Software

Introduction to Convolutional Neural Networks

Modified from Intel's original slides: ' Introduction to Convolutional Neural Networks '

Legal Notices and Disclaimers

Motivation – Image Data

- Traditional neural networks treat all inputs as interchangeable, ignoring relationships between individual inputs.
- Inputs are processed as an ordered set of variables, lacking spatial context.
- Goal: Integrate domain knowledge into the neural network architecture, to better capture structure and relationships in image data.

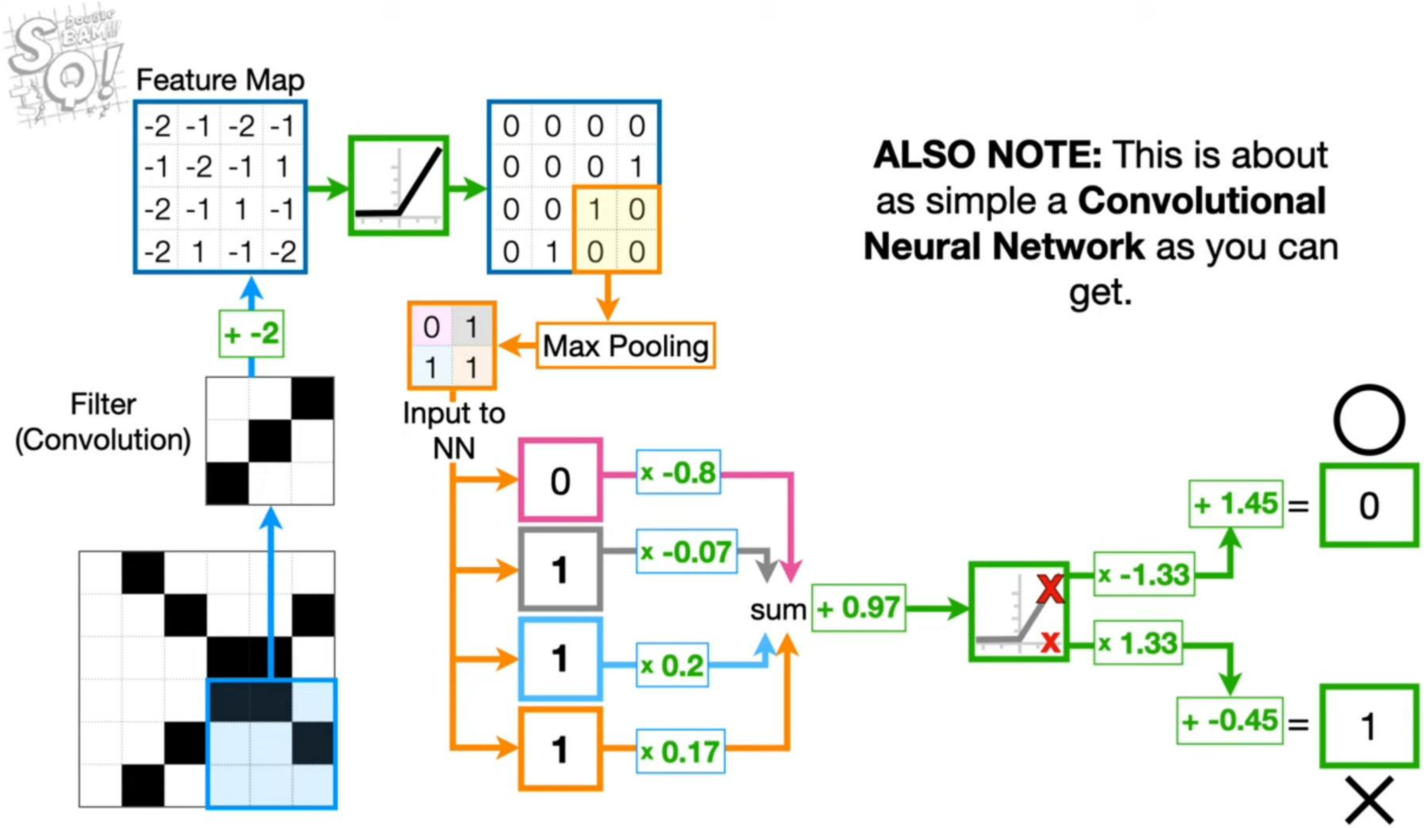
Motivation

- Image data has important structures, such as:
  - “**Topology**” of pixels: spatial relationships between pixels
  - **Local Similarity**: nearby pixels often have similar values
  - **Edges and Shapes**: fundamental building blocks of images
  - **Translation Invariance:** objects retain meaning when shifted
  - **Scale Invariance**: objects may appear at different sizes
  - **Lighting and Contrast:** variations affect pixel intensity
  - **Human Visual System Insights**: leverage natural perception

Motivation – Image Data

- Fully connected networks require an impractical number of parameters for image data.
- MNIST images (28×28 grayscale) are manageable, but typical images (e.g., 200×200 RGB) have ~120,000 input values.
- A single fully connected layer would require $(200\text{x}200\text{x}3)^2 =$ 14,400,000,000 weights! (~14.4 billion weights).
- High parameter count increases variance (overfitting).
- Convolutional layers reduce parameters by focusing on local patterns, introducing "bias" into the network.

Image Classification with Convolutional Neural Networks (CNNs)
(from StatQuest with Josh Starmer)



ALSO NOTE: This is about as simple a **Convolutional Neural Network** as you can get.

Motivation

- Features in images are hierarchically composed.
  - Edges → Shapes → Relationships between shapes → Textures.
- Example: Recognizing a cat.
  - **Cat**: two eyes in a specific arrangement + fur texture.
  - **Eyes**: dark circle (pupil) inside another circle.
  - **Circle**: combination of edge detectors.
  - **Fur**: repeating edge patterns.

Kernels (Filters)

- A *kernel* is a grid of weights applied to an image, centered on a pixel.
- Each weight is multiplied by the corresponding pixel value beneath it.
- Output for the centered pixel:

$$\sum_{p=1}^{P} W_p \cdot pixel_p$$

- Kernels are fundamental in traditional image processing techniques:
    - Blur
    - Sharpen
    - Edge detection
    - Emboss

Kernel: 3x3 Example

Input

3	2	1
1	2	3
1	1	1

Kernel

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

Output

Kernel: 3x3 Example

Kernel: 3x3 Example

Input

3	2	1
1	2	3
1	1	1

Kernel

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

Output

	2	

$$= (3 \cdot -1) + (2 \cdot 0) + (1 \cdot 1)$$
$$+(1 \cdot -2) + (2 \cdot 0) + (3 \cdot 2)$$
$$+(1 \cdot -1) + (1 \cdot 0) + (1 \cdot 1)$$

$$= -3 + 1 - 2 + 6 - 1 + 1 = 2$$

Kernels as Feature Detectors

Can think of kernels as a "local feature detectors"

Vertical Line Detector

-1	1	-1
-1	1	-1
-1	1	-1

Horizontal Line Detector

-1	-1	-1
1	1	1
-1	-1	-1

Corner Detector

-1	-1	-1
-1	1	1
-1	1	1

Edge Detection

0	1	0
1	-4	1
0	1	0

"Strong" Edge Detection

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

Try Convolution Yourself visit https://setosa.io/ev/image-kernels

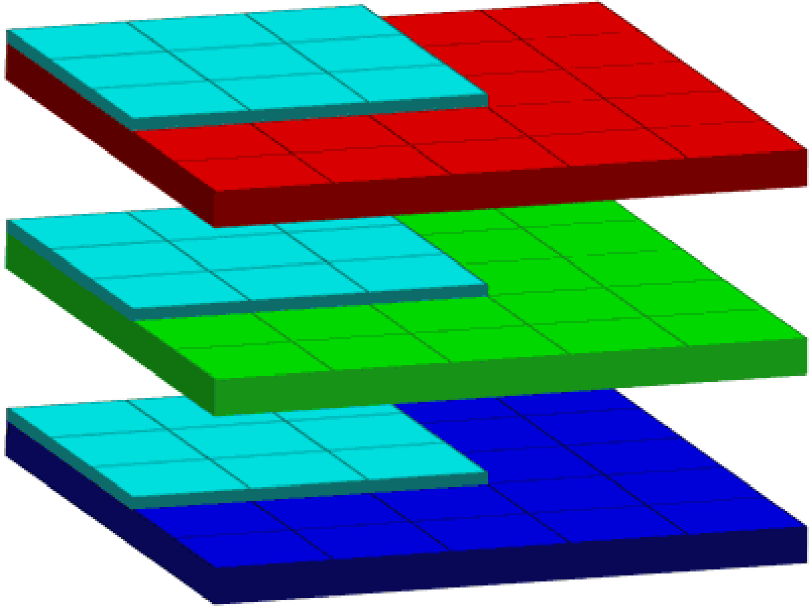
Convolutional Neural Nets

Primary Ideas behind Convolutional Neural Networks:

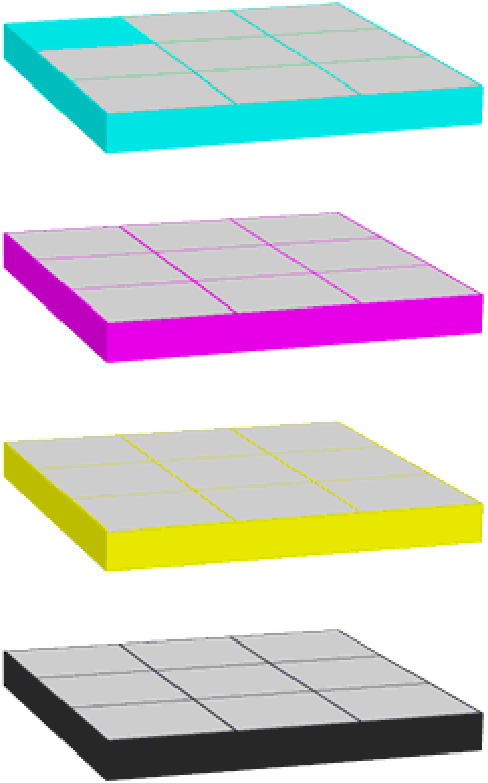
- Neural networks learn optimal kernels during training.
- Apply the same kernels across the entire image (translation invariance).
- Reduces parameters and minimizes overfitting by controlling variance (bias-variance tradeoff).

Convolutions

An Input RGB Image



4 Feature Maps
(from 4 Filters)

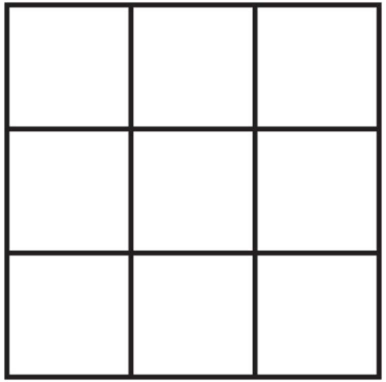


Convolution Settings – Grid Size or Kernel Size

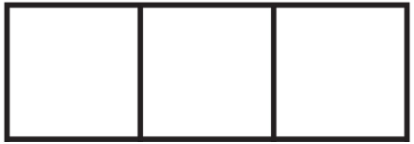
Grid Size or Kernel Size (Height x Width):

- Number of pixels the kernel processes at once.
- Typically use odd numbers to ensure a "center" pixel.
- Kernels can be rectangular, not necessarily square.

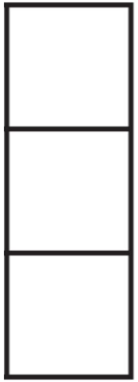
Height: 3, Width: 3



Height: 1, Width: 3



Height: 3, Width: 1



Convolution Settings - Padding

Padding

- Kernels cause an “edge effect” where edge pixels are not used as “center pixels” since there are not enough surrounding pixels
  - "center pixels" refer to the pixels in the input over which the kernel is centered
- Padding adds extra pixels around the image to include all original pixels as center pixels.
- Commonly used padding: zero-padding (added pixels have a value of zero), replication-padding (added pixels replicate the values of the nearest edge pixels).

Without Padding

padding="valid" --> no padding

padding="same" --> output size is the same as of input

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

input

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

kernel

-2		

output

output size = $n - f + 1$

With Padding

padding="valid" --> no padding

padding="same" --> output size is the same as of input

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

original input with zero-padding

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

kernel

-1				

output

output size = $n + 2p - f + 1$

Convolution Settings

Stride

- The step size for moving the kernel across the image.
- Can differ for vertical and horizontal steps (but typically the same).
- Larger strides (>1) reduce the output dimensions, effectively downsampling the **feature map**.
  - A **feature map** is the output of a convolutional layer in a neural network.
  - It represents the spatial activation of features detected by a specific filter (kernel) applied to the input data.

Stride 2 Example – No Padding

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

input

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

kernel

-2	

output

output size = $\left\lfloor \frac{n-f}{s} + 1 \right\rfloor$

Stride 2 Example – With Padding

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

input

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

kernel

-1	2	
3		

output

output size = $\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$

Convolutional Settings - Depth

- Images can have multiple values per pixel, called "channels" (e.g., RGB = 3 channels, CMYK = 4 channels).
- The number of channels is the depth of the image.
- The kernel itself will also have a "depth", the same size as the number of input channels
  - A kernel's depth matches the input's depth to process all channels simultaneously.
- The kernel performs a dot product between two 3D grids (input and kernel).
- Example: a 5x5 kernel on an RGB image has $5 \times 5 \times 3 = 75$ weights.

Convolutional Settings - Depth

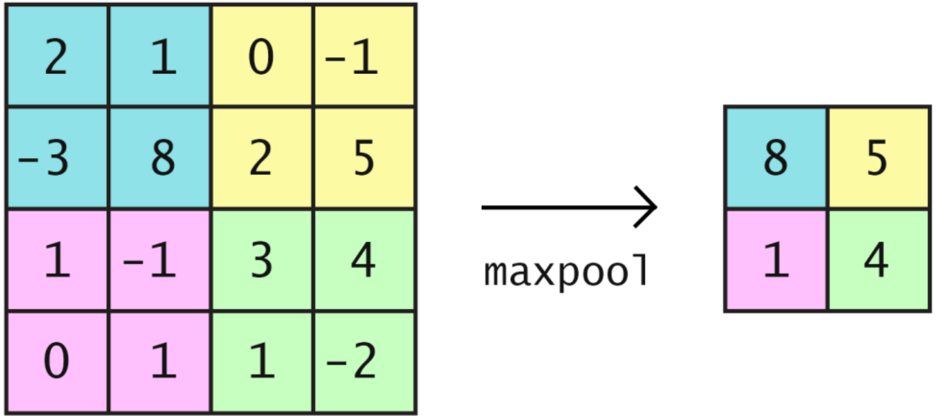
- The output depth equals the number of kernels applied in the layer.
- Each kernel produces one feature map.
- Multiple kernels allow the network to learn diverse features, such as edges, textures, and patterns, at different orientations and scales.
- Example: If a layer uses 10 kernels, the output depth will be 10.

Pooling

- **Purpose:** reduces image size by mapping a patch of pixels to a single value.
- **Effect:** shrinks the spatial dimensions of the feature map.
- **Key Feature:** no learnable parameters; applies predefined operations.
- **Types:** common pooling operations include max pooling and average pooling.

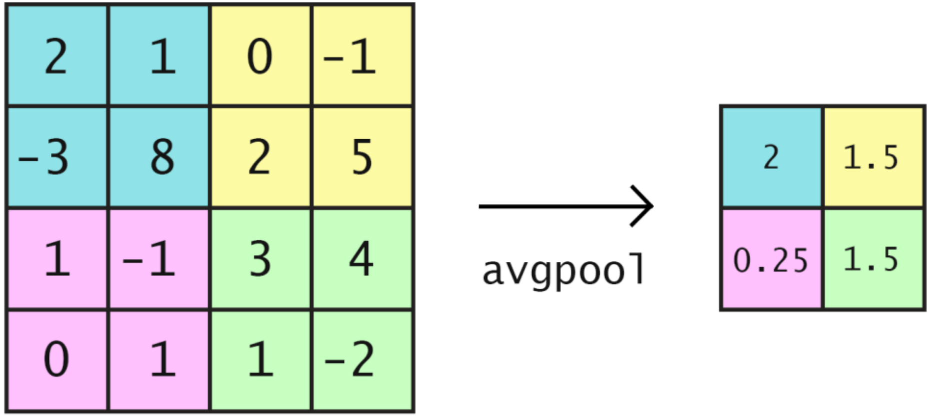
Pooling: Max-Pool

- For each distinct patch, represent it by the maximum
- 2x2 maxpool shown below



Pooling: Average-Pool

- For each distinct patch, represent it by the average
- 2x2 avgpool shown below.



Software