Computer Vision Projects with PyTorch

Design and Develop Production-Grade Models

—

Akshay Kulkarni
Adarsha Shivananda
Nitin Ranjan Sharma

Apress®

Computer Vision Projects with PyTorch

Design and Develop Production-Grade Models

Akshay Kulkarni
Adarsha Shivananda
Nitin Ranjan Sharma

Apress®

Computer Vision Projects with PyTorch: Design and Develop Production-Grade Models

Akshay Kulkarni
Bangalore, Karnataka, India

Adarsha Shivananda
Hosanagara tq, Shimoga dt, Karnataka, India

Nitin Ranjan Sharma
Bangalore, India

ISBN-13 (pbk): 978-1-4842-8272-4
ISBN-13 (electronic): 978-1-4842-8273-1
https://doi.org/10.1007/978-1-4842-8273-1

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: Laura Berendson
Coordinating Editor: Shrikant Vishwakarma
Copy Editor: Kezia Endsley

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-8272-4. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To our families.

Table of Contents

About the Authors xi

About the Technical Reviewer xiii

Introduction xv

Chapter 1: The Building Blocks of Computer Vision 1

What Is Computer Vision 3

Applications 3

Channels 8

Convolutional Neural Networks 12

Understanding the CNN Architecture Type 23

Working with Deep Learning Model Techniques 33

Introduction to PyTorch 37

Summary 41

Chapter 2: Image Classification 43

Topics to Cover 44

Defining the Problem 44

Overview of the Approach 45

Creating an Image Classification Pipeline 46

First Basic Model 47

Data 47

Data Exploration 51
Data Loader 53
Define the Model 55
The Training Process 63
The Second Variation of Model 69
The Third Variation of the Model 72
The Fourth Variation of the Model 80
Summary 83
Chapter 3: Building an Object Detection Model 85
Object Detection Using Boosted Cascade 86
R-CNN 90
The Region Proposal Network 92
Fast Region-Based Convolutional Neural Network 96
How the Region Proposal Network Works 98
The Anchor Generation Layer 101
The Region Proposal Layer 102
Mask R-CNN 103
Prerequisites 103
YOLO 104
YOLO V2/V3 106
Project Code Snippets 108
Step 1: Getting Annotated Data 110
Step 2: Fixing the Configuration File and Training 112
The Model File 113
Summary 128

Chapter 4: Building an Image Segmentation Model 129
Image Segmentation 130
Pretrained Support from PyTorch 133
Semantic Segmentation 133
Instance Segmentation 137
Fine-Tuning the Model 139
Summary 166

Chapter 5: Image-Based Search and Recommendation System 167
Problem Statement 168
Approach and Methodology 168
Implementation 170
The Dataset 171
Installing and Importing Libraries 172
Importing and Understanding the Data 173
Feature Engineering 178
Calculating Similarity and Ranking 186
Visualizing the Recommendations 189
Taking Image Input from Users and Recommending Similar Products 193
Summary 198

Chapter 6: Pose Estimation 199
Top-Down Approach 200
Bottom-Up Approach 200
OpenPose 201
Branch-1 201
Branch-2 201

HRNet (High-Resolution Net)....204
Higher HRNet....207
PoseNet....208
How Does PoseNet Work?....209
Pros and Cons of PoseNet....213
Applications of Pose Estimation....213
Test Cases Performed Retail Store Videos....214
Implementation....214
Step 1: Identify the List of Human Keypoints to Track....216
Step 2: Identify the Possible Connections Between the Keypoints....217
Step 3: Load the Pretrained Model from the PyTorch Library....218
Step 4: Input Image Preprocessing and Modeling....219
Step 5: Build Custom Functions to Plot the Output....220
Step 6: Plot the Output on the Input Image....223
Summary....227
Chapter 7: Image Anomaly Detection....229
Anomaly Detection....230
Approach 1: Using a Pretrained Classification Model....232
Step 1: Import the Required Libraries....233
Step 2: Create the Seed and Deterministic Functions....234
Step 3: Set the Hyperparameter....234
Step 4: Import the Dataset....235
Step 5: Image Preprocessing Stage....235
Step 6: Load the Pretrained Model....239
Step 7: Freeze the Model....241
Step 8: Train the Model....242
Step 9: Evaluate the Model....245

Approach 2: Using Autoencoder....255

Step 1: Prepare the Dataset Object....255

Step 2: Build the Autoencoder Network....256

Step 3: Train the Autoencoder Network....257

Step 4: Calculate the Reconstruction Loss Based on the Original Data....258

Step 5: Select the Most Anomalous Digit Based on the Error Metric Score....258

Output....258

Summary....259

Chapter 8: Image Super-Resolution....261

Up-Scaling Using the Nearest Neighbor Concept....263

Understanding Bilinear Up-Scaling....265

Variational Autoencoders....265

Generative Adversarial Networks....271

The Model Code....273

Model Development....274

Imports....275

Running the Application....291

Summary....294

Chapter 9: Video Analytics....297

Problem Statement....298

Approach....301

Implementation....302

Data....303

Uploading the Required Videos to Google Colab....304

Convert the Video to a Series of Images....304

Image Extraction....306

Data Preparation....306
Identify the Hotspots in a Retail Store....308
Importing Images....312
Getting Crowd Counts....312
Security and Surveillance....316
Identify the Demographics (Age and Gender)....321
Summary....322
Chapter 10: Explainable AI for Computer Vision....325
Grad-CAM....326
Grad-CAM++....328
NBDT....330
Step 1....331
Step 2....332
Steps 3 and 4....332
Grad-CAM and Grad-CAM++ Implementation....333
Grad-CAM and Grad-CAM++ Implementation on a Single Image....333
NBDT Implementation on a Single Image....337
Summary....340
Index....341

About the Authors

Akshay R Kulkarni is an AI and machine learning (ML) evangelist and a thought leader. He has consulted with several Fortune 500 and global enterprises to drive AI and data science-led strategic transformations. He is a Google developer, author, and regular speaker at major AI and data science conferences (including *Strata*, *O'Reilly AI Conf*, and *GIDS*). He has been a visiting faculty member for some of the top graduate institutes in India. In 2019, he was featured as one of the top 40 under 40 data scientists in India. In his spare time, he enjoys reading, writing, coding, and helping aspiring data scientists. He lives in Bangalore with his family.

Adarsha Shivananda is a data science and ML Ops leader. He is currently working on creating world-class ML Ops capabilities to ensure continuous value delivery from AI. He aims to build a pool of exceptional data scientists within and outside of the organization to solve problems through training programs, and he strives to stay ahead of the curve. He has worked extensively in the pharmaceutical, healthcare, CPG, retail, and marketing domains. He lives in Bangalore and loves to read and teach data science.

Nitin Ranjan Sharma is a manager at Novartis. He leads a team that develops products using multi-modal techniques. As a consultant, he has developed solutions for Fortune 500 companies and has been involved in solving complex business problems using machine learning and deep learning frameworks. His major focus area and core expertise is computer vision, including solving challenging business problems dealing with images and video data. Before Novartis, he was part of the data science team at Publicis Sapient, EY, and TekSystems Global Services. He is a regular speaker at data science community meetups and an open-source contributor. He also enjoys training and mentoring data science enthusiasts.

About the Technical Reviewer

Jalem Raj Rohit is a senior data scientist at Episource, where he leads all things computer vision. He co-founded ML communities like Pydata Delhi and Pydata Mumbai and organizes and speaks at meetups and conferences.

He has authored two books and a video lesson on the Julia language and serverless engineering. His areas of interest are computer vision, ML Ops, and distributed systems.

Introduction

This book explores various popular methodologies in the field of computer vision in order to unravel its mysteries. We use the PyTorch framework, because it's used by researchers, developers, and beginners to leverage the power of deep learning. This book explores multiple computer vision problems and shows you how to solve them. You can expect an introduction to some of the most critical challenges with hands-on code in PyTorch, which is suitable for beginner and intermediate Python users, along with various methodologies used to solve those business problems.

Production-grade code related to important concepts we present over the course of the book will help you get started quickly. These code snippets can be run on local systems, with or without GPUs (Graphics Processing Units) or on a cloud platform.

We'll introduce you to the concepts of image processing in stages, starting with the basic concepts of computer vision in the first chapter. We'll also delve into the field of deep learning and explain how models are developed for vision-related tasks. You'll get a quick introduction to PyTorch to prepare you for the example business challenges we'll be presenting later in the book. We explore concepts of the revolutionary convolutional neural networks, as well as architectures such as VGG, ResNet, YOLO, Inception, R-CNN, and many others.

The book dives deep into business problems related to image classification, object detection, and segmentation. We explore the concepts of super-resolution and GAN architectures, which are used in many industries. You learn about image similarity and pose estimation, which help with unsupervised problem sets. There are topics related to video analytics, which will help you develop the mindset of using the image and time-based concepts of frames. Adding to the list, the book ends by discussing how these deep learning models can be explained to your business partners. This book aims to be a complete suite for those pursuing computer vision business problems.

CHAPTER 1

The Building Blocks of Computer Vision

Humans have been part of a natural evolutionary pattern for centuries. According to the Flynn Effect, an average person born in recent times has a higher IQ than the average person born in the previous century. Human intelligence allows us to learn, decide, and make new decisions based on our learnings. We use IQ scores to quantify human intelligence, but what about machines? Machines are also part of this evolutionary journey. How have we moved our focus to machines and made them intelligent, as we know them today? Let's take a quick look at this history.

A breakthrough came in the 1940s when programmable digital computers became available, followed by the concept of the *Turing test,* which could measure the intelligence of machines. The concept of the *perceptron* goes back to 1958, when it was introduced as a powerful logical unit that could learn and predict. The perceptron is equivalent to a biological neuron that helps humans function. The 1970s saw fast growth in the field of artificial intelligence, and it has increased exponentially since that time.

Artificial intelligence is the intelligence showcased by a machine, more often when it is trained on historical events. Humans have been trained and conditioned their whole lives. We know, for example, that going too near a fire will cause us to be burned, which is painful and bad for our skin.

A. Kulkarni et al., *Computer Vision Projects with PyTorch*,
https://doi.org/10.1007/978-1-4842-8273-1_1

Similarly, a system can be trained to make distinctions between fire and water, based on the features or on historical evidence. Human intelligence is being replicated by machines, which gives rise to what we know as artificial intelligence.

Artificial intelligence encompasses machine learning and deep learning. Machine learning can be thought of as mathematical models that help algorithms learn from data/historical events and formulate decision-making processes. The machine learns the pattern of the data and enables the algorithms to create a self-sustaining system. Its performance can be limiting, such as in the case of huge complex data, which is where deep learning comes into the picture. Deep learning is another subset of artificial intelligence. It uses the concept of the perception, expands it to neural networks, and helps the algorithms learn from various complex data. Even though we have many modeling techniques at our disposal, it's best to find good and explainable results from the simplest of techniques, as stated by *Occam's razor* (the simplest answer is often the best).

Now that we have explored a bit of the history, let's browse through the applicable fields. There are two fields—Natural Language Processing (NLP) and Computer Vision (CV)—that use an immense amount of deep learning techniques to help solve problems. NLP caters to the problem sets defined by our language, essentially one of the most important modes of communication. CV, on the other hand, addresses vision-related problems. The world is full of data that humans can decipher with their senses. This includes vision from the eyes, smell from the nose, audio waves from the ear, taste from the tongue, and sensations from the skin. Using this sensory input, the connected neurons in our brains parse and process the information to make decisions on how to react. Computer vision is one field that addresses the visual side of the machine-learning problem.

This book takes you through the fundamentals and gives you a working knowledge of computer vision.

What Is Computer Vision

Computer vision deals with specific problem sets that rely on images and videos. It tries to decipher the information in the images/videos in order to make meaningful decisions. Just like humans parse an image or a series of images placed sequentially and make decisions about them, CV helps machines interpret and understand visual data. This includes object detection, image classification, image restoration, scene-to-text generation, super-resolution, video analysis, and image tracking. Each of these problems is important in its own way. Studying vision-related problems has gained a lot of attraction after the power of parallel computing came into play.

Applications

The applications of computer vision vary with respect to the industry being discussed. The following sections look at a few of these tasks.

Classification

The simplest form of an image-related problem involving a decision-making process is a supervised technique, called *classification*. Classification simply involves assigning classes to different images. The process can be as simple as an image having one class or it can more complex, when there are multiple classes within the same image. See Figures 1-1 and 1-2.

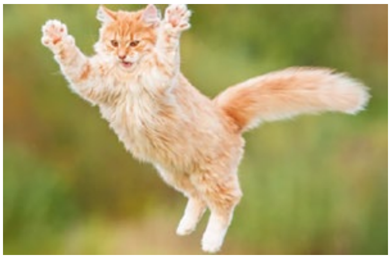


Figure 1-1. *The class in this case is a cat*

Figure 1-2. *The class in this case is a dog*

We can separate the content of such images based on whether they have an image of a cat or a dog. This is an example of how our eyes perceive differences. The background of the object we are trying to classify does not matter, so we need to make sure that it doesn't matter in the algorithms as well. For example, if we included a logo of some car company in front of all the dog images, the image classifier network might learn to classify dogs based on that logo and use it as a shortcut. We later describe in detail how to incorporate this information into the model. Classification can be used to identify objects in a production line of a manufacturing unit.

Object Detection and Localization

An interesting problem that is often encountered is the need to locate a particular image inside another image and even detect what that might be. Let's say there is a crowd of people and some are wearing masks and some are not. We can use a vision algorithm to learn the features of masks, then use that information to locate a mask relative to the image and detect the masks. See Figures 1-3 and 1-4.

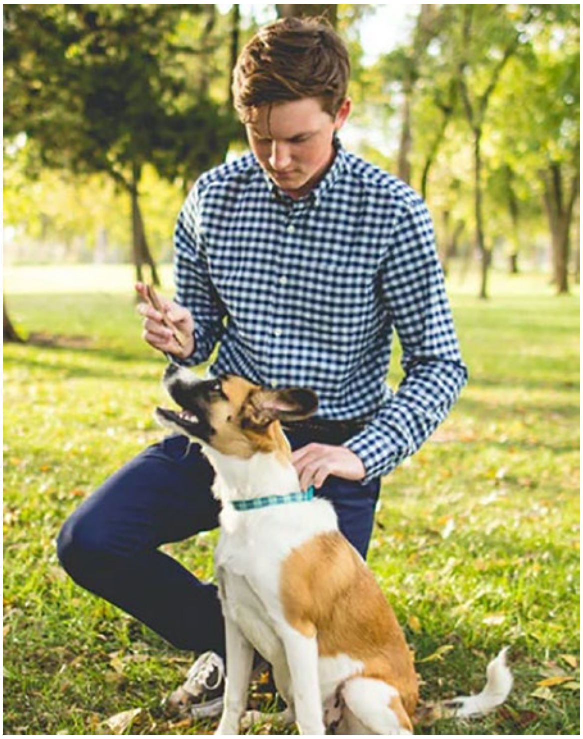


Figure 1-3. *Class: No mask detected*

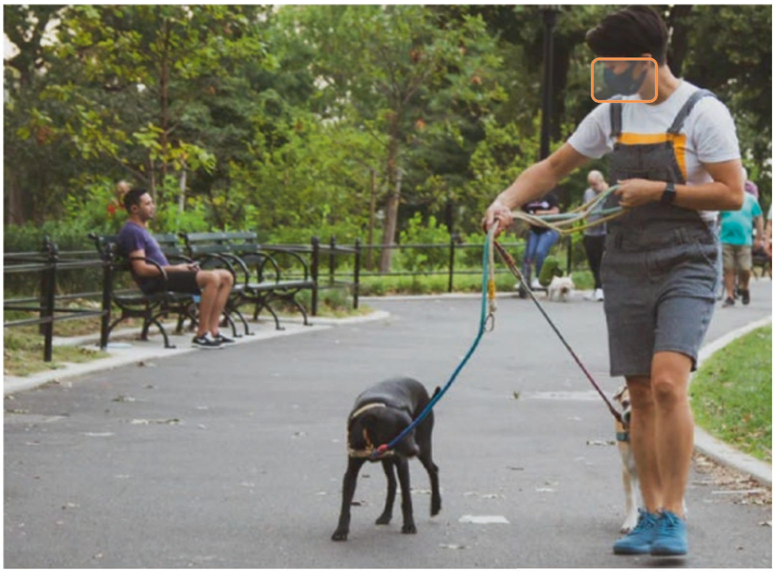


Figure 1-4. *Class: A mask is detected in the image*

This analysis can be helpful in detecting license plates of moving vehicles from traffic cameras. Sometimes, due to the resolution of the cameras and the moving traffic, the picture quality is not that great. Super-resolution is one technique that is sometimes used to enhance an image's quality and help identify the numbers on the plate.

Image Segmentation

This process is used to determine edges, curves, and gradients of similar objects placed together in order to separate different objects in an image. A classic unsupervised technique can be used here without the worry of finding good-quality, labeled data. The processed data can further be used as an input to an object detector. See Figure 1-5.

Figure 1-5. *Separating terrains in topological maps*

Anomaly Detection

Another classic, unsupervised way of determining changes is to compare an image to the usual, expected patterns from some training data. Anomaly detection can be used, for example, to determine imperfections in steel pipes when compared to training data. If the machine finds something odd, it will detect an anomaly and inform the line engineers to take care of it. See Figures 1-6a and 1-6b.

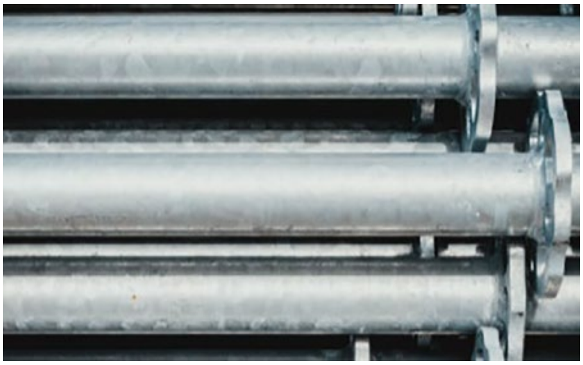


Figure 1-6a. *Perfect examples of steel pipes*

Figure 1-6b. *Anomalies showing up in the pipes*

Video Analysis

There are a lot of use cases for video or sequences of images. The task of object detection on running images can help with CCTV footage. It can also be used to detect abnormalities within the frames per video section.

We will be going through all of these applications in detail in the upcoming chapters. Before that, let's go through a few of the intrinsic concepts that lay the foundation for further understanding computer vision.

Channels

Figure 1-7. *Playing musicians*

One of the most basic and quintessential ideas around computer vision is the *channel*. Think of music being played with multiple instruments; we hear a combination of all the instruments playing together, which essentially constitutes the music in stereo (see Figure 1-7). If we break the music into single components, we can break the sound wave into individual sounds coming from the electric guitar, the acoustic guitar, the piano, and the vocals. After breaking the music into its components, we can modulate each component to get the desired music. There can be an infinite number of combinations if we learn all the musical modulations.

255	255	255	255
255	255	255	255
255	255	255	255
255	255	255	255

Figure 1-8a. *Pixel values corresponding to a white picture*

Figure 1-8b. *Sample white page*

2	36	40	200
195	190	20	180
40	54	200	200
30	40	200	180

Figure 1-8c. *Representational pixel values corresponding to a sketch*

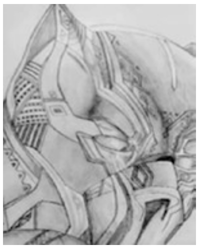


Figure 1-8d. *Sample sketch represented in one channel*

We can extrapolate these concepts to images, which we can break into components of colors. Pixels are the smallest containers of colors. If we zoom in on any digital image, we see small boxes (pixels), which make up the image. The general range of any pixel in terms of the intensity of a channel is 0-255, which is also the range defined by eight bits. Consider Figure 1-8b. We have a white page. If we convert that page to an array, it will give us a matrix of all 255 pixels, as shown in Figure 1-8a. On the other hand, Figure 1-8d, when converted to a matrix, will also have only one channel, and the intensity will be defined by the numbers in the range of 0-255, as depicted in Figure 1-8c. Closer to 0 gives us black and closer to 255 gives us white.

Let's consider a color image. We can break any full-color image into a combination of three primary components (channels)—red, green, and blue. We can break down any color image into some definite combination of red, green, and blue. Thus, RGB (red, green, and blue) becomes the channels of the colored image.

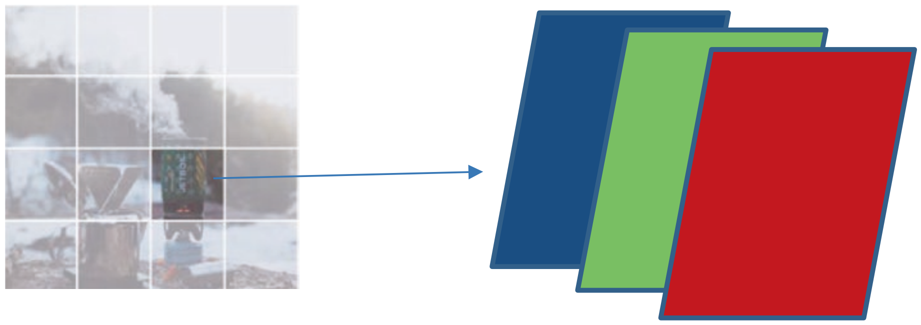


Figure 1-9. *Sample image to blue (left)=0, green (middle)=1, and red (right)=2*

The image in Figure 1-9 can be split into RGB, with the first channel being blue, then green, followed by red. Each pixel in the image can be a certain combination of RGB.

We are not restricted to using RGB as the color channels. We also have HSV (Hue, Saturation, and Value), LAB format, and CMYK (Cyan, Magenta, Yellow, and Black), which are a few representations of channels of an image. Color is a feature and its container is a channel, so every image is made of edges and gradients. We can create any image in the world with just edges and gradients. If you zoom in on a small circle, it should look like a combination of multiple edges and straight lines.

To summarize, channels can be called in as the container for a feature. The features can be the smallest individual characteristics of the image. Color channels are a specific example of channels. Since edges can be features, a channel that only caters to edges can be a channel too. Food for thought—if you were to create a model that would identify a cat or a dog, does the color of the animal affect the model behavior as much as edges and gradients can?

Convolutional Neural Networks

You now know that images have features, and those features need to be extracted for a better understanding of the data. If we consider a matrix of pixels, the pixels are related in all four directions. How do you do the extraction efficiently? Will the traditional methods of machine learning or deep learning help? Let's go through some problems:

1. Images can have huge dimensions. For example, a 2MP image, if it is allowed to capture a 1600x1200 image, will have 1.9 million pixels per image.
2. If we are capturing the data via the images, the data is not always centrally aligned. For example, a cat can be in one corner of one image, and on the next image, it can be in the center. The model should be able to capture the spatial changes in the information.
3. A cat in an image can be rotated along the vertical dimension or the horizontal dimension and still it remains a cat. Thus, we need a robust solution to capture such differences.

We need major upgrades from our regular tabular data approach. If we can break down a problem into smaller manageable pieces, anything can be solved. Herein, we use *convolutional neural networks.* We try to break the image into several feature maps via kernels and use those in sequence to build a model that can then be used for any downstream or pretext task.

Kernels are feature extractors. Features can be edges, gradients, patterns, or any of the small features discussed earlier in this chapter. A square matrix is generally used to operate a convolution task in the image on the first step and on feature maps from the second step. The convolution tasks performed by the kernels can be thought of as the simplest tasks in a dot product. See Figures 1-10a and 1-10b.

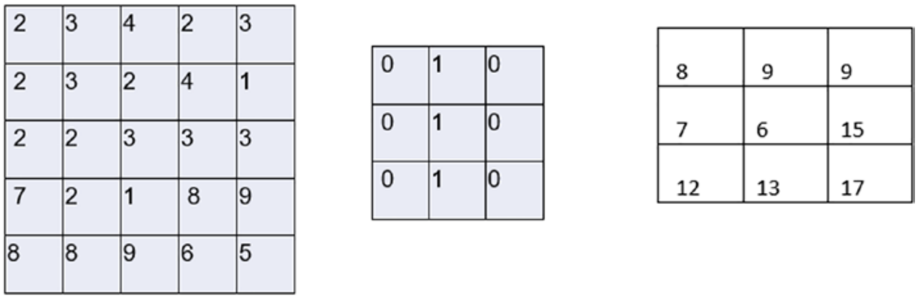
2	3	4
2	3	2
3	4	1

Figure 1-10a. *3x3 matrix of feature maps*

0	1	0
0	1	0
0	1	0

Figure 1-10b. *3x3 kernel*

Figure 1-10a is the image or feature map and Figure 1-10b is the kernel. The kernel is the feature extractor, so it will do a dot product on the feature map, resulting in the value 10. This is the first step of our convolution. The images or feature maps are going to be large and so the kernel might not operate on just a 3x3 matrix, but it will take a stride forward to calculate the next convolution operation. Let's look at an extrapolated example of this idea.



2	3	4	2	3
2	3	2	4	1
2	2	3	3	3
7	2	1	8	9
8	8	9	6	5

0	1	0
0	1	0
0	1	0

8	9	9
7	6	15
12	13	17

Figure 1-11. *Feature map, kernel, and resulting output*

As shown in Figure 1-11, a 5x5 feature map was convolved by a 3x3 kernel and that resulted in a 3x3 feature map. That map will again be convolved or converted to some features for a downstream task.

The convolution process also contains a concept of *stride*, which is a hyperparameter telling the kernel how to move around on the feature maps. Given our convolutional neural network, we have a stride of 1. A stride greater than 1 can cause a checkerboard issue in the feature map, with a few pixels getting more attention than the others. We might want or not want this effect, based on our business requirements. A higher stride value can also be used to lower the feature map dimensions.

There is an inherent problem with this convolution. The dimension keeps on shrinking when the convolutions happen. This can be desirable in some sense or some particular use cases but in a few use cases, we might want to retain the original dimensions. We can use the concept of padding on the image or the subsequent feature maps to avoid the issue of dimensionality reduction. Padding is another hyperparameter. We add layers around the images or feature maps.

Figure 1-12b shows how the padding essentially increases the space as well as allows the edge pixel values to be better noticed by the kernels. The convolution process will go through the edge values more than once, thus the information is being carried forward to the next feature maps with more effect. In the case of edges having pixel values, the convolution from kernels will take that value only once, whatever the stride values are on that occasion.

0	3	1
0	3	0
0	4	0

Figure 1-12a. *An edge detector*

0	0	0	0	0
0	4	1	4	0
0	5	1	2	0
0	5	1	1	0
0	0	0	0	0

Figure 1-12b. *A zero-padded feature map*

It is quite interesting how simple padding can change the identification of edges by a kernel. If we assume there is an important edge near the corner of the feature map and don't pad it, the edges won't be detected. This is because, to detect a line or an edge, the kernel or the feature extractor needs to have a similar pattern. In the case of the kernel in Figure 1-12a, which is an edge detector, it needs to find a proper gradient to detect an actual edge. It is because of the gradient from 0 to 4, 0 to 5, and 0 to 5, now it detects the edge. If the padding wasn't there, this gradient wouldn't be there and the kernel would have missed out on an important part.

Receptive Field

When we were working on the concepts of convolutions, there was a feature map and kernel stride. The kernel was extracting features in the space such that the model could interpret the information to its ease. Now we'll consider with an example an 56x56 image with one channel. (This is also written as 56x56x1.) If we try to convert the entire image to features, the entire span of pixels needs to be read. Let's look at a graphic example to sort out this concept.