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

1. Definition
2. Convolutional Layer
3. Activation Functions
4. Pooling Layers
5. Padding & Strides
6. Normalization (Batch Norm)
7. Regularization (Dropout)
8. Flattening & Fully Connected Layers
9. Loss Functions & Optimizers
10. Architectures
    - 10.1 LeNet
    - 10.2 AlexNet

Table of Contents (cont.)

10.3 VGG

10.4 InceptionNet

10.5 ResNet

10.6 MobileNet

11. Transfer Learning

12. Evaluation Metrics

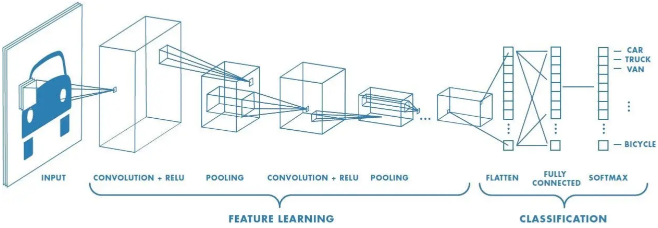
13. Further Reading

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.

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.

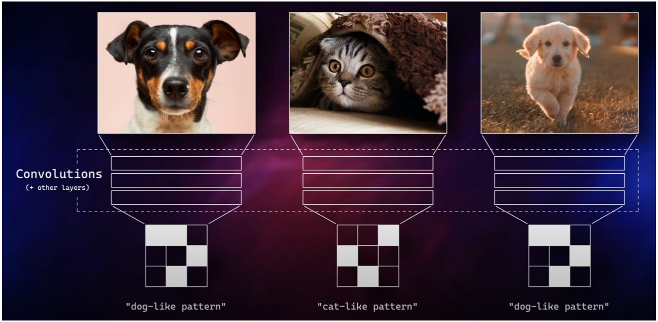


Why use CNNs?

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

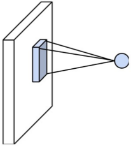
What makes a Convolutional Neural Network?

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

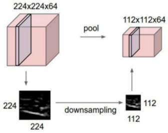


Components of a CNN

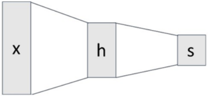
Convolution Layers



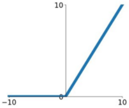
Pooling Layers



Fully-Connected Layers



Activation Function



Normalization

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

Operation:

- Element-wise multiply filter with image patch and sum $\rightarrow$ feature map.

Hyperparameters:

- kernel size
- number of filters
- stride
- padding

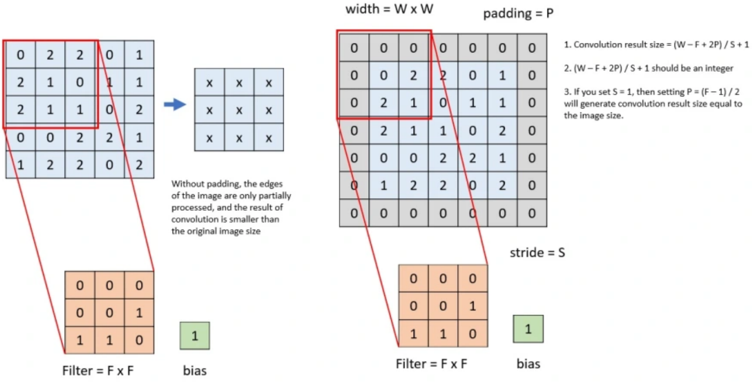
Listing 1: Code snippet (PyTorch)

```python
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

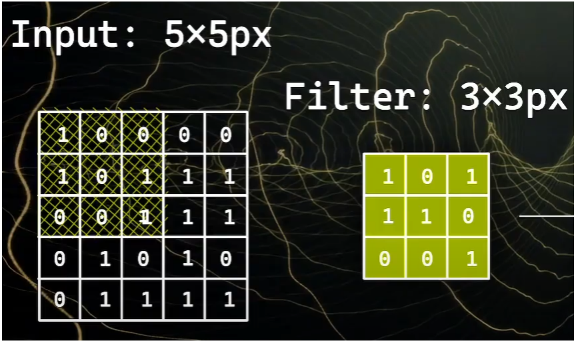


Interactive demo: cs231n.github.io/convolutional-networks

Quick Exercise (5 mins)

Let's find out what this can give us:

- Padding = 0
- Stride = 1



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

CNN - Activation Functions

Role:

- Introduce non-linearity so multiple conv layers can learn complex mappings.

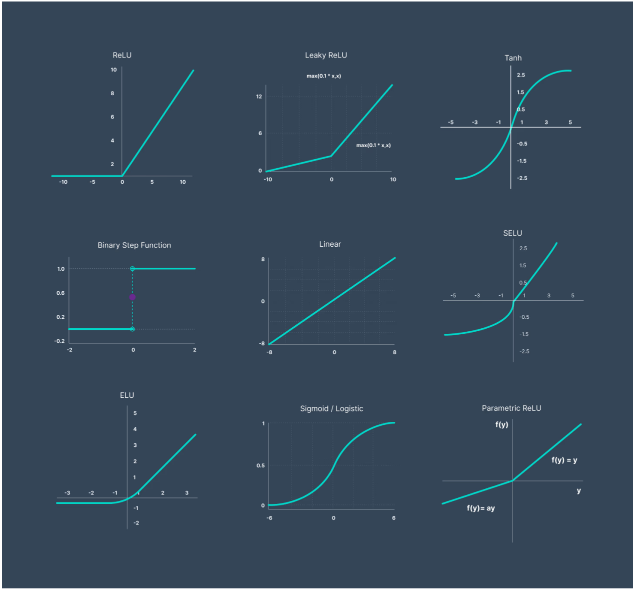
Common:

- ReLU (Rectified Linear Unit)
- Leaky ReLU
- Sigmoid
- Tanh

Listing 2: Code snippet (PyTorch)

```
import torch.nn.functional as F

x = conv(input_tensor)
x = F.relu(x)                # ReLU
x = F.leaky_relu(x, 0.1)     # Leaky ReLU
```



ReLU
Leaky ReLU
max(0.1 * x,x)
max(0.1 * x,x)
Tanh
Binary Step Function
Linear
SELU
ELU
Sigmoid / Logistic
Parametric ReLU
f(y)
f(y) = y
y
f(y)= ay

CNN - Pooling Layers

Purpose:

- Downsample feature maps, reduce spatial dims and parameters, add invariance.

Types:

- Max Pooling
- Average Pooling
- Global Average Pooling
- Global Max Pooling

Listing 3: Code snippet (PyTorch)

```
import torch.nn as nn

pool = nn.MaxPool2d(kernel_size=2, stride=2)
pooled = pool(x)  # halves H and W
```

CNN - Pooling Layers