DEEP LEARNING with Python

François Chollet

MEAP

MANNING

MEAP Edition
Manning Early Access Program

Deep Learning with Python

Version 6

For more information on this and other Manning titles go to
www.manning.com

Welcome

Thank you for purchasing the MEAP for *Deep Learning with Python*. If you are looking for a resource to learn about deep learning from scratch and to quickly become able to use this knowledge to solve real-world problems, you have found the right book. *Deep Learning with Python* is meant for engineers and students with a reasonable amount of Python experience, but no significant knowledge of machine learning and deep learning. It will take you all the way from basic theory to advanced practical applications. However, if you already have experience with deep learning, you should still be able to find value in the latter chapters of this book.

Deep learning is an immensely rich subfield of machine learning, with powerful applications ranging from machine perception to natural language processing, all the way up to creative AI. Yet, its core concepts are in fact very simple. Deep learning is often presented as shrouded in a certain mystique, with references to algorithms that "work like the brain", that "think" or "understand". Reality is however quite far from this science-fiction dream, and I will do my best in these pages to dispel these illusions. I believe that there are no difficult ideas in deep learning, and that's why I started this book, based on premise that all of the important concepts and applications in this field could be taught to anyone, with very few prerequisites.

This book is structured around a series of practical code examples, demonstrating on real-world problems every the notions that gets introduced. I strongly believe in the value of teaching using concrete examples, anchoring theoretical ideas into actual results and tangible code patterns. These examples all rely on Keras, the Python deep learning library. When I released the initial version of Keras almost two years ago, little did I know that it would quickly skyrocket to become one of the most widely used deep learning frameworks. A big part of that success is that Keras has always put ease of use and accessibility front and center. This same reason is what makes Keras a great library to get started with deep learning, and thus a great fit for this book. By the time you reach the end of this book, you will have become a Keras expert.

I hope that you will this book valuable —deep learning will definitely open up new intellectual perspectives for you, and in fact it even has the potential to transform your career, being the most in-demand scientific specialization these days. I am looking forward to your reviews and comments. Your feedback is essential in order to write the best possible book, that will benefit the greatest number of people.

— François Chollet

brief contents

PART 1: THE FUNDAMENTALS OF DEEP LEARNING

PART 2: DEEP LEARNING IN PRACTICE

1 What is Deep Learning?

1.1 Artificial intelligence, machine learning and deep learning

In the past few years, Artificial Intelligence (AI) has been a subject of intense media hype. Machine learning, deep learning, and AI come up in countless articles, often outside of technology-minded publications. We are being promised a future of intelligent chatbots, self-driving cars, and virtual assistants—a future sometimes painted in a grim light, and sometimes as an utopia, where human jobs would be scarce and most economic activity would be handled by robots or AI agents.

As a future or current practitioner of machine learning, it is important to be able to recognize the signal in the noise, to tell apart world-changing developments from what are merely over-hyped press releases. What is at stake is our future, and it is a future in which you have an active role to play: after reading this book, you will be part of those who develop the AIs. So let's tackle these questions—what has deep learning really achieved so far? How significant is it? Where are we headed next? Should you believe the hype?

First of all, we need to define clearly what we are talking about when we talk about AI. What is artificial intelligence, machine learning, and deep learning? How do they relate to each other?

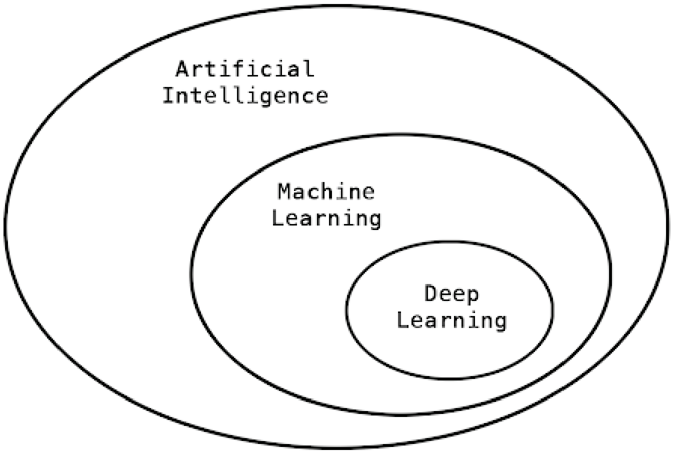

Figure 1.1 Artificial Intelligence, Machine Learning and Deep Learning

1.1.1 Artificial intelligence

Artificial intelligence was born in the 1950s, as a handful of pioneers from the nascent field of computer science started asking if computers could be made to "think"—a question whose ramifications we are still exploring today. A concise definition of the field would be: *the effort to automate intellectual tasks normally performed by humans*. As such, AI is a very general field which encompasses machine learning and deep learning, but also includes many more approaches that do not involve any learning. Early chess programs, for instance, only involved hard-coded rules crafted by programmers, and did not qualify as "machine learning". In fact, for a fairly long time many experts believed that human-level artificial intelligence could be achieved simply by having programmers handcraft a sufficiently large set of explicit rules for manipulating knowledge. This approach is known as "symbolic AI", and it was the dominant paradigm in AI from the 1950s to the late 1980s. It reached its peak popularity during the "expert systems" boom of the 1980s.

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, or language translation. A new approach to AI arose to take its place: machine learning.

1.1.2 Machine Learning

In Victorian England, Lady Ada Lovelace was a friend and collaborator of Charles Babbage, the inventor of the "Analytical Engine", the first known design of a general-purpose computer—a mechanical computer. Although visionary and far ahead of its time, the Analytical Engine wasn’t actually meant as a general-purpose computer when it was designed in the 1830s and 1840s, since the concept of general-purpose computation was yet to be invented. It was merely meant as a way to use mechanical operations to automate certain computations from the field of mathematical analysis—hence the name "analytical engine". In 1843, Ada Lovelace remarked on the invention:

"The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform… Its province is to assist us in making available what we are already acquainted with."

This remark was later quoted by AI pioneer Alan Turing as "Lady Lovelace’s objection" in his landmark 1950 paper "Computing Machinery and Intelligence", which introduced the "Turing test" as well as key concepts that would come to shape AI. Turing was quoting Ada Lovelace while pondering whether general-purpose computers could be capable of learning and originality, and he came to the conclusion that they could.

Machine learning arises from this very question: could a computer go beyond "what we know how to order it to perform", and actually "learn" on its own how to perform a specified task? Could a computer surprise us? Rather than crafting data-processing rules by hand, could it be possible to automatically learn these rules by looking at data?

This question opens up the door to a new programming paradigm. In classical programming, the paradigm of symbolic AI, humans would input rules (a program), data to be processed according to these rules, and out would come answers. With machine learning, humans would input data as well as the answers expected from the data, and out would come the rules. These rules could then be applied to new data to produce original answers.

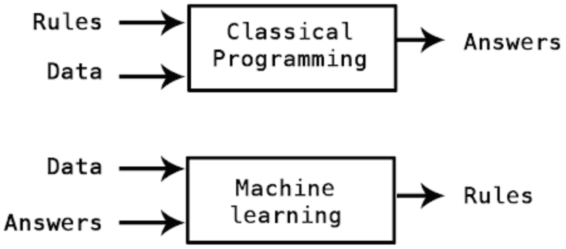

Figure 1.2 Machine learning: a new programming paradigm

A machine learning system is "trained" rather than explicitly programmed. It is presented with many "examples" relevant to a task, and it finds statistical structure in these examples which eventually allows the system to come up with rules for automating the task. For instance, if you wish 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.

Although machine learning only started to flourish in the 1990s, it has quickly become the most popular and most successful subfield of AI, a trend driven by the availability of faster hardware and larger datasets. 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 (e.g. a dataset of millions of images, each consisting of tens of thousands of pixels) for which "classical" statistical analysis such as bayesian analysis would simply be too impractical to be possible. As a result, machine learning, and especially deep learning, exhibits comparatively little mathematical theory—maybe too little—and is very engineering-oriented. It is a hands-on discipline where ideas get proven empirically much more often than theoretically.

1.1.3 Learning representations from data

To define deep learning, and understand the difference between deep learning and other machine learning approaches, first we need to get some idea of what machine learning algorithms really *do*. We just stated that machine learning discovers rules to execute a data-processing task, given examples of what is expected. So, to do machine learning, we need three things:

- Input data points. For instance, if the task is speech recognition, these data points could be sound files of people speaking. If the task is image tagging, they could be picture files.
- Examples of the expected output. In a speech recognition task, these could be

human-generated transcripts of our sound files. In an image task, expected outputs could tags such as "dog", "cat", and so on.

- A way to measure if the algorithm is doing a good job, to measure the distance between its current output and its expected output. This is used as a feedback signal to adjust the way the algorithm works. This adjustment step is what we call "learning".

A machine learning model transforms its input data into a meaningful output, a process which is "learned" from exposure to known examples of inputs and outputs. Therefore, the central problem in machine learning and deep learning is to *meaningfully transform data*, or in other words, to learn useful "representations" of the input data at hand, representations that get us closer to the expected output. Before we go any further: what's a representation? At its core, it's a different way to look at your data—to "represent", or "encode" your 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. Some tasks that may be difficult with one representation can become easy with another. For example, the task "select all red pixels in the image" is simpler in the RBG format, while "make the image less saturated" is simpler in the HSV format. 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.

Let's make this concrete. Let's consider an x axis, and y axis, and some points represented by their coordinates in the (x, y) system: our data, as illustrated in figure 3 1.3.

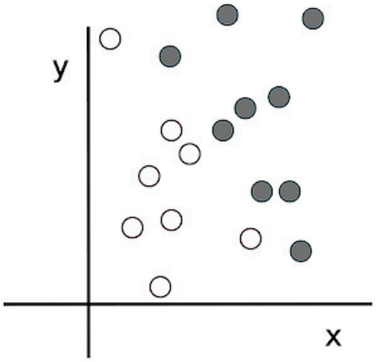

Figure 1.3 Some sample data

As you can see we have a few white points and a few black points. Let's say we want to develop an algorithm that could take the coordinates `(x, y)` of a point, and output whether the point considered is likely to be black or to be white. In this case:

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

What we need here is a new *representation* of our data that cleanly separates the white points from the black points. One transformation we could use, among many other possibilities, would be a coordinate change, illustrated in figure 1.4.

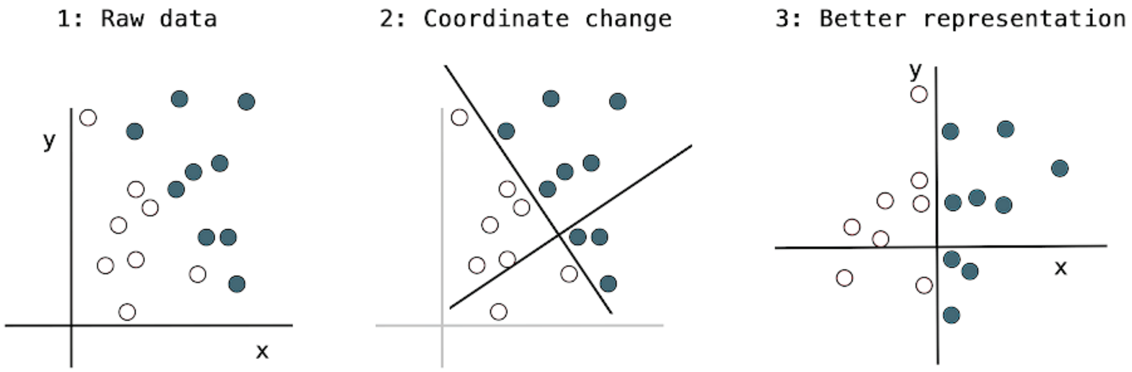

Figure 1.4 Coordinate change

In this new coordinate system, the coordinates of our points can be said to be a new "representation" of our data. And it’s a good one! With this representation, the black/white classification problem can be expressed as a simple rule: black points are such that x 0 or "white points are such that x < 0". Our new representation basically solves the classification problem.

In this case, we defined our coordinate change by hand. But if instead we tried systematically searching for different possible coordinate changes, and used as feedback the percentage of points being correctly classified, then we would be doing machine learning. "Learning", in the context of machine learning, describes an automatic search process for better representations.

All machine learning algorithms consist of automatically finding such transformations that turn data into more useful representations for a given task. These operations could sometimes be coordinate changes, as we just saw, or could be linear projections (which may destroy information), translations, non-linear operations (such as select all points such that x 0), etc. Machine learning algorithms are not usually very creative in finding these transformations, they are merely searching through a predefined set of operations, called an "hypothesis space".

So that’s what machine learning is, technically: searching for useful representations of some input data, within a pre-defined space of possibilities, using guidance from some feedback signal. This simple idea allows for solving a remarkably broad range of intellectual tasks, from speech recognition to autonomous car driving.

Now that you understand what we mean by *learning*, let’s take a look at what makes *deep learning* special.

1.1.4 The "deep" in deep learning

Deep learning is a specific subfield of machine learning, a new take on learning representations from data which puts an emphasis on learning successive "layers" of increasingly meaningful representations. The "deep" in "deep learning" is not a reference to any kind of "deeper" understanding achieved by the approach, rather, it simply stands for this idea of successive layers of representations—how many layers contribute to a model of the data is called the "depth" of the model. Other appropriate names for the field could have been "layered representations learning" or "hierarchical representations learning". Modern deep learning often involves tens or even hundreds of successive layers of representation—and they are all learned automatically from exposure to training data. Meanwhile, other approaches to machine learning tend to focus on learning only one or two layers of representation of the data. Hence they are sometimes called "shallow learning".

In deep learning, these layered representations are (almost always) learned via models called "neural networks", structured in literal layers stacked one after the other. The term "neural network" is a reference to neurobiology, but although some of the central concepts in deep learning were developed in part by drawing inspiration from our understanding of the brain, deep learning models are *not* models of the brain. There is no evidence that the brain implements anything like the learning mechanisms in use in modern deep learning models. One might sometimes come across pop-science articles proclaiming that deep learning works "like the brain", or was "modeled after the brain", but that is simply not the case. In fact, it would be confusing and counter-productive for new-comers to the field to think of deep learning as being in any way related to the neurobiology. You don’t need that shroud of "just like our minds" mystique and mystery. So you might as well forget anything you may have read so far about hypothetical links between deep learning and biology. For our purposes, deep learning is merely a mathematical framework for learning representations from data.

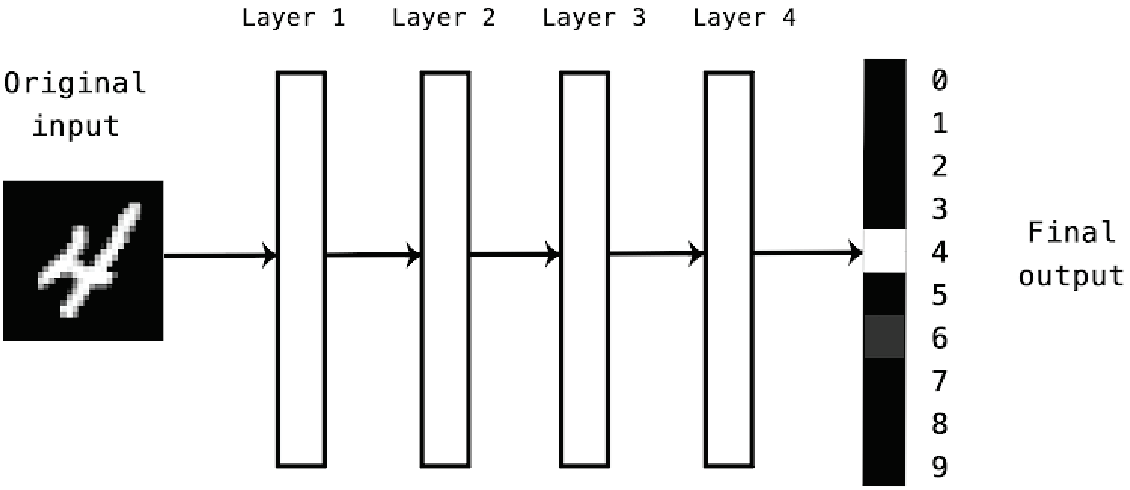

Figure 1.5 A deep neural network for digit classification

What do the representations learned by a deep learning algorithm look like? Let’s look at how a 3-layer deep network transforms an image of a digit in order to recognize what digit it is:

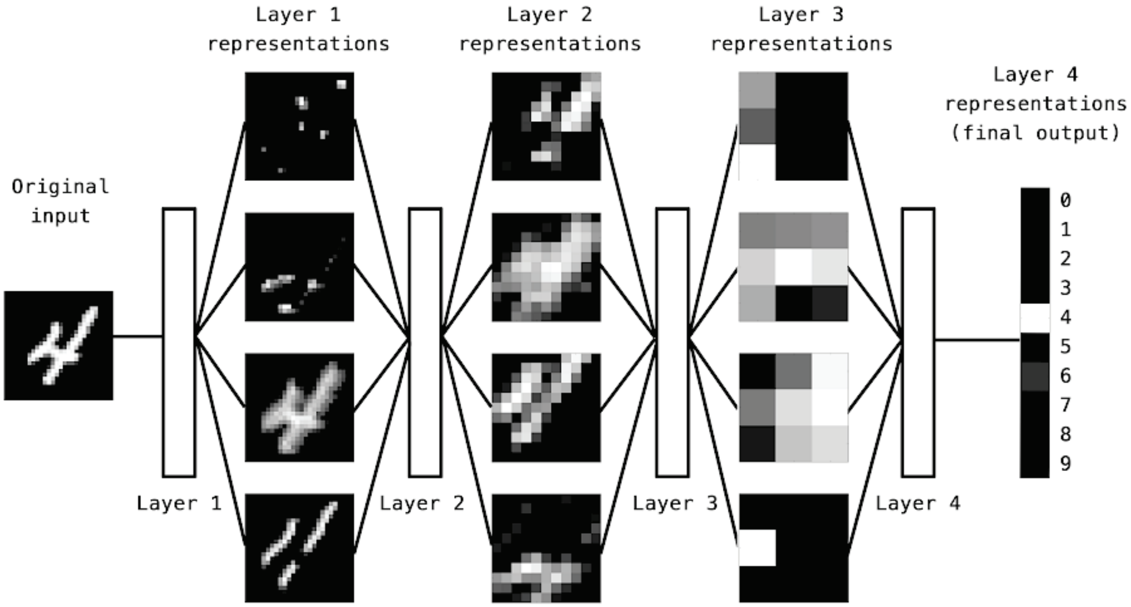

Figure 1.6 Deep representations learned by a digit classification model

As you can see, the network transforms the digit image into representations that are increasingly different from the original image, and increasingly informative about the final result. You can think of a deep network as a multi-stage information distillation operation, where information goes through successive filters and comes out increasingly "purified" (i.e. useful with regard to some task).

So that is what deep learning is, technically: a multi-stage way to learn data representations. A simple idea—but as it turns out, very simple mechanisms, sufficiently scaled, can end up looking like magic.

1.1.5 Understanding how deep learning works in three figures

At this point, you know that machine learning is about mapping inputs (e.g. images) to targets (e.g. the label "cat"), which is done by observing many examples of input and targets. You also know that deep neural networks do this input-to-target mapping via a deep sequence of simple data transformations (called "layers"), and that these data transformations are learned by exposure to examples. Now let’s take a look at how this learning happens, concretely.

The specification of what a layer does to its input data is stored in the layer’s "weights", which in essence are a bunch of numbers. In technical terms, you would say that the transformation implemented by a layer is "parametrized" by its weights. In fact, weights are also sometimes called the "parameters" of a layer. In this context, "learning" will mean finding a set of values for the weights of all layers in a network, such that the network will correctly map your example inputs to their associated targets. 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 since modifying the value of one parameter will affect the behavior of all others!

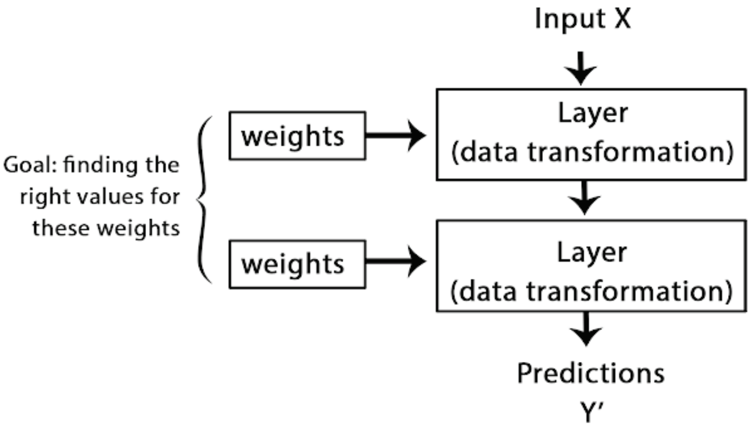
Input X
Goal: finding the right values for these weights
weights
Layer (data transformation)
weights
Layer (data transformation)
Predictions Y'

Figure 1.7 A neural network is parametrized by its weights

To control something, first, you need to be able to observe it. 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 "objective function". The loss function takes the predictions of the network and the true target (what you wanted the network to output), and computes a distance score, capturing how well the network has done on this specific example.

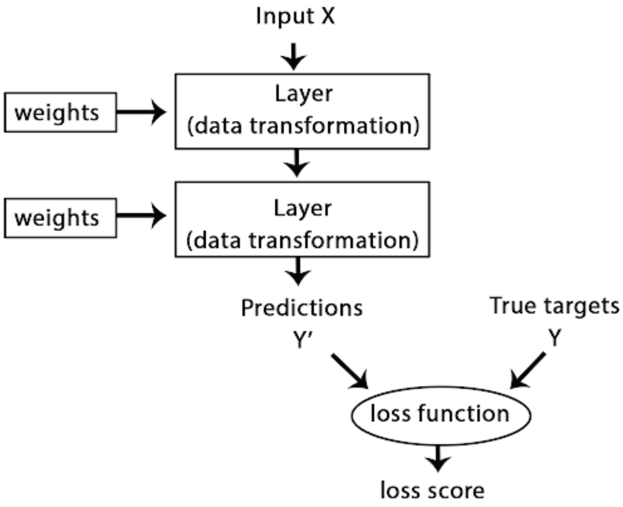

Figure 1.8 A loss function measures the quality of the network's output

The fundamental trick in deep learning is to use this score as a feedback signal to adjust the value of the weights by a little bit, in a direction that would lower the loss score for the current example. This adjustment is the job of the "optimizer", which implements what is called the "backpropagation" algorithm, the central algorithm in deep learning. In the next chapter we will explain in more detail how backpropagation works.

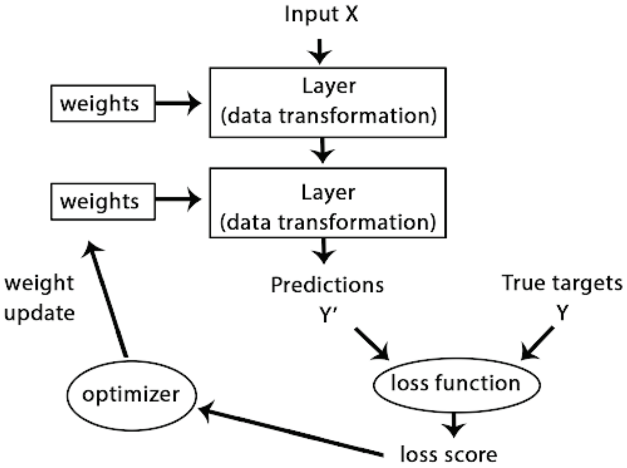

Figure 1.9 The loss score is used as a feedback signal to adjust the weights

Initially, the weights of the network are assigned random values, so the network merely implements a series of random transformations --naturally its output is very far from what it should ideally be, and the loss score is accordingly very high. But with every example that the network processes, the weights get adjusted just a little in the right direction, and the loss score decreases. This is the "training loop", which, repeated a sufficient number of times (typically tens of iterations overs thousands of examples), yields weight values that minimize the loss function. A network with a minimal loss is one for which the outputs are as close as they can be to the targets: a trained network.

Once again: a very simple mechanism, which once scaled ends up looking like magic.

1.1.6 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 all *perceptual* problems, such as "seeing" and "hearing"—problems which involve skills that seem very natural and intuitive to humans but have long been elusive for machines.

In particular, deep learning has achieved the following breakthroughs, all in historically difficult areas of machine learning:

- 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 or Amazon Alexa.
- Near-human level autonomous driving.
- Improved ad targeting, as used by Google, Baidu, and Bing.
- Improved search results on the web.
- Answering natural language questions.
- Superhuman Go playing.

In fact, we are still just exploring the full extent of what deep learning can do. We have started applying it to an even wider variety of problems outside of machine perception and natural language understanding, such as formal reasoning. If successful, this might herald an age where deep learning assists humans in doing science, developing software, and more.

1.1.7 Don’t believe the short-term hype

Although deep learning has led to remarkable achievements in recent years, expectations for what the field will be able to achieve in the next decade tend to run much higher than what will actually turn out to be possible. While some world-changing applications like autonomous cars are already within reach, many more are likely to remain elusive for a long time, such as believable dialogue systems, human-level machine translation across arbitrary languages, and human-level natural language understanding. In particular, talk of "human-level general intelligence" should not be taken too seriously. The risk with high expectations for the short term is that, as technology fails to deliver, research investment will dry up, slowing down progress for a long time.

This has happened before. Twice in the past, AI went through a cycle of intense optimism followed by disappointment and skepticism, and a dearth of funding as a result. It started with symbolic AI in the 1960s. In these early days, projections about AI were flying high. One of the best known pioneers and proponents of the symbolic AI approach was Marvin Minsky, who claimed in 1967: "Within a generation […] the problem of creating 'artificial intelligence' will substantially be solved". Three years later, in 1970, he also made a more precisely quantified prediction: "in from three to eight years we will have a machine with the general intelligence of an average human being". In 2016, such an achievement still appears to be far in the future, so far in fact that we have no way to predict how long it will take, but in the 1960s and early 1970s, several experts believed it to be right around the corner (and so do many people today). A few years later, as these high expectations failed to materialize, researchers and government funds turned away from the field, marking the start of the first "AI winter" (a reference to a nuclear winter, as this was shortly after the height of the Cold War).

It wouldn’t be the last one. In the 1980s, a new take on symbolic AI, "expert systems", started gathering steam among large companies. A few initial success stories triggered a wave of investment, with corporations around the world starting their own in-house AI departments to develop expert systems. Around 1985, companies were spending over a billion dollar a year on the technology, but by the early 1990s, these systems had proven expensive to maintain, difficult to scale, and limited in scope, and interest died down. Thus began the second AI winter.

It might be that we are currently witnessing the third cycle of AI hype and disappointment—and we are still in the phase of intense optimism. The best attitude to adopt is to moderate our expectations for the short term, and make sure that people less familiar with the technical side of the field still have a clear idea of what deep learning can and cannot deliver.

1.1.8 The promise of AI

Although we might have unrealistic short-term expectations for AI, the long-term picture is looking bright. We are only just getting started in applying deep learning to many important problems in which it could prove transformative, from medical diagnoses to digital assistants. While AI research has been moving forward amazingly fast in the past five years, in large part due to a wave of funding never seen before in the short history of A.I, so far relatively little of this progress has made its way into the products and processes that make up our world. Most of the research findings of deep learning are not yet applied, or at least not applied to the full range of problems that they can solve across all industries. Your doctor doesn't yet use AI, your accountant doesn't yet use AI. Yourself, you probably don't use AI technologies in your day-to-day life. Of course, you can ask simple questions to your smartphone and get reasonable answers. You can get fairly useful product recommendations on Amazon.com. You can search for "birthday" on Google Photos and instantly find those pictures of your daughter's birthday party from last month. That's a far cry from where such technologies used to stand. But such tools are still just accessory to our daily lives. AI has yet to transition to become central to the way we work, think and live.

Right now it may seem hard to believe that AI could have a large impact on our world, because at this point AI is not yet widely deployed—much like it would have been difficult to believe in the future impact of the Internet back in 1995. Back then most people did not see how the Internet was relevant to them, how it was going to change their lives. The same is true for deep learning and AI today. But make no mistake: AI is coming. In a not so distant future, AI will be your assistant, even your friend; it will answer your questions, it will help educate your kids, and it will watch over your health. It will deliver your groceries to your door and it will drive you from point A to point B. It will be your interface to an increasingly complex and increasingly information-intensive world. And even more importantly, AI will help humanity as a whole move forwards, by assisting human scientists in new breakthrough discoveries across all scientific fields, from genomics to mathematics.

On the road to get there, we might face a few setbacks, and maybe a new AI winter—in much the same way that the Internet industry got overhyped in 1998-1999 and suffered from a crash that dried up investment throughout the early 2000s. But we will get there eventually. AI will end up being applied to nearly every process that makes up our society and our daily lives, much like the Internet today.

Don't believe the short-term hype, but do believe in the long-term vision. It may take a while for AI to get deployed to its true potential—a potential the full extent of which no one has yet dared to dream—but AI is coming, and it will transform our world in a fantastic way.

1.2 Before deep learning: a brief history of machine learning

Deep learning has reached a level of public attention and industry investment never seen before in the history of AI, but it isn’t the first successful form of machine learning. In fact, it’s a safe bet to say that most of the machine learning algorithms in use in the industry today are still not deep learning algorithms. Deep learning isn’t always the right tool for the job—sometimes there just isn’t enough data for deep learning to be applicable, and sometimes the problem is simply better solved by a different algorithm. If deep learning is your first contact with machine learning, then you may find yourself in a situation where all you have is the deep learning hammer and every machine learning problem starts looking like a nail for this hammer. The only way not to fall into this trap is to be familiar with other approaches and practice them when appropriate.

A detailed exposure of classical machine learning approaches is outside of the scope of this book, but we will briefly go over them and describe the historical context in which they were developed. This will allow us to place deep learning in the broader context of machine learning, and better understand where deep learning comes from and why it matters.

1.2.1 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, yet it is 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 the Bayes Theorem while assuming that the features in the input data are all independent (a strong, or "naive" assumption, which is where the name comes from). This form of data analysis actually predates computers, and was applied by hand decades before its first computer implementation (most likely dating back to the 1950s). The Bayes Theorem and the foundations of statistics themselves date back to the 18th century, and these are all you need to start using Naive Bayes classifiers.

A closely related model is the Logistic Regression (logreg for short), which is sometimes considered to be the "hello world" of modern machine learning. Don’t be misled by its name—logreg is in fact a classification algorithm rather than a regression algorithm. Much like Naive Bayes, logreg predates computing by a long time, yet it is still very useful to this day, thanks to its simple and versatile nature. It is often the first thing a data scientist will try on a dataset to get a feel for the classification task at hand.

1.2.2 Early neural networks

Early iterations of neural networks have been completely supplanted by the modern variants that we cover in these pages; however, it is helpful to be aware of how deep learning originated. Although the core ideas of neural networks were investigated in toy forms as early as the 1950s, the approach took decades to really get started. For a long time, the missing piece was a lack of an efficient way to train large neural networks. This changed in the mid-1980s, as multiple people independently rediscovered the "backpropagation" algorithm, a way to train chains of parametric operations using gradient descent optimization (later in the book, we will go on to precisely define these concepts), and started applying it to neural networks.

The first successful practical application of neural nets came in 1989 from Bell Labs, when Yann LeCun combined together the earlier ideas of convolutional neural networks and backpropagation, and applied them to the problem of handwritten digits classification. The resulting network, dubbed "LeNet", was used by the US Post Office in the 1990s to automate the reading of ZIP codes on mail envelopes.

1.2.3 Kernel methods

As neural networks started gaining some respect among researchers in the 1990s thanks to this first success, a new approach to machine learning rose to fame and quickly sent neural nets back to oblivion: kernel methods.

Kernel methods are a group of classification algorithms, the best known of which is the Support Vector Machine (SVM). The modern formulation of SVM was developed by Vapnik and Cortes in the early 1990s at Bell Labs and published in 1995, although an older linear formulation was published by Vapnik and Chervonenkis as early as 1963.

SVM aims at solving classification problems by finding good "decision boundaries" (Figure 1.10) 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. To classify new data points, you just need to check which side of the decision boundary they fall on.

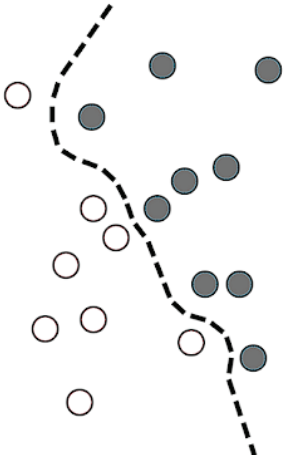

Figure 1.10 A decision boundary

SVMs proceed to find these boundaries in two steps:

- First, the data is mapped to a new high-dimensional representation where the decision boundary can be expressed as an hyperplane (if the data is two-dimensional like in our example, an "hyperplane" would simply be a straight line).
- Then 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". This allows the boundary to generalize well to new samples outside of the training dataset.

The technique of mapping data to a high-dimensional representation where a classification problem becomes simpler may look good on paper, but in practice it is often computationally intractable. That's where the "kernel trick" comes in, the key idea that kernel methods are named after. Here's the gist of it: for finding good decision hyperplanes in the new representation space, you don't have to explicitly compute the coordinates of your points in the new space, you just need to compute the distance between pairs of points in that space, which can be done very efficiently using what is called a "kernel function". A kernel function is a computationally tractable operation that maps any two points in your initial space to the distance between these points in your target representation space, completely by-passing the explicit computation of the new representation. Kernel functions are typically crafted by hand rather than learned from data—in the case of SVM, only the separation hyperplane is learned.

At the time they were developed, SVMs exhibited state of the art performance on simple classification problems, and were one of the few machine learning methods backed by extensive theory and amenable to serious mathematical analysis, making it well-understood and easily interpretable. Because of these useful properties, it became extremely popular in the field for a long time.

However, SVM proved hard to scale to large datasets and did not provide very good results for "perceptual" problems such as image classification. Since SVM is a "shallow" method, applying SVM to perceptual problems requires first extracting useful representations manually (a step called "feature engineering"), which is difficult and brittle.

1.2.4 Decision trees, Random Forests and Gradient Boosting Machines

Decision trees are flowchart-like structures that can allow to classify input data points or predict output values given inputs. They are easy to visualize and interpret. Decisions trees learned from data started getting significant research interest in the 2000s, and by 2010 they were often preferred to kernel methods.

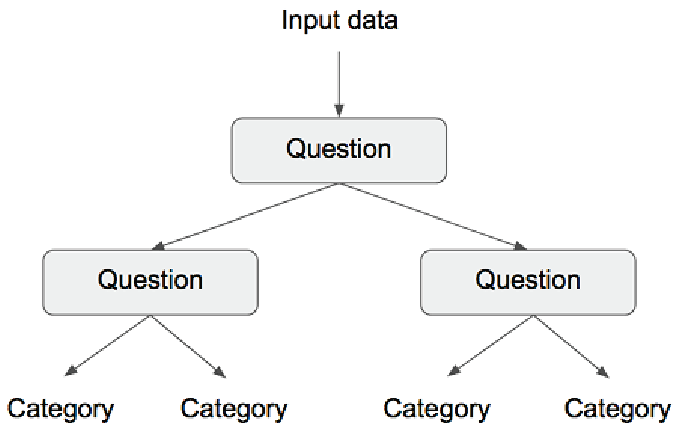

Figure 1.11 A decision tree: the parameters that are learned are the questions about the data. A question could be, for instance, "is coefficient 2 in the data higher than 3.5?".

In particular, the "Random Forest" algorithm introduced a robust and practical take on decision tree learning that involves building a large number of specialized decision trees then ensembling their outputs. Random Forests are applicable to a very wide range of problems --you could say that they are almost always the second-best algorithm for any shallow machine learning task. When the popular machine learning competition website Kaggle.com got started in 2010, Random Forests quickly became a favorite on the platform—until 2014, when Gradient Boosting Machines took over. Gradient Boosting Machines, much like Random Forests, is a machine learning technique based on ensembling weak prediction models, generally decision trees. It leverages "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. Applied to decision trees, the use of the "gradient boosting" technique results in models that strictly outperform Random Forests most of the time, while having very similar properties. It may be one of the best, if not the best, algorithm for dealing with non-perceptual data today. Alongside deep learning, it is one of the most commonly used technique in Kaggle competitions.

1.2.5 Back to neural networks

Around 2010, while neural networks were almost completely shunned by the scientific community at large, a number of people still working on neural networks started making important breakthroughs: the groups of Geoffrey Hinton at the University of Toronto, Yoshua Bengio at the University of Montreal, Yann LeCun at New York University, and IDSIA in Switzerland.

In 2011, Dan Ciresan from IDSIA started winning academic image classification competitions with GPU-trained deep neural networks—the first practical success of modern deep learning. But the watershed moment came in 2012, with the entry of Hinton's group in the yearly large-scale image classification challenge ImageNet. The ImageNet challenge was notoriously difficult at the time, consisting in classifying high-resolution color images into 1000 different categories after training on 1.4 million images. In 2011, the top-5 accuracy of the winning model, based on classical approaches to computer vision, was only 74.3%. Then in 2012, a team led by Alex Krizhevsky and advised by Geoffrey Hinton was able to achieve a top-5 accuracy of 83.6%—a significant breakthrough. The competition has been dominated by deep convolutional neural networks every year since. By 2015, we had reached an accuracy of 96.4%, and the classification task on ImageNet was considered to be a completely solved problem.

Since 2012, deep convolutional neural networks ("convnets") have become the go-to algorithm for all computer vision tasks, and generally all perceptual tasks. At major computer vision conferences in 2015 or 2016, it had become nearly impossible to find presentations that did not involve convnets in some form. At the same time, deep learning has also found applications in many other types of problems, such as natural language processing. It has come to completely replace SVMs and decision trees in a wide range of applications. For instance, for several years, the European Organization for Nuclear Research, CERN, used decision tree-based methods for analysis of particle data from the ATLAS detector at the Large Hadron Collider (LHC), but they eventually switched to Keras-based deep neural networks due to their higher performance and ease of training on large datasets.

1.2.6 What makes deep learning different

The reason why deep learning took off so quickly is primarily that it offered better performance on many problems. But that’s not the only reason. Deep learning is also making problem-solving much easier, because it completely automates what used to be the most crucial step in a machine learning workflow: "feature engineering".

Previous machine learning techniques, "shallow" learning, only involved transforming the input data into one or two successive representation spaces, usually via very simple transformations such as high-dimensional non-linear projections (SVM) or decision trees. But the refined representations required by complex problems generally cannot be attained by such techniques. As such, humans had to go to great length to make the initial input data more amenable to processing by these methods, i.e. they had to manually engineer good layers of representations for their data. This is what is called "feature engineering". Deep learning, on the other hand, completely automates this step: with deep learning, you *learn* all features in one pass rather than having to engineer them yourself. This has greatly simplified machine learning workflows, often replacing very sophisticated multi-stage pipelines with a single, simple, end-to-end deep learning model.

You may ask, if the crux of the issue is to have multiple successive layers of representation, could shallow methods be applied repeatedly to emulate the effects of deep learning? In practice, there are fast-diminishing returns to successive application of shallow learning methods, because *the optimal first representation layer in a 3-layer model is not the optimal first layer in a 1-layer or 2-layer model*. What is transformative about deep learning is that it allows a model to learn all layers of representation *jointly*, at the same time, rather than in succession ("greedily", as it is called). With joint feature learning, whenever the model adjusts one of its internal features, all other features that depend on it will automatically adapt to the change, without requiring human intervention. Everything is supervised by a single feedback signal: every change in the model serves the end goal. This is much more powerful than greedily stacking shallow models, as it allows for very complex and abstract representations to be learned by breaking them down into long series of intermediate spaces (layers), each space only a simple transformation away from the previous one.

These are the two essential characteristics of how deep learning learns from data: the *incremental, layer-by-layer way in which increasingly complex representations are developed*, and the fact *these intermediate incremental representations are learned jointly*, each layer being updated both to follow the representational needs of the layer above and the needs of the layer below. Together, these two properties have made deep learning vastly more successful than previous approaches to machine learning.

1.2.7 The modern machine learning landscape

A great way to get a sense of the current landscape of machine learning algorithms and tools is to look at machine learning competitions on `Kaggle.com`. Due to its highly competitive environment (some contests have thousands of entrants and million-dollar prizes) and to the wide variety of machine learning problems covered, Kaggle offers a realistic way to assess what works and what doesn't. So, what kind of algorithm is reliably winning competitions? What tools do top entrants use?

In 2016, Kaggle is dominated by two approaches: gradient boosting machines, and deep learning. Specifically, gradient boosting is used for problems where structured data is available, while deep learning is used for perceptual problems such as image classification. Practitioners of the former almost always use the excellent XGB library, which offers support for the two most popular languages of data science: Python and R. Meanwhile, most of the Kaggle entrants leveraging deep learning use the Keras library, due to its easy of use, flexibility and support of Python.

These are the two techniques that you should be the most familiar with in order to be successful in applied machine learning today: gradient boosting machines (for shallow learning problems), and deep learning (for perceptual problems). In technical terms, this means that you will need to be familiar with XGB and Keras—the two libraries that are currently dominating Kaggle competitions. With this book in hand, you are already one big step closer.

1.3 Why deep learning, why now?

The two key ideas of deep learning for computer vision, namely convolutional neural networks and backpropagation, were already well-understood in 1989. The LSTM algorithm, fundamental to deep learning for time series, was developed in 1997 and has barely changed since. So why did deep learning only take off after 2012? What changed in these two decades?

In general, there are three technical forces that are driving advances in machine learning:

- Hardware.

- Datasets and benchmarks.
- Algorithmic advances.

Because the field is guided by experimental findings rather than by theory, algorithmic advances only become possible when appropriate data and hardware is available to try new ideas (or just scale up old ideas, as is often the case). Machine learning is not mathematics or physics, where major advances can be done with a pen and a piece of paper. It is an engineering science.

So the real bottleneck throughout the 1990s and 2000s was data and hardware. But here is what happened during that time: the Internet took, and high-performance graphics chips were developed for the needs of the gaming market.

1.3.1 Hardware

Between 1990 and 2010, off-the shelf CPUs have gotten 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.

However, typical deep learning models used in computer vision or speech recognition require orders of magnitude more computational power than what your laptop can deliver. Throughout the 2000s, companies like NVIDIA and AMD have been investing billions of dollars into developing fast, massively parallel chips (graphical processing units, GPUs) for powering 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 a number of various highly-parallelizable applications, starting with physics modeling. Deep neural networks, consisting mostly of many small matrix multiplications, are also highly parallelizable, and around 2011, some researchers started writing CUDA implementations of neural nets—Dan Ciresan and Alex Krizhevsky were some of the first among them.

So what happened is that the gaming market has subsidized supercomputing for the next generation of artificial intelligence applications. Sometimes, big things start as games. Today, the NVIDIA Titan X, a gaming GPU that cost $1000 at the end of 2015, can deliver a peak of 6.6 TLOPS in single-precision, i.e. 6.6 trillion of `float32` operations per second. That’s about 350 times more than what you can get out of a modern laptop. On a Titan X, it only takes a couple of days to train an ImageNet model of the sort that would have won the competition a few years ago. Meanwhile, large companies train deep learning models on clusters of hundreds of GPUs of a type developed specifically for the needs of deep learning, such as the NVIDIA K80. The sheer computational power of such clusters is something that would never have been possible without modern GPUs.

What’s more, the deep learning industry is even starting to go beyond GPUs, and is investing into increasingly specialized and efficient chips for deep learning. In 2016, at its annual I/O convention, Google revealed its "TPU" project (tensor processing unit), a new chip design developed from the ground-up to run deep neural networks, reportedly 10x faster and far more energy-efficient than top-of-line GPUs.

1.3.2 Data

Artificial Intelligence is sometimes heralded as the new industrial revolution. If deep learning is the steam engine of this revolution, then data is its coal. The raw material that powers our intelligent machines, without which nothing would be possible. When it comes to data, besides the exponential progress in storage hardware over the past twenty years, following Moore's law, the game-changer has been the rise of the Internet, making it feasible to collect and distribute very large datasets for machine learning. Today, large companies work with image datasets, video datasets, and natural language datasets that could not have been collected without the Internet. User-generated image tags on Flickr, for instance, have been a treasure trove of data for computer vision. So were YouTube videos. And Wikipedia is a key dataset for natural language processing.

If there is one dataset that has been a catalyst for the rise of deep learning, it is the ImageNet dataset, consisting in 1.4 million images hand-annotated with 1000 images categories (one category per image). But what makes ImageNet special is not just its large size, but also the yearly competition associated with it. As `Kaggle.com` as been demonstrating since 2010, public competitions are an excellent way to motivate researchers and engineers to push the envelope. Having common benchmarks that researchers compete to beat has greatly helped the recent rise of deep learning.

1.3.3 Algorithms

Besides hardware and data, up until the late 2000s, we were still missing a reliable way to train very deep neural networks. As a result, neural networks were still fairly shallow, leveraging only one or two layers of representations, and so they were not able to shine against more refined shallow methods such as SVMs or Random Forests. 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 development of several simple but important algorithmic improvements that allowed for better gradient propagation:

- Better "activation functions" for neural layers.
- Better "weight initialization schemes". It started with layer-wise pre-training, which was quickly abandoned.
- Better "optimization schemes", such as *RMSprop* and *Adam*.

It is only when these improvements started allowing for training models with ten or more layers that deep learning really started to shine.

Finally, in 2014, 2015 and 2016, even more advanced ways to help gradient propagation were discovered, such as batch normalization, residual connections, and depthwise separable convolutions. Today we can train from scratch models that are thousands of layers deep.

1.3.4 A new wave of investment

As deep learning became the new state of the art for computer vision in 2012-2013, and eventually for all perceptual tasks, industry leaders took note. What followed was a gradual wave of industry investment far beyond anything previously seen in the history of AI.

In 2011, right before deep learning started taking the spotlight, the total venture capital investment in AI was around $19M, going almost entirely to practical applications of shallow machine learning approaches. By 2014, it had risen to a staggering $394M. Dozens of startups launched in these 3 years, trying to capitalize on the deep learning hype. Meanwhile, large tech companies such as Google, Facebook, Baidu and Microsoft have invested in internal research departments in amounts that would most likely dwarf the flow of venture capital money. Only a few numbers have surfaced. In 2013, Google acquired the deep learning startup DeepMind for a reported $500M—the largest acquisition of an AI company in history. In 2014, Baidu started a deep learning research center in Silicon Valley, investing $300M in the project. The deep learning hardware startup Nervana Systems was acquired by Intel in 2016 for over $400.

In fact, machine learning and in particular deep learning have become central to the product strategy of these tech giants. In late 2015, Sundar Pichai, Google CEO, stated:

"Machine learning is a core, transformative way by which we're rethinking how we're doing everything. We are thoughtfully applying it across all our products, be it search, ads, YouTube, or Play. And we're in early days, but you will see us?-- in a systematic way—apply machine learning in all these areas."

As a result of this wave of investment, the number of people working on deep learning went in just 5 years from a few hundreds, to tens of thousands, and research progress has reached a frenetic pace. There are currently no signs that this trend is going to slow anytime soon.

1.3.5 The democratization of deep learning

One the key factors driving this inflow of new faces in deep learning has been the democratization of the toolsets used in the field. In the early days, doing deep learning required significant C++ and CUDA expertise, which few people possessed. Nowadays, basic Python scripting skills suffice to do advanced deep learning research. This has been driven most notably by the development of Theano and then TensorFlow, two symbolic tensor manipulation frameworks for Python that support auto-differentiation, greatly simplifying the implementation of new models, and by the rise of user-friendly libraries such as Keras, which makes deep learning as easy as manipulating Lego bricks. After its release early 2015, Keras has quickly become the go-to deep learning solution for large numbers of new startups, grad students, and for many researchers pivoting into the field.

1.3.6 Will it last?

Is there anything special about deep neural networks that makes them the "right" approach for companies to be investing in and for researchers to flock to? Or is deep learning just a fashion that might not last? Will we still be using deep neural networks in 20 years?

The short answer is yes—deep learning does have several properties that justify its status as an AI revolution, and it is here to stay. We may not still be using neural networks two decades from now, but whatever we use will directly inherit from modern deep learning and its core concepts.

These important properties can be broadly sorted into 3 categories:

- Simplicity. Deep learning removes the need for feature engineering, replacing complex, brittle and engineering-heavy pipelines with simple end-to-end trainable models typically built using only 5 or 6 different tensor operations.
- Scalability. Deep learning is highly amenable to parallelization on GPUs or TPUs, making it capable of taking full advantage of Moore’s law. Besides, deep learning models are trained by iterating over small batches of data, allowing them to be trained on datasets of arbitrary size (the only bottleneck being the amount of parallel computational power available, which thanks to Moore’s law is a fast-moving barrier).
- Versatility and reusability. Contrarily to many prior machine learning approaches, deep learning models can be trained on additional data without restarting from scratch, making them viable for continuous online learning, an important property for very large production models. Furthermore, trained deep learning models are repurposable and thus reusable: for instance it is possible to take a deep learning model trained for image classification and drop it into a video processing pipeline. This allows us to reinvest previous work into increasingly complex and powerful models. This also makes deep learning applicable to fairly small datasets.

Deep learning has only been in the spotlight for a few years, and we haven’t yet established the full scope of what it can do. Every passing month we still come up with new use cases, or with engineering improvements lifting previously known limitations. Following a scientific revolution, progress generally follows a sigmoid curve: it starts with a period of fast progress than gradually stabilizes, as researchers start hitting against hard limitations and further improvements become more incremental. With deep learning in 2017, it seems that we are still in the first half of that sigmoid, and there is a lot more progress yet to come in the next few years.