Mathematical Engineering of Deep Learning

Book Draft

Benoit Liquet, Sarat Moka and Yoni Nazarathy

February 28, 2024

Contents

1 Introduction - DRAFT

Many methods and techniques of deep learning have been known for a long time. However, it is only recently that deep learning became a field of its own. At its core, deep learning is a collection of models, algorithms, and techniques, such that when assembled together, efficient automated detection and decision making can take place. Trained deep learning models are able to detect, classify, translate, create, and take part in systems that execute simple human like tasks and beyond. The basic building block of deep learning is the *artificial neural network*, or *neural network* for short. Deep learning is all about defining, training, and using such objects, often with multiple layers - hence the term “deep”.

While most of this book is about the mathematical engineering of deep learning, this introductory chapter takes a more general viewpoint. It aims to introduce the field and the terminology in broad terms. In Section 1.1 we introduce deep learning by discussing the general nature of the field and key terms involved. We continue in Section 1.2 where we present an overview of some of the most popular tasks and their associated deep learning architectures. These architectures then re-appear in detail in the remainder of the book. In Section 1.3 we discuss key ingredients of the field. It is here where we briefly touch on connections between deep learning and neuroscience, discuss computation power, and discuss the availability of large datasets. In the remainder of the book, since our focus is on mathematical formulation, these topics are seldom considered, yet we consider them here for completeness. In Section 1.4 we introduce common openly available datasets that are often used to train deep learning models, both in practice and industrially. In Section 1.5 we introduce the concept, **mathematical engineering of deep learning** which is behind the title of this book. We close with Section 1.6 where we introduce common mathematical notation used in the book.

1.1 The Age of Deep Learning

Classically, computers can be programmed to do complicated repetitive tasks very well. This includes automating calculations, sorting and filtering data, finding shortest paths between two locations on a map, and even executing extremely complicated weather simulations. For many of these tasks, one may consider specialized, well-defined algorithms whose specification is detailed by a sequence of steps including logical expression evaluation, conditional statements, iteration, and similar constructs.

With such a classical algorithmic approach, software can carry out computational tasks in a faster and more accurate manner than any living creature. No human can sort a large list of numbers faster than a computer, and no dog or other non-human animal can be expected to reliably sort numbers at all. However humans and other animals are able to think! This is something that computers (still) cannot do. On a more elementary level, many living beings can quickly learn to determine types of objects based on their appearance, sound, feel, or smell in a manner that up to **recently** was not possible by a machine.

The key here is **recently**. The second decade of the 21st century has witnessed incredible advances in the ability of computers to carry out (simple) human-like tasks. These include interpretation of images, voice, and text, learning how to execute smart decision making in uncertain environments such as playing games like Go, and highly impressive conversational agents which can communicate with humans in ways that appear human. The main vehicle for this success is **deep learning**.

The general computational area dealing with algorithms for classification, prediction, and decision making based on data is generally called *machine learning*. It has been around since the late 1950's together with the general phrase of *artificial intelligence*. Machine learning often involves programmed statistical techniques for tuning parameters of algorithms based on data, so that later when used on unseen data, the algorithms hopefully work well. There are dozens if not hundreds of well developed machine learning techniques and models. One class of models involves artificial neural networks which have recently also become known as **deep learning**.

Up to about 2010, deep learning models were generally perceived as just another class of machine learning models, which while being interesting, in many cases were often not the most successful or insightful tools to use. However, together with the advent of large collected datasets and the availability of GPUs (graphical processing units), deep learning has emerged as the norm in the machine learning world rather than the exception. There are now tens of thousands of companies around the world that are deploying and developing automated applications that use deep learning technology. Indeed, the emergence of deep learning has brought artificial intelligence and machine learning to the forefront of technology and it is by now an integral part of almost any application. Many human like tasks which include image processing, voice analysis, as well as dealing with general complex datasets, are handled exceptionally well by deep learning techniques.

The world of *machine learning (ML)*, *artificial intelligence (AI)*, and *deep learning (DL)* is often characterized via multiple competing terms. In addition to ML, AI, and DL which are often used interchangeably, other key terms include *data science* and *statistics*. Related terms that have somewhat stepped out of the spotlight include *statistical learning*, *data mining*, and *big data (analytics)*. When considered in isolation, each of these terms may be broadly defined with a slightly different meaning. However, when considered together there are significant intersections between the fields and meanings of the terms. One general way to describe the key terms is to say that **DL is a suite of very popular ML techniques which power much of today's AI**. Further, DL is one of the key components in the tool-box of data science and complements more traditional (as well as new and novel) methods from statistics. We generally do not use the ML, AI, and DL acronyms further in the book but rather use the general term deep learning to describe almost all of the methods and activities in the field.

A First Dive Into Deep Learning

A deep learning model is a very versatile function, denoted as $f_\theta(x)$, where the input x is typically high dimensional and the parameter θ is high dimensional as well. The function returns an output $y = f_\theta(x)$, where y is typically of significantly lower dimension than both θ and x. Given large datasets comprised of many x inputs matching desired outputs y, it is often possible to find good parameter values θ such that $f_\theta(\cdot)$ approximately satisfies the desired relationship between x and y. The process of finding such θ is called *training* or *learning*. Once trained, the model $f_\theta(\cdot)$ can be applied to *unseen input data*, x, with a hope of making good predictions, classifications, or decisions.

Depending on the application, the input data x can be in the form of an image, text, a sound waveform, tabular heterogeneous data, or some other variant. The output data y can be the probability of a label indicating the meaning/content of the image, a numerical value, or an object of similar form to x such as translated text in case that x is text, a masked image in case that x is an image, or similar.

Focusing on vision (images) for our initial example, one very popular source of data is the *ImageNet database* which has been used for the development and benchmarking of many deep learning models. The database has nearly 15 million color images. In many models a subset of about 1.5 million images are used for training where the basic form of y is given by a label which is one of $1,000$ categories indicating the content of the image. More on ImageNet is in Section 1.4 where we outline other popular datasets as well.

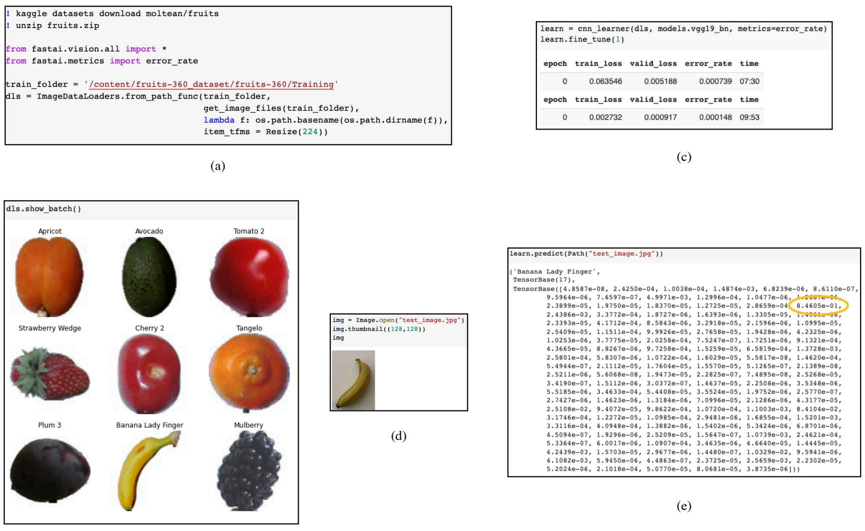
To get a feel for a deep learning model $f_\theta(\cdot)$, consider the *VGG19 model*[1] which is one of several popular (now classical) deep learning models for images. For this model, x is a 224×224 color image. It is thus comprised of $3 \times 224 \times 224 = 150,528$ values, with every coordinate of x representing the intensity of a specific pixel color (red, green, or blue). The output y is a $1,000$ dimensional vector where each coordinate of the vector corresponds to a different type of object, e.g., `car`, `banana`, etc. The numerical value of the coordinate y_i, where say i is the index which matches `banana` is the model's prediction of the probability that the input x is a picture of a banana.

Ideally when fed with an input image x of a banana, the output $y = f_\theta(x)$ will have y_i as a high probability value e.g., 0.85 for $i =$ `banana`, while the other y_j for j other than `banana` will be low. This then allows one to use $f_\theta(\cdot)$ to classify bananas and other objects by choosing the label (`banana`, `car`, etc...) with the highest output probability y_i. Such a machine learning task is called *classification.*

At around the year 2014 when VGG19 was introduced, it was fed about 1.5 million ImageNet images, x, each with a corresponding label, e.g., `banana`, which is essentially a desired output y. The process of training VGG19 then involved finding good parameters θ so that when the model is presented with a new unseen image x, it predicts the label of the image well. Note that in VGG19, θ has a huge number of parameters; 144 million!

So to recap, there were about 1.5×10^6 input data samples each of size of about 1.5×10^5 (pixel values). Hence the training data size has about 2.25×10^{11} values (numbers). This data was then used to learn about 1.44×10^8 parameters. This is a-lot of data, and a-lot of trained parameters, but the resulting trained VGG19 model, $f_\theta(\cdot)$, works well. At the time when this specific model was introduced it took days to train and much longer to fine tune. Today such a model may take around 8 hours to train on current state of the art hardware and software. Further, it can take about a fifths of a second to make a prediction with this model, that is to evaluate $f_\theta(x)$. This is not an insignificant duration and can be improved upon by other models.

[1] Note that we only chose the VGG19 model here as an illustration. More efficient modern models are discussed later in the book, with image models discussed specifically in Chapter 6.



```
! kaggle datasets download moltean/fruits
! unzip fruits.zip

from fastai.vision.all import *
from fastai.metrics import error_rate

train_folder = '/content/fruits-360_dataset/fruits-360/Training'
dls = ImageDataLoaders.from_path_func(train_folder,
                                      get_image_files(train_folder),
                                      lambda f: os.path.basename(os.path.dirname(f)),
                                      item_tfms = Resize(224))
```

(a)

```
dls.show_batch()
```

(b)

```
learn = cnn_learner(dls, models.vgg19_bn, metrics=error_rate)
learn.fine_tune(1)

epoch  train_loss  valid_loss  error_rate  time
    0    0.063546    0.005188    0.000739  07:30
epoch  train_loss  valid_loss  error_rate  time
    0    0.002732    0.000917    0.000148  09:53
```

(c)

```
img = Image.open("test_image.jpg")
img.thumbnail((128,128))
img
```

(d)

```
learn.predict(Path("test_image.jpg"))

('Banana Lady Finger',
 TensorBase(17),
 TensorBase([4.8587e-08, 2.4250e-04, 1.0038e-04, 1.4874e-03, 6.8239e-06, 8.6110e-07,
         9.5964e-06, 7.6597e-07, 4.9971e-03, 1.2996e-04, 1.0477e-06, 1.2687e-06,
         2.3899e-05, 1.9750e-05, 1.8370e-05, 1.2725e-05, 2.8659e-04, 8.4605e-01,
         2.4386e-03, 3.3772e-04, 1.8727e-06, 1.6393e-06, 1.3305e-05, 1.4961e-06,
         2.3393e-05, 4.1712e-04, 8.5843e-06, 3.2918e-05, 2.1596e-06, 1.0995e-05,
         2.5409e-05, 1.1511e-04, 9.9926e-05, 2.7658e-05, 1.9428e-06, 4.2325e-06,
         1.0253e-06, 3.7775e-05, 2.0258e-04, 7.5247e-07, 1.7251e-06, 9.1321e-04,
         4.3665e-05, 8.8267e-06, 9.7258e-04, 1.5259e-05, 6.5819e-04, 1.3728e-03,
         2.5801e-04, 5.8307e-06, 1.0722e-04, 1.6029e-05, 5.5817e-08, 1.4620e-04,
         5.4944e-07, 2.1112e-05, 1.7604e-05, 1.5570e-05, 5.1265e-07, 2.1389e-08,
         2.5211e-06, 5.6068e-08, 1.9473e-05, 2.2825e-07, 7.4895e-08, 2.5268e-05,
         3.4190e-07, 1.5112e-06, 3.0372e-07, 1.4637e-05, 2.2508e-06, 3.5348e-06,
         5.5185e-06, 3.4633e-04, 5.4408e-05, 3.5524e-05, 1.9752e-06, 2.5770e-07,
         2.7427e-06, 1.4623e-06, 1.3184e-06, 7.0996e-05, 2.1286e-06, 4.3177e-05,
         2.5108e-02, 9.4072e-05, 9.8622e-04, 1.0720e-04, 1.1003e-03, 8.4104e-02,
         3.1746e-04, 1.2272e-05, 1.0985e-04, 2.9481e-06, 1.6855e-04, 1.5201e-03,
         3.3116e-04, 4.0948e-04, 1.3882e-06, 1.5402e-06, 5.3424e-06, 6.8701e-06,
         4.5094e-07, 1.9296e-06, 2.5209e-05, 1.5647e-07, 1.0739e-03, 2.4621e-04,
         5.3364e-07, 6.0017e-06, 1.0907e-04, 3.4635e-06, 4.6640e-05, 1.4445e-05,
         4.2439e-03, 1.5703e-05, 2.9677e-06, 1.4480e-07, 1.0329e-02, 9.5941e-06,
         4.1082e-03, 5.9450e-06, 4.4863e-07, 2.3725e-05, 2.5659e-03, 2.2302e-05,
         5.2024e-06, 2.1018e-04, 5.0770e-05, 8.0681e-05, 3.8735e-06]))
```

(e)

Figure 1.1: Illustrative fast.ai Python code for training (fine tuning) a VGG19 model to recognize fruits. The VGG19 model was originally trained using ImageNet and this pre-trained model is easily downloaded. Retraining this model from scratch using ImageNet would take several hours. Instead, in this example VGG19 is adapted to data from the Fruits 360 dataset. This adaptation takes about 10 minutes using a GPU. We then use the trained model on a single ad hoc image of a banana and it is correctly classified as `Banana Lady Finger`.

As a matter of illustration Figure 1.1 presents code that uses the VGG19 model which was previously trained on ImageNet. This training of VGG19 from scratch on ImageNet is not something one would typically do in practice. Instead, we use the pre-trained VGG19 model parameters and adapt them based on another dataset called *Fruits 360*[2] which has nearly 100, 000 images of fruits, most of which were taken with repetitions by rotating a single fruit of each type and taking pictures at different angles. Here we use the *fast.ai* library with the *Python* language which also uses *PyTorch* under the hood. However, we could have presented alternatives with other languages (e.g., Julia or R) as well as other deep learning libraries such as for example *Keras* which uses *TensorFlow*. Indeed, there are many books dealing with deep learning libraries and code with notable ones mentioned in the notes and references at the end of the chapter. This current book is not about using such libraries and the practicalities of deep learning, it is rather about the concepts and key ideas.

In Figure 1.1 (a) we present setup code including downloading Fruits 360 from Kaggle. See also Section 1.4 where we discuss publicly available datasets. In (b) we present a few Fruits 360 images and their labels. In (c) there is the actual code and output needed for training (fine tuning) of the pre-trained VGG19 model (`models.vgg19_bn`) using fast.ai's `fine_tune()` method. This takes about 10 minutes to execute. This practice is called *transfer learning* and is also known as *fine-tuning*. In (d) we present an example banana image not related to the Fruits 360 dataset. Finally in (e) we execute model prediction using fast.ai's `learn.predict()` method where the probability associated with `Banana Lady Finger` is highest among all possible fruits. It is highlighted in index `17`. This code takes a fraction of

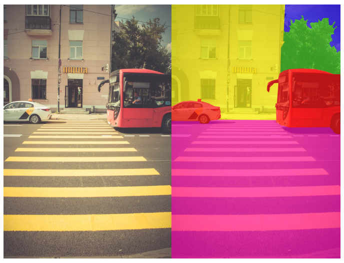
[2]See https://www.kaggle.com/datasets/moltean/fruits.

a second to execute. We note that Figure 1.1 is the first and last example in this book that includes computer code.

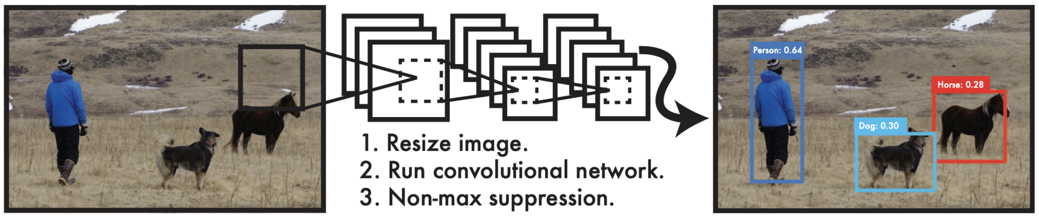
Beyond Classification

Deep learning models can be used to perform various forms of *tasks*. The classification task described above is one type of these tasks and is often considered the most basic task. Other tasks include *regression* where y is a numerical value associated with x which needs to be predicted. Chapter 2 presents an introduction to regression and classification tasks in the context of general machine learning, together with an overview of other aspects of machine learning.

There are many more involved tasks as well. One example in the context of image data is *localization* where the goal is to determine the location of an object within an image. This task can also be handled by a variant of VGG19 where the input data is still a $3 \times 224 \times 224$ image but the output data y encodes the location of an object in the image (as well as possibly the type of object). Such a model is no longer $f_\theta(\cdot)$ above, but some other function, say $\tilde{f}_{\tilde{\theta}}(\cdot)$. While the models differ, one of the useful things about deep learning is that the parameters θ for the classification task and the parameters $\tilde{\theta}$ for the localization task are often similar and the bulk of the training effort can be used to learn both θ and $\tilde{\theta}$.



(a)



(b)

Figure 1.2: Different types of image tasks.[3] (a) Semantic segmentation of images. (b) Object detection and localization.

In the context of images other tasks include *semantic segmentation* where all individual pixels of the input image x are marked as belonging to specific categories or classes. In such a case, y is of a similar form to x since it includes an indication of the class of each individual pixel. This case also clearly requires training data where each individual input

[3]Image (a) is attributed to B. Palac under the creative commons license and available via Wikimedia Commons. Image (b) is thanks to "You Only Look Once: Unified, Real-Time Object Detection" by J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, [346].

pixel is considered. See for example Figure 1.2 (a). Further one may wish to identify multiple objects in an image as in Figure 1.2 (b). There are other tasks as well when considering text, video, sound, or tabular data and these are discussed in Section 1.2 and throughout the book.

Deep Learning: Where is it Applied?

If you are reading this book then you probably already know that deep learning is applied in many diverse contexts. By around the time of publication of this book, many automated systems that involve sensing of noisy data already use some variant of deep learning. Many statistical modeling arenas also make use of deep learning. And finally, at the time of publication of this book, some mundane programming or writing tasks are beginning to be replaced by deep learning.

A medical doctor's decision can be viewed as $y = f_\theta(x)$. Here x may be the full medical history of the patient, or more specifically it may be the pixel information of medical imaging. In this context, y may simply be an indication of `benign` vs. `malignant` or may involve more complicated outputs such as `rest for one week and then get checked up again`. On much smaller time scales, the steering wheel actions of a driver are also of the form $y = f_\theta(x)$. Here x can be based on the fusion of multiple sensory data of a car and y is the steering command. Similarly, a farm worker that decides which tomatoes to pick today and which to wait on also repeatedly uses a relationship of the form $y = f_\theta(x)$. The same goes for automated voice recognition systems, for determination of the importance of text messages, and for comprehending hand written text or hand written mathematical equations. Even the task of converting a *prompt* such as `please write code for creating a web server for e-commerce for...`, can be carried out via deep learning where the output is a complete set of actions that create the web-server.

In all of these application domains a trained human makes decisions based on sensory inputs and determines an outcome. It is true that other statistical and machine learning techniques for $y = f_\theta(x)$ may also do the job instead of deep learning. However, experience of the past decade has shown that with ample training data and multiple features, deep learning methods work exceptionally well and in many cases surpass the performance of other machine learning methods such as support vector machines, random forests, or other methods.

Who are the Personas Involved?

Due to its success, deep learning has been dubbed the "new electricity". If that is the case then who are the "new electricians"? The answer to this question is still evolving because the age of deep learning has just begun. Nevertheless we can try and answer this question based on where deep learning stands as of 2024. As a first go you may use general labels such as *data scientists*, *statisticians*, *computer scientists*, and *machine learning engineers*. People that train deep learning models would probably classify themselves as one of these. If their work involves using modeling to make predictions, gain business insight from data, or help finding scientific relationships between variables then the "data scientist" label is probably most appropriate. Further, if they use more statistical rigour for model selection and making conclusions then "a statistician" is probably the right persona classification to use. This is especially true when the work of individuals involves experimental design and analysis of experimental data. Finally, if the work involves training and integrating deep learning solutions in software, then a "machine learning engineer" is probably the correct description.

In terms of research and core development of ideas, many of the developers of deep learning ideas and technology are probably rightly called "computer scientists". Computer science is obviously a broad field which at one extreme deals with the discrete mathematical theoretical questions such as $P \overset{?}{=} NP$, and at the other extreme includes ideas from artificial intelligence pioneers such as Frank Rosenblatt, the initial creator of the perceptron in the 1950's and many others that worked on neural networks during the second half of the 20th century. In recent years computer scientists responsible for the deep learning revolution include names[4] such as Yann LeCun, Yoshua Bengio, Geoffrey Hinton and many others. Other important names related both to development of ideas, development of software, and education include Ian Goodfellow, Jeremey Howard, and Andrew Ng. These individuals, together with thousands of others, are responsible for recent advances and education of deep learning that made it what it is today. Most of these researchers would probably feel comfortable with the title "computer scientists", however you will also find "mathematicians", "statisticians", and "neuroscientists" heavily involved.

In addition to some of the deep learning pioneers mentioned above, many *researchers* from other domains are also now focusing heavily on deep learning. This involves experts from the world of *pure mathematics*, *probability*, *statistics*, *information theory*, and *control theory*. In the years to come we may witness development of theoretical results that describe deep learning; what works, what does not, and why. The focus of this book is not on such theoretical results. Our focus is rather on *mathematical engineering*, a phrase that we define in Section 1.5 below.

1.2 A Taste of Tasks and Architectures

A reader completing this book is to gain a mathematically aided understanding of deep learning. This includes understanding deep learning model architectures designed for a variety of tasks as well as an understanding of the motivation behind several architectural choices. Further, since deep learning architectures and their training algorithms go hand in hand, the journey also encompasses key methods used to train such models.

To get a feel for the models and methods covered, we now present an overview of the key tasks and architectures. Figure 1.3 presents schematics which include the simplest deep models called *feedforward fully connected neural networks*[5]; *autoencoders* which combine such feedforward networks with an *encoder* and a *decoder*; *convolutional neural networks* useful for image analysis; *recurrent neural networks* useful for sequence data; *transformer models* useful for *large language models* and other applications internally using the *attention mechanism* and also utilizing an encoder and decoder like autoencoders; *diffusion models* that are useful for image generation; *generative adversarial network* architectures also useful for fake data generation; and the paradigm of *reinforcement learning* useful for control of dynamic systems.

[4]In general this book aims to minimize historical notes, yet you may find more historical information in the notes and references at the end of this chapter.

[5]Such models are also known as *fully connected networks*, *feedforward networks*, or *dense neural networks* where each of these terms may also be augmented with the phrase "deep" as well as the phrase "general". Another name for such models is *multi-layer perceptrons* (MLP).

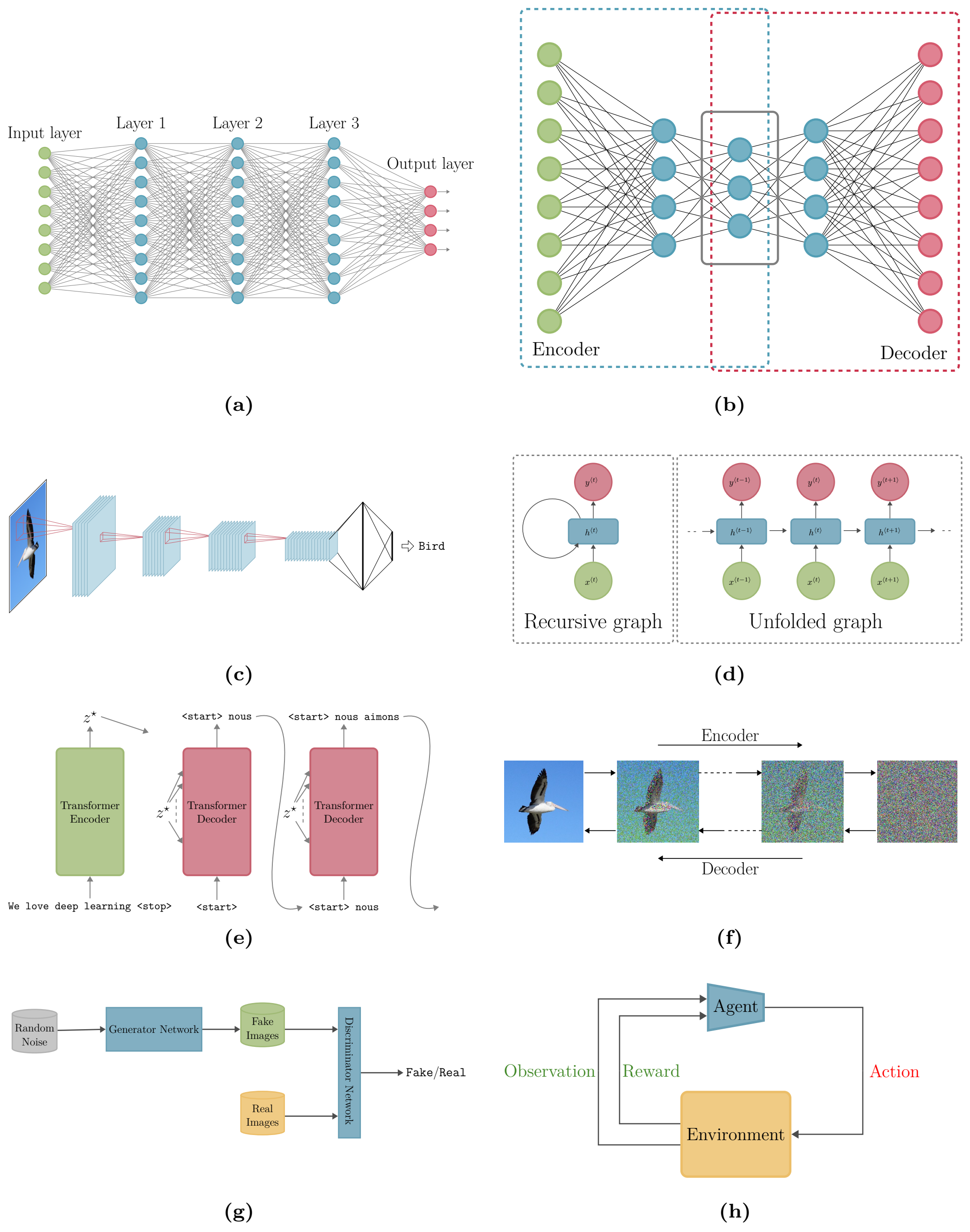


Figure 1.3: Illustrations of some common deep learning architectures and paradigms: (a) Feedforward fully connected neural networks covered in Chapter 5. (b) Autoencoders with shallow versions covered in Chapter 3. (c) Convolutional neural networks covered in Chapter 6. (d) Recurrent neural networks covered in Chapter 7. (e) Transformer architectures with an encoder and repeated application of a decoder covered in Chapter 7. (f) Diffusion models for image generation covered in Chapter 8. (g) Generative adversarial networks covered in Chapter 8. (h) Reinforcement learning covered in Chapter 8.

Feedforward Fully Connected Neural Network

The most basic deep neural network is the *feedforward fully connected neural network*. It is illustrated in Figure 1.3 (a) and covered in detail in Chapter 5. Simple special cases of this network are the *linear model*, analyzed in Chapter 2, as well as *logistic regression* (sigmoid) and *multinomial regression model* (softmax), both covered in Chapter 3.

Mathematically, feedforward fully connected neural networks are simply combinations of affine (linear) transformations and non-linear activation functions. They constitute a very basic mechanism for enhancing the classical linear model with non-linearities. It turns out that this enhancement gives the model, $f_\theta(\cdot)$, an incredible ability to express complex relationships, $y = f_\theta(x)$, while supporting an algorithmically tractable way of finding θ (training). Classically these models are also called *multi-layer perceptrons* since they are descendants of the first ever neural network model, the *perceptron*, developed by Frank Rosenblatt in the late 1950's.

This architecture is useful for tasks such as classification, regression, or feature extraction. The models provide a highly expressive ability, which is especially useful when the data does not have a specific structure. While the models are heavily parameterized, they often work well for such ad-hoc tasks. Components of these networks, called *fully connected layers*, can also be components of more complex architectures such as convolutional networks, transformer models, and others.

Understanding training of these feedforward fully connected architectures, where gradients are computed via the famous *backpropagation algorithm*, covered in Chapter 5 (after automatic differentiation is introduced in Chapter 4), is key to understanding the essence of deep learning. Fully connected networks are also useful for understanding key ideas such as *dropout*, *batch normalization*, and *weight initialization*, which are all at the heart of deep learning.

Autoencoders

A simple *fully connected deep autoencoder architecture* is illustrated in Figure 1.3 (b). Shallow variants of this paradigm are studied at the end of Chapter 3. In many cases one may wish to use the internal representation of the architecture to extract meaningful features from the inputs. This is called *feature extraction* and we call the outputs of this process *computed features* or *derived features*. After carrying out feature extraction, the computed features may then be transformed into outputs, used for clustering the data, or used for other manipulations of data.

Variants of architectures that make use of feature extraction fall under the name of *encoders* since they encode inputs into computed features; *decoders* since they decode computed features to outputs; as well as *autoencoders* since these architectures combine the tasks with an aim of having an output which matches the input. Particularly with an autoencoder, the goal is to find parameters θ, such that $x = f_\theta(x)$ is (approximately) maintained. The application of encoders, decoders, and autoencoders to different tasks is omnipresent in deep learning architecture design. There are hundreds of applications and variations with a few applications outlined at the end of Chapter 3.

Convolutional Neural Networks

The VGG19 model used in the discussion of Section 1.1 is one example of a *convolutional neural network architecture*. This class of models is illustrated in Figure 1.3 (c) and is covered in detail in Chapter 6. These types of models constitute the most famous specialization of fully connected neural networks. Convolutional models are primarily used for image analysis and it is fair to say that their recent success has shuffled the cards in the broad field of *image processing*. Beyond images, these models can be adapted for other domains such as radiology data or audio. As alluded to in Section 1.1, in the context of images, there are multiple related tasks including classification, semantic segmentation, and localization, and all of this can be handled via convolutional neural networks.

Convolutional neural networks can be viewed as adaptations of fully connected networks, where the action of each layer is not based on the full connections between activations but rather on smaller trainable convolutions. These convolutions maintain a spatially homogenous structure in the network. Such a setup significantly reduces the number of parameters, enables deeper architectures, and most importantly capitalizes on spatial relationships present in the input. As a consequence, for a similar size of θ (e.g., 144 million parameters as in the VGG19 model), one may have a much deeper architecture than would have been possible with a fully connected network. This results in training that is much more efficient and the model is more efficient in production as well.

In addition to the core *trainable convolutions* idea, convolutional networks introduce additional architectural concepts such as the use of *channels* and the use of *pooling*. Huge leaps with convolutional neural networks were made during the first half of the second decade of this century. The incredible success of the so-called *AlexNet* model in the 2012 ImageNet challenge boosted neural networks within the world of machine learning. This in many ways started the deep learning revolution and put deep learning at the forefront of ML after the field was on the sidelines for many years.

Recurrent Neural Networks, LSTMs, and GRUs

Figure 1.3 (d) illustrates a recurrent neural network where on the left side of (d) we see the basic architectural components and on the right side we see what is called an unfolded representation of the network, illustrating its recursive operation. While key advances during 2010–2015 were in the convolutional domain focusing on images, the second half of that decade witnessed deep learning becoming an integral part of *natural language processing* (NLP). By today, automatic translation engines, language generation models, and other solutions for tasks associated with text almost always involve deep neural networks or are entirely based on deep learning models. The use of recurrent neural networks is the most rudimentary modeling paradigm for such purposes. These models are covered extensively in Chapter 7.

There are multiple variations of recurrent neural networks where the most basic one is illustrated in Figure 1.3 (d). However, the internal structure can vary and some popular and powerful variations include *long short term memory* (LSTM) models and *gated recurrent unit* (GRU) models. These architectures, also surveyed in Chapter 7, have been very impactful. In addition to NLP, there are many other domains where such models for sequence data is a natural choice. These domains include genomic sequencing, multivariable time series, audio, and even video.

Transformers and the Attention Mechanism

It is probably fair to say that the greatest advances in deep learning at the start of the third decade of the current century are centered around *large language models*. These systems, including the highly popularized *ChatGPT* service, among others, are able to execute many language tasks and are reshaping our view on intelligence at large. To date, the underlying model in most of these systems is the *transformer architecture*. Figure 1.3 (e) is a loose sketch of how such an architecture engages in the task of *machine translation*, i.e. translating from one human language to another. The reader should keep in mind that the illustrated encoder and decoder blocks are each composed of multiple sub components (not appearing in the figure). Some of these components include feedforward layers as in Figure 1.3 (a), and other components are based on a concept called the *attention mechanism*. The full details of the attention mechanism and transformer models are in Chapter 7.

Large language models that use transformers appear to comprehend and generate human-like text. They engage in natural language understanding, extracting information from textual data, answering questions, and providing contextually relevant responses. Machine translation, our core example activity of Chapter 7, can also be handled by large language models. Beyond machine translation, large language models are versatile as they can handle multiple tasks including summarization of text, facilitating efficient communication across diverse languages, and more. Recent advances also include *multimodal models* which handles text, images, and other formats both as input and output.

Diffusion Models and other Variational Autoencoders

In Figure 1.3 (f) we see a schematic of a diffusion model which here simply appears as a process of either adding noise to an image in an *encoder* or alternatvily removing noise from an image with a *decoder*. The overarching idea of a diffusion model is to learn not just how to add noise, but also how to create an image out of noise. With this, a trained decoder can generate realistic looking images that are actually random. Diffusion models recently arose as extremely powerful image generation models and are able to generate images that are both realistic looking and highly creative in their style and nature.

Diffusion models and their generalizations are probabilistic in nature. The complete details are in Chapter 8 where we first describe *variational autoencoder* models, then modify them to a class of models called *hierarchical Markovian variational autoencoders* of which diffusion models are a special case.

Generative Adversarial Networks

A *generative adversarial network* (GAN) architecture is illustrated in Figure 1.3 (g) and introduced in Chapter 8. Like diffusion models, GANs are very useful for creating random data that is realistic in nature. The rise and popularity of GANs predated that of diffusion models and today GANs and diffusion models compete for the state of the art in artificial data generation. GANs and diffusion models differ in their architecture and analysis. While diffusion models are probabilistic, the analysis and study of GANs is close to the field of *game theory*.

The key idea of a GAN is to simultaneously train two deep neural networks, a *generator* and a *discriminator*. The former generates fake data, while the latter attempts to determine if the data is `fake` or `real`. As the training of both of these networks progresses, the generator is ultimately able to fool the discriminator and as a consequence, it also creates "real looking" data. Much of the choice of architecture is then with finding measures of the quality of the data in the discriminator as well as with the algorithms for jointly training these networks.

Deep Reinforcement Learning

In Figure 1.3 (h) we illustrate the paradigm of reinforcement learning. Here the basic setup is that a system, or *environment*, is controlled by an *agent*. For example, one may think of the environment as a home, and the agent as a cleaning robot traversing and cleaning the home. As time progresses, the agent makes decisions in the form of *actions*, for example `move right 5 cm`, and these are interfaced with the environment. The agent in turn receives *reward* from the environment as well as *observations*, where the reward is a mechanism that helps to drive towards better goals, for example "cleaning in a quick and energy efficient manner", and the observations can include sensory input. The goal of reinforcement learning is to develop meaningful ways for the agent to choose actions.

One of the great leaps of AI during the second decade of this century is in the game of Go; see references at the end of this chapter. This strategic board game was long considered much more difficult "to program" in comparison to other games such as Chess.[6] Yet in 2015 a team from DeepMind through a series of advances and competitions designed a system called AlphaGo which beat the world's best Go players. This highly publicized achievement made the dream of artificial intelligence a bit more concrete by showing the ability of neural networks to solve complicated tasks. The key ideas of this achievement are from the field of *reinforcement learning*. We outline basic ideas in Chapter 8.

Graph Neural Networks

An additional category of neural network models that we explore in Chapter 8 are *graph neural networks*. These models operate on graphical structures, i.e., nodes and edges with attributes. Graph neural networks are suitable for social networks data, for the study of chemical compounds, and for many other applications where relationships between entities are well described via graph structures. In contrast to other types of neural networks, graph neural network models are often not directly used to try and mimic human level performance but rather for discovery and insight within data.

1.3 Key Ingredients of Deep Learning

Having gotten a taste of deep learning, we now discuss the key ingredients that leverage its success. These are notably the availability of large datasets, advances in computer architectures, advances in software, the internet, and the interplay of cognitive science and artificial intelligence research. While the remainder of the book focuses on the mathematical description of deep learning, this section aims to overview the non-mathematical key ingredients which are attributed to the success of the field.

[6] Computers have shown their superiority in the game of Chess since the mid 1990's with a notable victory of the Deep Blue Chess playing expert system defeating the champion Garry Kasparov over a six-game match in 1996.

Neural Networks as Artificial Brains?

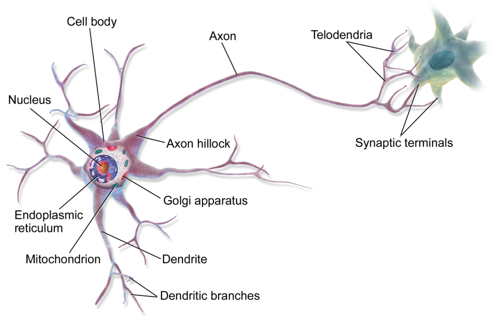
Brains are composed of (biological) neurons that are interconnected in unstructured ways; see Figure 1.4 where display (a) illustrates a single biological neuron and display (c) illustrates an interconnected network of biological neurons. A human brain has an estimated 85 billion neurons. A single human action such as movement of an arm may induce the firing of around 80 million such neurons, whereas the identification of a visual object may use the bulk of the estimated 150 million neurons that are in the *visual cortex*. Unquestionably, brains are fascinating organs whose scientific understanding, while still at its infancy, will undoubtedly grow in the years to come and have profound affects on human endeavour.

Deep neural network models are neither brains nor attempts to create artificial brains. Nevertheless, the development of these models is highly motivated by the biological structure of the brain. The basic building block of a deep neural network model is the (artificial) neuron abstracting the synapse connection between neurons via a single number called an activation value. See display (b) of Figure 1.4 for a single (artificial) neuron and display (d) which presents a combination of multiple neurons as part of a feedforward (artificial) neural network similar to Figure 1.3 (a). Pioneering and landmark work in AI research was inspired by neuroscience since brains are essentially the only complete proof we have for the existence of what we call “general intelligence”. Further, many tasks of deep learning models involve the mimicking of human level (or animal level) tasks such as understanding images or conversational tasks. Thus for example one of the most well-known benchmarks in the world of artificial intelligence is the *Turing test*, originally named the *imitation game* when introduced by Alan Turing in 1950. It is essentially a test to see if a computer can engage in long conversation with a human, without another observing human distinguishing between the computer and the human.

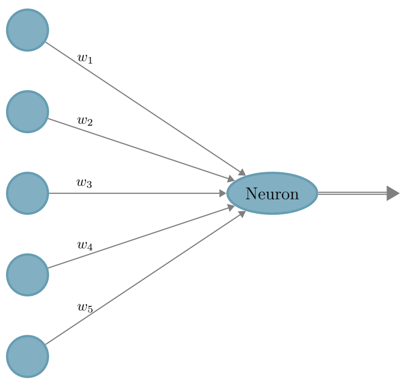
At the time of publishing of this book, the state-of-the-art *large language models* are on the verge of passing the Turing test and in fact, researchers are seeking alternative more suitable criteria. This is because while the test may appear to be (nearly) achieved, it is still believed that these language models do not constitute general intelligence. In fact, *artificial general intelligence* (AGI) systems are at best at their infancy. Deep learning models generally only achieve *narrow tasks* such as pattern recognition, conversational agents, or playing specific games, as opposed to general intelligence tasks of creative problem solving. Nevertheless, large language models of 2024 and onwards appear to be very powerful in multi-modal activities.

With both the fields of neuroscience and AGI still awaiting major breakthroughs, it is natural for researchers to continue to draw parallels between neuroscience and artificial intelligence. On the cognitive sciences side of the spectrum an increasing number of researchers are making use of artificial neural networks as abstractions for understanding cognitive tasks. However on the AI side of the spectrum, while in the early years, neuroscientific-motivated models were central, today they have become more of a niche research area. In our context, the mathematical description of deep learning in this book is completely agnostic to biological brains.

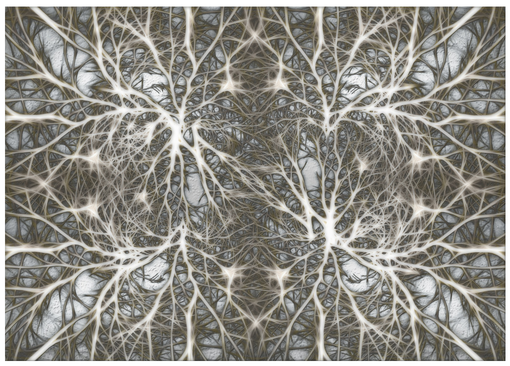
[7] Image (a) is attributed to B. Blaus under the creative commons license and available via Wikimedia Commons. Image (c) is sourced from pixabay.com.



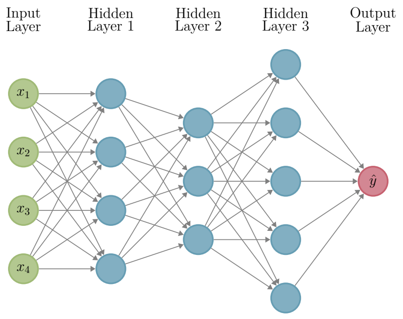
(a)



(b)



(c)



(d)

Figure 1.4: Biological and artificial neurons and networks.[7] (a) A single biological neuron. (b) A neuron in an artificial neural network. (c) Connection of multiple biological neurons in a brain. (d) An artificial neural network connecting multiple neurons in a feedforward structured manner.

Computational Power

It is well known that deep learning models require fast computers with plenty of memory. Training deep learning models can take hours or days using current state of the art hardware and would not have been practically possible on machinery of the 1990's or earlier. Similarly, the large scale application of (trained) deep learning models, such as for example in self driving cars, requires massive resources as is attested by the fact that the power consumption for computing in a self driving car can sometimes equate or exceed the power used by the engine. At around the first decade of the 21st century a technological threshold was passed and the availability of fast computing hardware made earlier deep learning ideas realizable and successful.

A complete description of hardware advances and their relationship with deep learning is beyond our scope and is not the focus of this book. Indeed, practical machine learning engineers often need to consider the computing power and hardware at play to train or implement effective deep learning models. A core component that has made a huge difference to the field is the development and availability of *graphical processing units* (GPUs). In contrast to standard *central processing units* (CPUs) which are optimized for logical operations, branching, and general computations, GPUs are optimized for repetitive large scale matrix operations and can execute deep learning training or prediction in the order of 20 to 200 times faster than state-of-the-art CPUs. The GPU industry initially grew due to demand from the video gaming market, however by 2020 their importance in the AI revolution is well understood. Today the needs of deep learning influence the design and development of future generation GPUs. In summary it is fair to say that without GPUs, deep learning would not be anywhere where it is today.

For example, the aforementioned success of the so-called *AlexNet* model in the 2012 ImageNet challenge, was based on a neural network model specifically designed to be trained on two parallel GPUs which were the state of the art of the time. From a programming perspective, utilizing such GPUs required considerable effort at that time but since then they have become much more accessible with better software. In the past decade, more specialized computing systems, including GPUs, the similar *tensor processing units* (TPUs), and dedicated driver software were specifically adapted for deep learning applications. Indeed today, a machine learning engineer engaging in deep learning almost always needs to make use of such tools. Also central in this arena is the availability of *cloud computing*. Today many deep learning applications use dedicated cloud computing services both for training and model application.

For light applications including *scientific machine learning* with small datasets, or for pedagogical purposes, one may often use non-GPU machines (as well as laptops). See the notes and references at the end of the chapter for recommended reading of applied deep learning.

Large Datasets

Deep learning models work exceptionally well in cases where input data has a lot of features, a setting that is loosely called *high dimensional*. In a more statistical context a proper definition of high dimensionality is that the number of features exceeds the number of datapoints. However, in deep learning, such a distinction is not common. Large datasets with many samples are ubiquitous and often critical to the success of deep learning.

Large enough and annotated datasets were rarely available prior to the turn of the century, yet in recent times humanity has witnessed an explosion of the volume of data stored. As a general guide consider Figure 1.5 showing a projection of the total stored data on earth. While clearly not all of this data is open, available, and suitable for deep learning models, this trend in total data storage is also characteristic with data suitable for deep learning models.

A key example which was pivotal in the success of the field is the aforementioned ImageNet database which contains nearly 15 million images where early deep learning models were trained with 1.5 million annotated images out of the total. Another example is with large scale text models trained on all of Wikipedia which as of early 2024 includes over 59 million

[8] As there is not one credible openly available data source, we crowd sourced estimates via search results for a few years that had estimates searchable via Google. With this, we obtained estimates for the years, 2006, 2007, 2010, 2012, 2018, 2021, and 2022, some of which were for the total data stored and some for the data generated during that year. We then fit an exponential model to the data under a few minor additional assumptions. Details of our extrapolation are in the source code notebook available through `https://deeplearningmath.org/`.

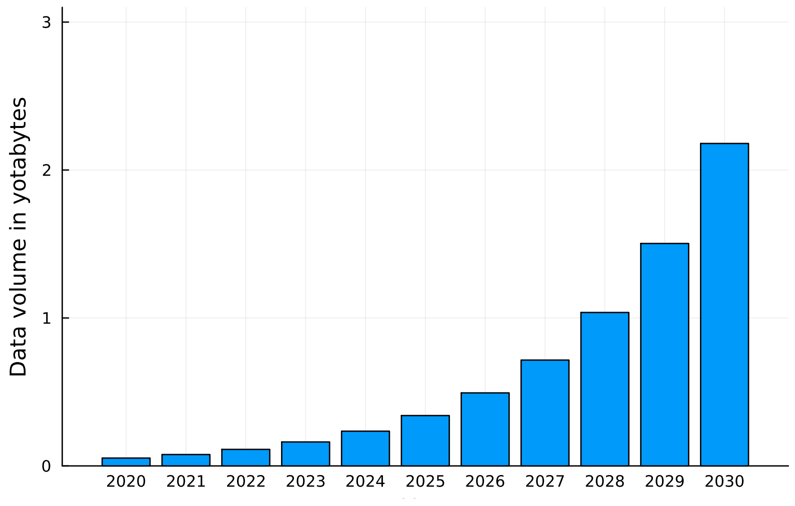


Figure 1.5: Predictions of the world's total data storage during the third decade of the 21st century. The graph is in yotabytes where each yotabyte is 2^{80} bytes or approximately 10^{24} bytes. Note that at the time of publishing of this book, estimates still use the zetabyte unit (a zetabyte is 2^{70} bytes, or 1/1024 of a yotabyte). Current estimates at the time of publishing of the book are at around 150 zetabytes. The predictions in this plot are speculative and are based on a simple extrapolation that we carried out.[8]

pages with a compressed text size of about 22 gigabytes. Up to the turn of the century, such datasets were much harder to come by, were rarely openly available, and disk sizes of most computer workstations were often too small to accommodate them. However in recent years the availability of plenty of rich datasets has been pivotal for the success of deep learning. More information on annotations and popular datasets is in Section 1.4 below.

The Internet, Software Practices, and Open Source

In addition to fast computation and the availability of large datasets, the recent success of deep learning was also fueled by new software development practices that evolved around the proliferation of the internet. Up to the last decade of the 20th century, most software developments involved relatively isolated groups working in companies, in research groups, or individually. As such, there was not much sharing of ideas, information, packages, and modules. However, by the end of the first decade of the current century, global collaborative community practices solidified and eventually resulted in (generally) free services such as Stack Overflow, GitHub, GitLab, Kaggle, and many more which today are a natural part of software development and data science culture. A key attribute of these new services is that they incentivize individuals (and groups) to share their ideas and source with the global community. These developments went hand in hand with the growth of the open source ethos which is shared and respected by many.

By the middle of the second decade of the 21st century, as the strength of deep learning became evident, new age collaborative global software practices were already quite mature. The timing of these events greatly helped the deep learning revolution as it allowed thousands of contributors around the globe to develop software, supporting documentation, and examples suited for deep learning applications. As a consequence, within a period of a few years, deep learning software frameworks such as the now popular TensorFlow, PyTorch, Keras, fast.ai, Flux.jl, and others became available, quickly matured, and are now widely used. By today, new deep learning ideas stemming from research are often published together with open source software. As a consequence, machine learning engineers and other users of deep learning, are able to easily and quickly use of the state of the art models and methods.

All of these practices are key ingredients not just of deep learning, but of data science, and software development at large. However, in terms of the deep learning revolution, the timing of events was just right.

1.4 DATA, Data, data!

Effective training of deep learning models requires large datasets. We now present examples of various forms of data as well as a few popular datasets that are used for educational purposes, for training of real models, or for benchmarking. Our focus here is on *annotated* datasets typically used for *supervised learning*, a concept discussed at greater depth in Chapter 2. These are datasets consisting of *feature vectors*, each denoted by x, and associated *labels*, each denoted by y.

One should keep in mind that single data values are typically either *numerical* or *categorical*. The latter is the case when the values come from small discrete sets. In some cases categorical data has order such as for example the level of customer satisfaction which may be recorded in the range `unsatisfied`, ..., `very satisfied`. These are called *ordinal categorical variables* and in certain cases they may be directly converted to numerical variables which encode the order. However, other categorical variables such as `banana`, `car`, etc. have no specific order. These are called *nominal categorical variables* and they are typically treated by expanding the values into unit vectors. More on dealing with such data is in Chapter 2.

Here are a few generic examples for the features x and labels y. In some cases there is a natural *dimension* to the data, such as (mono) audio being, one dimensional, black and white images being two dimensional, and color images being three dimensional. When treating this data mathematically, we often attempt to represent x as a one dimensional vector of length p, where p is the number of features.

Audio recording: A simple representation of audio is the vector of amplitudes of the recording at regular intervals. For example at $44,100$ samples per second is a common sampling rate for high quality audio. The amplitude can be a positive or negative number.[9] Hence for example, an audio recording taking a snapshot of music for 5 seconds will be a vector x with $p = 220,500$ entries (features). An associated label y, in case this is music audio, may be the genre of the music such as `jazz`, `hip-hop`, etc.

A monochrome image: Here consider an image of p_1 by p_2 pixels with each pixel signifying the intensity e.g., 0 is black and 255 (or some other maximal value) is white. We can consider the image as a $p_1 \times p_2$ matrix and the vector x of length $p = p_1 \cdot p_2$ as a vectorized form of the matrix using either a *column-major* or *row-major* representation of the matrix. As an example consider a 200×100 (portrait) image. In this case the vector x has $p = 20,000$ features (pixels). A simple example for y may be an indication

[9] Often such amplitudes are two bytes each, or 16 bits each, and this means that they obtain values from a finite range of $2^{16} = 65,536$ possibilities.

of the class of the image, such as `banana`, `car`, etc.. Alternatively y may represent a bounding box inside the image where the object we are looking for is localized.

A color image: Here each pixel is not just an intensity but rather an RGB (Red, Green, Blue) 3-tuple. One way to represent this image is via a 3-tensor which is $3 \times p_1 \times p_2$ dimensional. A vectorized image would be a $p = 3 \cdot p_1 \cdot p_2$ dimensional vector. Note that while color images are typically stored in a compressed format such as JPEG, for deep learning purposes images are typically considered in a bit map format as described here.

A text corpus: Text encoded in the ASCII format, the Unicode format, or other means is a sequence of characters. However treated as a datasource for deep learning we may sometimes break up the text into words (or tokens), associate a unique vector with each word, and represent the text as a sequence of vectors. Here y may be the associated text in a different language, or it may be a level of *sentiment* of the text which is a number between -1 and 1 indicating the text is of a negative tone (angry, critical, etc...), or is of a positive tone.

Heterogeneous datasets: In many cases the features are heterogeneous in nature as one would expect for example in the case of individual customer records in a database. For such a case, each customer can be associated with dozens, hundreds, or thousands of features, either numerical or categorical. Here y may be a continuous variable indicating the propensity (probability) of a customer to leave the service.

A Few Popular Datasets Examples

In most cases, datasets are created using a *manual annotation process* where after the x's are collected, standardized, and cleaned, human annotators look at each x and assign the matching value to y. Needless to say, the process of annotating datasets is generally time consuming and expensive. The deep learning revolution was sparked by the creation of a few large annotated datasets and to date, the process of annotating datasets for new purposes is often a costly barrier for engaging in new deep learning ventures or research.

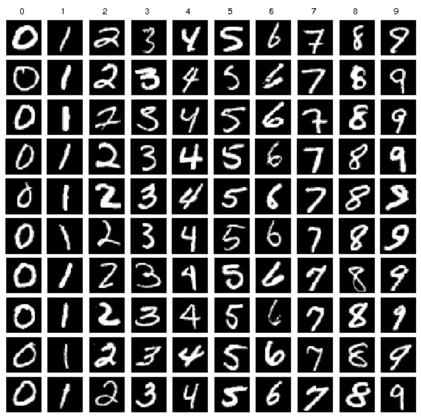
We now list a few selected notable datasets that are likely to appear in elementary deep learning practical examples, in research papers, and are sometimes also used professionally. See also Figure 1.6. Of the examples which we list here, the ImageNet database is the most industrially applicable dataset since many vision models are trained via this dataset and can then be ported to other models using transfer learning.

Some datasets break up the data a-priori into some random but fixed partition between a *train set* and a *test set*. This, in principle, allows one to benchmark models by training them on only the training set and evaluating performance on the test set. Such practices are discussed in greater detail in Chapter 2.

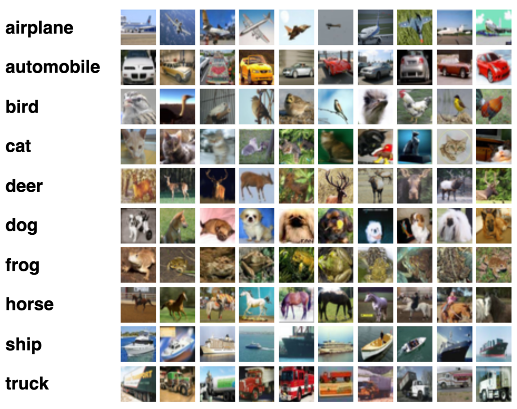
MNIST[11] digit images: One of the most basic machine learning datasets is the *MNIST database*. In this dataset, each sample x is a 28×28 black and white image of a handwritten digit, and when vectorized it is a $p = 784$ long vector. Each label y is

[10]Image (b) is sourced from the website https://www.cs.toronto.edu/~kriz/cifar.html. Image (c) is thanks to A. Krizhevsky, I. Sutskever, and G. E. Hinton, sourced from "ImageNet Classification with Deep Convolutional Neural Networks", [239]. Image (d) is from https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews.

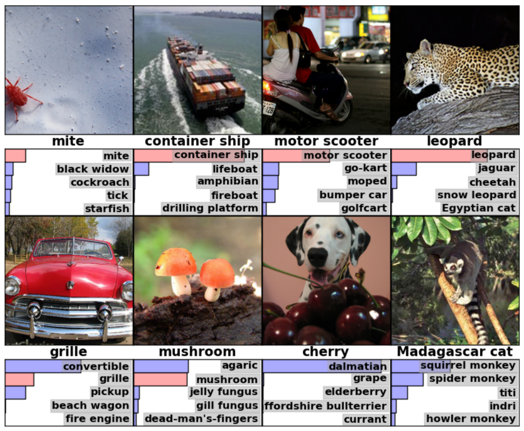
[11]Modified National Institute of Standards and Technology.



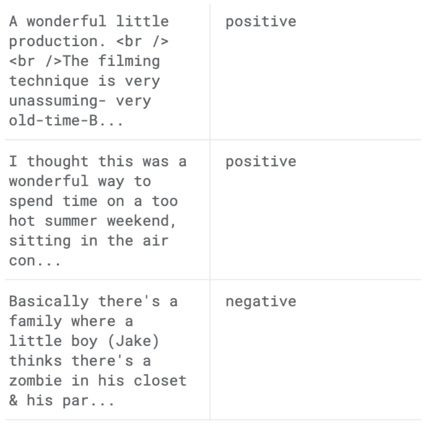
(a)



(b)



(c)



A wonderful little production. <br /><br />The filming technique is very unassuming- very old-time-B...	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his par...	negative

(d)

Figure 1.6: Some popular datasets.[10] (a) Handwritten digits from the MNIST dataset. Each digit is 28×28 pixels. (b) The CIFAR-10 dataset comprised of 32×32 color images from 10 classes. (c) Some images from the ImageNet dataset together with the top-5 classified labels. The correct label is in pink. (d) An extract from the IMDB movie reviews dataset. Each review is labeled as either `positive` or `negative`.

an element from `0`,`1`, ..., `9` indicating the digit. The distribution of digits is *balanced*, meaning that there are approximately the same number of images for each digit (class). The dataset is broken into a train set comprised of 60,000 images, and a test set with an additional 10,000 images.

It is fair to say that this dataset does not have much industrial value, but is rather used for educational and academic purposes. Training a machine learning model to classify MNIST images is one of the most basic practice tasks one can consider. Further, many machine learning papers use MNIST to illustrate ideas and benchmarks.

A similar dataset with the exact same characteristics, is the *fashion MNIST dataset* where the images and labels are `shirt`, `shoe`, etc... and the dimensions of the data are set exactly like MNIST. The digit MNIST can be trained for over 99.8% accuracy with state of the art models, yet fashion MNIST is slightly more challenging and can be trained to achieve around 96.5% accuracy.

CIFAR[12] - 10 small color images: Similarly to MNIST, the *CIFAR-10 dataset* has small images (32×32 pixels in this case) each broken up into 10 classes, namely `airplane`, `frog`, etc... as appearing in Figure 1.6 (b). However, the images are in color and hence $p = 3 \times 32 \times 32 = 3,072$. Here there are $50,000$ train images and $10,000$ test images and like MNIST this is a balanced dataset between classes. This dataset is popular with research papers and is occasionally used as a benchmark dataset. State of the art models achieve around 99% prediction accuracy.

ImageNet: The *ImageNet database* is undoubtedly the most popular dataset for deep learning benchmarking, as well as for industrial use in image analysis. It was created at the end of the first decade of this century and coupled with the associated *ImageNet challenge*[13]. The challenge uses $1,000$ non-overlapping image classes. ImageNet has nearly 15 million color images with varying resolutions and an average resolution of around 470×385. The images are labelled very specifically to more than $20,000$ categories where some images are also labelled with bounding boxes around objects.

In 2012 the AlexNet deep learning model achieved a top-5 accuracy of nearly 85%, meaning that in 85% of the tested images the correct label out of the $1,000$ labels is one of the top-5 predicted probabilities. See also Figure 1.6 (c) where the top 5 label probabilities are presented for a few example images with the correct label marked in pink. More on top-5 accuracy and other performance measures is in Chapter 2. By today, state of the art performance for top-5 accuracy is over 99% and for top-1 (or absolute accuracy) state of the art is just over 90%.

IMDb[14] movie reviews: This textual dataset has $50,000$ movie reviews. It is split into $25,000$ for training and the remainder for testing. For each review an indication of either `positive` or `negative` indicates the sentiment of the review. The typical task with such a dataset is to predict the sentiment for an unseen review. This dataset is often used as a practice dataset for introductory tasks within *natural language processing* (NLP).

Beyond these datasets there are now thousands of additional quality available open sourced datasets, some of which are only really useful for practice or experimentation, while others, like ImageNet have real industrial value. Deep learning frameworks and specialized libraries often come with example datasets, and Kaggle is considered as the most popular general source of openly available datasets since it also holds competitions and educational activities associated with the data.

1.5 Deep Learning as a Mathematical Engineering Discipline

While computing power and the abundance of data are key to the success of deep learning, the importance of mathematics cannot be underplayed. We now explain the term **mathematical engineering** and justify its use in the book's title.

[12] Canadian Institute For Advanced Research.
[13] More formally known as ILSVRC (ImageNet Large Scale Visual Recognition Challenge).
[14] IMDb is short for Internet Movie Database. The dataset was made publicly available by the authors of [275].

Human engineering of systems has relied on mathematics almost since the dawn of time. The pyramids of Egypt could not have been constructed without some prior geometric calculations and later down the road, the machines that drove the industrial revolution were designed with the aid of trigonometry, algebra, calculus, and other mathematical techniques. For such reasons and others, the term **engineering mathematics** is often used to describe the combination of calculus, linear algebra, basic differential equations, Fourier analysis, Laplace transforms, and many other mathematical tools that can interplay to help design and develop physical engineered systems. So why does this book use the permuted term **mathematical engineering**?

Our answer lies in the fact that in the field of deep learning, mathematics is used directly as an engineered component. When one designs an electrical circuit the flow of electricity is engineered and hence the field is called **electrical engineering**. When one designs a robot arm, the mechanics are directly considered and hence the term used is **mechanical engineering**. What about the design of a deep neural network for deep learning? The specification of deep learning models is a mathematical specification about the flow of data through a combination of affine functions, non-linear functions, and at times other mathematical components. Hence the design of deep learning models is the act of **mathematical engineering**.

To further understand the phrase **mathematical engineering** consider the following display borrowed from Chapter 5. It represents an action that is at the heart of deep learning:

$$\begin{tikzcd} a^{[\ell-1]} \arrow[r, "\text{Affine Transformation}"] \arrow[rr, bend right, "f^{[\ell]}_{\theta^{[\ell]}}"'] & z^{[\ell]} := W^{[\ell]}a^{[\ell-1]} + b^{[\ell]} \arrow[r, "\text{Activation}"] & a^{[\ell]} := S^{[\ell]}(z^{[\ell]}). \end{tikzcd}$$

The function $f^{[\ell]}_{\theta^{[\ell]}}$ represents the mathematical action of a single layer of a deep neural network on neurons/activations/inputs $a^{[\ell-1]}$ to obtain neurons/activations/outputs $a^{[\ell]}$. It involves an affine transformation to reach an intermediate value $z^{[\ell]}$ via a matrix multiplication by the (weight) matrix $W^{[\ell]}$ and an addition of a (bias) vector $b^{[\ell]}$. It then involves the application of a non-linear (activation) function $S^{[\ell]}$. The trainable parameters of the layer are $W^{[\ell]}$ and $b^{[\ell]}$ and their combination is denoted via $\theta^{[\ell]}$. A deep neural network $f_\theta(\cdot)$ is then a functional composition of many such layers, say L,

$$y = f_\theta(x) = f^{[L]}_{\theta^{[L]}}(f^{[L-1]}_{\theta^{[L-1]}}(\ldots(f^{[1]}_{\theta^{[1]}}(x))\ldots)),$$

where $x = a^{[0]}$ and $y = a^{[L]}$. The parameters of the whole model, θ are comprised of the individual parameters of the layers $\theta^{[1]}, \ldots, \theta^{[L]}$.

Some aspects of the mathematical engineering of deep learning involve choosing the non-linear functions $S^{[\ell]}(\cdot)$ and other aspects involve defining the dimensions of $W^{[\ell]}$ and $b^{[\ell]}$ as well as any sparsity structures in these. Importantly, deep learning training is essentially the iterative solution of an optimization problem where the decision variables are the many components of θ. For such a problem, in addition to specifying efficient optimization algorithms, one needs to determine the optimization objective which in the language of deep learning is called the loss function. All these activities and many more comprise the **mathematical engineering of deep learning**.

The Mathematics Used

While deep learning is "deep", the application of mathematics in the field is relatively shallow. The mathematical engineering of deep learning is mostly based on linear algebra, multivariate calculus, and basic probability. It thus uses mathematics at a level comparable to the first two years of university studies of an engineering degree or a similar field. In fact, even within these fields, one mostly requires only elementary operations and there is not much reliance on theoretical results. For example in terms of linear algebra, matrix multiplication operations are key, but vector spaces, eigenvalues, and matrix decompositions are mostly not essential. Similarly in terms of calculus, derivatives and their multi-variable counterparts are central to the field, but integration, vector fields, manifolds, or properties of real functions are mostly not used. Finally in terms of probability, one simply needs to account for basic probabilities and occasionally evaluate expectations or variances for random variables.

This accessibility of deep learning has motivated pedagogical approaches focusing on a practical coding perspective as in "Deep Learning for Coders with fast.ai and PyTorch" [190] as well as many other resources; see also the notes at the end of this chapter. In contrast to such code-centric approaches, our approach relies on mathematical notation as a means of communicating ideas. With our approach, even though theoretical mathematical results are mostly not essential, mathematical notation plays a central role in conveying ideas concisely. See also Section 1.6. where we introduce basic notation for the remainder of the book as well as a few supporting mathematical results summarized in the appendices.

Development and Investigation of Deep Learning via Advanced Mathematics

It is important to note that while the mathematics used in this book is quite simple, many of the models and techniques that we present have advanced theoretical underpinnings. These advanced counterparts have often played a role in the development of the current simple models. One such case which will become evident in Chapter 3 is the logistic regression model. With this model one can analyze the model as either a simple neural network, or alternatively position it as a statistical model leveraging on the (slightly more advanced) theory of maximum likelihood estimation (MLE). Interestingly, historically logistic regression was first developed using MLE and only later appeared as a basic machine learning model. Similar trends in the machine learning community are also in association with support vector machines (SVM), a topic that we do not cover further in this book. The theory and application of SVMs hinges on beautiful mathematical results from functional analysis and their popularity has certainly inspired multiple ideas in deep learning. Nevertheless, deep learning models can be understood without considering SVMs. Other such examples where more advanced mathematics are lurking behind the scenes is in our consideration of optimization algorithms in Chapter 4. In this context, while most of todays methods are simple first-order optimization methods, the path to realize that (at the moment) these techniques work best has also gone through the study of much more advanced second-order optimization techniques and their variations.

We also mention that there is ongoing research on the theoretical properties of deep neural networks. In this domain researchers make use of functional analysis, measure theory, information theory, stochastic processes, differential geometry, topology, category theory, and other mathematical fields to describe and understand theoretical properties of deep learning. While such research efforts are very exciting, they are not the topic of this book.

1.6 Notation and Mathematical Background

It is assumed that the reader has an understanding of basic mathematical notation and operations including sets, function notation, basic probability, and basic matrix algebra. Appendix A reviews key results from multi-variable calculus that are used, but other basics are assumed known. We now highlight a few notational elements that we use throughout and you may also consult the notes and references at the end of this chapter for suggestions of a few resources that may help with a review.

Vectors are in general considered as elements of $\mathbb{R}^n$ with n being some positive integer. We can denote a vector $u \in \mathbb{R}^n$ as $u = (u_1, \ldots, u_n)$ where the scalar u_i is the i-th coordinate of the vector. Written in this *tuple* form, we consider the vector as a column vector. That is, we may consider for example a matrix $A \in \mathbb{R}^{m \times n}$ and then have the matrix-vector product $v = Au$ with $v = (v_1, \ldots, v_m) \in \mathbb{R}^m$, and each v_i given via,

$$v_i = \sum_{j=1}^{n} A_{i,j} u_j.$$

An alternative way to represent a vector is using square brackets in which case we take its orientation literally. For example $w = [w_1 \ \cdots \ w_m]$ is a row vector and its transpose, $w^\top$ is a column vector and an element of $\mathbb{R}^m$. In this case, we may, for example consider the vector-matrix product $x = wA$ which yields x, an m dimensional row vector. Hence in summary, with our vector notation for $u \in \mathbb{R}^n$ we have,

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = [u_1 \ \cdots \ u_n]^\top \equiv (u_1, \ldots, u_n).$$

Some key vectors we consider are $\mathbf{1} \in \mathbb{R}^n$ which is a vector of all 1's. Similarly 0 can be taken as a vector of all 0's. However, the same symbol, 0, is clearly also used for scalars, and matrices with the actual meaning clear from context. There are also the unit vectors e_i with dimension taken from context and $i = 1, \ldots, n$ in case they are n dimensional. Each such unit vector has 0 entries everywhere except for 1 in the i'th coordinate. So as an example of this notation, see that the identity matrix $I \in \mathbb{R}^{n \times n}$ can be represented as a sum of outer products,

$$I = \sum_{i=1}^{n} e_i e_i^\top.$$

The norm $\|\cdot\|$ is often used and unless stated otherwise it is the vector L_2 norm. That is, for $u \in \mathbb{R}^n$, $\|u\| = \sqrt{u^\top u}$. Similarly $\|u\|^2 = u^\top u$.

Surprisingly, beyond matrix vector arithmetic, norms, and a few basic operations, we only seldom use linear algebra machinery such as eigenvalues, and the singular value decomposition. This is because most of the book is focused on the basic mathematical engineering of deep learning which at its core, does not use very sophisticated mathematics but rather employs basic mathematical concepts to create sophisticated models. We thus leave it to the reader to seek a self-refresher on such topics as they arise. For example, some of our discussion of unsupervised learning in Chapter 2 builds on understanding of linear algebra. However, these topics are in general tangental to the core material of the book.

In terms of concepts of optimization theory, Chapter 4 covers these in a self contained manner that does not require further background beyond a few multivariate calculus results, reviewed in Appendix A. Hence for example, convexity is briefly introduced and summarized directly in Chapter 4. At certain points, the reader may want to dig deeper and review topics individually elsewhere.

Finally, in terms of probabilistic concepts, it is assumed the reader is aware of the informal definition of probability $\mathbb{P}(\cdot)$ and basics of random variables. This also includes the expectation $\mathbb{E}[X]$ of a random variable X, the variance, denoted $\text{Var}(\cdot)$, and covariance. Similarly to the linear algebra concepts, we do not make heavy use of probability theory, however certain probabilistic computations occasionally arise.

Notes and References

Since this book is purposefully agnostic both to implementation issues and to the historical account of deep learning, we use an unnumbered section such as this at the end of every chapter to provide key notes and references. More additional information is also available via the book's website.[15] The website also includes links to articles and references, tutorials, videos, and mathematical background.

In terms of implementation, at the time of writing this book, almost any introductory (or advanced) work in deep learning uses the *Python* language. See for example [342] for a general intermediate guide to Python. Neat examples in Python for general machine learning are in [240], and that resource can also serve as a comprehensive mathematical introduction to the whole field of machine learning and data science. Another language slowly stepping into the spotlight is *Julia*, see [303] for a statistical introduction to the language and [338] to get a taste for scientific machine learning, where Julia is often used. Further, the *R statistical computing system* has a vast user base in statistics and supports tens of thousands of packages as well as multiple interfaces to deep learning frameworks. See for example [245] for a classical introduction to statistics with R and [10] for ways of using R with deep learning. Until around 2015, programming in *C/C++* for deep learning was the norm. However, these days it is much more of a speciality and is only required in very specific cases. See for example [319] for a classical text involving pattern recognition and C/C++. Paid language platforms such as *Mathematica*, *Maple*, and *MATLAB* are attempting to retain some of their user base by incorporating deep learning libraries. However to date, these commercial systems have shown to be much less mature with deep learning in comparison to the open source arena.

Beyond the computer languages in use, a key aspect of deep learning is the usage of deep learning frameworks. These are software packages for training and execution of the model. In the near past, the most popular framework was *TensorFlow* supported with the higher level interface of *Keras*. See for example [133]. However more recently *PyTorch*, see for example [320], has gained much more popularity and it has higher level interfaces such as *fast.ai* or *PyTorch Lightning*. See also *Flux.jl* in the Julia arena. In general, [190] provides a comprehensive applied introduction to deep learning with fast.ai and PyTorch. Further, consider the video lecture series coordinated by Andrew Ng[16]; see also [306]. Our book's website provides more information and suggested resources.

Some quantitive measures of the size of the human brain and other brains is in [175]. As we do not provide a complete historical account of deep learning and its relationship with neuroscience, we recommend the article [364] for a taste of recent work on neuroscience inspired by deep learning. Also see [165] for a comprehensive condensed review of the interface of the fields, both historically and today. An interesting style of neural network architecture which attempts to resemble the brain more closely than standard architectures is the *spiking neural network*; see for example [398]. More relationships with neuroscience as well as other views of deep learning can be found in other deep learning texts. A now classical deep learning book is [142], and a slightly more recent comprehensive textbook is [4]. See also the book [308], as well as a more recent book taking a mathematical approach [69].

A recognized reference for the *Turing test* from 1950 is [404]. The first major work on neural network architectures is the *perceptron* by Frank Rosenblatt [353] in 1958. A major landmark that helped spark general attention in deep learning is the *AlexNet* success in the 2012 ImageNet challenge, [239]. The literature and work between Rosenbladtt's early work and the 2012 AlexNet success is too numerous to list here. It involved an evolution of many ideas and experiments over decades. A good starting point for the key references is the nature paper [249]. Further key references are mentioned at the end of the chapters in the sequel.

The creation of the *ImageNet* database by Fei-Fei Li, her team, and collaborators has had a profound affect on deep learning in the second decade of this century. ImageNet, [102], was created with the aid of Amazon's *Mechanical Turk* service and with the *WordNet*, [289], hierarchical word database. The *VGG19* model (see [380] as well as the *VGG16* model) mentioned in this chapter is one of several models that at the time of creation had near-top performance in the ImageNet challenge, associated with the ImageNet database. There are many additional models that are both more efficient and outperform VGG19 and VGG16. The frontier keeps improving. See for example the empirical analysis [70] to get a feel for the performance metrics at play. Chapter 6 provides more references to landmark convolutional architectures.

[15] See `https://deeplearningmath.org/`.
[16] See `https://www.deeplearning.ai/`.

Diffusion models, gained significant prominence following [183]. The *generative adversarial network* idea and the association to game theory first appeared in Ian Goodfellow's co-authored work in 2015, [143]. For a key reference dealing with *AlphaGo* see [377] from 2016. An article describing the earlier (1999) *deep blue* chess playing system is [164]. Modern approaches to deep reinforcement learning are surveyed extensively in the texts [125] and [393]. The more recent *GPT-3* model is described in [66]. General texts about graph neural networks are [157] and [273]. The subsequent chapters contain more detailed references for each of these subject areas.

Our exposition has completely ignored the many *ethical aspects* associated with machine learning, deep learning, and artificial intelligence. This includes issues of disinformation, bias and fairness (also in datasets), privacy and surveillance, algorithmic colonialism, and the many negative applications that one can cook up with deep learning and generative models. Ethical considerations are critical for anyone embarking on an applied deep learning project. As this is not the topic of this book, we suggest the resources and content from the fast.ai website[17] as a starting point on ethics in deep learning.

Since the rest of the chapters of this book are more mathematical than this chapter, some readers may wish to strengthen or revive their mathematical background. In the context of *linear algebra* for data science the introductory book [56] is recommended. Further [391] provides more context with a rich variety of linear algebra topics and their interface with data science. A general mathematical review for machine learning is in [101]. Finally, in addition to exploring the appendices of the current book we also recommend the appendices of [240]. See also further resources in our book's website.

The research world of deep learning is expanding very quickly with multiple important directions that are beyond the scope of the book. One direction is *Bayesian neural networks*, see for example [213]. Another component is *causal modeling*; see for example [325]. Further, in the field of quantum computing there is already research on *quantum deep learning*; see for example [130]. The applied world of deep learning is currently in a massive growth phase with large language models being one of the most exciting directions. A comprehensive survey relevant for the time of publishing of this book is [449].

[17]See `https://ethics.fast.ai/`.