Convolutional Neural Network (Recap)

Naeemullah Khan

naeemullah.khan@kaust.edu.sa

Prashant Aparajeya

KAUST Academy
King Abdullah University of Science and Technology

May 20, 2025

Table of Contents

Table of Contents (cont.)

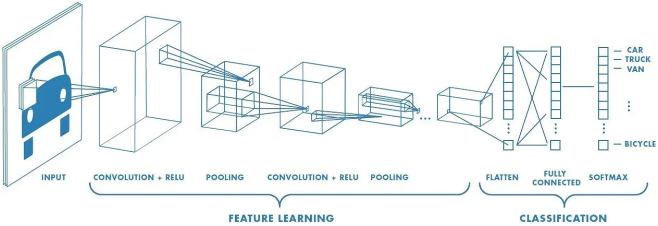
CNN - Definition

- Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing structured grid data, such as images.
- They are particularly effective for tasks like image classification, object detection, and segmentation.
- CNNs leverage the spatial structure of images by using convolutional layers to automatically learn hierarchical features.
- The architecture typically consists of convolutional layers, activation functions, pooling layers, and fully connected layers.
- CNNs are known for their ability to capture local patterns and translate them into higher-level representations.

CNN Architecture

What is a CNN?

A CNN is a deep network of neurons with learnable filters that perform convolution operations on inputs, usually images, to extract hierarchical features.



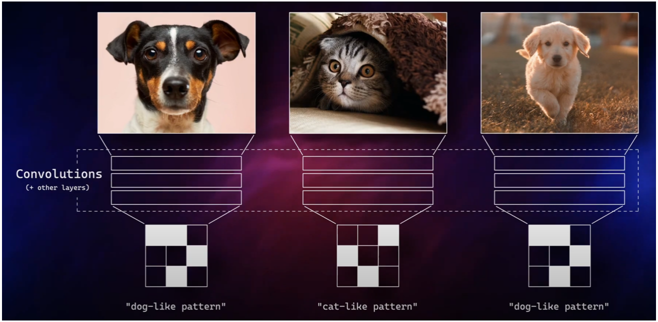
Why use CNNs?

Parameter sharing and sparse connectivity reduce number of parameters and improve spatial feature extraction.

CNN Architecture

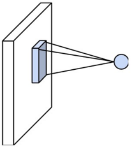
What makes a Convolutional Neural Network?

Characterised by “Convolutional Layer” – they are able to detect “abstract features” and “almost ideas within the image”

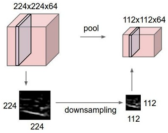


Components of a CNN

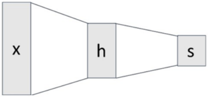
Convolution Layers



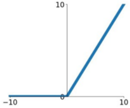
Pooling Layers



Fully-Connected Layers



Activation Function



Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

CNN - Convolutional Layer

Operation:

- Element-wise multiply filter with image patch and sum → feature map.

Hyperparameters:

- kernel size
- number of filters
- stride
- padding

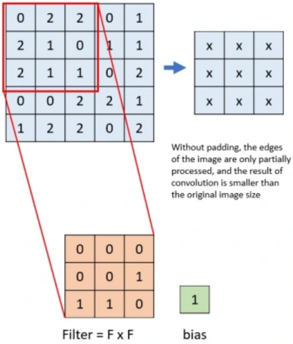
Listing 1: Code snippet (PyTorch)

```
import torch.nn as nn

conv = nn.Conv2d(in_channels=3, out_channels=16,
    kernel_size=3, stride=1, padding=1)

output = conv(input_tensor)  # input_tensor: [
    batch_size, 3, H, W]
```

CNN - Convolutional Layer



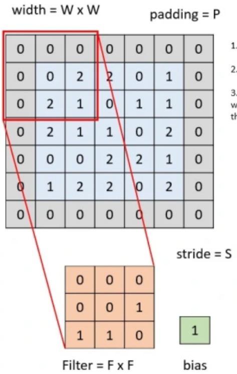
0	2	2	0	1
2	1	0	1	1
2	1	1	0	2
0	0	2	2	1
1	2	2	0	2

x	x	x
x	x	x
x	x	x

0	0	0
0	0	1
1	1	0

1

Without padding, the edges of the image are only partially processed, and the result of convolution is smaller than the original image size



0	0	0	0	0	0	0
0	0	2	2	0	1	0
0	2	1	0	1	1	0
0	2	1	1	0	2	0
0	0	0	2	2	1	0
0	1	2	2	0	2	0
0	0	0	0	0	0	0

0	0	0
0	0	1
1	1	0

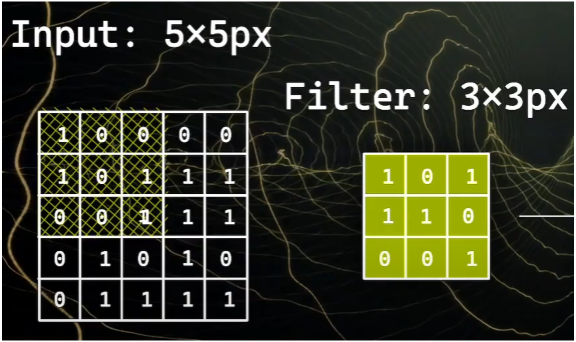
1

1. Convolution result size = (W – F + 2P) / S + 1
2. (W – F + 2P) / S + 1 should be an integer
3. If you set S = 1, then setting P = (F – 1) / 2 will generate convolution result size equal to the image size.

Quick Exercise (5 mins)

Let's find out what this can give us:

- Padding $= 0$
- Stride $= 1$



Input: 5×5px

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

Filter: 3×3px

1	0	1
1	1	0
0	0	1

Note: Once you traverse entire image/matrix it will give you a matrix calls Feature Map or Activation Map.