



2. What Is Deep Learning?

“Deep learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but nonlinear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. [...] The key aspect of deep learning is that these layers are not designed by human engineers: they are learned from data using a general-purpose learning procedure” – Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, Nature 2015. [7]

Deep learning is a subfield of machine learning, which is, in turn, a subfield of artificial intelligence (AI). For a graphical depiction of this relationship, please refer to Figure 2.1.

The central goal of AI is to provide a set of algorithms and techniques that can be used to solve problems that humans perform *intuitively* and *near automatically*, but are otherwise very challenging for computers. A great example of such a class of AI problems is interpreting and understanding the contents of an image – this task is something that a human can do with little-to-no effort, but it has proven to be *extremely difficult* for machines to accomplish.

While AI embodies a large, diverse set of work related to automatic machine reasoning (inference, planning, heuristics, etc.), the machine learning subfield tends to be *specifically interested* in **pattern recognition** and **learning from data**.

Artificial Neural Networks (ANNs) are a class of machine learning algorithms that learn from data and specialize in pattern recognition, inspired by the structure and function of the brain. As we’ll find out, deep learning belongs to the family of ANN algorithms, and in most cases, the two terms can be used interchangeably. In fact, you may be surprised to learn that the deep learning field has been around for over *60 years*, going by different names and incarnations based on research trends, available hardware and datasets, and popular options of prominent researchers at the time.

In the remainder of this chapter, we’ll review a brief history of deep learning, discuss what makes a neural network “deep”, and discover the concept of “hierarchical learning” and how it has made deep learning one of the major success stories in modern day machine learning and computer vision.

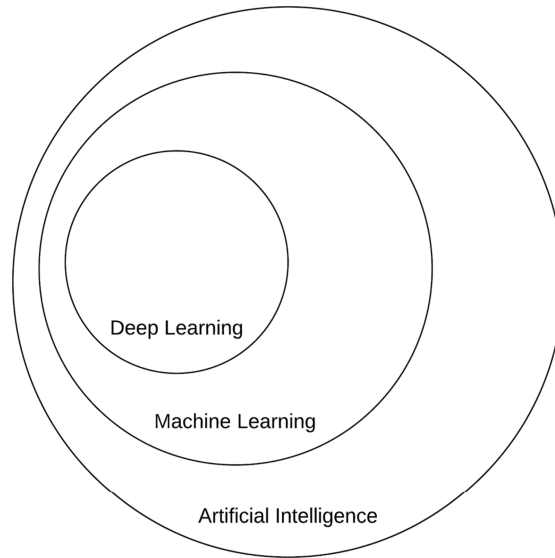


Figure 2.1: A Venn diagram describing deep learning as a subfield of machine learning which is in turn a subfield of artificial intelligence (Image inspired by Figure 1.4 of Goodfellow et al. [8]).

2.1 A Concise History of Neural Networks and Deep Learning

The history of neural networks and deep learning is a long, somewhat confusing one. It may surprise you to know that “deep learning” has existed since the 1940s undergoing various name changes, including *cybernetics*, *connectionism*, and the most familiar, *Artificial Neural Networks* (ANNs).

While *inspired* by the human brain and how its neurons interact with each other, ANNs are *not* meant to be realistic models of the brain. Instead, they are an inspiration, allowing us to draw parallels between a very basic model of the brain and how we can mimic some of this behavior through artificial neural networks. We’ll discuss ANNs and the relation to the brain in Chapter 10.

The first neural network model came from McCulloch and Pitts in 1943 [9]. This network was a *binary classifier*, capable of recognizing two different categories based on some input. The problem was that the *weights* used to determine the class label for a given input needed to be *manually tuned* by a human – this type of model clearly does not scale well if a human operator is required to intervene.

Then, in the 1950s the seminal Perceptron algorithm was published by Rosenblatt [10, 11] – this model could *automatically* learn the weights required to classify an input (no human intervention required). An example of the Perceptron architecture can be seen in Figure 2.2. In fact, this automatic training procedure formed the basis of Stochastic Gradient Descent (SGD) which is still used to train *very deep* neural networks today.

During this time period, Perceptron-based techniques were all the rage in the neural network community. However, a 1969 publication by Minsky and Papert [12] effectively stagnated neural network research for nearly a decade. Their work demonstrated that a Perceptron with a linear activation function (regardless of depth) was merely a linear classifier, unable to solve nonlinear problems. The canonical example of a nonlinear problem is the XOR dataset in Figure 2.3. Take a second now to convince yourself that it is *impossible* to try a *single line* that can separate the blue

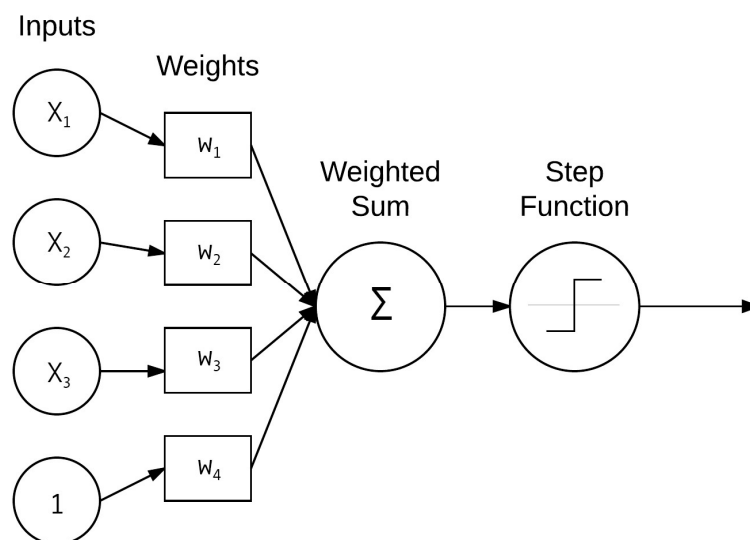


Figure 2.2: An example of the simple Perceptron network architecture that accepts a number of inputs, computes a weighted sum, and applies a step function to obtain the final prediction. We'll review the Perceptron in detail inside Chapter 10.

stars from the red circles.

Furthermore, the authors argued that (at the time) we did not have the computational resources required to construct large, deep neural networks (in hindsight, they were absolutely correct). This single paper alone *almost killed* neural network research.

Luckily, the backpropagation algorithm and the research by Werbos (1974) [13], Rumelhart (1986) [14], and LeCun (1998) [15] were able to resuscitate neural networks from what could have been an early demise. Their research in the backpropagation algorithm enabled *multi-layer feedforward* neural networks to be trained (Figure 2.4).

Combined with nonlinear activation functions, researchers could now learn nonlinear functions and solve the XOR problem, opening the gates to an entirely new area of research in neural networks. Further research demonstrated that neural networks are *universal approximators* [16], capable of approximating any continuous function (but placing no guarantee on whether or not the network can actually *learn* the parameters required to represent a function).

The backpropagation algorithm is the cornerstone of modern day neural networks allowing us to efficiently train neural networks and “teach” them to learn from their mistakes. But even so, at this time, due to (1) slow computers (compared to modern day machines) and (2) lack of large, labeled training sets, researchers were unable to (reliably) train neural networks that had more than two hidden layers – it was simply computationally infeasible.

Today, the latest incarnation of neural networks as we know it is called **deep learning**. What sets deep learning apart from its previous incarnations is that we have faster, specialized hardware with more available training data. We can now train networks with *many more hidden layers* that are capable of hierarchical learning where simple concepts are learned in the lower layers and more abstract patterns in the higher layers of the network.

Perhaps the quintessential example of applied deep learning to feature learning is the *Convo-*

XOR Dataset (Nonlinearly Separable)

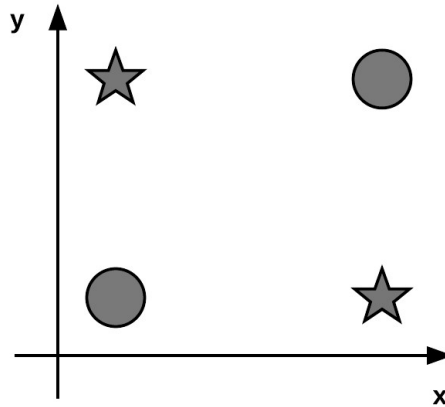


Figure 2.3: The XOR (E(X)clusive Or) dataset is an example of a nonlinear separable problem that the Perceptron *cannot* solve. Take a second to convince yourself that it is impossible to draw a single line that separates the blue stars from the red circles.

lutional Neural Network (LeCun 1988) [17] applied to handwritten character recognition which *automatically* learns discriminating patterns (called “filters”) from images by sequentially stacking layers on top of each other. Filters in lower levels of the network represent edges and corners, while higher level layers use the edges and corners to learn more abstract concepts useful for discriminating between image classes.

In many applications, CNNs are now considered the most powerful image classifier and are currently responsible for pushing the state-of-the-art forward in computer vision subfields that leverage machine learning. For a more thorough review of the history of neural networks and deep learning, please refer to Goodfellow et al. [8] as well as this excellent blog post by Jason Brownlee at Machine Learning Mastery [18].

2.2 Hierarchical Feature Learning

Machine learning algorithms (generally) fall into three camps – *supervised*, *unsupervised*, and *semi-supervised* learning. We’ll discuss supervised and unsupervised learning in this chapter while saving semi-supervised learning for a future discussion.

In the supervised case, a machine learning algorithm is given both a set of *inputs* and *target outputs*. The algorithm then tries to learn patterns that can be used to automatically map input data points to their correct target output. Supervised learning is similar to having a teacher watching you take a test. Given your previous knowledge, you do your best to mark the correct answer on your exam; however, if you are incorrect, your teacher guides you toward a better, more educated guess the next time.

In an unsupervised case, machine learning algorithms try to automatically discover discriminating features *without* any hints as to what the inputs are. In this scenario, our student tries to group similar questions and answers together, even though the student does not know what the correct answer is *and* the teacher is not there to provide them with the true answer. Unsupervised

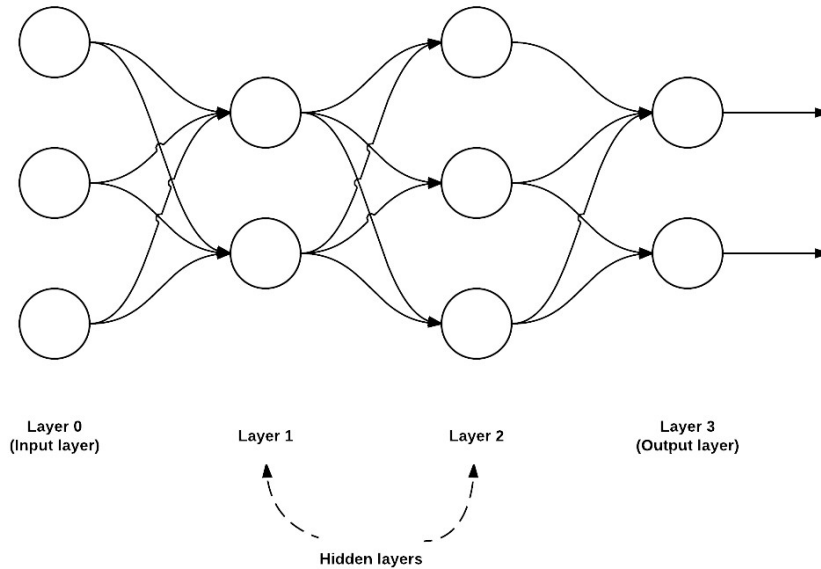


Figure 2.4: A multi-layer, feedforward network architecture with an input layer (3 nodes), two hidden layers (2 nodes in the first layer and 3 nodes in the second layer), and an output layer (2 nodes).

learning is clearly a more challenging problem than supervised learning – by knowing the answers (i.e., target outputs), we can more easily define discriminate patterns that can map input data to the correct target classification.

In the context of machine learning applied to image classification, the goal of a machine learning algorithm is to take these sets of images and identify patterns that can be used to discriminate various image classes/objects from one another.

In the past, we used *hand-engineered features* to quantify the contents of an image – we rarely used raw pixel intensities as inputs to our machine learning models, as is now common with deep learning. For each image in our dataset, we performed *feature extraction*, or the process of taking an input image, quantifying it according to some algorithm (called a *feature extractor* or *image descriptor*), and returning a vector (i.e., a list of numbers) that aimed to quantify the contents of an image. Figure 2.5 below depicts the process of quantifying an image containing prescription pill medication via a series of blackbox color, texture, and shape image descriptors.

Our hand-engineered features attempted to encode texture (Local Binary Patterns [19], Haralick texture [20]), shape (Hu Moments [21], Zernike Moments [22]), and color (color moments, color histograms, color correlograms [23]).

Other methods such as keypoint detectors (FAST [24], Harris [25], DoG [26], to name a few) and local invariant descriptors (SIFT [26], SURF [27], BRIEF [28], ORB [29], etc.) describe *salient* (i.e., the most “interesting”) regions of an image.

Other methods such as Histogram of Oriented Gradients (HOG) [30] proved to be very good at detecting objects in images when the viewpoint angle of our image did not vary dramatically from what our classifier was trained on. An example of using the HOG + Linear SVM detector method

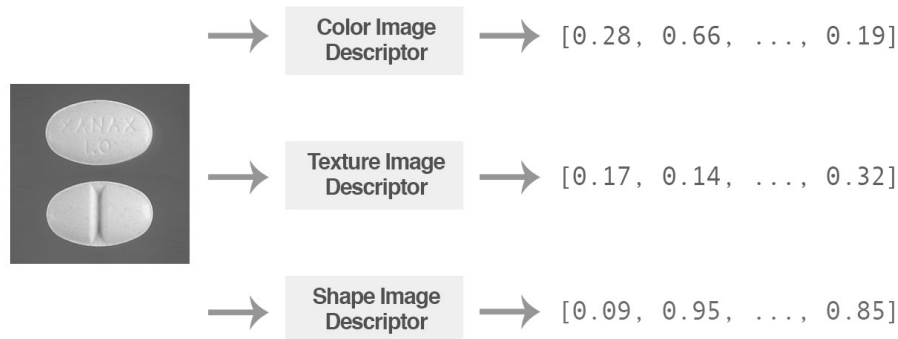


Figure 2.5: Quantifying the contents of an image containing a prescription pill medication via a series of blackbox color, texture, and shape image descriptors.

can be seen in Figure 2.6 where we detect the presence of stop signs in images.

For a while, research in object detection in images was guided by HOG and its variants, including computationally expensive methods such as the Deformable Parts Model [32] and Exemplar SVMs [33].

R For a more in-depth study of image descriptors, feature extraction, and the process it plays in computer vision, be sure to refer to the *PyImageSearch Gurus* course [31].

In each of these situations, an algorithm was *hand-defined* to quantify and encode a particular aspect of an image (i.e., shape, texture, color, etc.). Given an input image of pixels, we would apply our hand-defined algorithm to the pixels, and in return receive a feature vector quantifying the image contents – the image pixels themselves did not serve a purpose other than being inputs to our feature extraction process. The feature vectors that resulted from feature extraction were what we were truly interested in as they served as inputs to our machine learning models.

Deep learning, and specifically Convolutional Neural Networks, take a different approach. Instead of hand-defining a set of rules and algorithms to extract features from an image, **these features are instead automatically learned from the training process.**

Again, let's return to the goal of machine learning: *computers should be able to learn from experience (i.e., examples) of the problem they are trying to solve.*

Using deep learning, we try to understand the problem in terms of a hierarchy of concepts. Each concept builds on top of the others. Concepts in the lower level layers of the network encode some basic representation of the problem, whereas higher level layers *use these basic layers* to form more abstract concepts. This hierarchical learning allows us to *completely remove* the hand-designed feature extraction process and treat CNNs as end-to-end learners.

Given an image, we supply the pixel intensity values as **inputs** to the CNN. A series of **hidden layers** are used to extract features from our input image. These hidden layers build upon each other in a hierarchical fashion. At first, only edge-like regions are detected in the lower level layers of the network. These edge regions are used to define corners (where edges intersect) and contours (outlines of objects). Combining corners and contours can lead to abstract “object parts” in the next layer.

Again, keep in mind that the types of concepts these filters are learning to detect are *automatically learned* – there is no intervention by us in the learning process. Finally, **output** layer is



Figure 2.6: The HOG + Linear SVM object detection framework applied to detecting the location of stop signs in images, as covered inside the PyImageSearch Gurus course [31].

used to classify the image and obtain the output class label – the output layer is either *directly* or *indirectly* influenced by every other node in the network.

We can view this process as hierarchical learning: each layer in the network uses the output of previous layers as “building blocks” to construct increasingly more abstract concepts. These layers are learned *automatically* – there is *no hand-crafted feature engineering* taking place in our network. Figure 2.7 compares classic image classification algorithms using hand-crafted features to representation learning via deep learning and Convolutional Neural Networks.

One of the primary benefits of deep learning and Convolutional Neural Networks is that it allows us to skip the feature extraction step and instead focus on process of training our network to learn these filters. However, as we’ll find out later in this book, training a network to obtain reasonable accuracy on a given image dataset isn’t always an easy task.

2.3 How "Deep" Is Deep?

To quote Jeff Dean from his 2016 talk, *Deep Learning for Building Intelligent Computer Systems* [34]:

“When you hear the term deep learning, just think of a large, deep neural net. Deep refers to the number of layers typically and so this kind of the popular term that’s been adopted in the press.”

This is an excellent quote as it allows us to conceptualize deep learning as large neural networks where layers build on top of each other, gradually increasing in depth. **The problem is we still don’t have a concrete answer to the question, “How many layers does a neural network need to be considered deep?”**

The short answer is there is *no consensus* amongst experts on the depth of a network to be considered deep [8].

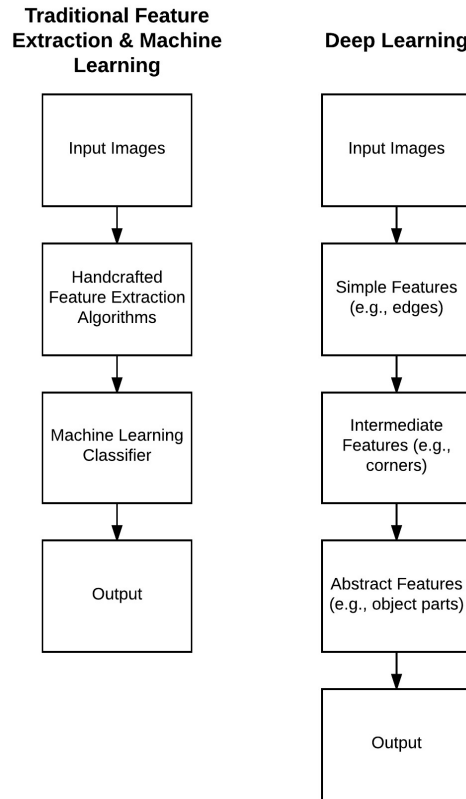


Figure 2.7: **Left:** Traditional process of taking an input set of images, applying hand-designed feature extraction algorithms, followed by training a machine learning classifier on the features. **Right:** Deep learning approach of stacking layers on top of each other that *automatically* learn more complex, abstract, and discriminating features.

And now we need to look at the question of network type. By definition, a Convolutional Neural Network (CNN) is a type of deep learning algorithm. But suppose we had a CNN with only one convolutional layer – is a network that is shallow, but yet still belongs to a family of algorithms inside the deep learning camp considered to be “deep”?

My personal opinion is that any network with greater than two hidden layers can be considered “deep”. My reasoning is based on previous research in ANNs that were heavily handicapped by:

1. Our lack of large, labeled datasets available for training
2. Our computers being too slow to train large neural networks
3. Inadequate activation functions

Because of these problems, we could not easily train networks with more than two hidden layers during the 1980s and 1990s (and prior, of course). In fact, Geoff Hinton supports this sentiment in his 2016 talk, *Deep Learning* [35], where he discussed why the previous incarnations of deep learning (ANNs) did not take off during the 1990s phase:

1. Our labeled datasets were thousands of times too small.

2. Our computers were millions of times too slow.
3. We initialized the network weights in a stupid way.
4. We used the wrong type of nonlinearity activation function.

All of these reasons point to the fact that training networks with a depth larger than two hidden layers were a futile, if not a computational, impossibility.

In the current incarnation we can see that the tides have changed. We now have:

1. Faster computers
2. Highly optimized hardware (i.e., GPUs)
3. Large, labeled datasets in the order of millions of images
4. A better understanding of weight initialization functions and what does/does not work
5. Superior activation functions and an understanding regarding why previous nonlinearity functions stagnated research

Paraphrasing Andrew Ng from his 2013 talk, *Deep Learning, Self-Taught Learning and Un-supervised Feature Learning* [36], we are now able to construct deeper neural networks and train them with more data.

As the *depth* of the network increases, so does the *classification accuracy*. This behavior is different from traditional machine learning algorithms (i.e., logistic regression, SVMs, decision trees, etc.) where we reach a plateau in performance even as available training data increases. A plot inspired by Andrew Ng's 2015 talk, *What data scientists should know about deep learning*, [37] can be seen in Figure 2.8, providing an example of this behavior.

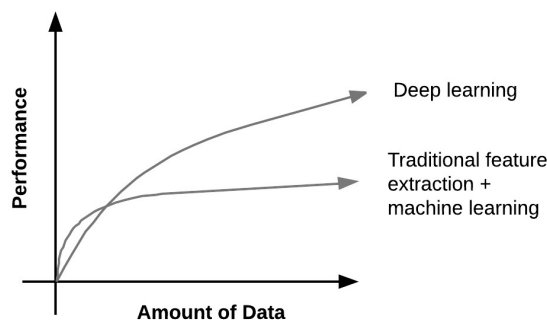


Figure 2.8: As the amount of data available to deep learning algorithms increases, accuracy does as well, substantially outperforming traditional feature extraction + machine learning approaches.

As the amount of training data increases, our neural network algorithms obtain higher classification accuracy, whereas previous methods plateau at a certain point. Because of the relationship between higher accuracy and more data, we tend to associate *deep learning* with *large datasets* as well.

When working on your own deep learning applications, I suggest using the following rule of thumb to determine if your given neural network is deep:

1. Are you using a *specialized* network architecture such as Convolutional Neural Networks, Recurrent Neural Networks, or Long Short-Term Memory (LSTM) networks? If so, **yes, you are performing deep learning**.
2. Does your network have a depth > 2 ? If yes, **you are doing deep learning**.
3. Does your network have a depth > 10 ? If so, **you are performing very deep learning** [38].

All that said, try not to get caught up in the buzzwords surrounding deep learning and what is/is not deep learning. At the very core, deep learning has gone through a number of different

incarnations over the past 60 years based on various schools of thought – **but *each* of these schools of thought centralize around artificial neural networks inspired by the structure and function of the brain.** Regardless of network depth, width, or specialized network architecture, you’re *still* performing machine learning using artificial neural networks.

2.4 Summary

This chapter addressed the complicated question of “*What is deep learning?*”.

As we found out, deep learning has been around since the 1940s, going by different names and incarnations based on various schools of thought and popular research trends at a given time. At the very core, deep learning belongs to the family of Artificial Neural Networks (ANNs), a set of algorithms that learn patterns inspired by the structure and function of the brain.

There is no consensus amongst experts on exactly what makes a neural network “deep”; however, we know that:

1. Deep learning algorithms learn in a hierarchical fashion and therefore stack multiple layers on top of each other to learn increasingly more abstract concepts.
2. A network should have > 2 layers to be considered “deep” (this is my anecdotal opinion based on decades of neural network research).
3. A network with > 10 layers is considered *very deep* (although this number will change as architectures such as ResNet have been successfully trained with over 100 layers).

If you feel a bit confused or even overwhelmed after reading this chapter, don’t worry – the purpose here was simply to provide an extremely high-level overview of deep learning and what exactly “deep” means.

This chapter also introduced a number of concepts and terms you may be unfamiliar with, including pixels, edges, and corners – our next chapter will address these types of image basics and give you a concrete foundation to stand on. We’ll then start to move into the fundamentals of neural networks, allowing us to graduate to deep learning and Convolutional Neural Networks later in this book. While this chapter was admittedly high-level, the rest of the chapters of this book will be extremely hands-on, allowing you to master deep learning for computer vision concepts.