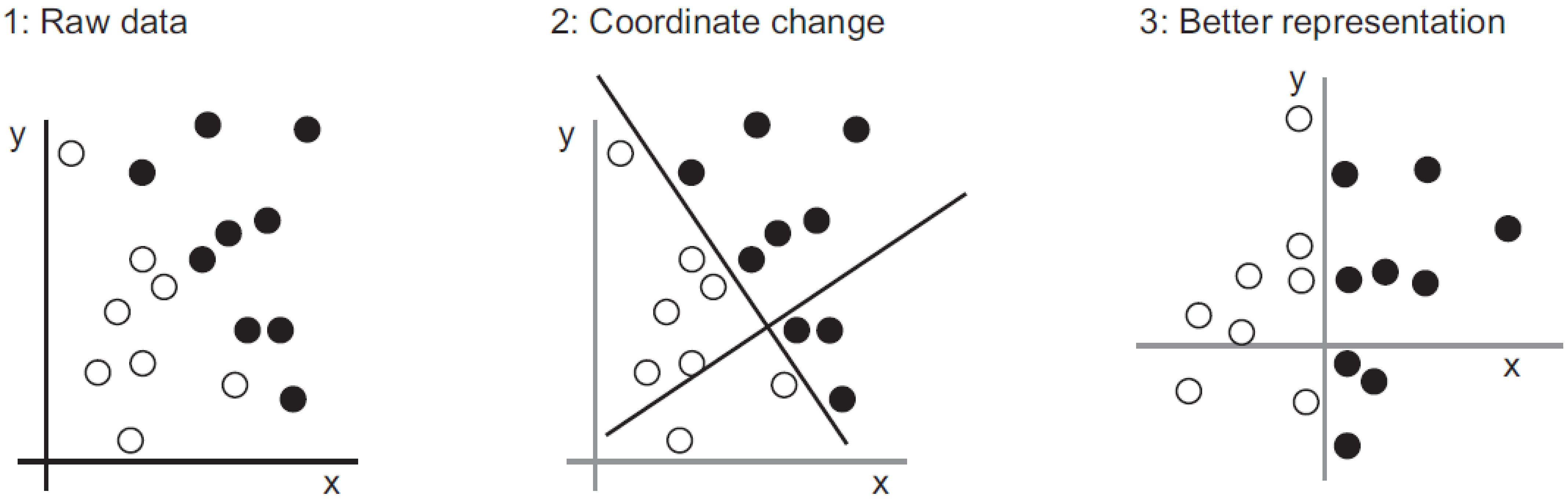


**Figure 1.4 Coordinate change**

“Black points are such that x > 0,” or “White points are such that x < 0.”

# *The “deep” in deep learning*

- Deep learning is a specific subfield of machine learning: a new take on learning representations from data that puts an emphasis on learning successive *layers* of increasingly meaningful representations.
- The *deep* in *deep learning* stands for this idea of successive layers of representations.
- Modern deep learning often involves tens or even hundreds of successive layers of representations— and they’re all learned automatically from exposure to training data.
- In deep learning, these layered representations are learned via models called *neural networks*, structured in literal layers stacked on top of each other.

# *The "deep" in deep learning*

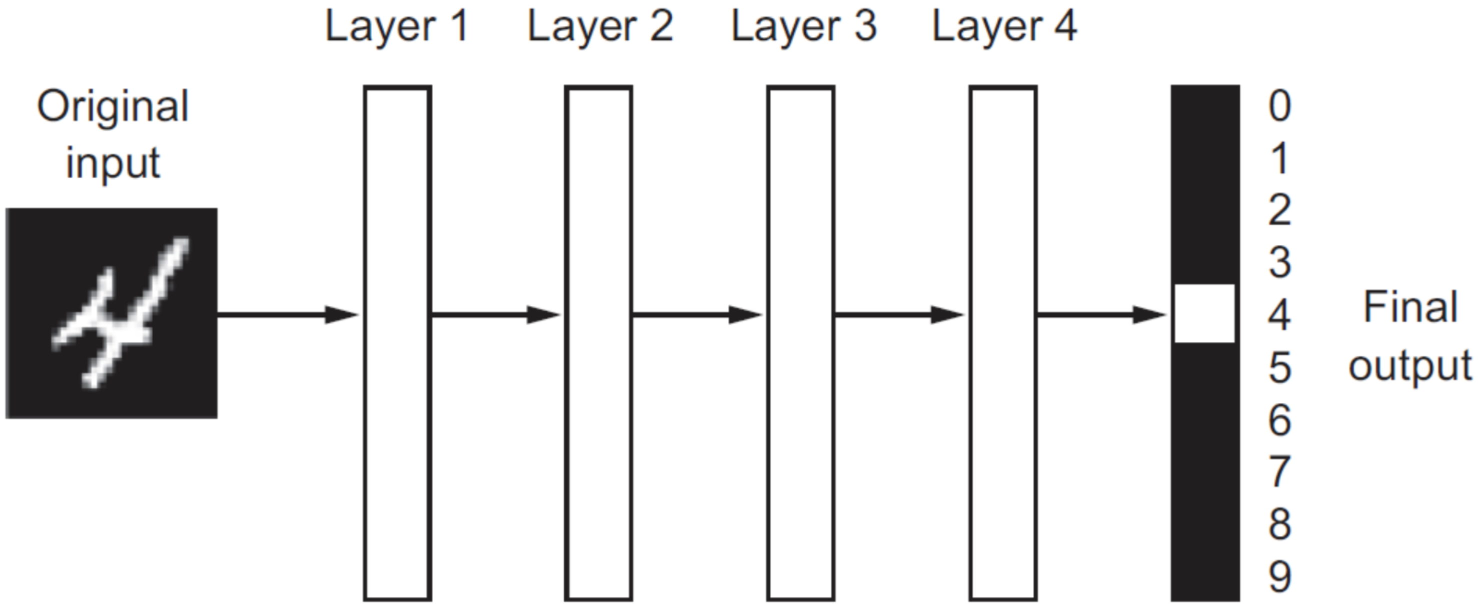


**Figure 1.5 A deep neural network for digit classification**

# ***The "deep" in deep learning***

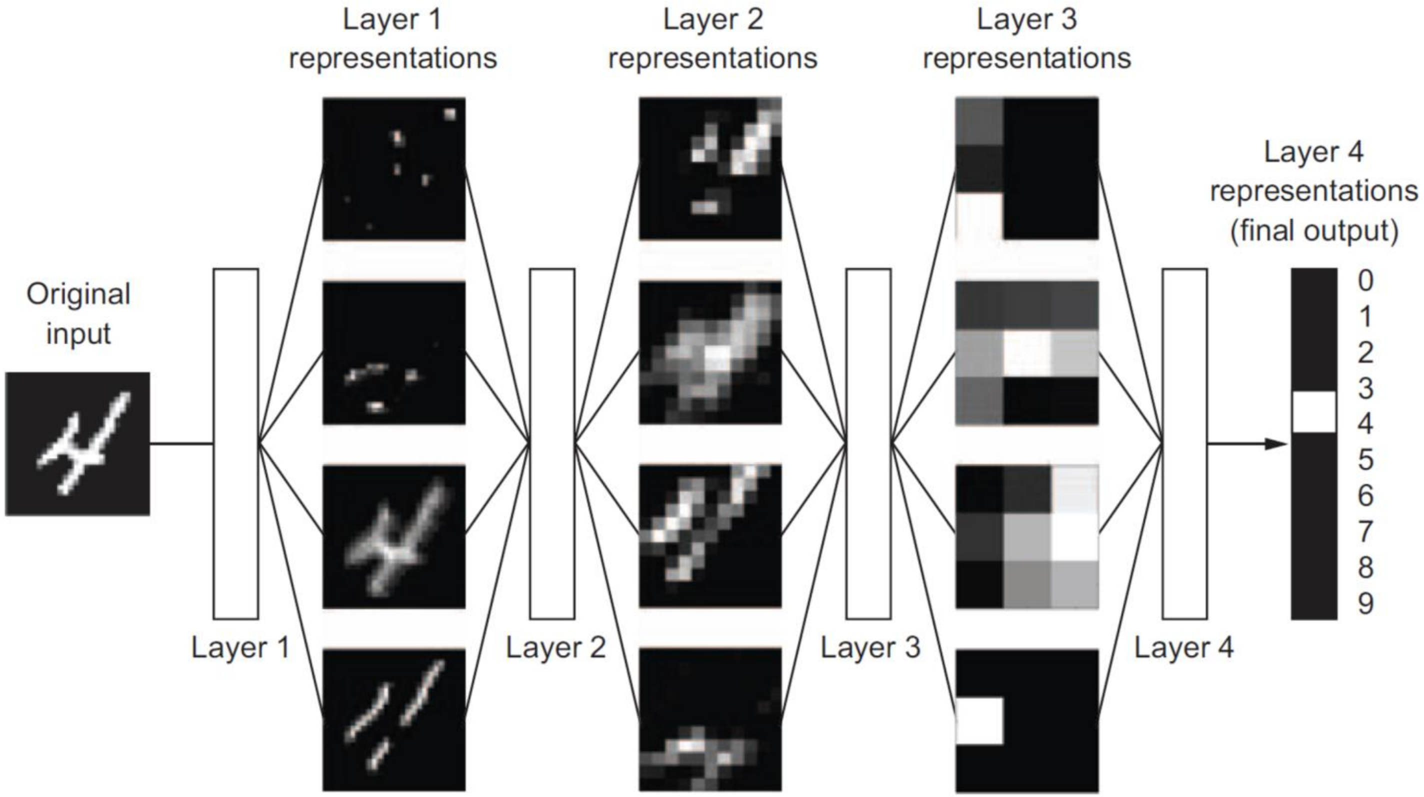


**Figure 1.6 Deep representations learned by a digit-classification model**

A multistage way to learn data representations. It's a simple idea—but, as it turns out, very simple mechanisms, sufficiently scaled, can end up looking like magic.

# ***Understanding how deep learning works, in three figures***

- machine learning is about mapping inputs (such as images) to targets (such as the label “cat”), which is done by observing many examples of input and targets.

- deep neural networks do this input-to-target mapping via a deep sequence of simple data transformations (layers) and that these data transformations are learned by exposure to examples.

# *Understanding how deep learning works, in three figures*

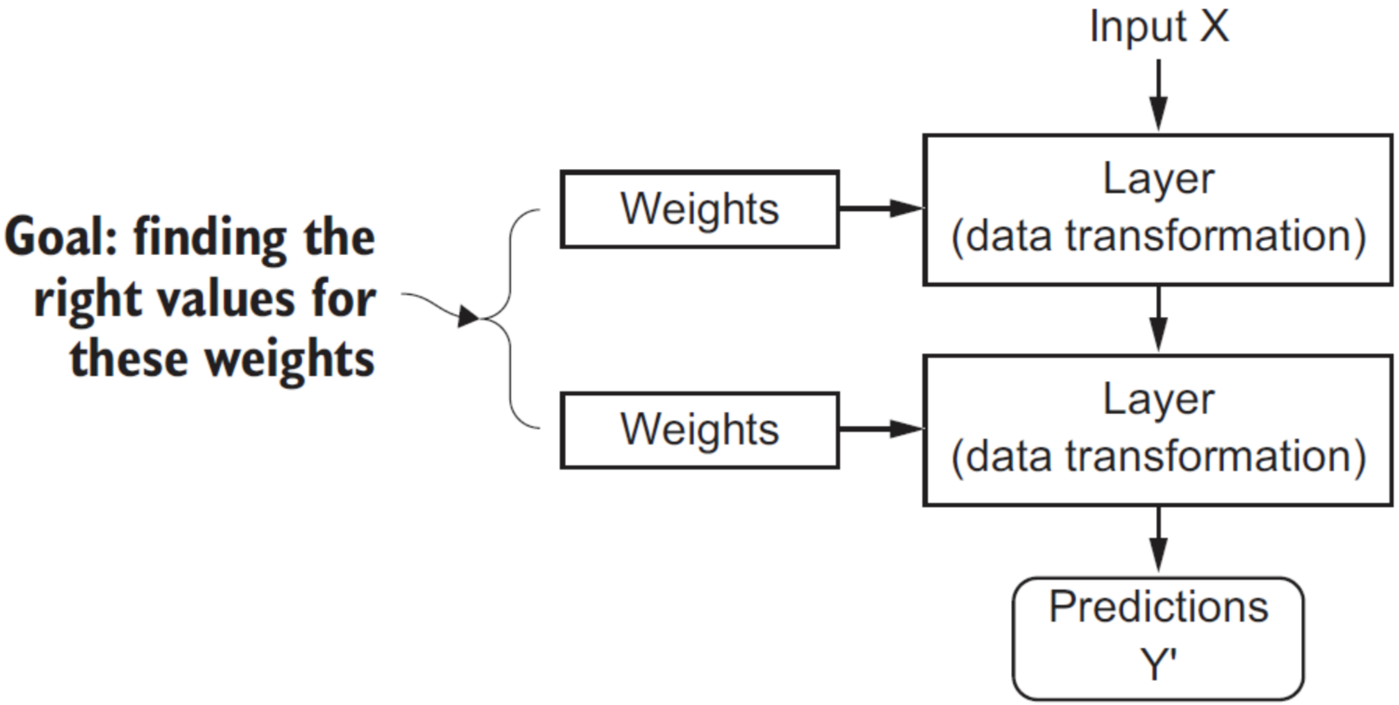


**Figure 1.7 A neural network is parameterized by its weights.**

*Learning* means finding a set of values for the weights of all layers in a network, such that the network will correctly map example inputs to their associated targets.

# *Understanding how deep learning works, in three figures*

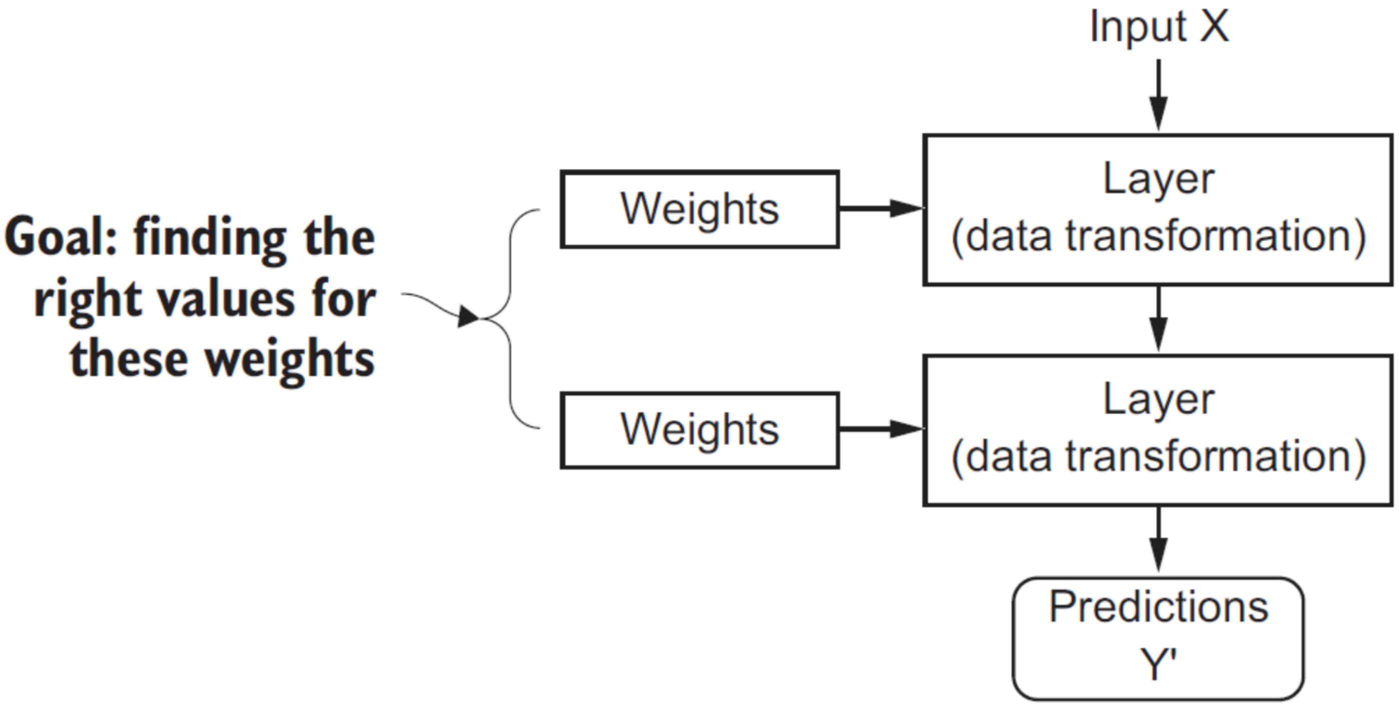


**Figure 1.7 A neural network is parameterized by its weights.**

But here’s the thing: a deep neural network can contain tens of millions of parameters. Finding the correct value for all of them may seem like a daunting task, especially given that modifying the value of one parameter will affect the behavior of all the others!

# ***Understanding how deep learning works, in three figures***

- To control the output of a neural network, you need to be able to measure how far this output is from what you expected.
- This is the job of the *loss function* of the network, also called the *objective function*.
- The loss function takes the predictions of the network and the true target and computes a distance score, capturing how well the network has done on this specific example

# *Understanding how deep learning works, in three figures*

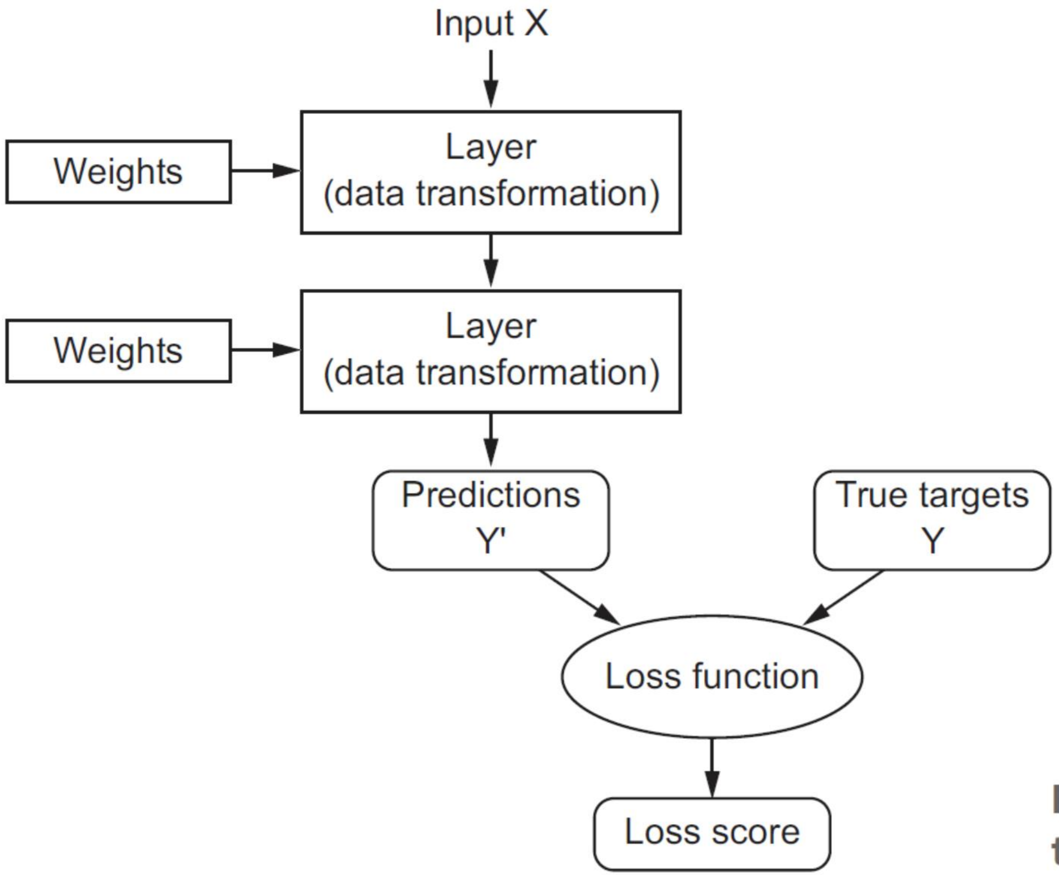


**Figure 1.8 A loss function measures the quality of the network's output.**

# *Understanding how deep learning works, in three figures*

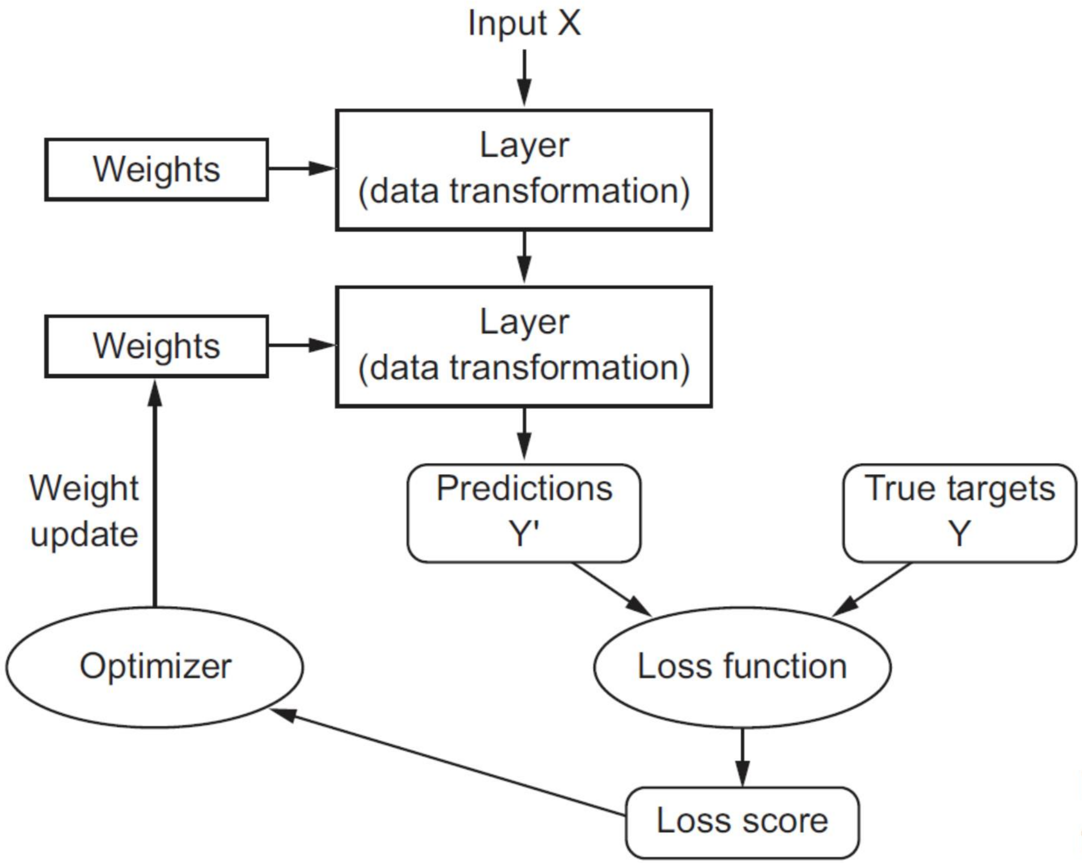


**Figure 1.9 The loss score is used as a feedback signal to adjust the weights.**

The fundamental trick in deep learning is to use this score as a feedback signal to adjust the value of the weights a little, in a direction that will lower the loss score for the current example

# *Understanding how deep learning works, in three figures*

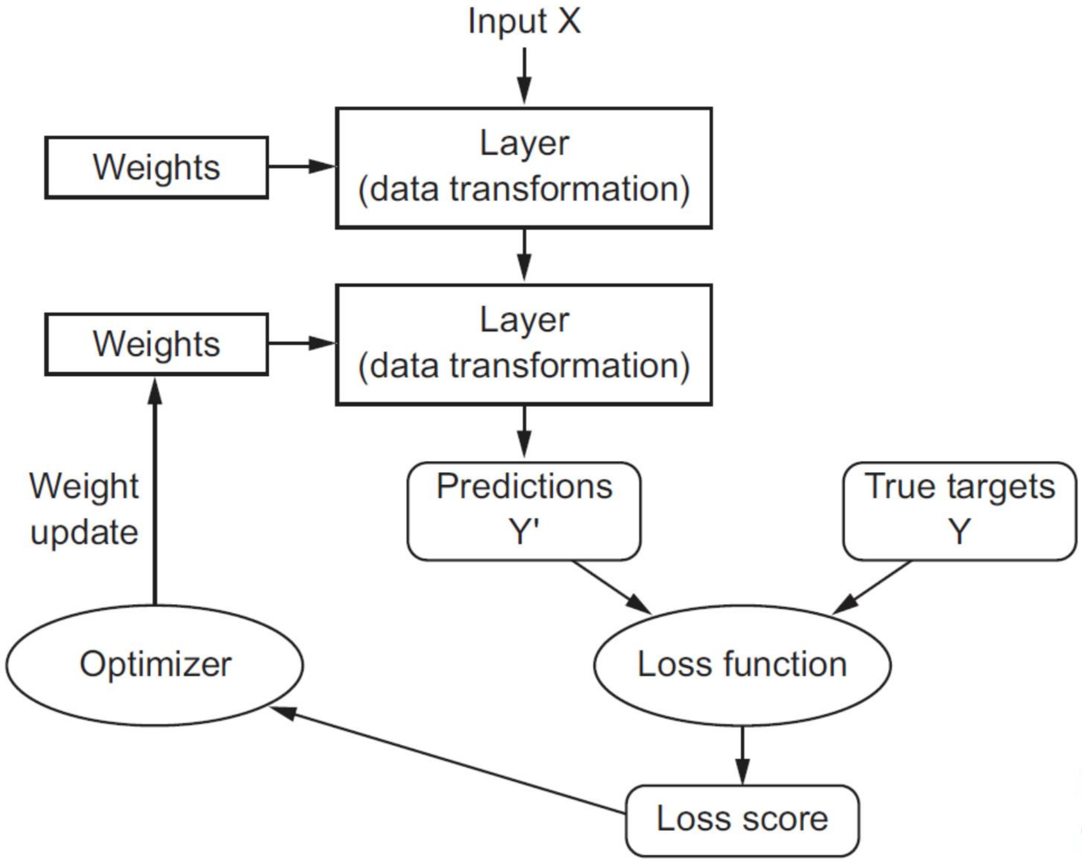


**Figure 1.9 The loss score is used as a feedback signal to adjust the weights.**

This adjustment is the job of the *optimizer*, which implements what's called the *Backpropagation* algorithm: the central algorithm in deep learning.

# ***Understanding how deep learning works, in three figures***

- Initially, the weights of the network are assigned random values, so the network merely implements a series of random transformations.
- Naturally, its output is far from what it should ideally be, and the loss score is accordingly very high.
- But with every example the network processes, the weights are adjusted a little in the correct direction, and the loss score decreases.
- This is the *training loop*, which, repeated a sufficient number of times, yields weight values that minimize the loss function.

# ***What deep learning has achieved so far***

- Although deep learning is a fairly old subfield of machine learning, it only rose to prominence in the early 2010s.

- In the few years since, it has achieved nothing short of a revolution in the field, with remarkable results on perceptual problems such as seeing and hearing—problems involving skills that seem natural and intuitive to humans but have long been elusive for machines.

# *What deep learning has achieved so far*

- Near-human-level image classification
- Near-human-level speech recognition
- Near-human-level handwriting transcription
- Improved machine translation
- Improved text-to-speech conversion
- Digital assistants such as Google Now and Amazon Alexa
- Near-human-level autonomous driving
- Improved ad targeting, as used by Google, Baidu, and Bing
- Improved search results on the web
- Ability to answer natural-language questions
- Superhuman Go playing

# *Before deep learning: a brief history of machine learning*

- Deep learning has reached a level of public attention and industry investment never before seen in the history of AI, but it isn’t the first successful form of machine learning. It’s safe to say that most of the machine-learning algorithms used in the industry today aren’t deep-learning algorithms.
- Deep learning isn’t always the right tool for the job—sometimes there isn’t enough data for deep learning to be applicable, and sometimes the problem is better solved by a different algorithm.

# ***Probabilistic modeling***

- *Probabilistic modeling* is the application of the principles of statistics to data analysis. It was one of the earliest forms of machine learning, and it’s still widely used to this day.
- One of the best-known algorithms in this category is the Naive Bayes algorithm.
- Naive Bayes is a type of machine-learning classifier based on applying Bayes’ theorem while assuming that the features in the input data are all independent.
- A closely related model is the *logistic regression.*

# *Early neural networks*

- The core ideas of neural networks were investigated in toy forms as early as the 1950s, the approach took decades to get started. For a long time, the missing piece was an efficient way to train large neural networks. This changed in the mid-1980s, when multiple people independently rediscovered the Backpropagation algorithm.

- The first successful practical application of neural nets came in 1989 from Bell Labs, when Yann LeCun combined the earlier ideas of convolutional neural networks and backpropagation, and applied them to the problem of classifying handwritten digits.

# *Kernel methods*

- *Kernel methods* are a group of classification algorithms, the best known of which is the *support vector machine* (SVM).
- The new approach to machine learning rose to fame and quickly sent neural nets back to oblivion.
- SVMs aim at solving classification problems by finding good *decision boundaries* between two sets of points belonging to two different categories.
- A decision boundary can be thought of as a line or surface separating your training data into two spaces corresponding to two categories.

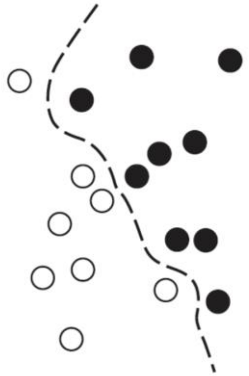


**Figure 1.10**
**A decision boundary**

# *Kernel methods*

- SVMs proceed to find these boundaries in two steps:
  - The data is mapped to a new high-dimensional representation where the decision boundary can be expressed as a hyperplane.
  - A good decision boundary (a separation hyperplane) is computed by trying to maximize the distance between the hyperplane and the closest data points from each class, a step called *maximizing the margin*.
- But SVMs proved hard to scale to large datasets and didn’t provide good results for perceptual problems such as image classification. Because an SVM is a shallow method, applying an SVM to perceptual problems requires first extracting useful representations manually, which is difficult and brittle.

# *Decision trees, random forests, and gradient boosting machines*

- *Decision trees* are flowchart-like structures that let you classify input data points or predict output values given inputs.

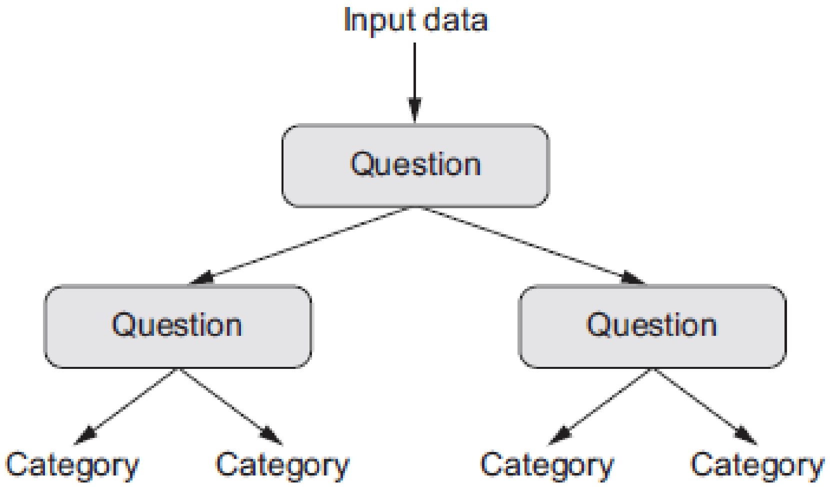


**Figure 1.11 A decision tree: the parameters that are learned are the questions about the data. A question could be, for instance, "Is coefficient 2 in the data greater than 3.5?"**

# ***Decision trees, random forests, and gradient boosting machines***

- *Random Forest* algorithm introduced a robust, practical take on decision-tree learning that involves building a large number of specialized decision trees and then ensembling their outputs.
- A gradient boosting machine, much like a random forest, is a machine-learning technique based on ensembling weak prediction models, generally decision trees. It uses *gradient boosting*, a way to improve any machine-learning model by iteratively training new models that specialize in addressing the weak points of the previous models.

# ***Back to neural networks***

- Around 2010, although neural networks were almost completely shunned by the scientific community at large, a number of people still working on neural networks started to make important breakthroughs.
- Since 2012, deep convolutional neural networks (*convnets*) have become the go-to algorithm for all computer vision tasks; more generally, they work on all perceptual tasks.

# ***What makes deep learning different***

- The primary reason deep learning took off so quickly is that it offered better performance on many problems.

- Deep learning also makes problem-solving much easier, because it completely automates what used to be the most crucial step in a machine-learning workflow: feature engineering.

# ***Why deep learning? Why now?***

- The two key ideas of deep learning for computer vision—convolutional neural networks and backpropagation—were already well understood in 1989.

- The Long Short-Term Memory (LSTM) algorithm, which is fundamental to deep learning for timeseries, was developed in 1997 and has barely changed since.

# ***Why deep learning? Why now?***

- So why did deep learning only take off after 2012? What changed in these two decades?

- In general, three technical forces are driving advances in machine learning:
    - Hardware
    - Datasets and benchmarks
    - Algorithmic advances

# *Why deep learning? Why now?*

***Hardware***

- Between 1990 and 2010, off-the-shelf CPUs became faster by a factor of approximately 5,000. As a result, nowadays it’s possible to run small deep-learning models on your laptop, whereas this would have been intractable 25 years ago.

- But typical deep-learning models used in computer vision or speech recognition require orders of magnitude more computational power than what your laptop can deliver.

# *Why deep learning? Why now?*

***Hardware***

- Throughout the 2000s, companies like NVIDIA and AMD have been investing billions of dollars in developing fast, massively parallel chips (graphical processing units [GPUs]) to power the graphics of increasingly photorealistic video games— cheap, single-purpose supercomputers designed to render complex 3D scenes on your screen in real time.

- This investment came to benefit the scientific community when, in 2007, NVIDIA launched CUDA, a programming interface for its line of GPUs. A small number of GPUs started replacing massive clusters of CPUs in various highly parallelizable applications, beginning with physics modeling.

# *Why deep learning? Why now?*

***Hardware***

- Deep neural networks, consisting mostly of many small matrix multiplications, are also highly parallelizable; and around 2011, some researchers began to write CUDA implementations of neural nets.

- The NVIDIA TITAN X, a gaming GPU that cost $1,000 at the end of 2015, can deliver a peak of 6.6 TFLOPS in single precision: 6.6 trillion float32 operations per second.

# ***Why deep learning? Why now?***

***Data***

- AI is sometimes heralded as the new industrial revolution. If deep learning is the steam engine of this revolution, then data is its coal.

# *Why deep learning? Why now?*

***Algorithms***

- The key issue was that of *gradient propagation* through deep stacks of layers. The feedback signal used to train neural networks would fade away as the number of layers increased.
- This changed around 2009–2010 with the advent of several simple but important algorithmic improvements that allowed for better gradient propagation:
  - Better *activation functions* for neural layers
  - Better *weight-initialization schemes*, starting with layer-wise pretraining, which was quickly abandoned
  - Better *optimization schemes*, such as RMSProp and Adam