MODERN COMPUTER VISION

BY RAJEEV RATAN

Welcome to Modern Computer Vision™

A Comprehensive Course on Computer Vision for 2022 onwards!

So what exactly is Computer Vision?

"An interdisciplinary field that aims to enable computers to gain ***understanding of what is being seen in*** *images and videos."*

Source - Terminator 2: Judgement Day (1991)

Is Computer Vision Artificial Intelligence?

Relation between Artificial Intelligence, Machine Learning and Deep Learning,Computer Vision.
https://www.researchgate.net/figure/Relation-between-Artificial-Intelligence-Machine-Learning-and-Deep-Learning-Computer_fig1_342978934

Computer Vision is an amalgamation of many fields

So what can Computer Vision do?

You might be familiar with these…

- **Snapchat and Instagram filters**
- Optical Character Recognition (OCR)
- Licence Plate Reading
- Self-driving cars
- Sporting Analysis
- Facial Recognition

Source - Cnet - Snapchat's Time Machine AR lens creepily shows what you'll look like old

So what can Computer Vision do?

You might be familiar with these…

- Snapchat and Instagram filters
- **Optical Character Recognition (OCR)**
- Licence Plate Reading
- Self-driving cars
- Sporting Analysis
- Facial Recognition

Source -AT&T's LeNet OCR for Handwritten Digits

So what can Computer Vision do?

You might be familiar with these…

- Snapchat and Instagram filters
- Optical Character Recognition (OCR)
- **Licence Plate Reading**
- Self-driving cars
- Sporting Analysis
- Facial Recognition

So what can Computer Vision do?

You might be familiar with these…

- Snapchat and Instagram filters
- Optical Character Recognition (OCR)
- Licence Plate Reading
- **Self-driving cars**
- Sporting Analysis
- Facial Recognition

Source Inc. - The Tech That Powers Your Self-Driving Car Might Be Built Using People Playing Games on Their Phones

So what can Computer Vision do?

You might be familiar with these…

- Snapchat and Instagram filters
- Optical Character Recognition (OCR)
- Licence Plate Reading
- Self-driving cars
- **Sporting Analysis**
- Facial Recognition

HawkEye in Cricket

Source -Roboflow - AI Coach

So what can Computer Vision do?

You might be familiar with these…

- Snapchat and Instagram filters
- Optical Character Recognition (OCR)
- Licence Plate Reading
- Self-driving cars
- Sporting Analysis
- **Facial Recognition**

But there’s so much more!

- **Image Recognition**
- **Object Detection**
- **Segmentation**
- **AI Art**
- Image Similarity
- Deep Fakes
- Body Pose Detection
- Image Generation

But there’s so much more!

- Image Recognition
- Object Detection
- Segmentation
- AI Art
- **Image Similarity**
- **Deep Fakes**
- Body Pose Detection
- Image Generation

But there’s so much more!

- Image Recognition
- Object Detection
- Segmentation
- AI Art
- Image Similarity
- Deep Fakes
- **Body Pose Detection**
- **Image Generation**

Computer Vision Applications are endless!

Electric Scooter ID
Gas Leak Detection
Document Digitization
Plant Phenotyping
Flare Stack Monitoring
Resume Parsing
Augmented Reality
Weed Detection
Microscopy
Bean Counting
Garbage Cleanup
Drone Video Analysis
Conveyer Belt Debris
Traffic Counter
Pothole Identification
Soccer Player Tracker

Steelyard Throughput
Security Cam Analysis
Self Driving Cars
Fish Measuring
Remote Tech Support
Tennis Line Tracking
Know Your Customer
Endangered Species Tracking
Inventory Management
Hard Hat Detection
Pest Identification
OCR Math
Basketball Shot Tracking
Logo Identification
Satellite Imagery
Traffic Cone Finder

Airplane Maintenance
Tumor Detection
D&D Dice Counter
Plant Disease Finder
X-Ray Analysis
Roof Damage Estimator
City Bus Tracking
Board Game Helpers
Dental Cavity Detection
Drought Tracking
Hog Confinements
Sushi Identifier
Oil Storage Estimator
Car Wheel Finder
License Plate Reader
Exercise Counter

roboflow

Why do this course?

What are you going to learn exactly?

- Foundation in **Classical Computer** Vision with **OpenCV**
- **Deep Learning** applied to Computer Vision
  - PyTorch
  - TensorFlow Keras

Who should do this course?

- **College Students** (BSc., MSc or PhD) looking to start understanding and implementing computer vision applications
- **High School Students** or hobbyists
- **Software Engineers/Developers & Data Scientists** looking to get into Computer Vision

About me

- **Electrical & Computer Engineer** with **7 years** experience as a Radio Frequency Engineer
- **MSc in Artificial Intelligence** from the **University of Edinburgh**
- **6 years experience** working in **Computer Vision**
- Worked at several **London startups** and even co-founded my own company
- Created several **Udemy courses in Computer Vision**
- Now **Senior Computer Vision Engineer**

My Udemy Courses

Instructor Rating of 4.3/5

INSTRUCTOR

Rajeev D. Ratan

Data Scientist, Computer Vision Expert & Electrical Engineer

Total students	Reviews
49,022	7,559

Facebook

Linkedin

About me

Hi I'm Rajeev, a **Data Scientist**, and **Computer Vision Engineer.**

I have a BSc in Computer & Electrical Engineering and an **MSc in Artificial Intelligence from the University of Edinburgh** where I gained extensive knowledge of machine learning, computer vision, and intelligent robotics.

I have published research on using data-driven methods for **Probabilistic Stochastic Modeling for Public Transport** and even was part of a group that won a robotics competition at the University of Edinburgh.

I launched my own computer vision startup that was based on using deep learning in education since then I've been contributing to 2 more startups in computer vision domains and one multinational company in Data Science.

Previously, I worked for 8 years at two of the Caribbean's largest telecommunication operators where he gained experience in managing technical staff and deploying complex telecommunications projects.

Show less

My courses (7)

Deep Learning Computer Vision™ CNN, OpenCV, YOLO, SSD & GANs

Rajeev D. Ratan

4.3 ★★★★☆ (1,979)

14.5 total hours • 178 lectures • Intermediate

£21.99 ~~£79.99~~

Data Science, Analytics & AI for Business & the Real World™

Rajeev D. Ratan, Nidia Sahjara

4.4 ★★★★☆ (316)

30.5 total hours • 248 lectures • Beginner

£47.99 ~~£174.99~~

Bestseller

Data Science & Deep Learning for Business™ 20 Case Studies

Rajeev D. Ratan, Nidia Sahjara

4.5 ★★★★☆ (867)

21 total hours • 190 lectures • All Levels

£15.99 ~~£59.99~~

Bestseller

Master Computer Vision™ OpenCV4 in Python with Deep Learning

Rajeev D. Ratan

4.2 ★★★★☆ (3,506)

11 total hours • 116 lectures • All Levels

£15.99 ~~£59.99~~

Why did I make this Course?

- My previous courses launched in (2016 & 2018) were good but had many deficiencies. This course aims to fix everything that was wrong or broken in my older courses.
  - Taught using **Google Colab** (no messy installs and Virtual Machine setups)
  - Code and methodology are **up to date**
  - Covers **all key areas** in Computer Vision
  - Taught using both **TensorFlow Keras** and **PyTorch**
  - Includes both **OpenCV** and **Deep Learning** Modules in **One Single** 27+ hour course!

Requirements?

- **Internet connection** (preferably at least 2mb) and web browser
- **High school level math** knowledge (preferred, not required)
- **Basic programming** knowledge is **not required** but preferred (especially if it's Python)
- **Enthusiasm** about Artificial Intelligence!

High Level Overview

OpenCV - Classical Computer Vision Outline

1. Image Manipulations
2. Segmentation
3. Feature Extraction
4. Object Detection with Haar Cascade Classifiers
5. Image analysis and Transformations
6. Background Removal
7. Tracking - Optical Flow and MeanShift
8. Face Swapping
9. OCR and text detection
10. QR and Barcode reading
11. YOLOv3 & SSDs in OpenCV
12. Neural Style Transfer
13. Colorising Black and White Photos
14. Computation Photography (noise removal, in-painting)
15. Facial Recognition
16. Working with Video and Video Stream

High Level Overview

Deep Learning Outline

1. Introduction to Deep Learning in Computer Vision
2. Deep Learning Basics with PyTorch and TensorFlow Keras
3. Comparing Libraries and Analysing Performance
4. Regularisation and Overfitting
5. Understanding What CNN's See
6. Designing CNNs and Overview of Modern CNN Architectures
7. PyTorch Lightning
8. Transfer Learning
9. Google DeepDream
10. Neural Style Transfer
11. Autoencoders
12. Generative Adversarial Networks
13. Siamese Networks
14. Facial Recognition
15. Object Detection
16. Deep Segmentation
17. Tracking with Deep Learning
18. Vision Transformers
19. Depth Estimation
20. Video Classification
21. Point Cloud Classification and Segmentation
22. 3D Image Classification
23. OCR Readers
24. Image Captioning
25. Computer Vision APIs

MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

Course Outline Overview

MODERN COMPUTER VISION

BY RAJEEV RATAN

Course Overview

This is a BIG course! Let’s get started!

So how do we do Computer Vision?

What makes it possible?

- We need tools! Namely a programming language
- Many exist such as:
  - Matlab
  - C++ & Java
  - **Python**

What makes Python so great for Computer Vision?

- Easy to learn and accessible
- Language of Artificial Intelligence
- The best and easiest to use Libraries, such as:
  - **OpenCV** for Classical Computer Vision
  - **PyTorch** (Facebook) and **TensorFlow with Keras** (Google)

Classical Computer Vision?

- What is meant by **Classical Computer Vision?**
- It encompasses Computer Vision algorithms that **do not** involve Machine Learning
- Before the advent of Machine Learning and Deep Learning, Computer Vision was a deeply explored field and many useful algorithms were developed for things like **feature extraction, OCR, Segmentation and simple transformations.**
- **OpenCV** is the Classical Computer Vision library of choice!

Deep Learning Computer Vision

- Deep Learning was used in Computer Vision since the 1990s, however due to the computational requirements and intricate design, it remained on the sidelines for decades.
- Until the the mid 2010s…which brought **two important building blocks** together.
  - **Mature Deep Learning libraries** (TensorFlow, Keras, Theano, Caffe)
  - **Accessible GPU processing** (NVIDIA's CUDA)

NVIDIA
CUDA

Deep Learning Examples

Image Classification

{Dog}

Object Detection

{Dog, Dog, Dog, Person, Person}

Deep Learning Examples

Deep Learning Examples

Deep Learning CV vs Classical CV

Deep Learning	Classical Computer Vision
Adapts to new images well (assuming it's similar to the data it was trained on)	Small changes can have big negative impacts
Requires Models to be trained	Doesn't require training and can be used once coded
Model weights learn to adapt to varying image conditions	Relies on hardcode features and parameters
Requires GPU hardware (most times)	Can be run on CPU

Our OpenCV Outline

1. Getting Started with OpeNCV
2. Grey-scaling Images
3. Color Spaces (HSV & RGB)
4. Drawing on Images
5. Transformations - Translations and Rotations
6. Scaling and re-sizing and Cropping
7. Arithmetic and Bitwise Operations
8. Convolutions, Blurring and Sharpening
9. Thresholding & Binarization
10. Dilation, Erosion and Edge Detection
11. Contours - Drawing and Hierarchy and Modes
12. Moments, Matching and Sorting Contours
13. Line, Circle, Blob Detection
14. Counting Circles, Ellipses and Finding Waldo
15. Finding Corners
16. Face and Eye Detection with HAAR Cascade Classifiers
17. Vehicle & Pedestrian Detection
18. Perspective Transforms
19. Histograms and K-means clustering for finding dominant colours
20. Comparing Images with MSE and Structural Similarity

Our OpenCV Outline

21. Filtering Colors
22. Watershed Algorithm marker-based image segmentation
23. Background and Foreground Subtraction
24. Motion tracking using Mean Shift and CAM-Shift
25. Optical Flow Object Tracking
26. Simple Object Tracking by Colour
27. Facial Landmarks Detection with Dlib
28. Face Swapping with Dlib
29. Tilt Shift Effect
30. Grabcut Algorithm for Background Removal
31. OCR with PyTesseract and EasyOCR
32. Barcode and QR generation and reading
33. YOLOv3 in OpenCV
34. Neural Style Transfer with OpenCV
35. SSDs in OpenCV
36. Colorise Black and White Photos
37. Repair Damaged Photos with Inpainting
38. Add and remove Noise, Fix Contrast with Histogram Equalisation
39. Detect Blur in Images
40. Facial Recognition

Our OpenCV Outline

Video Section

1. Using Webcam to do Sketch
2. Opening Video Files
3. Saving and Recording Videos
4. Video Streams RTSP and IP
5. Auto Reconnect to Stream
6. Capturing video using screenshots
7. Import Youtube Video into OpenCV

Deep Learning Outline

Introduction to Deep Learning in Computer Vision

- Convolution Neural Networks
- Feature Detectors
- Convolution on Color Images
- Kernel Size and Depth
- Padding
- Stride
- Activation Layer ReLU
- Pooling
- Fully Connected Layer
- Softmax Layer
- Building a CNN
- Parameter Counts in CNNs
- Why CNNs work so well for Images
- How to Train a CNN
- Loss Functions
- Back Propagation
- Gradient Descent
- Optimisers
- Summary of CNNs

Deep Learning Outline

Deep Learning Basics with PyTorch and TensorFlow Keras

- Deep Learning History
- Deep Learning Libraries
- Building and Training Your First CNN with PyTorch
- PyTorch Transformations
- PyTorch Inspect and Visualise Dataset
- PyTorch Dataloader
- PyTorch Building our CNN
- PyTorch Loss Function and Optimiser
- PyTorch Training
- PyTorch Results
- PyTorch Plot Results
- Building and Training Your First CNN with Keras in TensorFlow 2 - Load Data
- Keras - Inspecting and Visualising
- Keras - Preprocessing
- Keras Constructing our CNN
- Keras - Training our Model
- Keras - Plotting our Results
- Keras - Saving and Loading, Visualising Results

Comparing Libraries and Analysing Performance

- Deep Learning Library Comparisons
- Assessing Model Performance
- Confusion Matrix and Classification Reports
- Keras - Loading Data and Viewign Misclassifications
- Keras - Confusion Matrix and Classification Report
- PyTorch - Loading Data and Viewign Misclassifications

Deep Learning Outline

Regularisation and Overfitting

- Overfitting and Generalisation
- Regularisation Methods
- L1 and L2 Regularisation
- Dropout
- Data Augmentation
- Early Stopping
- Batch Normalisation
- When to use regularisation
- Keras - FNIST No Regularisation
- Keras - FNIST with Regularisation
- PyTorch - FNIST No Regularisation
- PyTorch - FNIST with Regularisation

Understanding What CNN's See

- Visualising Filters of CNNs
- Filter Activations of CNNS
- Keras - Visualise Filter and Feature Maps
- Maximising Filters
- Maximising Class Activations
- Keras - Filter and Class Maximisation
- GradCAM
- Keras - GradCAM

Deep Learning Outline

Designing CNNs and Overview of Modern CNN Architectures

- Basic CNN Design Principles
- CNN History
- LeNet
- AlexNet
- VGGNet
- ResNet
- Why do ResNets Work
- MobileNet
- Inception
- SqueezeNet
- EfficientNet
- DenseNet
- ImageNet Dataset
- Keras LeNET and AlexNet
- PyTorch Pretrained Networks
- Keras Pretrained Networks
- Top-1 and Top-5 Accuracies
- PyTorch Rank N
- Keras Rank N
- Callbacks with PyTorch
- Callbacks with Keras

Deep Learning Outline

PyTorch Lightning

- PyTorch Lightning Setup and Class Design
- AutoBatch Selection & Auto LR Selection, Tensor-boards
- Callbacks, Saving and Inference
- mGPU, TPU more
- Transfer Learning Lightning

Transfer Learning

- Keras Transfer Learning and Fine Tuning
- Keras Using CNN as a Feature Extactor
- PyTorch Transfer Learning and Fine Tuning
- PyTorch Feature Extraction Transfer Learning
- PyTorch Feature Extraction

Google DeepDream

- Google DeepDream with PyTorch
- Google DeepDream with Keras

Neural Style Transfer

- Neural Style Transfer with PyTorch
- Neural Style Transfer with Keras

Autoencoders

- Autoencoders with Keras
- Autoencoders with PyTorch

Deep Learning Outline

Generative Adversarial Networks

- How do GANs Work?
- Training GANs
- Where do we use GANs?
- Keras DCGAN MNIST
- PyTorch DCGAN MNIST
- Super Resolution Keras
- Anime Characters - StyleGAN
- Horses to Zebras - CycleGAN
- ArcaneGAN

Siamese Networks

- Training Siamese Networks
- Keras Siamese
- PyTorch Siamese

Facial Recognition

- Facial Similarity Keras VGGFace
- Face Recognition Keras One Shot learning and Friends
- Facial Similarity PyTorch - FaceNet
- DeepFace - Age, Gender, Pose

Deep Learning Outline

Object Detection

- Early Object Detectors
- Intersection Over Union - Assessing Object Detector Performance
- Mean Average Precision (mAP)
- Non-Maximum Suppression
- Object Detection - R-CNNs - COCO
- Object Detection - SSDs
- Object Detection - Introduction to the YOLO Object Detectors
- Object Detectors - How Does YOLO Work?
- Object Detectors - Training YOLO
- Object Detectors - YOLO Architecture and Evolution from YOLOv3 to v5
- Object Detection - EfficentDet
- Object Detection - Detectron2
- Object Detection - Weapons Recognition - Guns SclaedYOLOv4
- Object Detection - Mask Detection - MobileNetV2 SSD
- Object Detection - Sign Language - TFODAPI - EfficientNetD0-D7
- Object Detection - Mushroom Detection - Detectron2
- Object Detection - Pothole Detection - TinyYOLOv4
- Object Detection - Website Region - YOLOV4 Darknet
- Object Detection - Drone Maritime Faster R-CNN
- Object Detection - Chess Pieces YOLOv3 PyTorch
- Object Detection - Hardhat Detection using EfficentDet
- Object Detection - Bloodcell Detector YOLOv3
- Object Detection - Plant Doctor YOLOv5

Deep Learning Outline

Deep Segmentation

- Deep Segmentation - U-Net & SegNet - Nuclie
- Deep Segmentation - SegNet Medical
- MaskCNN Demo
- Detectron2 MaskRCNN
- Detectron2 Bodybose
- MaskCNN Shapes

Tracking with Deep Learning

- DeepSort - Track Cars

Deep Learning Outline

Miscellaneous Deep Learning Tutorials

- DeepFakes
- Vision Transformers
- ViT PyTorch
- ViT Keras
- BiT Keras
- Monocular depth estimation
- Image Similarity using Metric Learning
- Image Captioning
- Video Classification with a CNN-RNN Architecture
- Video Classification using Transformers
- Point Cloud Classification
- Point Cloud Segmentation
- 3D image classification from CT scans
- X-Ray Pneumonia Classification
- OCR model for reading Captchas
- WebApp - Computer Vision using Flask
- WebApp - Computer Vision using Flask Site

MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

Introduction to Computer Vision

MODERN COMPUTER VISION

BY RAJEEV RATAN

What Makes Computer Vision Hard?

Nothing worth having comes easy

Our Brains are amazing at Vision

The **visual cortex** (located in the occipital lobe) is the primary cortical region of the brain that receives, integrates, and processes visual information relayed from the retinas

Can Artificial Intelligence Come Close?

The AI Dream, a machine that can see and understand better than humans.

Source - Terminator 2: Judgement Data (1991)

What Makes Computer Vision Hard?

Camera and Senor Limitations

What Makes Computer Vision Hard?

Viewpoint Variations

What Makes Computer Vision Hard?

Changing Lighting Conditions

What Makes Computer Vision Hard?

Scaling Issues

What Makes Computer Vision Hard?

Non-rigid Deformations

What Makes Computer Vision Hard?

Occlusion

What Makes Computer Vision Hard?

Clutter

What Makes Computer Vision Hard?

Object Class Variation

What Makes Computer Vision Hard?

Ambiguous Optical Illusions

MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

What are Images?

MODERN COMPUTER VISION

BY RAJEEV RATAN

What Are Images?

A foundation in understanding digital images

How do Computers ‘see’ Images?

What are Images?

- Images are 2-D representations of the **visible light spectrum**

Image Creation

- Digital images are created either by using software on computer (think clip art or logos) or by digital cameras (or scanners).

- The pinhole camera model is the basis of how we form an image using a photosensitive sensor

Modern Cameras

Our eyes as well as cameras use an **adaptive lens** to control many aspects of image formation such as:

- **Aperture Size**
  - Controls the amount of light allowed through (f-stops in cameras)
  - Depth of Field (Bokeh)
- **Lens width** - Adjusts focus distance (near or far)

Digital Images

- Computers store images in a variety of formats, however they all involve mixing some of the following such as: colours, hues, brightness or saturation.

- The one most commonly used in Computer Vision is **RGB (Red, Green, Blue)**

- Each pixel is a combination of the brightness of blue, green and red, ranging from 0 to 255 (brightest)

- Yellow is represented as:
  - Red – 255
  - Green – 255
  - Blue - 0

Digital Images Format

Images are stored in Multi-Dimensional Arrays

- A 1-Dimensional array looks like this

- A 2-Dimensional array looks like this

- A **3-Dimensional** array looks like this

A Grayscale Image

Sometimes referred to as black and white image

- Info is stored in a 2-Dim array

255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	220	146	237	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	91	[illegible]	170	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	253	68	[illegible]	220	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	238	30	62	241	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	218	[illegible]	64	241	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	145	[illegible]	64	241	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	81	[illegible]	72	244	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	72	[illegible]	128	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	73	[illegible]	146	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	64	[illegible]	184	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	63	[illegible]	188	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	71	[illegible]	187	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	68	[illegible]	190	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	83	[illegible]	160	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	181	48	184	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255

Image File Formats

MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

Let’s get Started with Google Colab and the Course setup

MODERN COMPUTER VISION

BY RAJEEV RATAN

Introduction to Convolutional Neural Networks

A Gentle Overview of CNNs

Understanding Images

Predicted to be a Cat

Predicted to be a Dog

What Do We See?

- Many things indicated it was a cat or dog
- Whiskers, shape, eyes, fur, colour etc.
- Can an Algorithm or Predictive Model do this?

Convolutional Neural Networks

Building an Intuition for CNNs

- What digit is this?
- How would we program a computer to know this?
  - Overall shape?

  - Maybe if something lies in this region only it's a one?

- But what if it's shifted?

What If We Had Filters that Could Scan All Parts of An Image?

Overview of our CNN Chapter

- Convolutions
- Feature Detectors
- 3D Conv Filters - Convolution on Color Images
- Kernel Size and Depth
- Padding
- Stride
- Activation Layer - ReLU
- Pooling
- Fully Connected Layer
- Softmax
- Building a CNN
- Parameter Counts in CNNs
- Why CNNs Work so Well For Images
- The Training Process Part 1 - Loss Functions
- The Training Process Part 2 - Back Propagation
- The Training Process Part 3 - Gradient Descent
- Optimisers and Learning Rate Schedules
- Summary of CNNs

MODERN COMPUTER VISION

BY RAJEEV RATAN

Convolutions

How We Detect Features in Images

The Convolution Operation

- Mathematical term to describe the process of combining **two** functions to produce a **third**
- In our situation, the output is called a Feature Map
- We use a matrix, called a Filter or Kernel that is applied to our Image
- So the first 'Function' is the image that is combined with the Kernel or Filter which produces a Feature Map

$$\text{Image} \times \text{Kernel} = \text{Feature Map}$$

Example of a Convolution Operation

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3
2	1	1

Output or Feature Map

Recall grayscale images are just intensity values ranging from black to white

255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	220	146	237	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	91	[illegible]	170	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	253	68	[illegible]	220	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	238	30	62	241	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	218	[illegible]	64	241	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	145	[illegible]	64	241	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	81	[illegible]	72	244	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	72	[illegible]	128	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	73	[illegible]	146	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	64	[illegible]	104	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	63	[illegible]	188	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	71	[illegible]	187	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	68	[illegible]	190	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	83	[illegible]	160	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	181	48	184	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255

Example of a Convolution Operation

$$(1 \times 0) + (0 \times 1) + (1 \times 0) + (1 \times 1) + (0 \times 0) + (0 \times -1) + (0 \times 0) + (1 \times 1) + (1 \times 0) = 2$$

1x0	0x1	1x0	0	1
1x1	0x0	0x-1	1	1
0x0	1x1	1x0	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2		

Output or Feature Map

Example of a Convolution Operation

$$(0 \times 0) + (1 \times 1) + (0 \times 0) + (0 \times 1) + (0 \times 0) + (1 \times -1) + (1 \times 0) + (1 \times 1) + (0 \times 0) = 1$$

1	0x0	1x1	0x0	1
1	0x1	0x0	1x-1	1
0	1x0	1x1	0x0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	

Output or Feature Map

Example of a Convolution Operation

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1

Output or Feature Map

Example of a Convolution Operation

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1		

Output or Feature Map

Example of a Convolution Operation

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	

Output or Feature Map

Example of a Convolution Operation

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3

Output or Feature Map

Example of a Convolution Operation

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3
2		

Output or Feature Map

Example of a Convolution Operation

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3
2	1	

Output or Feature Map

Example of a Convolution Operation

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3
2	1	1

Output or Feature Map

Image Features

- Our feature maps are actually **Feature Detectors**
- **Why did we do this?**
- Because Convolution Filters or Kernels **detect features** in images

Our Convolution Filter as an Edge Detector

1	1	0	0	0
1	1	0	0	0
1	1	0	0	0
1	1	0	0	0
1	1	0	0	0

Input Image

*

1	0	-1
1	0	-1
1	0	-1

Filter or Kernel

=

3	3	0
3	3	0
3	3	0

Output or Feature Map

Convolution Filters Detect Many Features

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

Convolution Filters Detect Many Features

Example filters learned by Krizhevsky et al.

Convolution Filters Detect Many Features

Source – Deep Learning Methods for Vision
https://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

Calculating Feature Map Size

Feature Map Size = $n - f + 1 = m$

Feature Map Size = $5 - 3 + 1 = 3$

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

5 x 5

$n \times n$

*

0	1	0
1	0	-1
0	1	0

3 x 3

$f \times f$

=

2	1	-1
-1	1	3
2	1	1

3 x 3

$m \times m$

MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

Feature Detectors

MODERN COMPUTER VISION

BY RAJEEV RATAN

Feature Detectors

How Feature Detectors Help Classify Images

How Classifiers Use Feature Detectors

We perform Convolution operations with the input image and our filters

Uses the Feature Maps we output to determine the image class

How Classifiers Use Feature Detectors

Conv Filters

Uses the Feature Maps we output to determine the image class

Cat = 0.99
Dog = 0.01

We perform Convolution operations with the input image and our filters

In the Past Hand Engineered Features Were Often Used

HoGs, SIFT, SURF, ORB, PHOG, Haar etc.

Yan & Huang
(Winner of PASCAL 2010 classification competition)

Felzenszwalb, Girshick,
McAllester and Ramanan, PAMI 2007

Hand crafting Features is VERY hard, messy and leads to often poor results…

CNNs solved this by having the ability to Learn Features

The Layers of a CNN

- Convolution
- ReLU
- Pooling Layers
- Fully Connected or Dense Layer
- Output

MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

Convolution on Color Images

MODERN COMPUTER VISION

BY RAJEEV RATAN

Convolution on Color Images

How we use 3D Volumes of Convolution Filters

Convolution on our Grey Scale Image

1	1	0	0	0
1	1	0	0	0
1	1	0	0	0
1	1	0	0	0
1	1	0	0	0

Input Image

*

1	0	-1
1	0	-1
1	0	-1

Filter or Kernel

=

3	3	0
3	3	0
3	3	0

Output or Feature Map

Convolution Operations on Color Images

Sum all the numbers to get the Feature Map Output

Input Image

Filter or Kernel

Output or Feature Map

Advantages of Having a Filter For Each Colour

- We can detect features that are specific to a colour

Considered 3D Volumes

Input Image
5 x 5 x **3**

*

Filter or Kernel
3 x 3 x **3**

=

3	3	0
3	3	0
3	3	0

Output or Feature Map
3 x 3

How Multiple Filters Affect Our Output

Input Image
5 x 5 x 3

*

2 Filters or Kernels
3 x 3 x 3 x 2

=

Output or Feature Map
3 x 3 x 2

Calculating Output Size for 3D Conv Volumes

$$(n \times n \times n_c) * (f \times f \times n_c) = (n - f + 1) \times (n - f + 1) \times n_f$$

$$(5 \times 5 \times 3) * (3 \times 3 \times 3) = 3 \times 3 \times 2$$

Input Image
5 x 5 x **3**

*

2 Filters or Kernels
3 x 3 x **3** x **2**

=

Output or Feature Map
3 x 3 x **2**

Next…

Kernel Size and Depth

MODERN COMPUTER VISION

BY RAJEEV RATAN

Kernel Size and Depth

We look at some of parameters that define our Conv Filter

Parameters that Control the Conv Filter

- Kernel Size (k x k)
- Depth (1 for grayscale or 3 for RGB)
- Stride
- Padding

Sizing Convolution Filters

- In the previous example we used a 3 x 3 Filter or Kernel
- Can we use other sizes?

1	1	0	0	0
1	1	0	0	0
1	1	0	0	0
1	1	0	0	0
1	1	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

3	3	0
3	3	0
3	3	0

Sizing Convolution Filters

- Yes we can use larger filters/kernels

Feature Map Size = $n - f + 1 = m$

Feature Map Size = $6 - 5 + 1 = 2$

1	1	0	0	0	0
1	1	0	0	0	0
1	1	0	0	0	0
1	1	0	0	0	0
1	1	0	0	0	0
1	1	0	0	0	0

6 x 6

*

1	0	-1	0	1
1	0	-1	0	1
1	0	-1	0	1
1	0	-1	0	1
1	0	-1	0	1

5 x 5

=

3	3
3	3

Output or Feature Map

Odd Number vs Even Number for Filter Dimensions

- Odd sized filters are symmetrical around the centre pixel or anchor point
- A lack of symmetry here results in distortions across layers

source:https://towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f9363

Depth

- Depth typically refers to the **colour channels**
- However, in some nomenclature it can refer to the 3rd dimension of any layer in our CNN e.g. our Feature Map has depth of 4

Input Image
5 x 5 x **3**

*

4 Filters or Kernels
3 x 3 x **3** x **4**

=

Output or Feature Map
3 x 3 x **4**

Next…

Padding

MODERN COMPUTER VISION

BY RAJEEV RATAN

Padding

Manipulating the input size

Padding

Notice How Conv Filters Produce an Output Smaller than the Input

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

5 x 5

$n \times n$

*

0	1	0
1	0	-1
0	1	0

3 x 3

$f \times f$

=

2	1	-1
-1	1	3
2	1	1

3 x 3

$m \times m$

CNNs can have several sequences of Convolution layers

Consecutive Conv layers would keep shrinking the output

Can we preserve our image size?

We've added a 1 pixel pad of zeros (zero padding) around our input

0	0	0	0	0	0	0
0	1	0	1	0	1	0
0	1	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0

*

0	1	0
1	0	-1
0	1	0

=

7 x 7

$n \times n$

3 x 3

$f \times f$

Padding

Let's Perform our Convolution with the Padding

0	0	0	0	0	0	0
0	1	0	1	0	1	0
0	1	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0

*

0	1	0
1	0	-1
0	1	0

=

7 x 7

$n \times n$

3 x 3

$f \times f$

Padding

Let's Perform our Convolution with the Padding

0	0	0	0	0	0	0
0	1	0	1	0	1	0
0	1	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0

7 x 7

$n \times n$

*

0	1	0
1	0	-1
0	1	0

3 x 3

$f \times f$

=

Padding

Let's Perform our Convolution with the Padding

0	0	0	0	0	0	0
0	1	0	1	0	1	0
0	1	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0

*

0	1	0
1	0	-1
0	1	0

=

7 x 7

$n \times n$

3 x 3

$f \times f$

Padding

Let's Perform our Convolution with the Padding

0	0	0	0	0	0	0
0	1	0	1	0	1	0
0	1	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0

7 x 7

$n \times n$

*

0	1	0
1	0	-1
0	1	0

3 x 3

$f \times f$

=

Padding

Let's Perform our Convolution with the Padding

0	0	0	0	0	0	0
0	1	0	1	0	1	0
0	1	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0

*

0	1	0
1	0	-1
0	1	0

=

7 x 7

$n \times n$

3 x 3

$f \times f$

Padding

Let's Perform our Convolution with the Padding

0	0	0	0	0	0	0
0	1	0	1	0	1	0
0	1	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0

7 x 7

$n \times n$

*

0	1	0
1	0	-1
0	1	0

3 x 3

$f \times f$

=

Padding

Let's Perform our Convolution with the Padding

Feature Map Size = $n - f + 1 = m$

Feature Map Size = $7 - 3 + 1 = 5$

0	0	0	0	0	0	0
0	1	0	1	0	1	0
0	1	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0

7 x 7

$n \times n$

*

0	1	0
1	0	-1
0	1	0

3 x 3

$f \times f$

=

2	1	-1	2	2
-1	1	3	2	1
2	1	1	1	2
1	1	1	0	2
2	0	2	3	1

5 x 5

$m \times m$

Why Use Padding?

- For very deep networks we don't want to keep reducing the size
- Pixels at the edges contribute less to the output Feature Maps, thus we're throwing away information from them

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Without padding, our top left pixel is only touched by the Conv Filter once

Whereas, our centre pixel is passed over numerous times

Next…

Stride

MODERN COMPUTER VISION

BY RAJEEV RATAN

Stride

We look at a parameter that defines how we move our Conv Filter

Stride

Our Step Size

- Stride defines how many steps we take when sliding our Convolution Window across the input image

What a Stride of 1 Looks Like

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2		

Output or Feature Map

What a Stride of 1 Looks Like

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	

Output or Feature Map

What a Stride of 1 Looks Like

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1

Output or Feature Map

What a Stride of 1 Looks Like

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1		

Output or Feature Map

What a Stride of 1 Looks Like

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	

Output or Feature Map

What a Stride of 1 Looks Like

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3

Output or Feature Map

What a Stride of 1 Looks Like

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3
2		

Output or Feature Map

What a Stride of 1 Looks Like

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3
2	1	

Output or Feature Map

What a Stride of 1 Looks Like

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3
2	1	1

Output or Feature Map

What about a Stride of 2?

What a Stride of 2 Looks Like

We start off in the same position

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	

Output or Feature Map

What a Stride of 2 Looks Like

Now we jump two spots to the left

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	-1

Output or Feature Map

What a Stride of 2 Looks Like

Now we go down, but by also 2 spots

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	-1
2	

Output or Feature Map

What a Stride of 2 Looks Like

Now we jump two spots to the left

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	-1
2	1

Output or Feature Map

Stride Observations

- A larger Stride produced a **smaller** Feature Map output
- Larger Stride has **less overlap**
- We can use stride to **control the size of the Feature Map output**

Calculating Output Size

Using Stride and Padding

Stride = 2
Padding = 0

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image
5 x 5

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel
3 x 3

=

2	-1
2	1

$$(n \times n) * (f \times f) = (\frac{n + 2p - f}{s} + 1) \times (\frac{n + 2p - f}{s} + 1) = (\frac{5 + (2 \times 0) - 3}{2} + 1) \times (\frac{5 + (2 \times 0) - 3}{2} + 1) = 2 \times 2$$

Calculating Output Size

Using Stride and Padding

Stride = 1
Padding = 0

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image
5 x 5

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel
3 x 3

=

2	-1	1
2	1	0
1	-1	1

$$(n \times n) * (f \times f) = (\frac{n + 2p - f}{s} + 1) \times (\frac{n + 2p - f}{s} + 1) = (\frac{5 + (2 \times 0) - 3}{1} + 1) \times (\frac{5 + (2 \times 0) - 3}{1} + 1) = 3 \times 3$$

Calculating Output Size

When using Stride & Padding

Stride = 1

Padding = 1

0	0	0	0	0	0	0
0	1	0	1	0	1	0
0	1	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0

Input Image
7 x 7

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel
3 x 3

=

2	-1	1	1	0
2	1	0	1	2
1	-1	1	0	1
0	1	2	1	1
2	0	1	0	2

$$(n \times n) * (f \times f) = (\frac{n + 2p - f}{s} + 1) \times (\frac{n + 2p - f}{s} + 1) = (\frac{5 + (2 \times 1) - 3}{1} + 1) \times (\frac{5 + (2 \times 1) - 3}{1} + 1) = 5 \times 5$$

Note if we get non integer value for our output size, we found it down to the nearest integer

MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

Activation Layer ReLU

MODERN COMPUTER VISION

BY RAJEEV RATAN

Activation Layer ReLU

What are Activation Functions and their importance

Purpose of Activation Functions

To enable the learning of **complex patterns** in our data

- Biological neurons fire (activate) on certain inputs, these are then fed into other neurons
- Introduces **non-linearity** to our network
- This allows a non-linear decision boundary via non-linear combinations of the weight and inputs

A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Stanford's CS231 Course

Types of Activation Functions

There are several activation functions we can use in our CNN. However, Rectified Linear Units (ReLU) have become the activation function of choice for CNNs.

ReLU is advantageous in CNN Training:

- Simple Computation (fast to train)
- Does not saturate

Sigmoid

$$\phi(z) = \frac{1}{1+e^{-z}}$$

Hyperbolic Tangent

$$\phi(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

Rectified Linear

$$\phi(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$

Radial Basis Function

$$\phi(z, c) = e^{-(\epsilon\|z-c\|)^2}$$

The ReLU Operation

- Change all negative values to 0
- Leave all positive Values alone

$$f(x) = max(0,x)$$

Applying the ReLU Activation

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3
2	1	-5

Output or Feature Map

ReLU

2	1	0
0	1	3
2	1	0

Applying the ReLU Activation

Example of a Rectified Linear Map

Source - http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

Pooling

MODERN COMPUTER VISION

BY RAJEEV RATAN

Pooling

We explore the Max Pool Layer and it’s purpose

Pooling

- Pooling is the process whereby we reduce the size or dimensionality of the Feature Map
- This allows us to reduce the number of Parameters in our Network whilst retaining important features
- Also called Subsampling or Downsampling

Example of Max Pooling

4	123	1	34
56	99	222	253
45	122	165	12
21	187	133	124

MaxPool Operation

Stride = 2

Kernel = 2x2

123	167
187	165

More on Pooling

- Typically we use 2x2 kernels and a Stride of 2 with no padding.
- With the above setting, pooling reduces the dimensionality by a factor of 2 (width and height).
- Pooling makes our model more invariant to minor transformations and distortions in our image.
- We can also use Average Pooling or Sum Pooling

How Max Pooling Achieves Translation Invariance

Shifting Our C down one pixel

Why Pooling Works

Pooling reduces our feature map size by half in most cases, is that ok?

- Neighbouring pixels are **strongly correlated**, especially in lowest layers
- Remember further apart two pixels are from each other, the less correlated
- Therefore, we can **reduce the size** of the output by subsampling (**pooling**) the filter response **without losing information**
- A big stride in the pooling layer leads to high information loss
- In practice, a stride of 2 and a kernel size 2x2 for the pooling layer was found to be effective in practice

Next…

Fully Connected Layer

MODERN COMPUTER VISION

BY RAJEEV RATAN

Fully Connected Layer

We explore the Fully Connected Layer and it’s purpose

What does Fully Connected Mean?

- It means all nodes in one layer are connected to the outputs of the next layer.
- It takes the 3D Volume output of the previous layer and flattens it into a single vector that is used for input in the next layer.
- It's sometimes called the Dense Layer

FC Layer - The Max Pool Layer is Flattened

Purpose of the Fully Connected Layer

- It compiles the data/outputs extracted from previous layers to form the final output
- It's an easy way of learning non-linear combinations of these features

Next…

Softmax Layer

MODERN COMPUTER VISION

BY RAJEEV RATAN

Softmax Layer

Softmax Layer to Probabilities

The Fully Connected Layer

Softmax Layer

- We now need to produce probability outcomes for each class in Network.
- Softmax converts the Logs into Probabilities
- It takes the exponents of every output and the normalises each output by the sum of the exponents.
- It guarantees a well behaved distribution i.e. all sum to 1 and no values are zero.

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j))}$$

Softmax Layer

Logits Scores

2.0

1.0

0.1

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j))}$$

Probabilities

0.7

0.2

0.1

MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

Building a CNN