CSE 471: MACHINE LEARNING

Convolutional Neural Network (CNN)

Image classification task

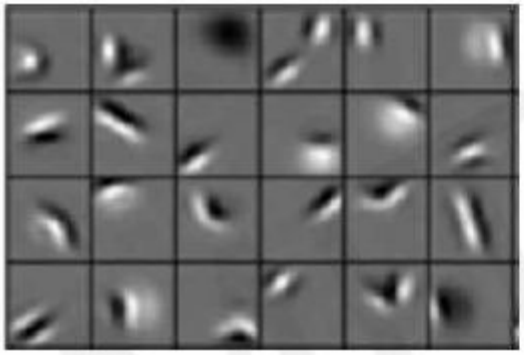
- Image data exhibits spatial invariance
  - Anything that is detectable in one small, local region of the image look the same if it appeared in another small, local region of the image.

Image classification task

Learning Feature Representations

Can we learn a **hierarchy of features** directly from the data instead of hand engineering?

Low level features



Edges, dark spots

Mid level features

Eyes, ears, nose

High level features

Facial structure

Taken from MIT 6.S191 lecture slides

Convolution

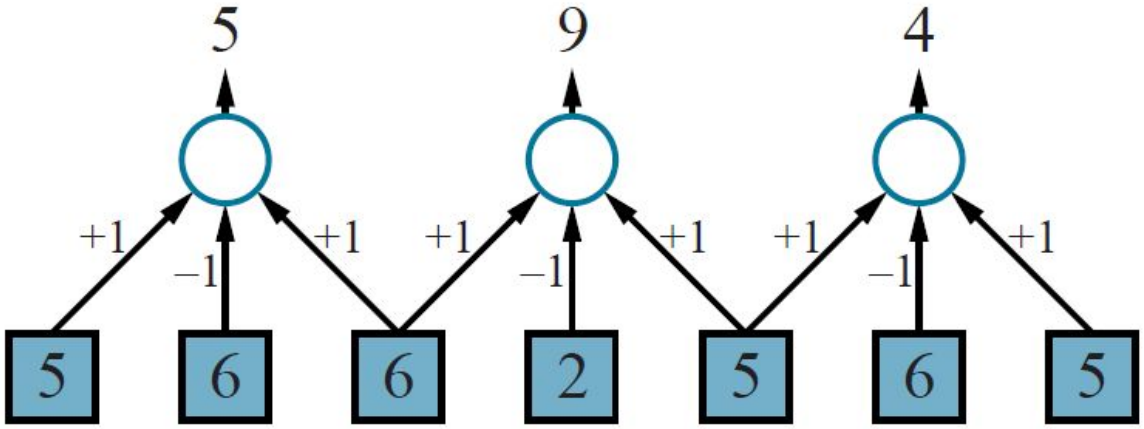


Figure 21.4 An example of a one-dimensional convolution operation with a kernel of size $l=3$ and a stride $s=2$. The peak response is centered on the darker (lower intensity) input pixel. The results would usually be fed through a nonlinear activation function (not shown) before going to the next hidden layer.

$$z_i = \sum_{j=1}^{l} k_j x_{j+i-(l+1)/2}$$

Convolution

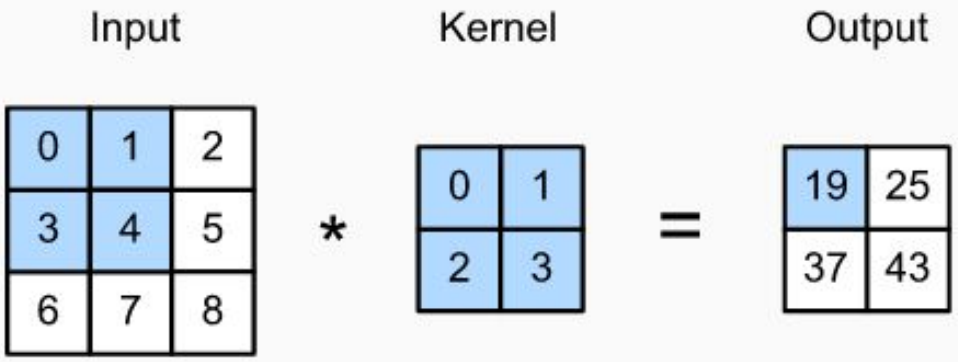


Fig. 7.2.1 Two-dimensional cross-correlation operation. The shaded portions are the first output element as well as the input and kernel tensor elements used for the output computation: $0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$.

Convolution

```
import torch
from torch import nn
from d2l import torch as d2l

def corr2d(X, K):  #@save
    """Compute 2D cross-correlation."""
    h, w = K.shape
    Y = torch.zeros((X.shape[0] - h + 1, X.shape[1] - w + 1))
    for i in range(Y.shape[0]):
        for j in range(Y.shape[1]):
            Y[i, j] = (X[i:i + h, j:j + w] * K).sum()
    return Y
```

Stride

- Distance between successive pixels on which the kernels are centered.
- Reduces the output size
  - Roughly from n to n/s

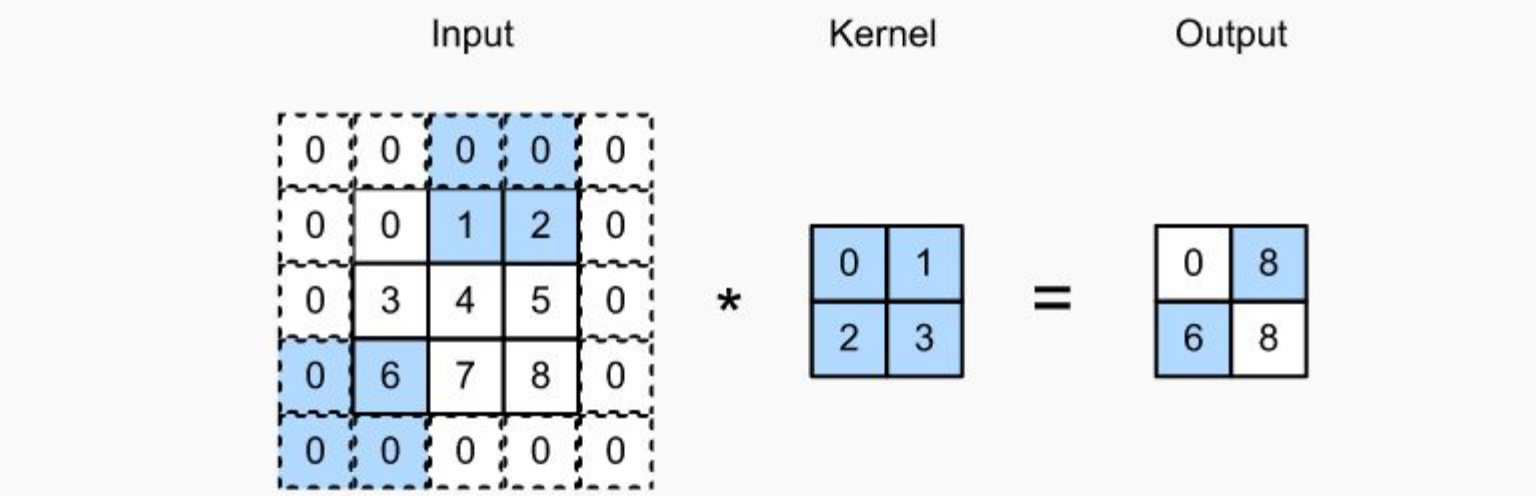


Fig. 7.3.3 Cross-correlation with strides of 3 and 2 for height and width, respectively.

Padding

- 0s OR
- Copies of the outer pixels

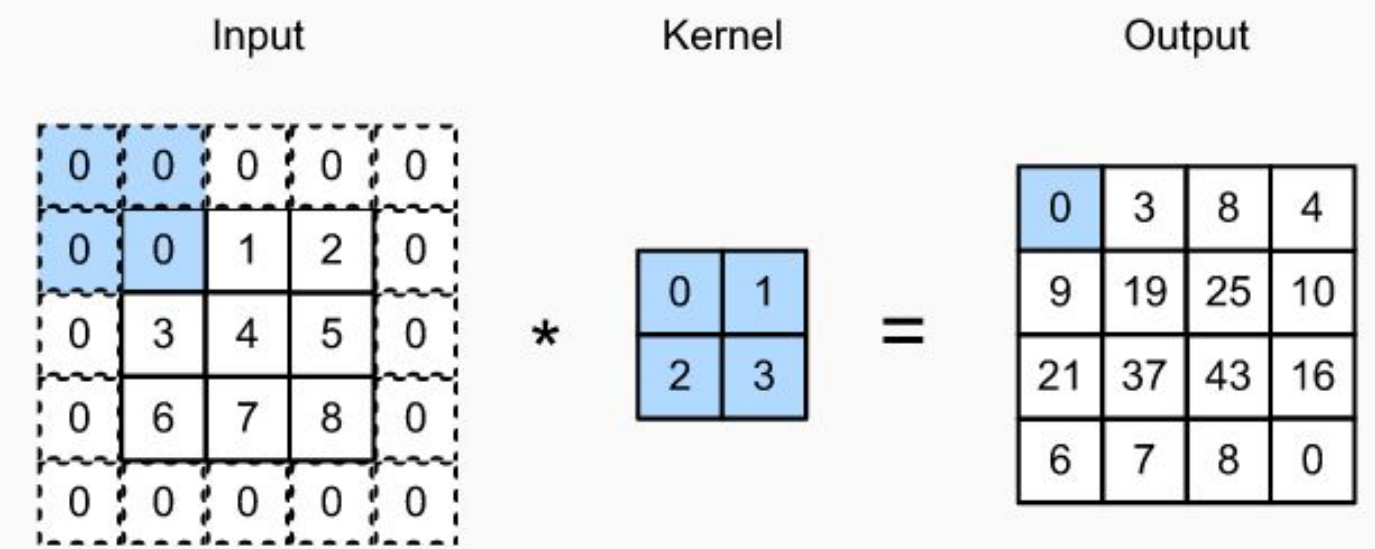


Fig. 7.3.2 Two-dimensional cross-correlation with padding.

Multiple Input Channels

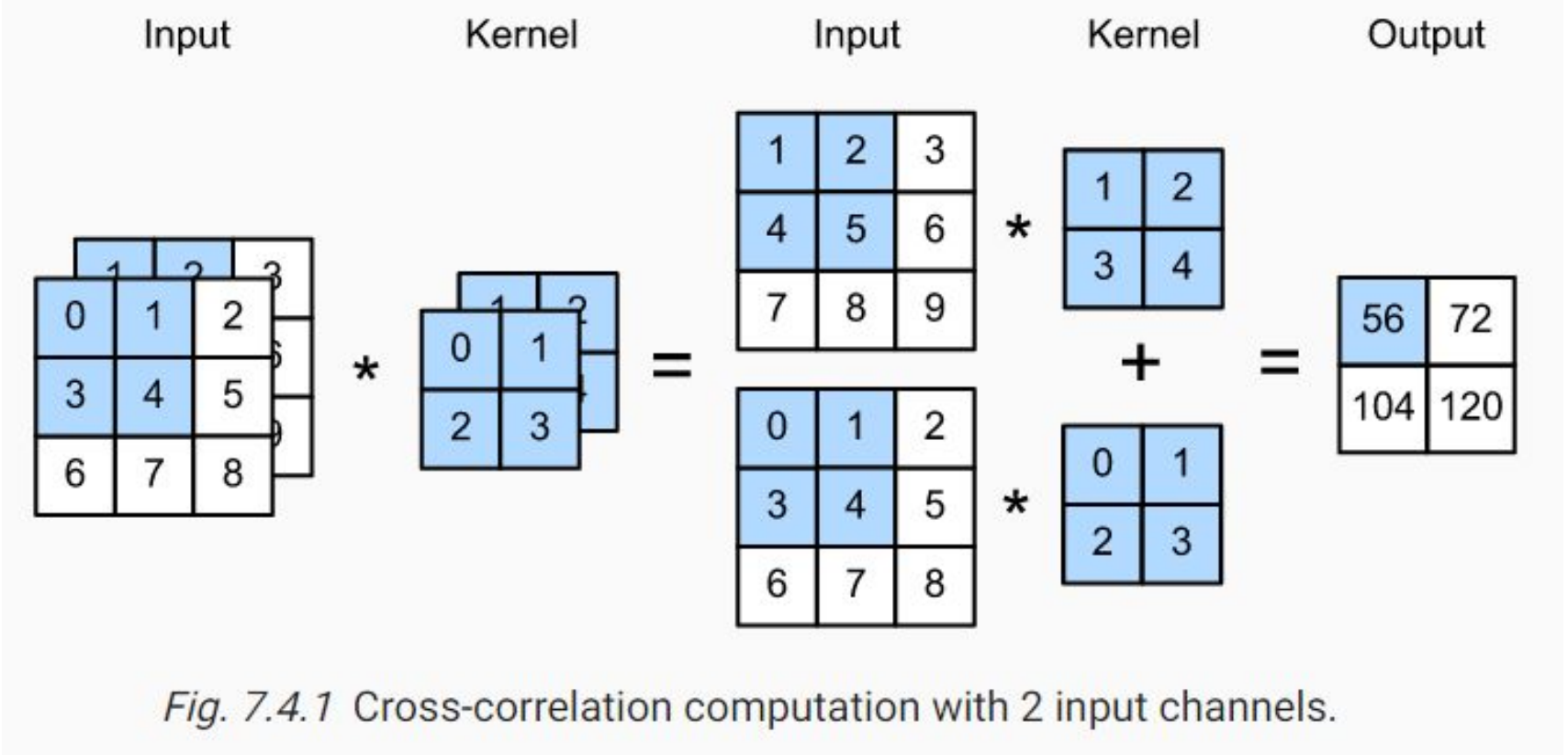


Fig. 7.4.1 Cross-correlation computation with 2 input channels.

See the animation from https://cs231n.github.io/convolutional-networks/

Receptive field

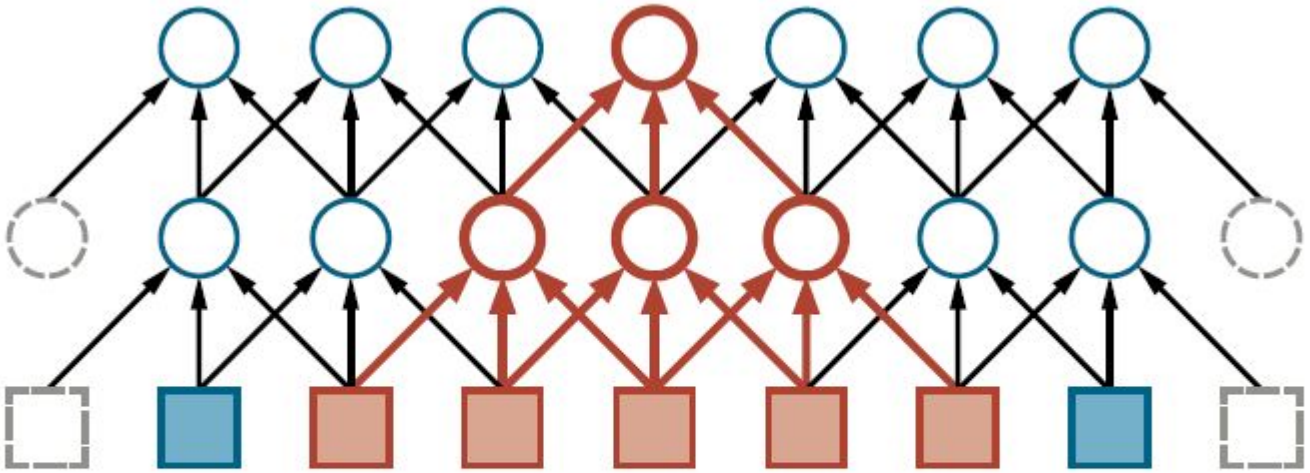
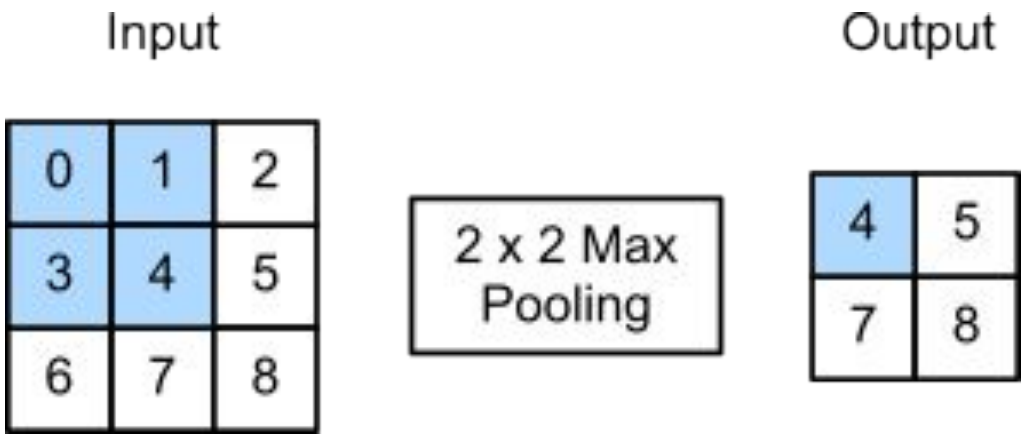


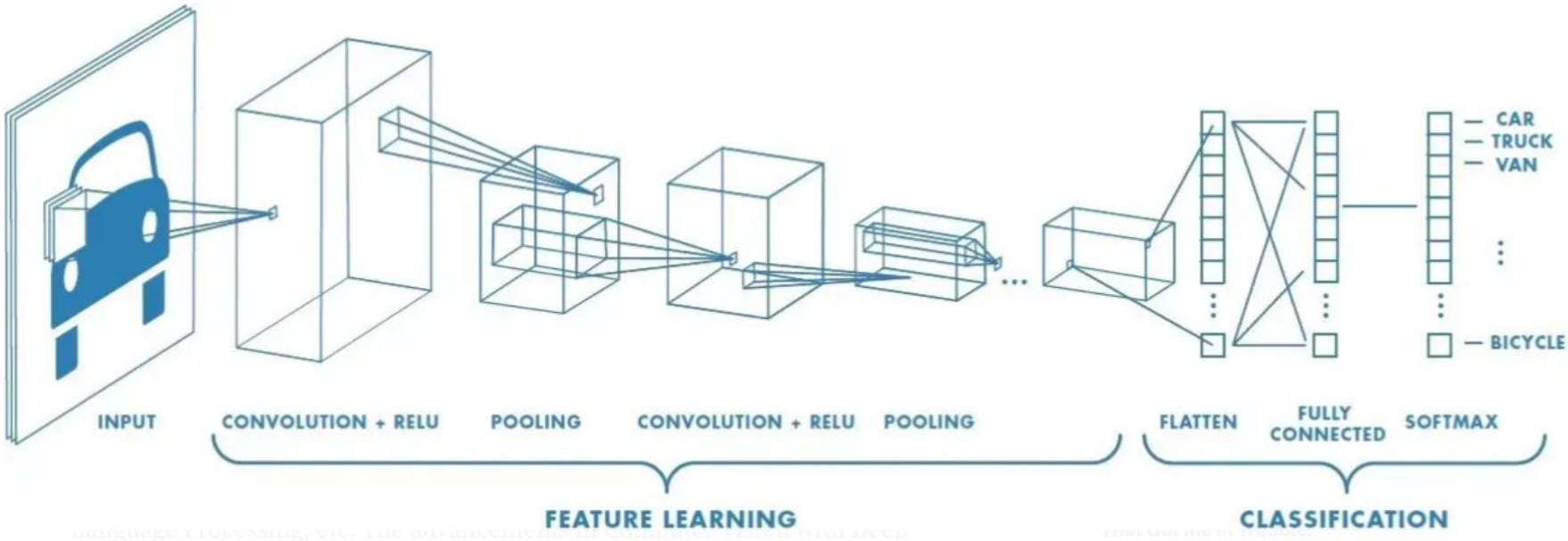
Figure 21.5 The first two layers of a CNN for a 1D image with a kernel size $l=3$ and a stride $s=1$. Padding is added at the left and right ends in order to keep the hidden layers the same size as the input. Shown in red is the receptive field of a unit in the second hidden layer. Generally speaking, the deeper the unit, the larger the receptive field.

Pooling

- Average pooling
  - Average value of *l* inputs
  - l=s Down sampling an image
- Max pooling
  - Maximum value of *l* inputs
  - Feature detector



Putting it all together



Tensors

- Minibatch size = 64
- 256 x 256 RGB image
  - A tensor of dimension: 256×256×3×64
- 5 x 5 x 3 kernel, 96 of them. When applied with stride 2 in both x and y, we get
  - A tensor of 128 x 128 x 96 x 64
  - Called a Feature Map
  - Each channel carries information from one feature

Batch Normalization

$$\hat{z}_i = \gamma \frac{z_i - \mu}{\sqrt{\epsilon + \sigma^2}} + \beta$$

- z_i are values of z for each instance of the minibatch
- μ - mean
- σ – standard deviation
- ε – small constant
- Ɣ and β are learned parameters

Gradient of cross entropy with softmax

- https://fanzengau.com/myblog/content/machine_learning/gradient_of_categorical_cross_entropy/gradient_of_categorical_cross_entropy.html

CNN Back propagation

- See hand written notes