Convolutional Neural Network (Recap)

Naeemullah Khan
naeemullah.khan@kaust.edu.sa
Prashant Aparajeya

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

May 20, 2025

Table of Contents

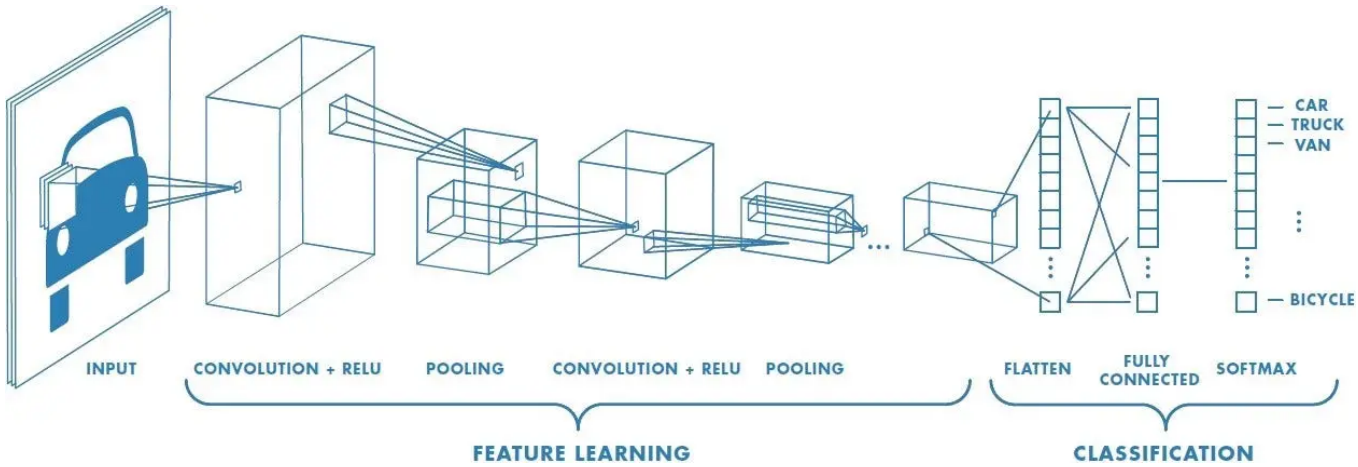
Table of Contents (cont.)

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.

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.

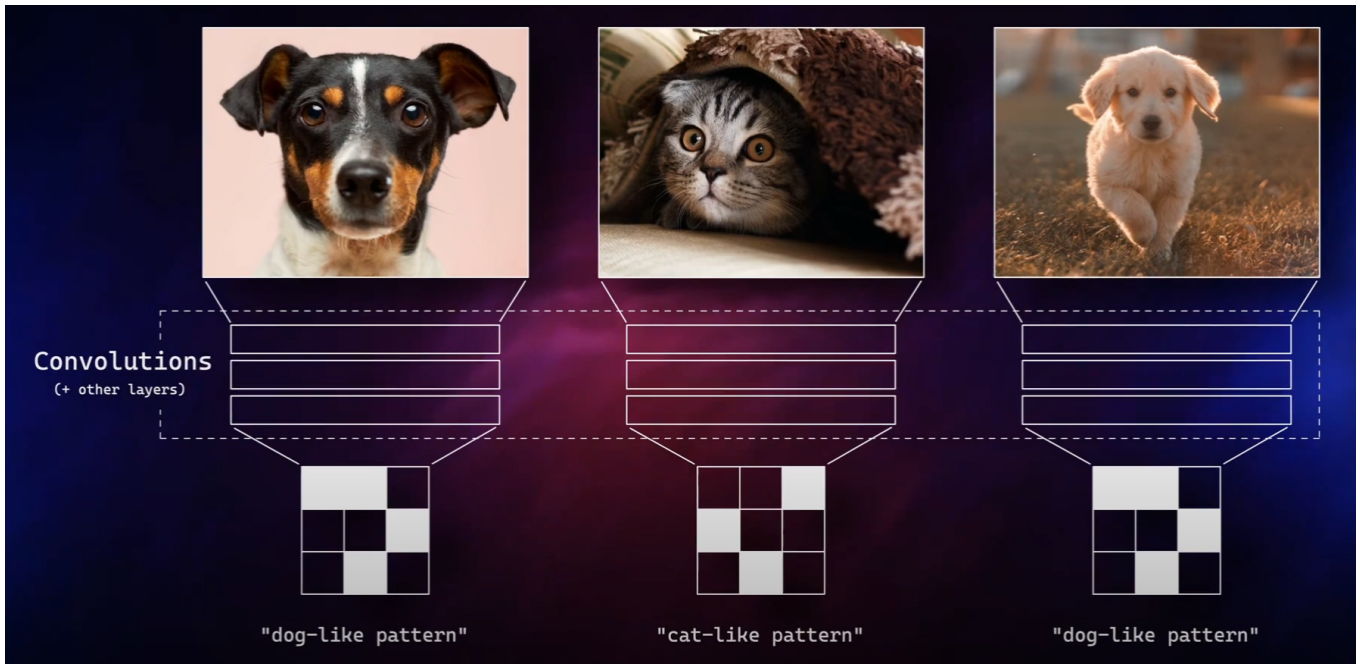


Why use CNNs?

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

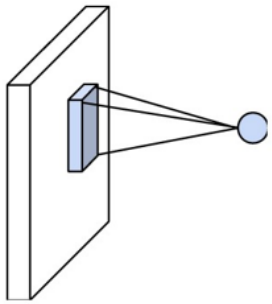
What makes a Convolutional Neural Network?

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

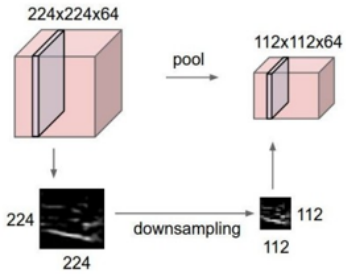


Components of a CNN

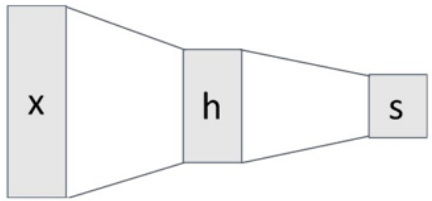
Convolution Layers



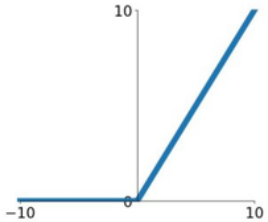
Pooling Layers



Fully-Connected Layers



Activation Function



Normalization

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

CNN - Convolutional Layer

أكاديمية كاوست
KAUST ACADEMY

Operation:

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

Hyperparameters:

- kernel size
- number of filters
- stride
- padding

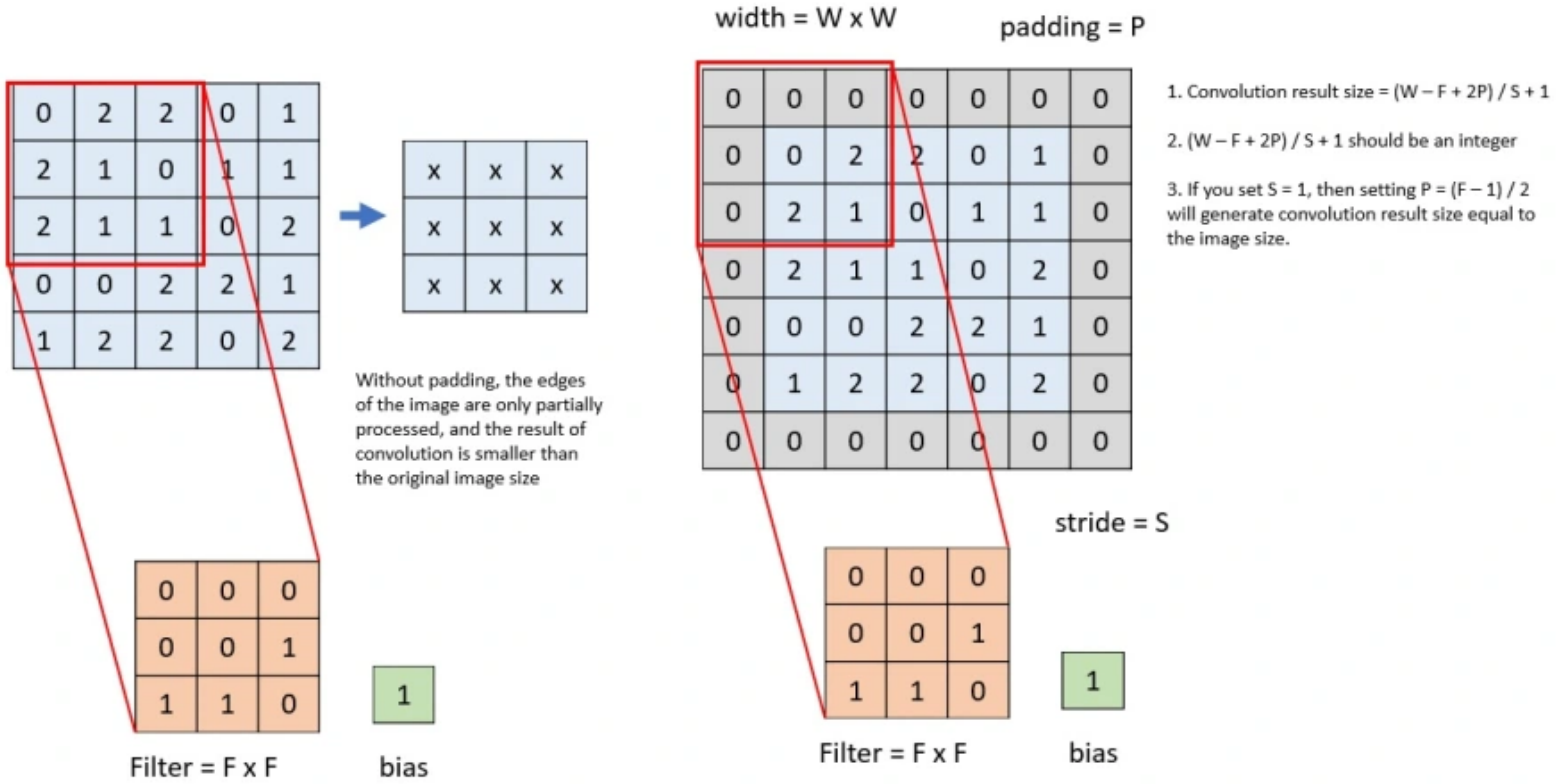
Listing 1: Code snippet (PyTorch)

```
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

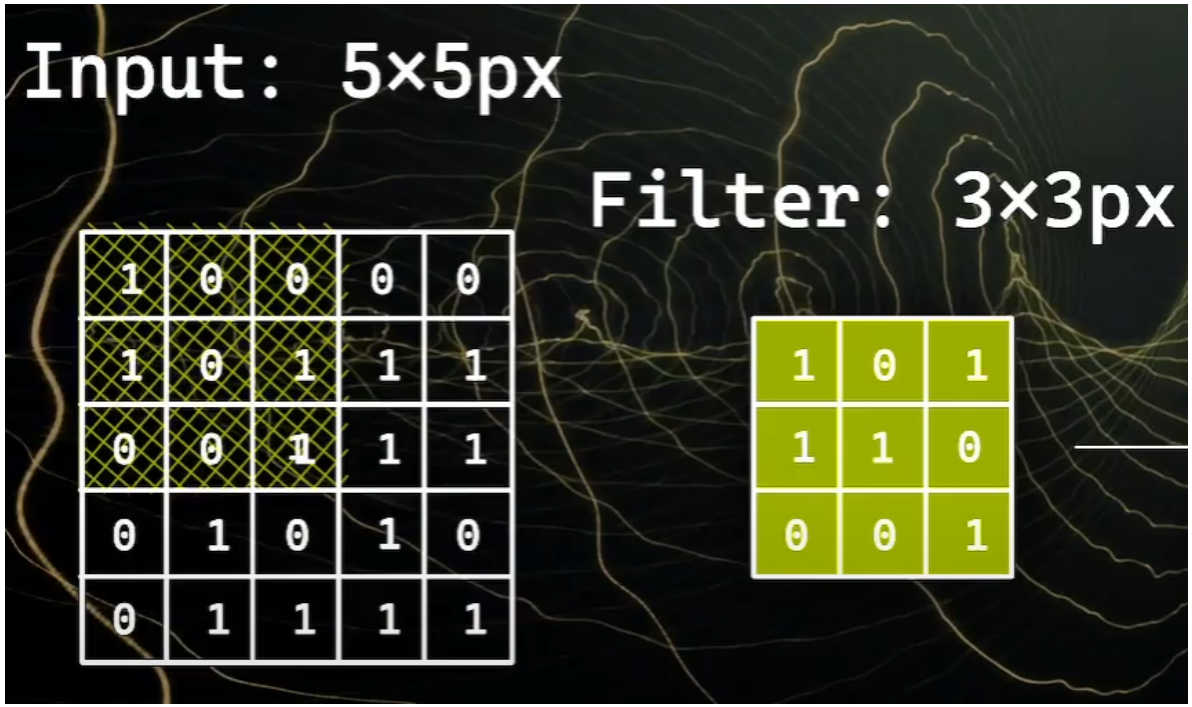


Interactive demo: cs231n.github.io/convolutional-networks

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.

Role:

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

Common:

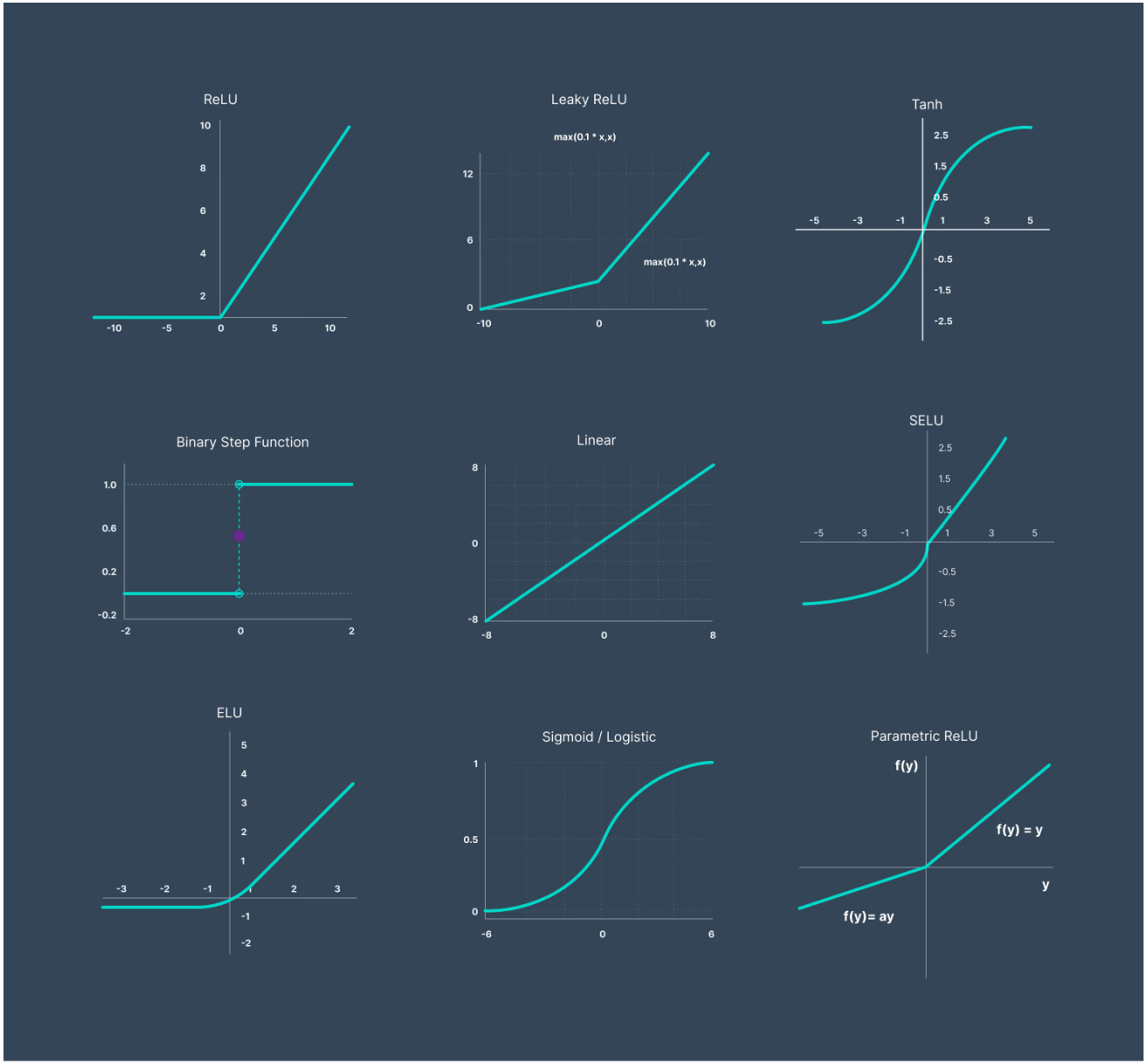
- ReLU (Rectified Linear Unit)
- Leaky ReLU
- Sigmoid
- Tanh

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
```

CNN - Activation Functions



CNN - Pooling Layers

أكاديمية كاوست
KAUST ACADEMY

Purpose:

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

Types:

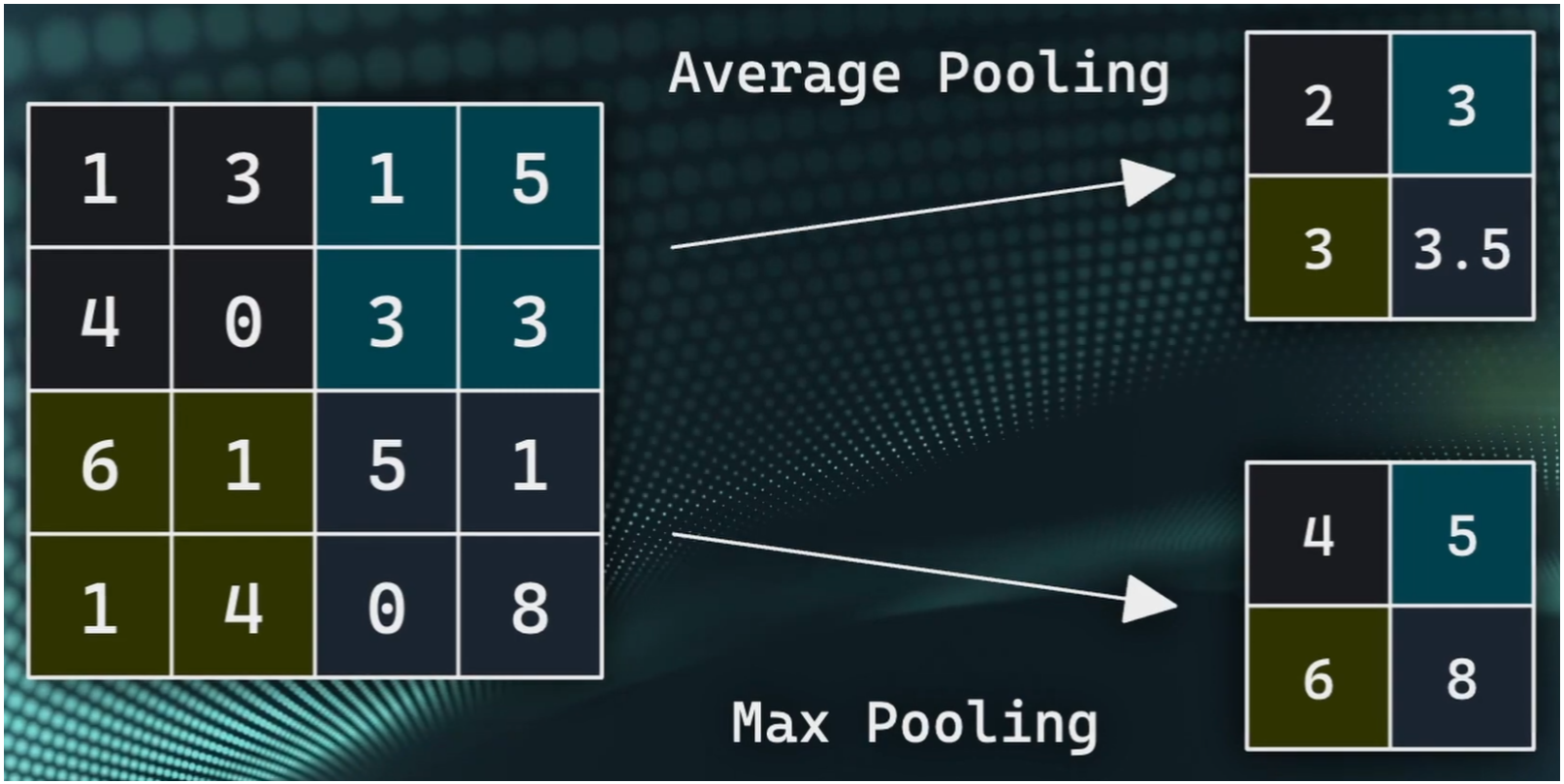
- Max Pooling
- Average Pooling
- Global Average Pooling
- Global Max Pooling

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



CNN - Padding Strides

أكاديمية كاوست
KAUST ACADEMY

Padding:

- "same" preserves spatial size by adding zeros around input;
- "valid" reduces spatial size by not adding any padding.
- Padding is used to control the spatial size of the output feature map.
- Padding is added to the input image before applying the convolution operation.

Strides:

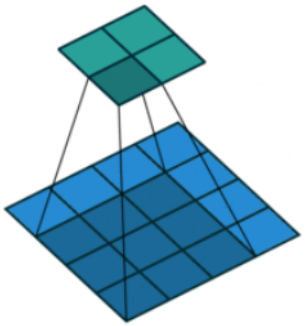
- Strides control how much the filter moves across the input image.
- Strides can be set independently for height and width.
- Strides are used to control the spatial size of the output feature map.
- Strides are set in the convolutional layer.

Listing 4: Code snippet (PyTorch)

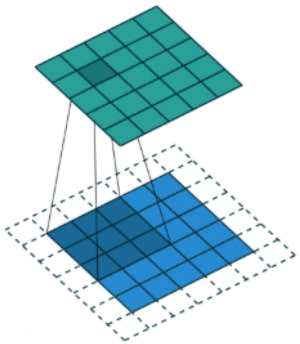
```
import torch.nn as nn

conv_valid = nn.Conv2d(3, 16, 3, stride=2, padding=0) #
        Valid    : no padding (padding=0)
conv_same  = nn.Conv2d(3, 16, 3, stride=1, padding=1) #
        Same     : preserve size
```

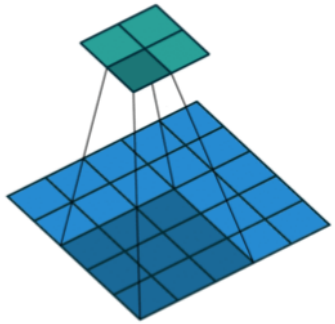
CNN - Padding Strides



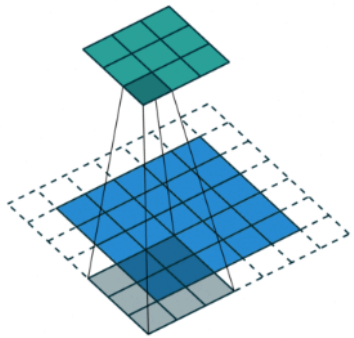
padding = 0, stride = 1



padding = 1, stride = 1



padding = 0, stride = 2



padding = 1, stride = 2

CNN - Normalization (Batch Norm)

أكاديمية كاوست
KAUST ACADEMY

What:

- Normalize layer inputs over mini-batch, then scale shift.

Benefits:

- Faster training,
- Allows higher learning rates,
- Reduces sensitivity to initialization,
- Acts as a regularizer.

Listing 5: Code snippet (PyTorch)

```
import torch.nn as nn

bn = nn.BatchNorm2d(16)
x = bn(x)
```

Note:

- BatchNorm is typically used after convolutional layers and before activation functions.
- BatchNorm normalizes activations per-batch, then scales/shifts them.

CNN - Normalization (Batch Norm)

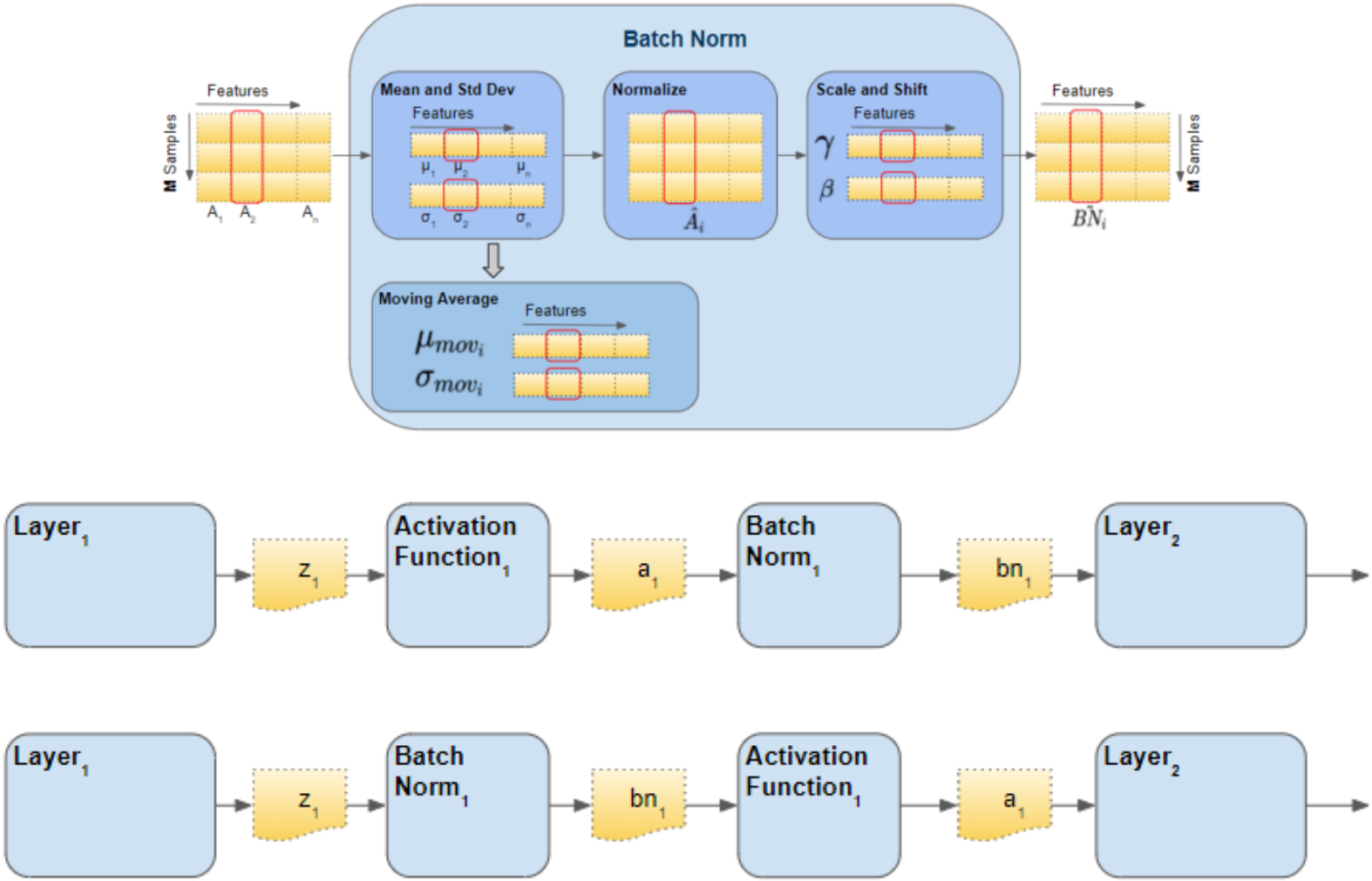


Figure 2: Order of placement of Batch Norm layer

More on Batch Norm from Towards Data Science

CNN - Regularization (Dropout)

What:

- Randomly drop units with probability p during training to prevent co-adaptation.

Where:

- Often after FC layers, sometimes after conv layers.

Effect on images:

- Regularization does not alter the input images but modifies the training process to improve generalization.

Listing 6: Code snippet (PyTorch)

```
import torch.nn as nn

drop = nn.Dropout(p=0.5)
x = drop(x)
```