BOOK REVIEW: THE HUNDRED-PAGE MACHINE LEARNING BOOK

Scott O’Hara
Metrowest Developers Machine Learning Group
11/18/2020

Andriy Burkov

THE HUNDRED-PAGE MACHINE LEARNING BOOK

- Concise definitions and well-thought-out examples.
- ~140 pages.
- Covers mostly supervised learning.
- Provides pointers to things it doesn't cover.
- Generally, a good summary of things you should know about ML.

CONTENTS

CHAPTER 3: FUNDAMENTAL ALGORITHMS

- Linear Regression
- Logistic Regression
- Decision Trees
- Support Vector Machine
- k-Nearest Neighbors

CHAPTER 5: BASIC PRACTICE

- Feature Engineering
  - One-Hot Encoding
  - Binning
  - Normalization
  - Standardization
  - Dealing with Missing Features
  - Data Imputation Techniques
- Learning Algorithm Selection
- Three Sets (Training Set, Validation Set, Test Set)
- Underfitting and Overfitting
- Regularization
- Model Performance Assessment
- Hyperparameter Tuning

CHAPTER 6: NEURAL NETWORKS AND DEEP LEARNING

- Multilayer Perceptron Example
- Feed-Forward Neural Network Architecture
- Convolutional Neural Network
- Recurrent Neural Network

THE BOOK HAS A WEBSITE

- http://themlbook.com/

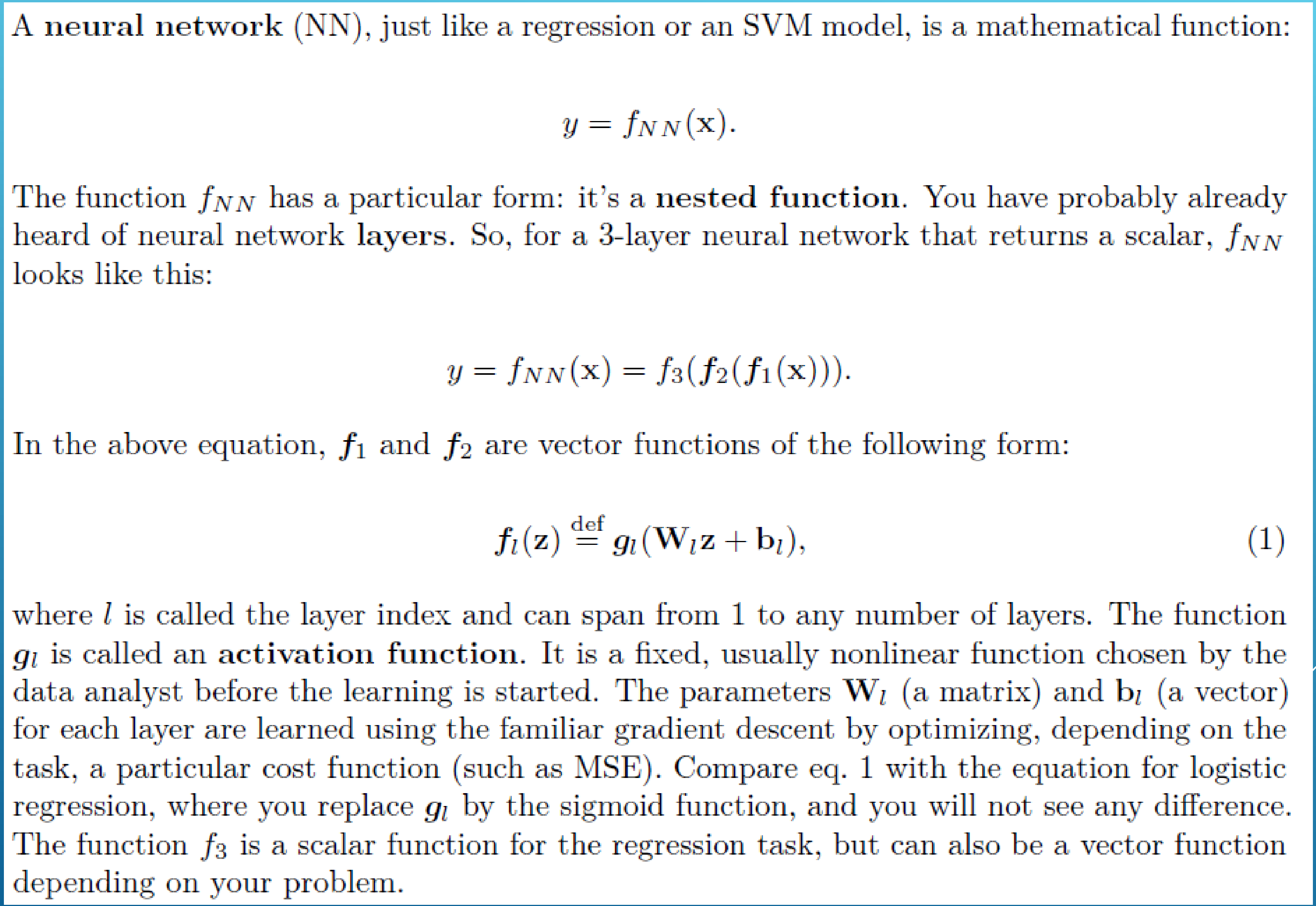
THE BOOK HAS WIKI PAGE

- http://themlbook.com/wiki/doku.php

THE BOOK HAS GITHUB REPOSITORY

- https://github.com/aburkov/theMLbook

NEURAL NETWORKS



A **neural network** (NN), just like a regression or an SVM model, is a mathematical function:

$$y = f_{NN}(\mathbf{x}).$$

The function f_{NN} has a particular form: it's a **nested function**. You have probably already heard of neural network **layers**. So, for a 3-layer neural network that returns a scalar, f_{NN} looks like this:

$$y = f_{NN}(\mathbf{x}) = f_3(\boldsymbol{f}_2(\boldsymbol{f}_1(\mathbf{x}))).$$

In the above equation, $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are vector functions of the following form:

$$\boldsymbol{f}_l(\mathbf{z}) \overset{\text{def}}{=} \boldsymbol{g}_l(\mathbf{W}_l\mathbf{z} + \mathbf{b}_l), \tag{1}$$

where l is called the layer index and can span from 1 to any number of layers. The function $\boldsymbol{g}_l$ is called an **activation function**. It is a fixed, usually nonlinear function chosen by the data analyst before the learning is started. The parameters $\mathbf{W}_l$ (a matrix) and $\mathbf{b}_l$ (a vector) for each layer are learned using the familiar gradient descent by optimizing, depending on the task, a particular cost function (such as MSE). Compare eq. 1 with the equation for logistic regression, where you replace $\boldsymbol{g}_l$ by the sigmoid function, and you will not see any difference. The function f_3 is a scalar function for the regression task, but can also be a vector function depending on your problem.

NEURAL NETWORKS

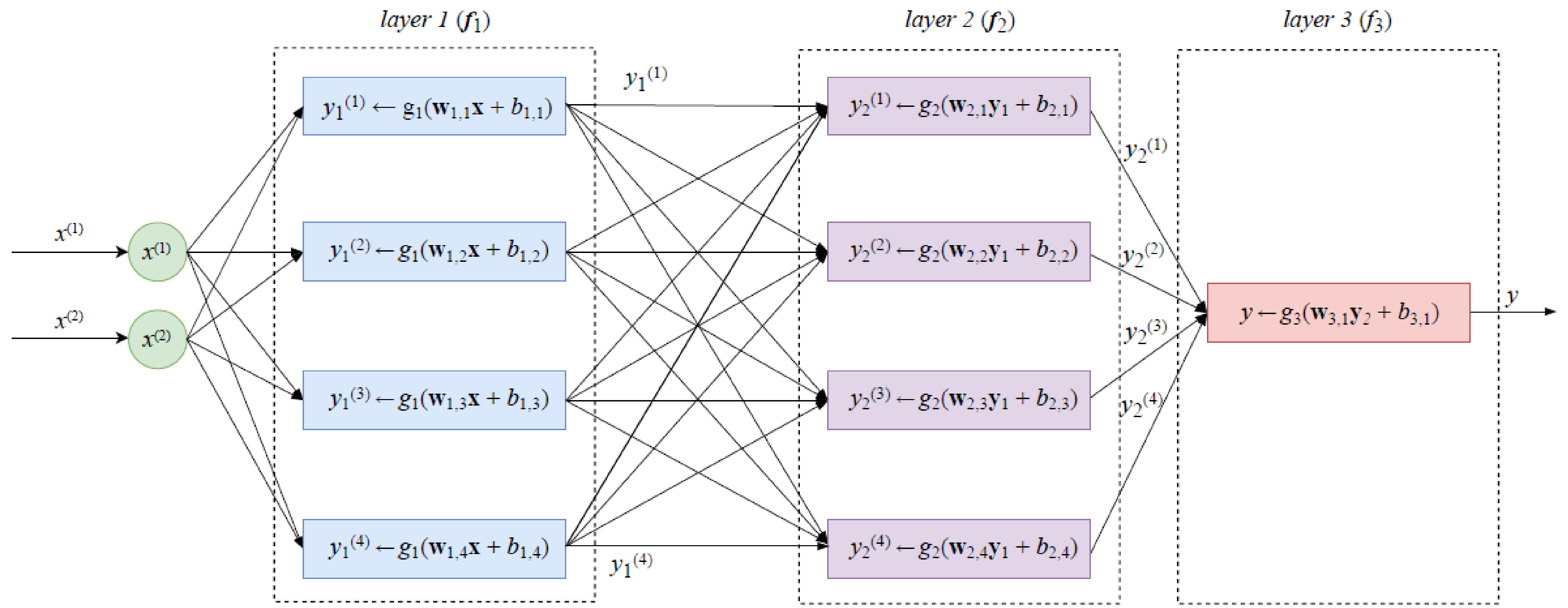
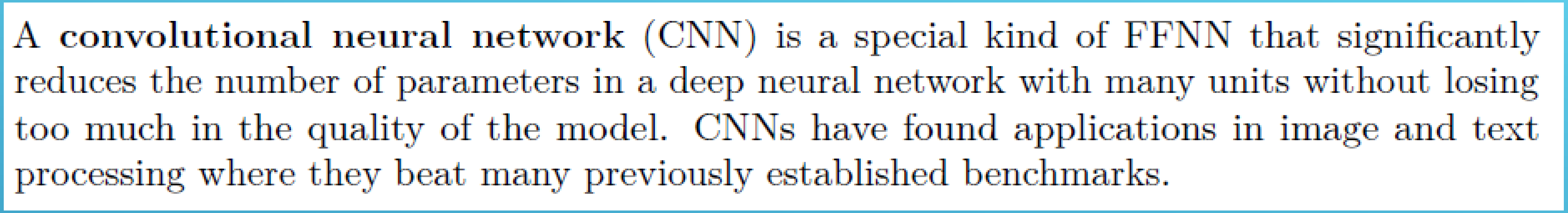
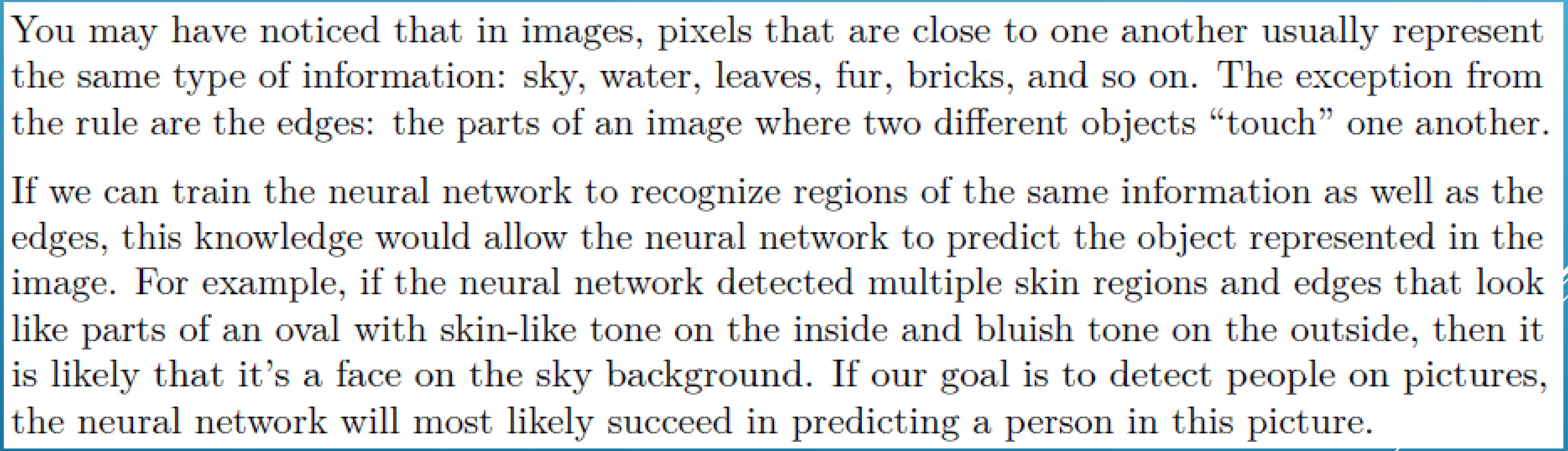

layer 1 (f_1)
layer 2 (f_2)
layer 3 (f_3)
$x^{(1)}$
$x^{(2)}$
$y_1^{(1)} \leftarrow g_1(\mathbf{w}_{1,1}\mathbf{x} + b_{1,1})$
$y_1^{(2)} \leftarrow g_1(\mathbf{w}_{1,2}\mathbf{x} + b_{1,2})$
$y_1^{(3)} \leftarrow g_1(\mathbf{w}_{1,3}\mathbf{x} + b_{1,3})$
$y_1^{(4)} \leftarrow g_1(\mathbf{w}_{1,4}\mathbf{x} + b_{1,4})$
$y_1^{(1)}$
$y_1^{(4)}$
$y_2^{(1)} \leftarrow g_2(\mathbf{w}_{2,1}\mathbf{y}_1 + b_{2,1})$
$y_2^{(2)} \leftarrow g_2(\mathbf{w}_{2,2}\mathbf{y}_1 + b_{2,2})$
$y_2^{(3)} \leftarrow g_2(\mathbf{w}_{2,3}\mathbf{y}_1 + b_{2,3})$
$y_2^{(4)} \leftarrow g_2(\mathbf{w}_{2,4}\mathbf{y}_1 + b_{2,4})$
$y_2^{(1)}$
$y_2^{(2)}$
$y_2^{(3)}$
$y_2^{(4)}$
$y \leftarrow g_3(\mathbf{w}_{3,1}\mathbf{y}_2 + b_{3,1})$
y

Figure 1: A multilayer perceptron with two-dimensional input, two layers with four units and one output layer with one unit.

CONVOLUTIONAL NEURAL NETWORKS



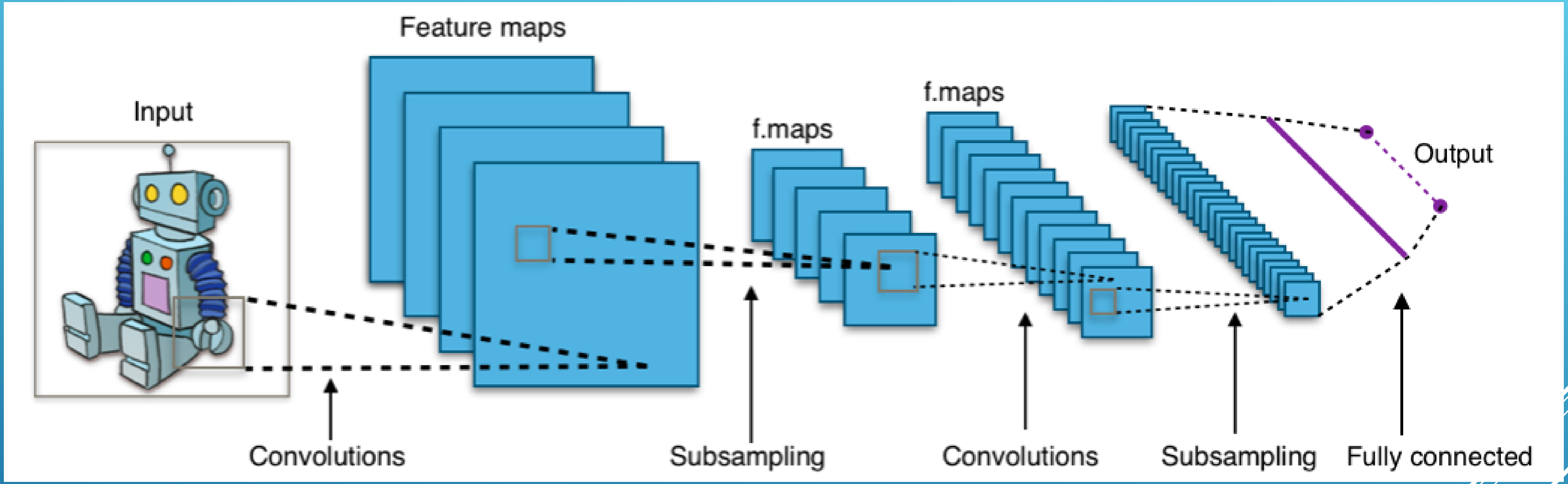
A **convolutional neural network** (CNN) is a special kind of FFNN that significantly reduces the number of parameters in a deep neural network with many units without losing too much in the quality of the model. CNNs have found applications in image and text processing where they beat many previously established benchmarks.



You may have noticed that in images, pixels that are close to one another usually represent the same type of information: sky, water, leaves, fur, bricks, and so on. The exception from the rule are the edges: the parts of an image where two different objects "touch" one another.

If we can train the neural network to recognize regions of the same information as well as the edges, this knowledge would allow the neural network to predict the object represented in the image. For example, if the neural network detected multiple skin regions and edges that look like parts of an oval with skin-like tone on the inside and bluish tone on the outside, then it is likely that it's a face on the sky background. If our goal is to detect people on pictures, the neural network will most likely succeed in predicting a person in this picture.

TYPICAL CNN ARCHITECTURE



Wikipedia contributors. "Convolutional neural network." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 12 Nov. 2020. Web. 19 Nov. 2020.

CONVOLUTIONAL NEURAL NETWORKS

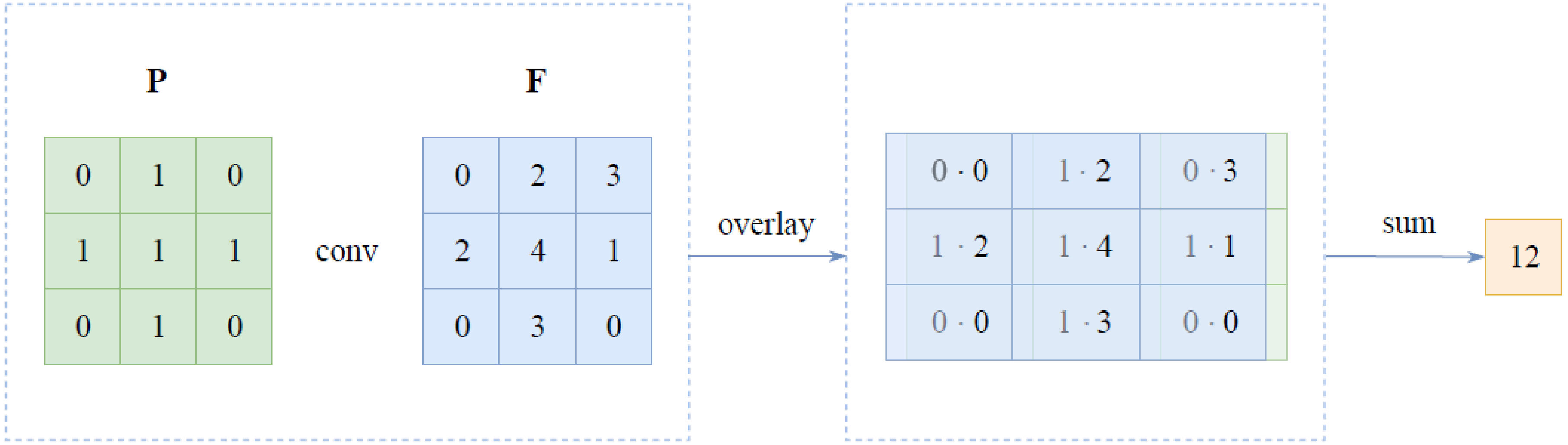


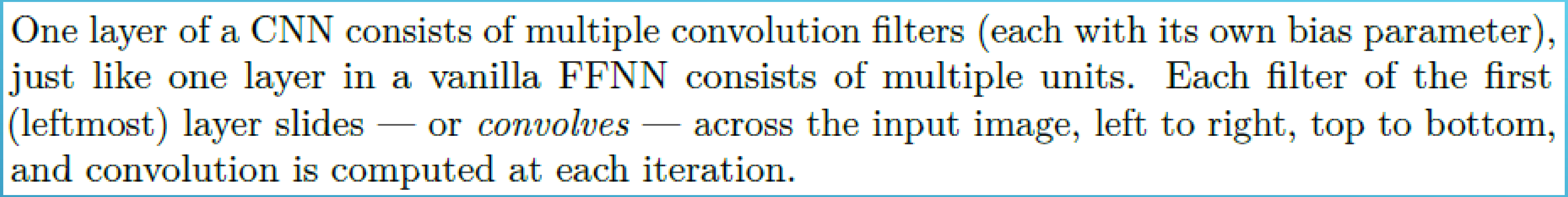
Figure 2: A convolution between two matrices.

If our input patch **P** had a different patten, for example, that of a letter L,

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix},$$

then the convolution with **F** would give a lower result: 5. So, you can see the more the patch "looks" like the filter, the higher the value of the convolution operation is. For convenience, there's also a bias parameter b associated with each filter **F** which is added to the result of a convolution before applying the nonlinearity (activation function).

A FILTER CONVOLVING ACROSS AN IMAGE



One layer of a CNN consists of multiple convolution filters (each with its own bias parameter), just like one layer in a vanilla FFNN consists of multiple units. Each filter of the first (leftmost) layer slides — or *convolves* — across the input image, left to right, top to bottom, and convolution is computed at each iteration.

A FILTER CONVOLVING ACROSS AN IMAGE

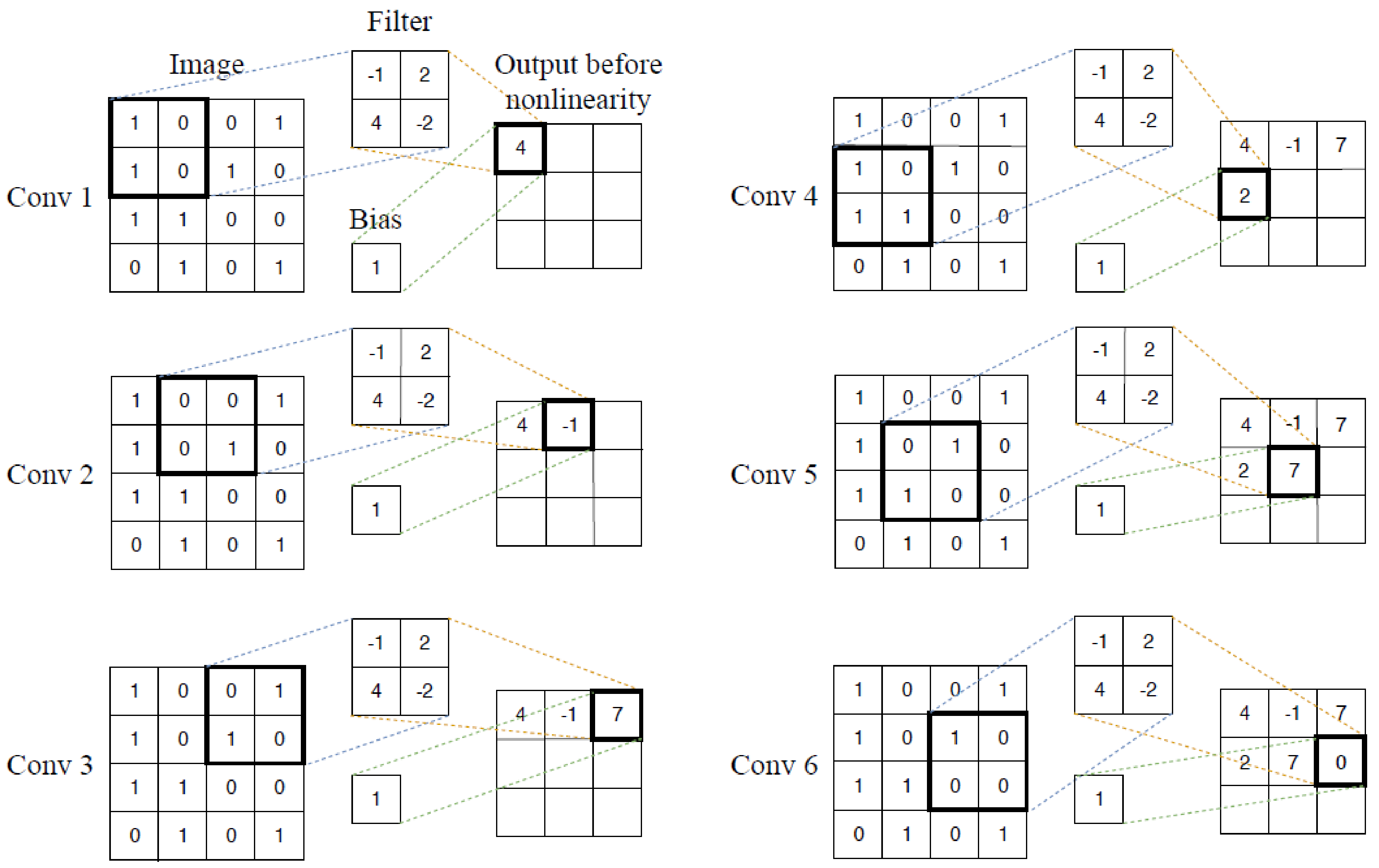


Figure 3: A filter convolving across an image.

A FILTER CONVOLVING ACROSS AN IMAGE

The filter matrix (one for each filter in each layer) and bias values are trainable parameters that are optimized using gradient descent with backpropagation.

A nonlinearity is applied to the sum of the convolution and the bias term. Typically, the ReLU activation function is used in all hidden layers. The activation function of the output layer depends on the task.

Since we can have $size_l$ filters in each layer l, the output of the convolution layer l would consist of $size_l$ matrices, one for each filter.

If the CNN has one convolution layer following another convolution layer, then the subsequent layer $l+1$ treats the output of the preceding layer l as a collection of $size_l$ image matrices. Such a collection is called a **volume**. The size of that collection is called the volume's depth. Each filter of layer $l+1$ convolves the whole volume. The convolution of a patch of a volume is simply the sum of convolutions of the corresponding patches of individual matrices the volume consists of.

CONVOLUTION OF A VOLUME

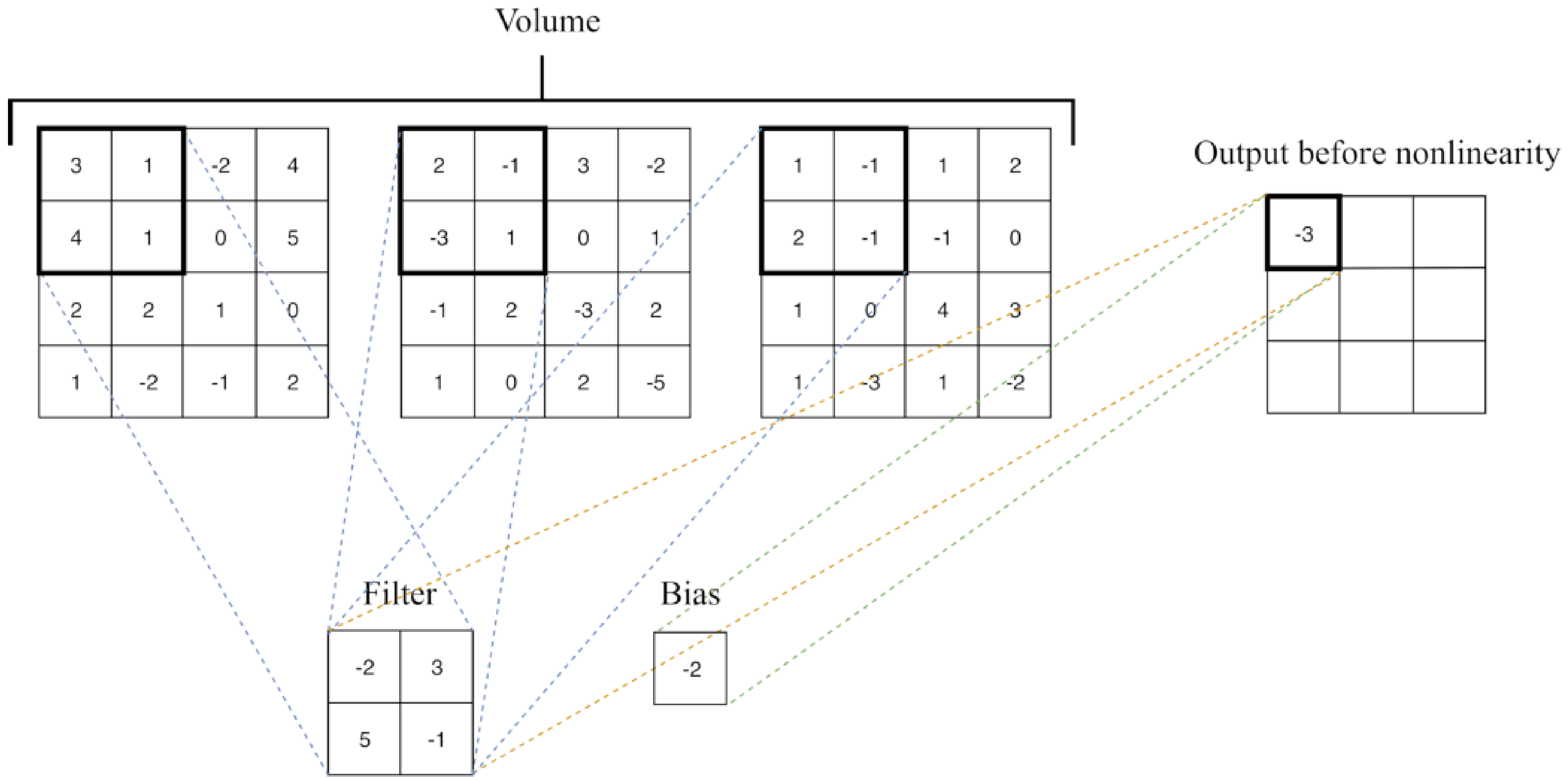


Figure 4: Convolution of a volume consisting of three matrices.

STRIDE

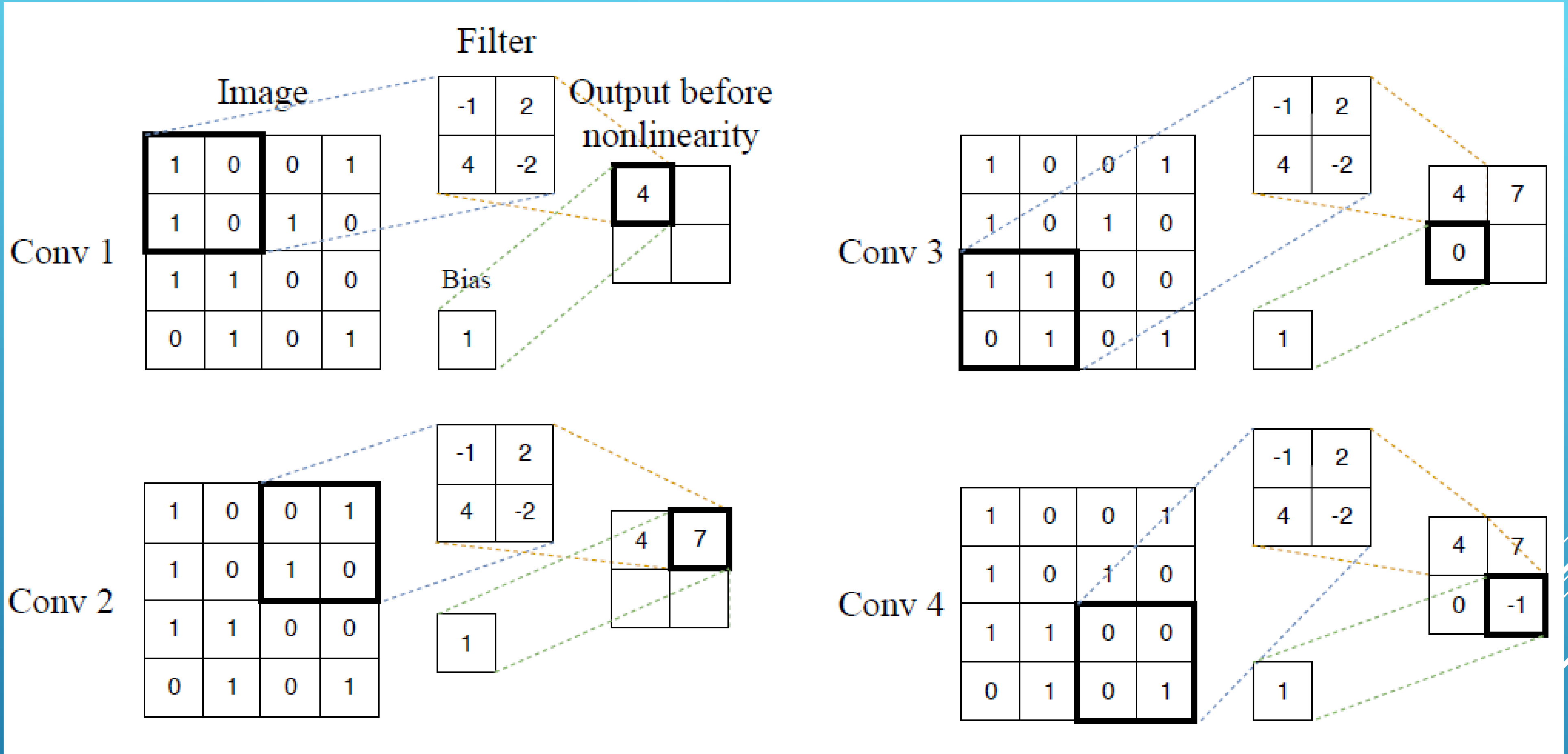


Figure 5: Convolution with stride 2.

Stride is the step-size of the moving window.

PADDING

Padding allows getting a larger output matrix; it's the width of the square of additional cells with which you surround the image (or volume) before you convolve it with the filter.

The cells added by padding usually contain zeroes.

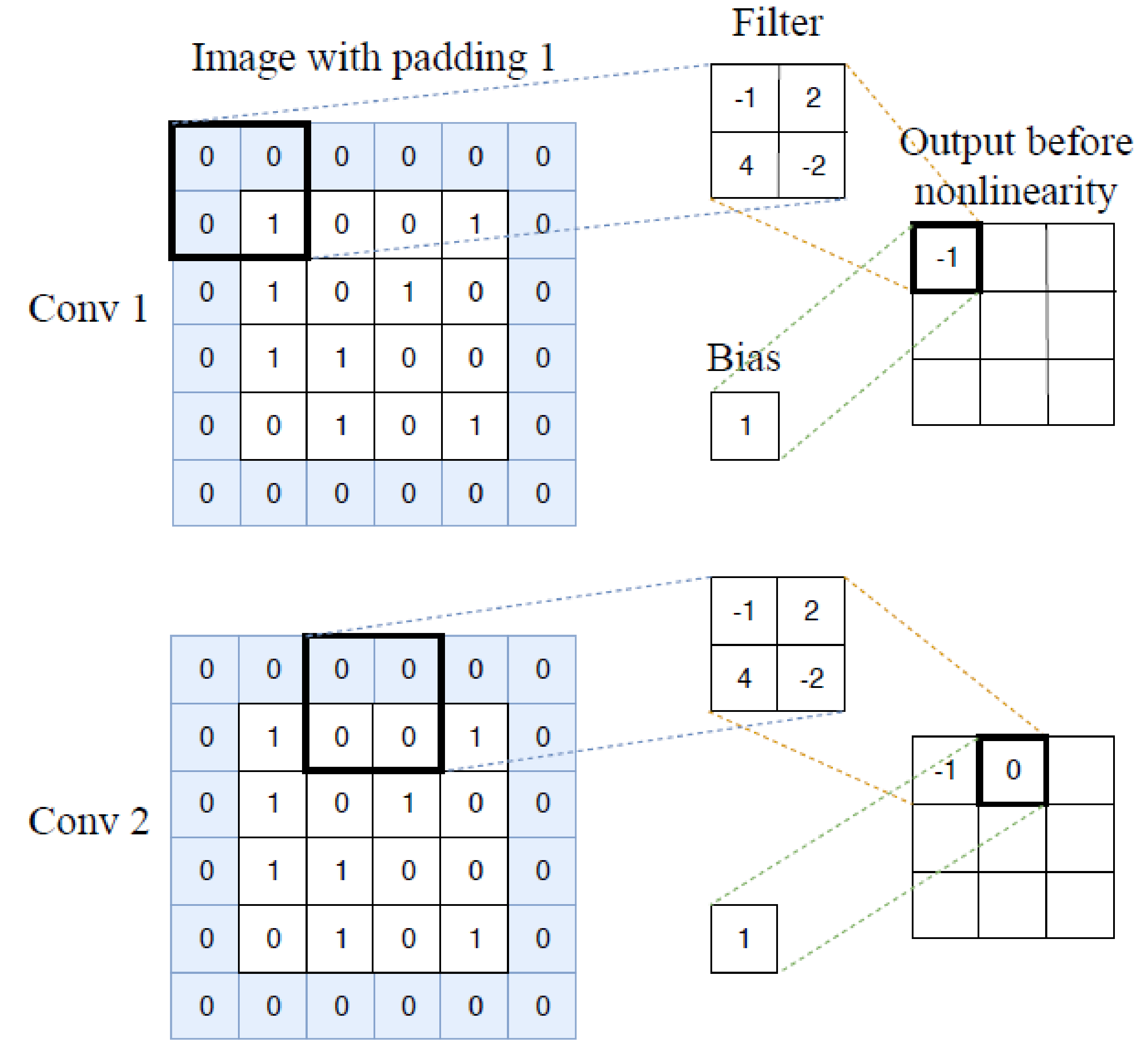


Figure 6: Convolution with stride 2 and padding 1.

POOLING

Pooling works in a way very similar to convolution, as a filter applied using a moving window approach. However, instead of applying a trainable filter to an input matrix or a volume, pooling layer applies a fixed operator, usually either max or average.

POOLING

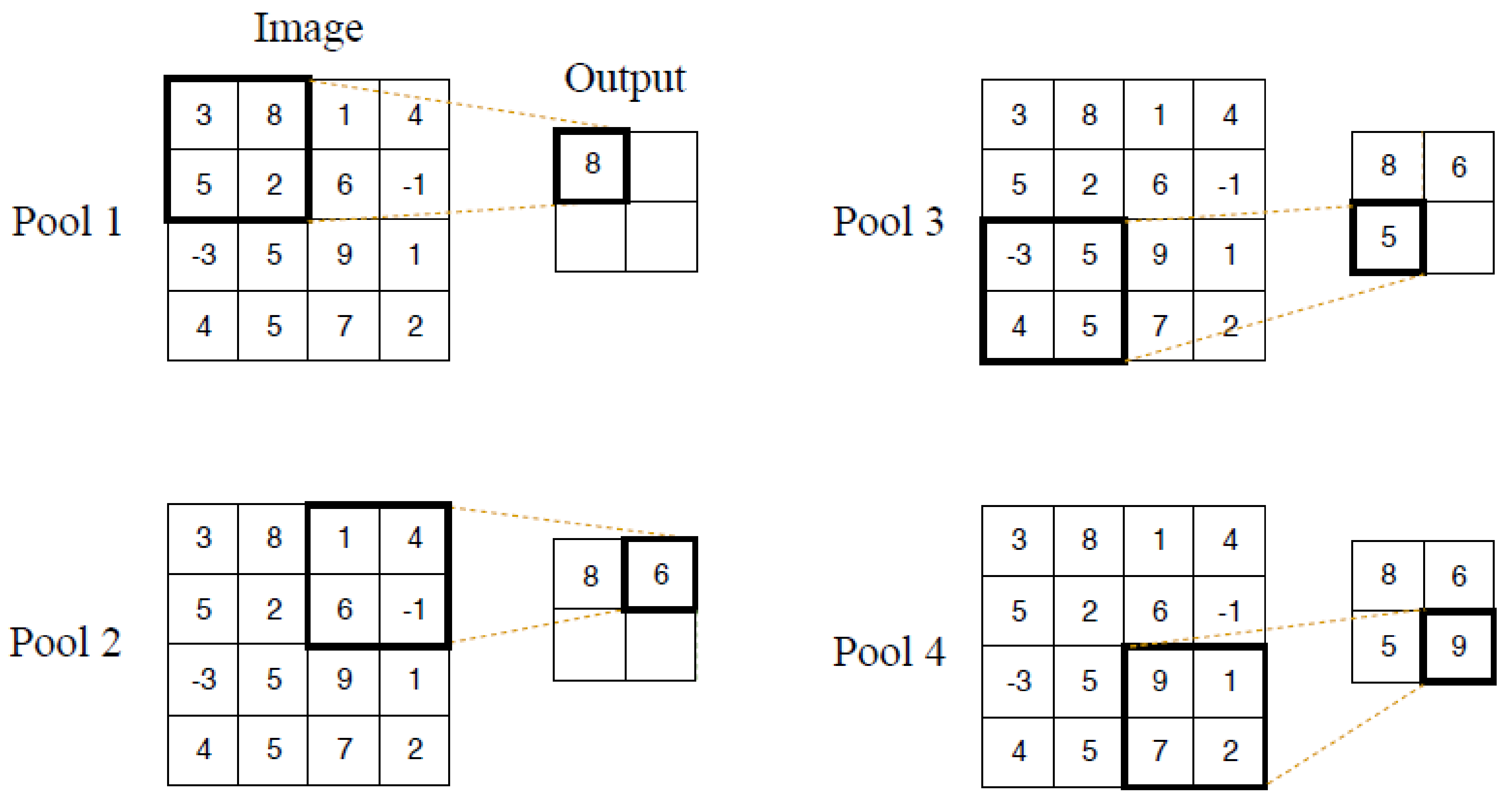


Figure 8: Pooling with filter of size 2 and stride 2.

POOLING

Usually, a pooling layer follows a convolution layer, and it gets the output of convolution as input. When pooling is applied to a volume, each matrix in the volume is processed independently of others. Therefore, the output of the pooling layer applied to a volume is a
volume of the same depth as the input.

Typically pooling contributes to the increased accuracy of the model. It also improves the speed of training by reducing the number of parameters of the neural network.

CONCLUSIONS

- I recommend the book for the care in which Burkov has paid to defining terms and choosing instructive examples.
- Machine learning can’t really be covered in a hundred pages, but the attempt to boil things down is worth it.
- This might be a good book to review if you are preparing for an interview.