CHECK-IN

INTRODUCTION TO MACHINE LEARNING (PART 5 OF 5)

TUESDAY, JANUARY 23 AT 2:00PM

https://cglink.me/2gj/c19387721116154635

1. **Open** the **My PrincetonU** app.
2. Select a Hub
3. Click on QR Code scanner.
4. Scan this QR Code and you are checked-in!

Download on the App Store

GET IT ON Google Play

Computer Vision

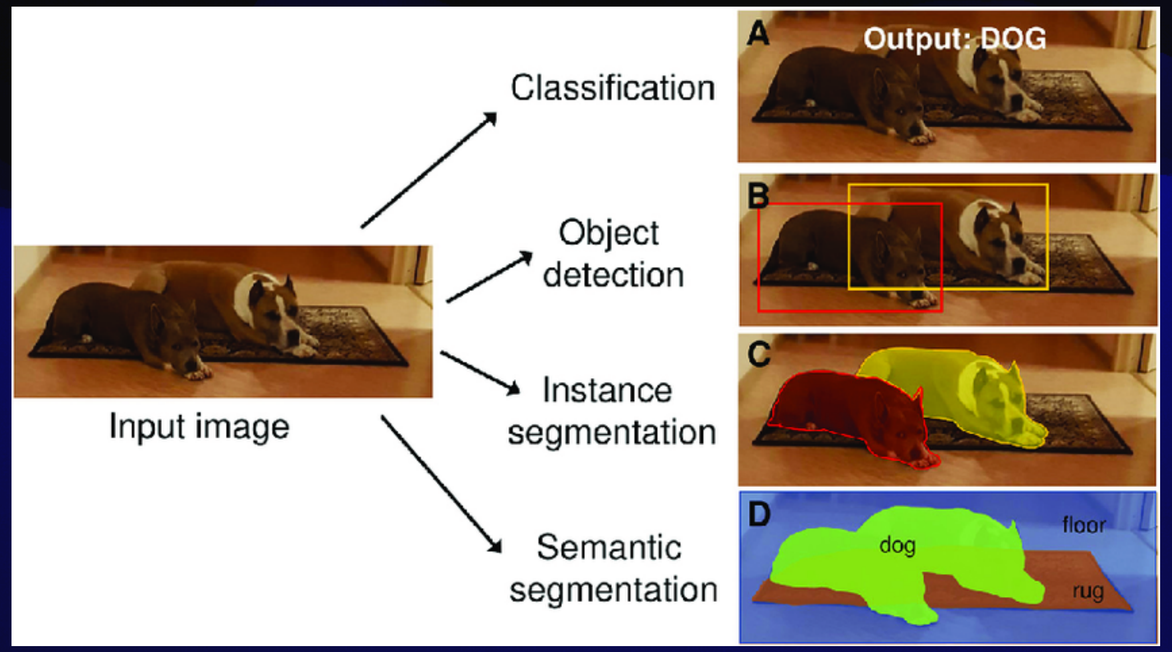
with Convolutional Neural Networks

Gage DeZoort | 1/23/2024

Computer Vision

Common Tasks

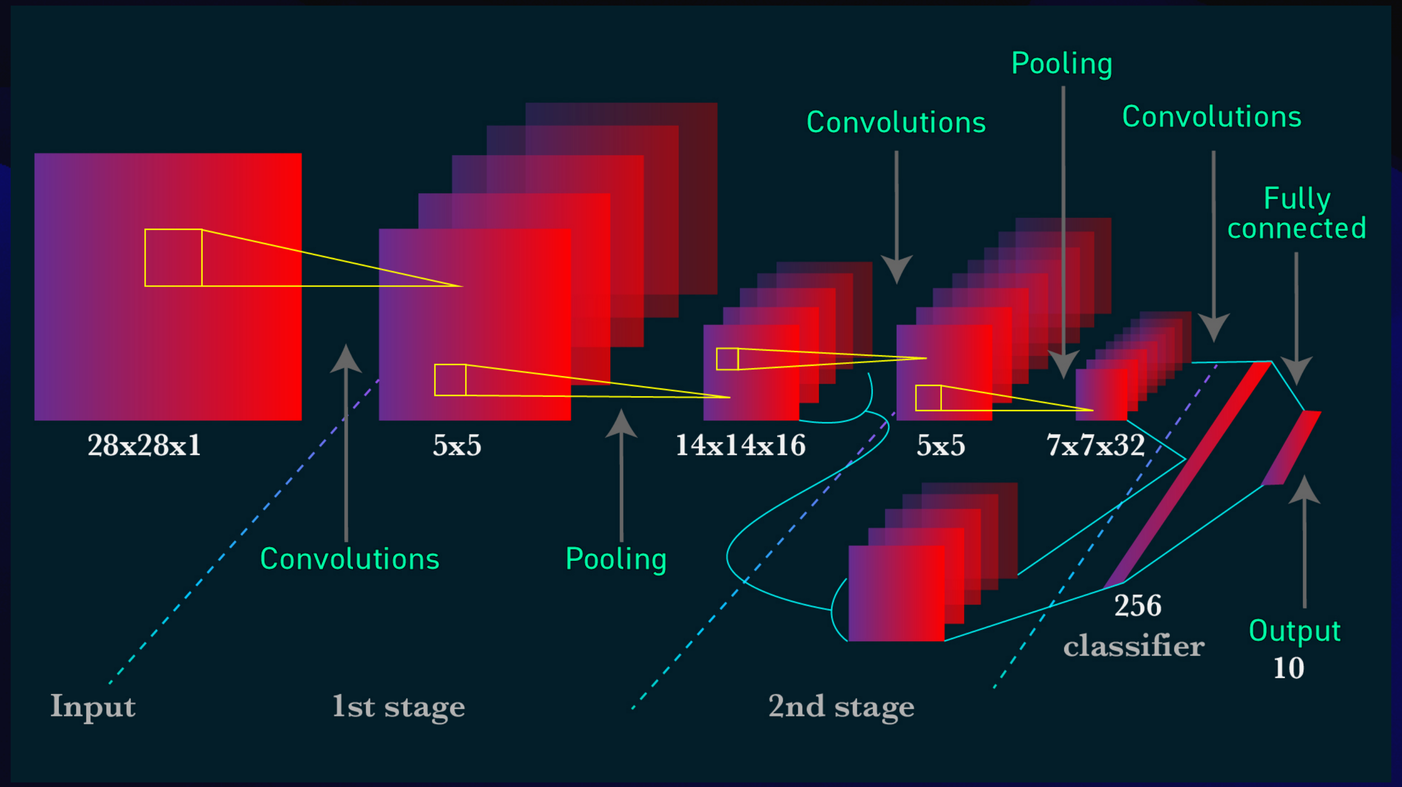
- **Classification**: predict one image label
- **Object detection**: predict bounding boxes for objects in image
- **Instance segmentation**: predict pixel boundaries for objects in images
- **Semantic segmentation**: predict class labels per pixel of objects in image



10.1556/1647.2022.00104

Image Classification

A Simple Introduction

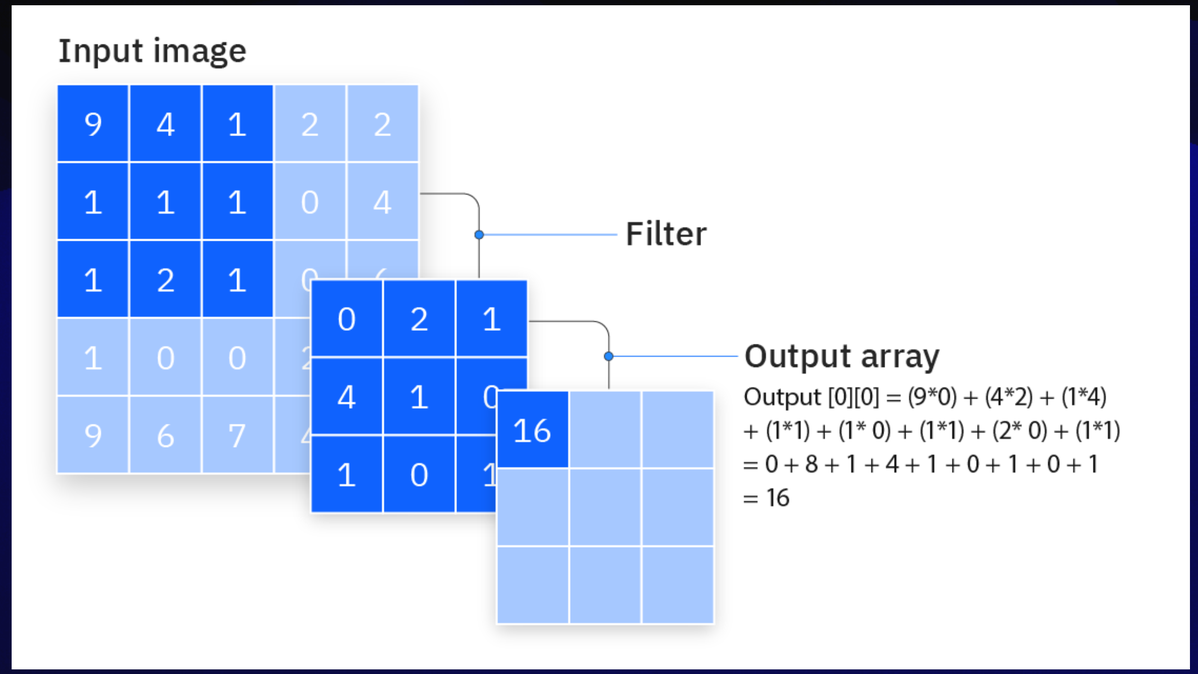


https://analyticsindiamag.com/convolutional-neural-network-image-classification-overview/

Image Classification

A Simple Introduction

- **Convolutional layers:** learnable filters (or “kernels”) are applied to patches of the image
- Filters look for specific features in the image —> feature learning
- Parameter sharing: same filter applied to every image patch



https://analyticsindiamag.com/convolutional-neural-network-image-classification-overview/

Image Classification
Formalities

- **Input Shape:** (N, C, H, W)
  - N: batch size
  - C: number channels (1 for greyscale, 3 for RGB)
  - H: image height (# pixels)
  - W: image height (# pixels)

- PyTorch Convolutional Layer:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

  - Convolution over one image at batch dimension *I* and one output feature dimension *j*
  - weight(C_{out_j}, k) is the filter applied to the input image input(N_i, k) in the *k*-th feature
  - The resulting feature maps are summed and a bias is applied

See: https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

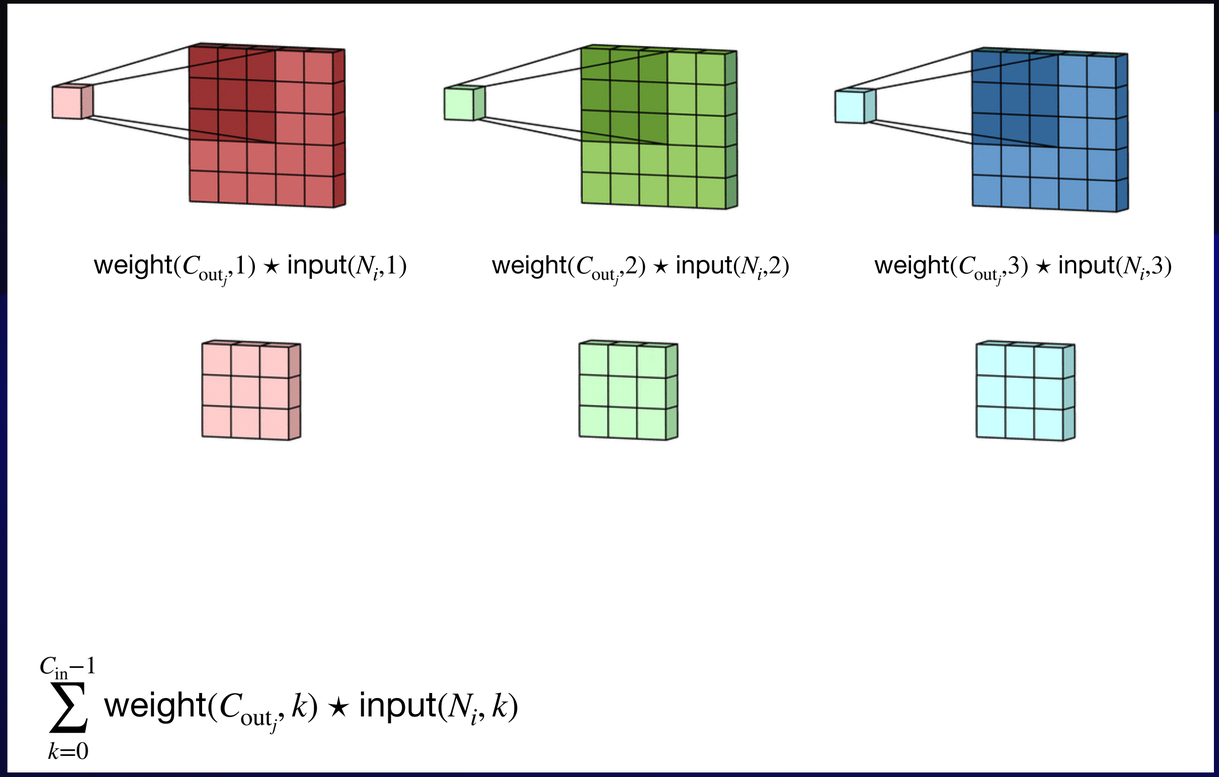
Image Classification

Visualizing the Formalism

- PyTorch Convolutional Layer:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

- Start with a (1x3x5x5) image batch containing 1 RGB image (3 input channels)
- Use a 3x3 Conv2D filter to produce a 1x1x3x3 representations in each input channel
- Sum the representations to achieve a 1x3x3x3 representation. Add a bias.
- Repeat for every requested output channel

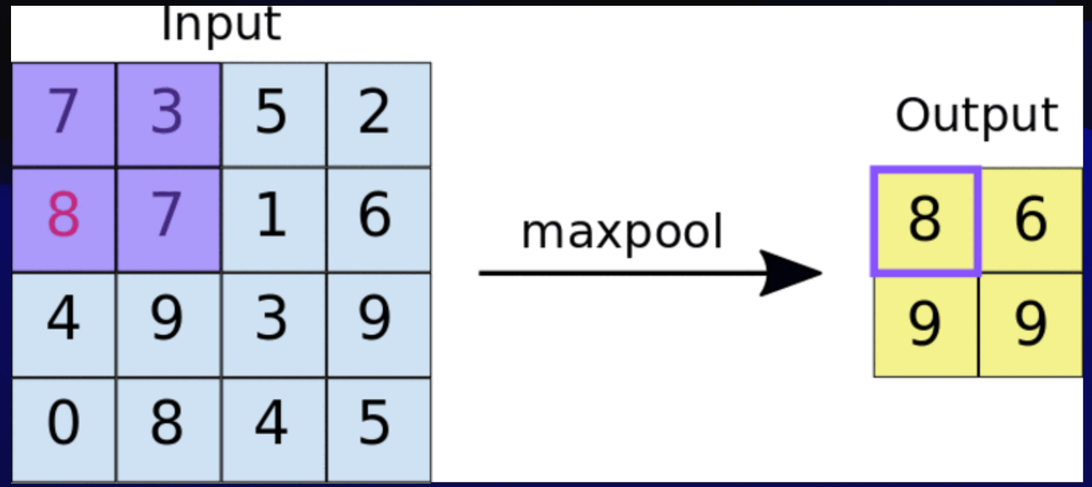


https://analyticsindiamag.com/convolutional-neural-network-image-classification-overview/

Image Classification

Pooling Layers

- **Convolutional Layers** produce downsampled feature maps via learnable weights
- **Pooling Layers** produce downsampled feature maps via fixed operations (max, mean)
  - In PyTorch, the right operation uses MaxPool2D(2,2) - i.e. a 2x2 kernel with stride=2.
  - This reduces a 1x4x4 image to a 1x2x2 image



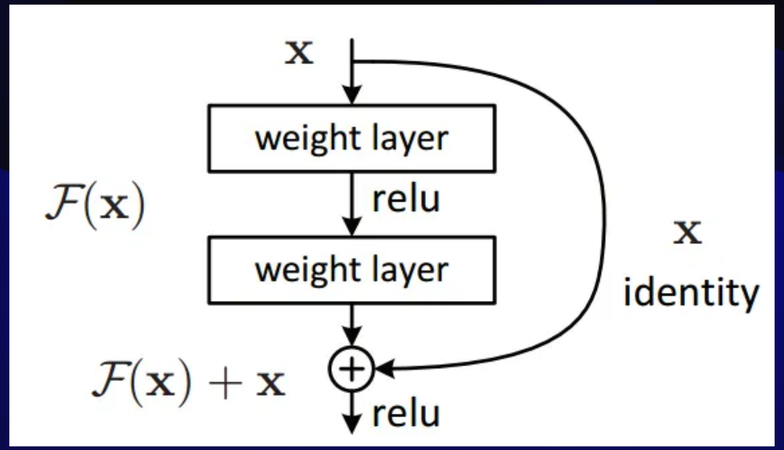
ResNet50
Quick Overview

arXiv
https://arxiv.org › cs

[1512.03385] Deep Residual Learning for Image Recognition

by K He · 2015 · Cited by 197465 — Title:**Deep Residual Learning** for **Image Recognition**. Authors:**Kaiming He**, **Xiangyu Zhang**, **Shaoqing Ren**, **Jian Sun**. Download a PDF of the…

- 50-layer CNN composed of 48 convolutional layers, one max pool layer, and one average pool layer
- The “Res” means it uses residual blocks
  - Residual blocks allow training to very large depth by making it possible to “skip” layers when necessary (skip connections)
  - Avoids the exploding/vanishing gradient problem




SCAN ME