UNC CHARLOTTE

The WILLIAM STATES LEE COLLEGE of ENGINEERING

Introduction to ML
Lecture 4: CNNs

Hamed Tabkhi

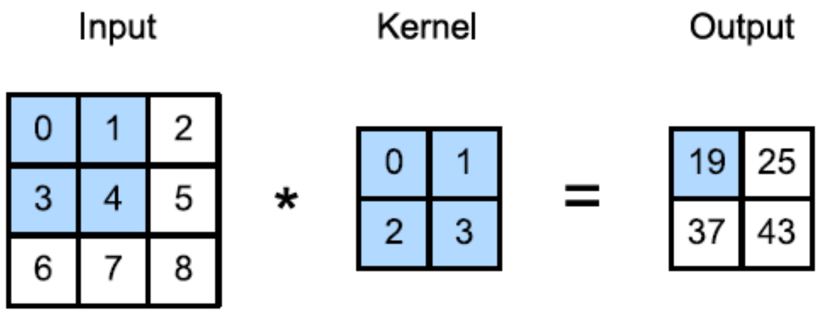
Department of Electrical and Computer Engineering,

University of North Carolina Charlotte (UNCC)

htabkhiv@uncc.edu

UNC CHARLOTTE

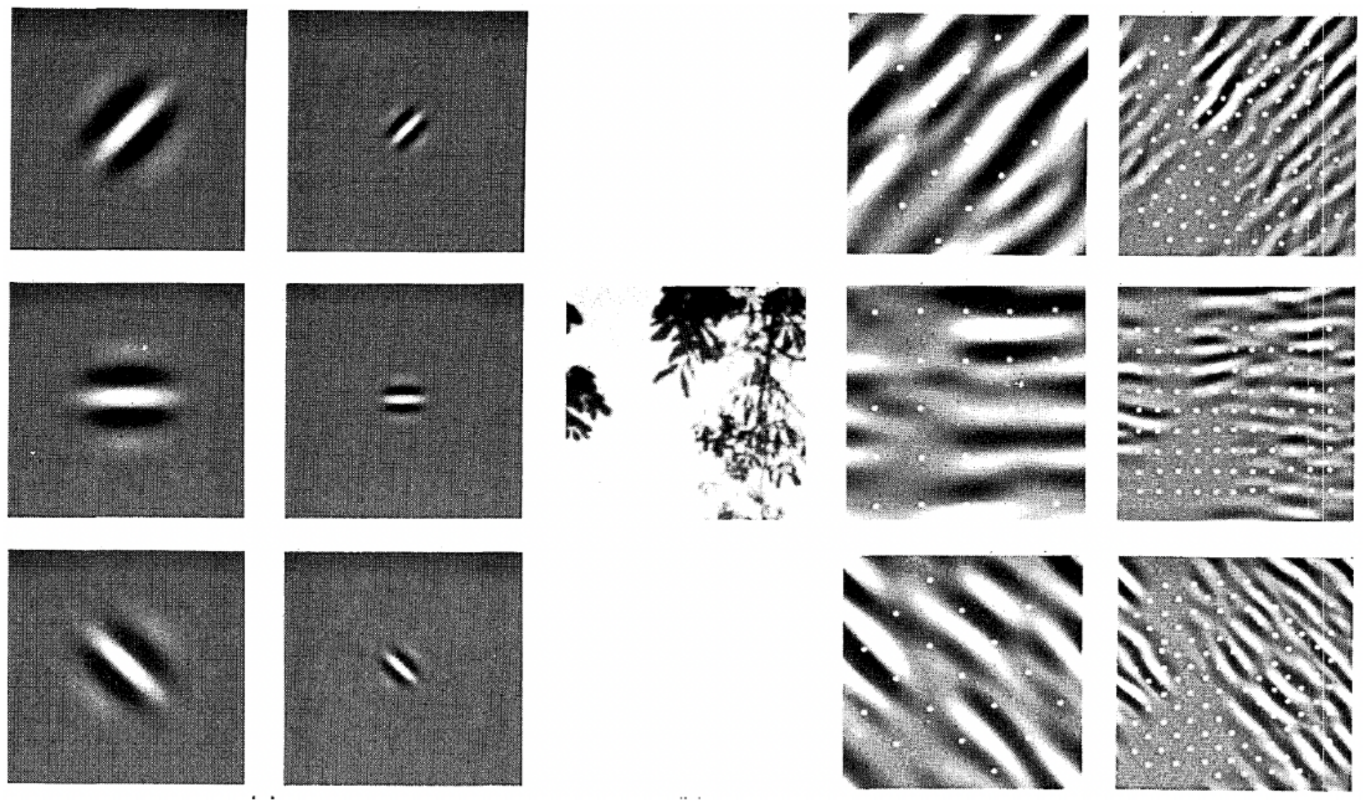
Convolution



$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$

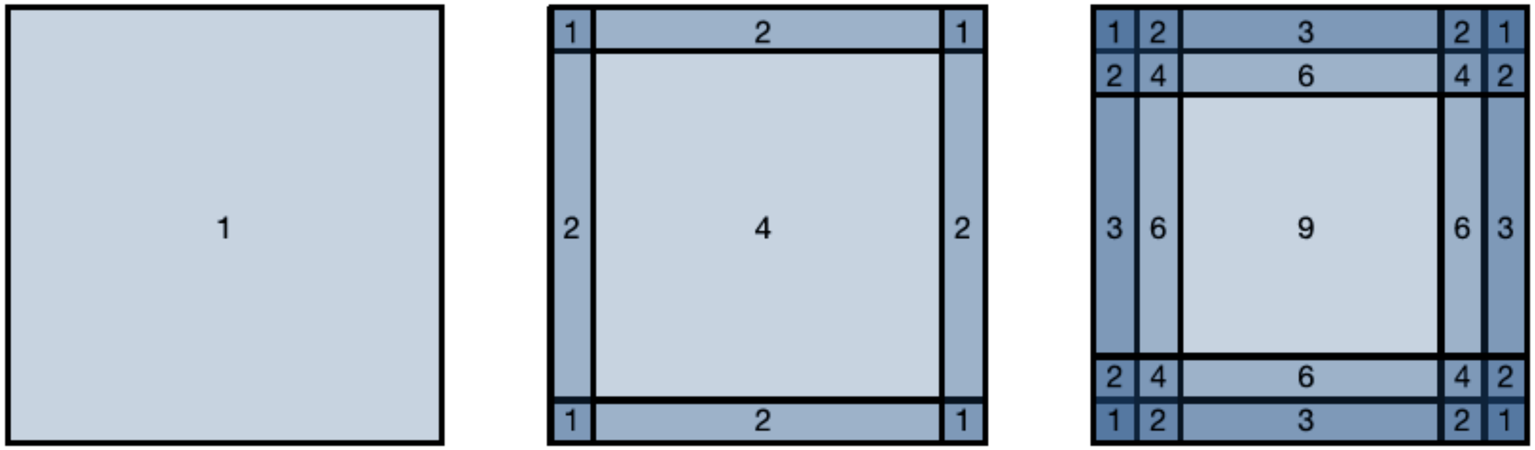
UNC CHARLOTTE

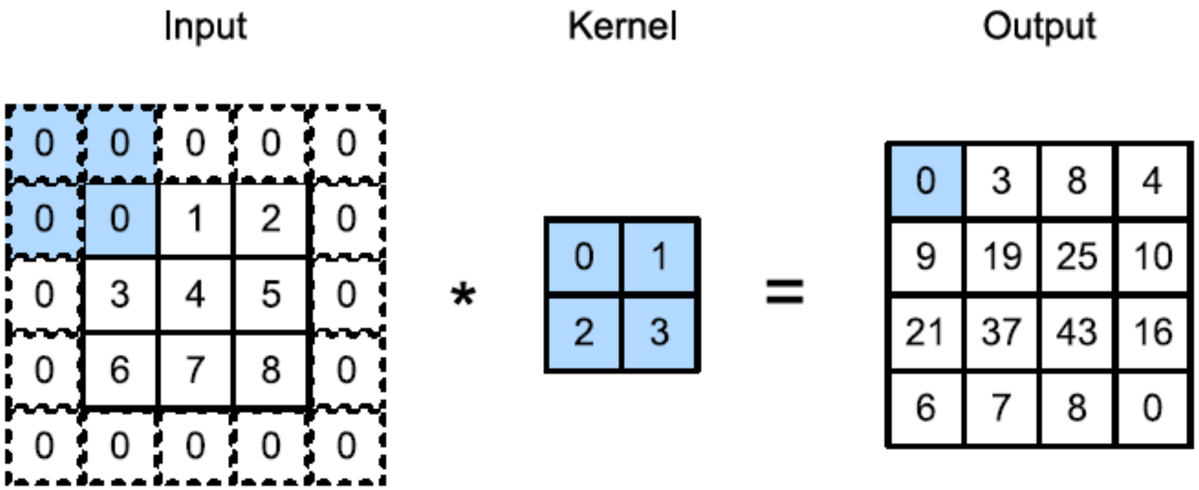
Impact of Conclusion



Convolution of the image in (Middle) with the six (filters) shown in (Left).

Zero Padding





Stride

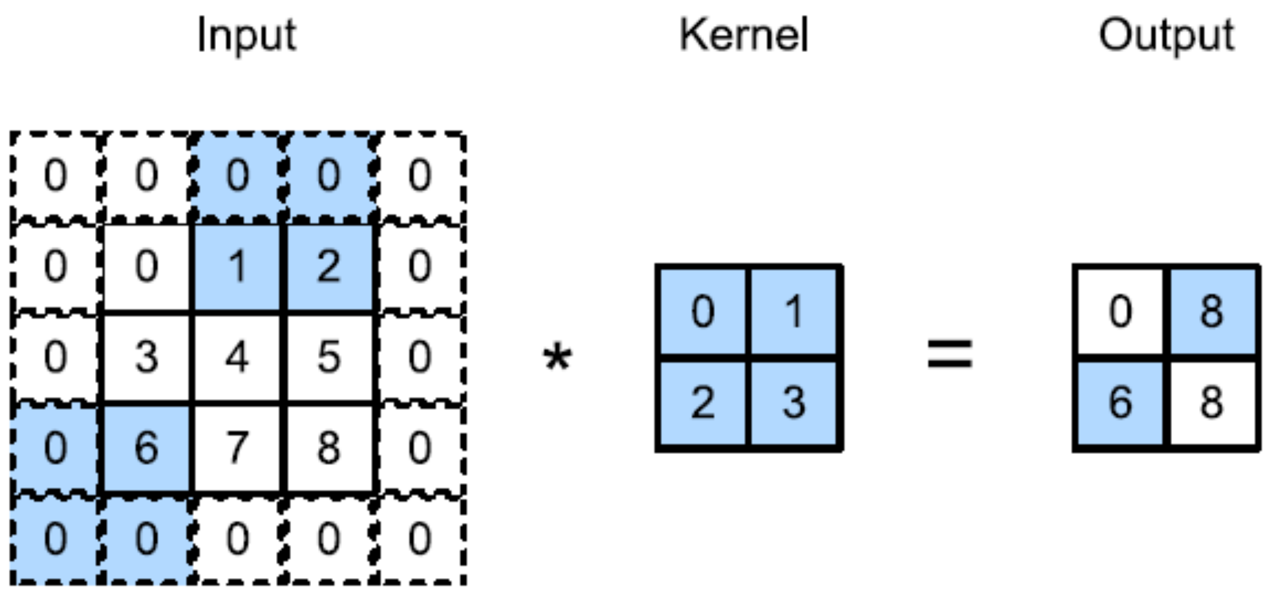


Figure shows a two-dimensional cross-correlation operation with a stride of 3 vertically and 2 horizontally.

Convolution in Pytorch

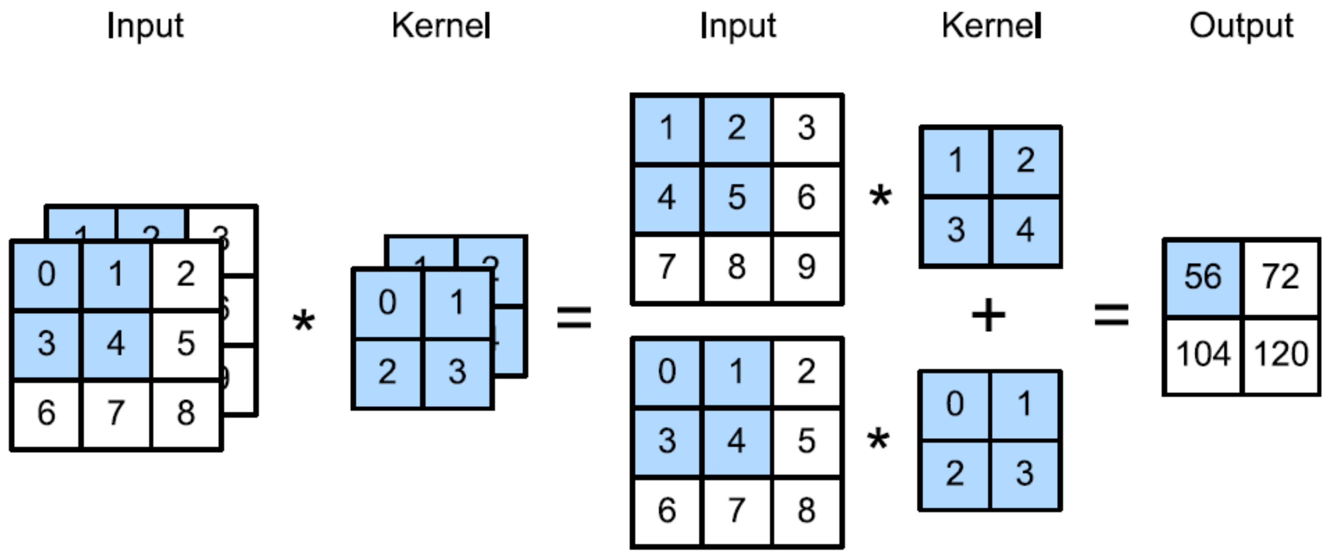
```
import torch
from torch import nn
from d2l import torch as d2l
```

```
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
```

```
X = torch.tensor([[0.0, 1.0, 2.0], [3.0, 4.0, 5.0], [6.0, 7.0, 8.0]])
K = torch.tensor([[0.0, 1.0], [2.0, 3.0]])
corr2d(X, K)
```

```
tensor([[19., 25.],
        [37., 43.]])
```

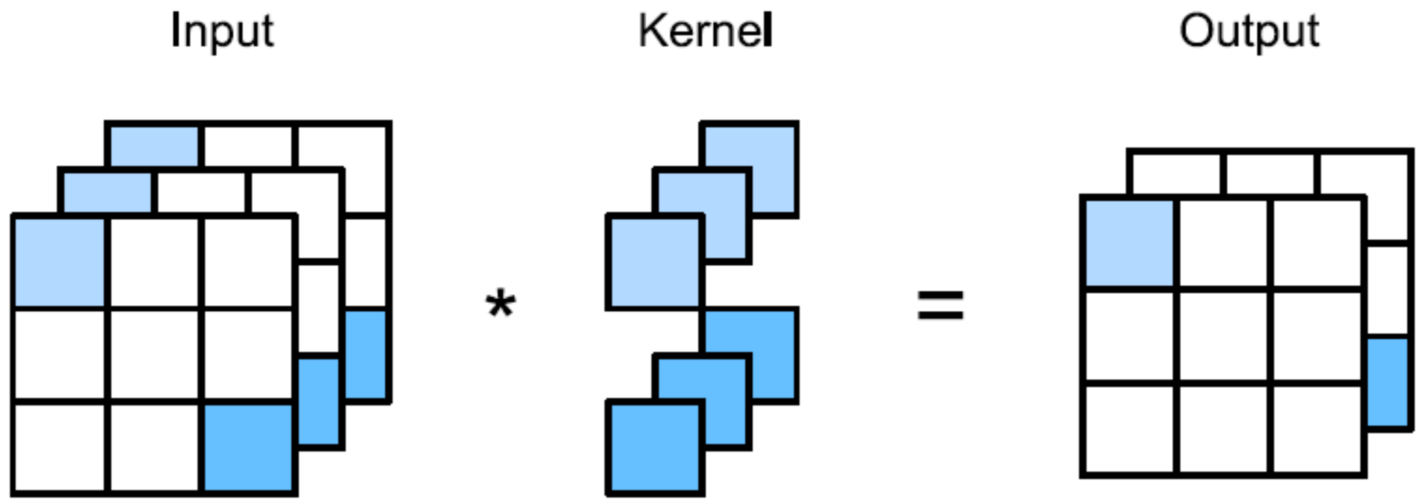
Multiple Input and Multiple Output Channels



- When the input data contains multiple channels, we need to construct a convolution kernel with the same number of input channels as the input data, so that it can perform cross-correlation with the input data. Assuming that the number of channels for the input data is *ci*, the number of input channels of the convolution kernel also needs to be *ci* .
- Adding the *ci* results together (summing over the channels) to yield a two-dimensional tensor. This is the result of a two-dimensional cross-correlation between a multi-channel input and a multi-input-channel convolution kernel.

Multiple Input and Multiple Output Channels

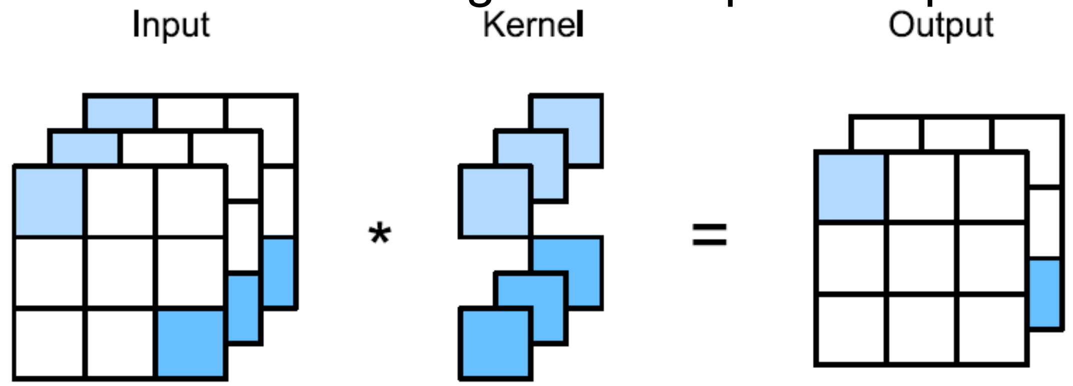
- In the most popular neural network architectures, we actually increase the channel dimension as we go deeper in the neural network, typically downsampling to trade off spatial resolution for greater *channel depth*.
- Intuitively, you could think of each channel as responding to a different set of features.



UNC CHARLOTTE

1*1 Convolution

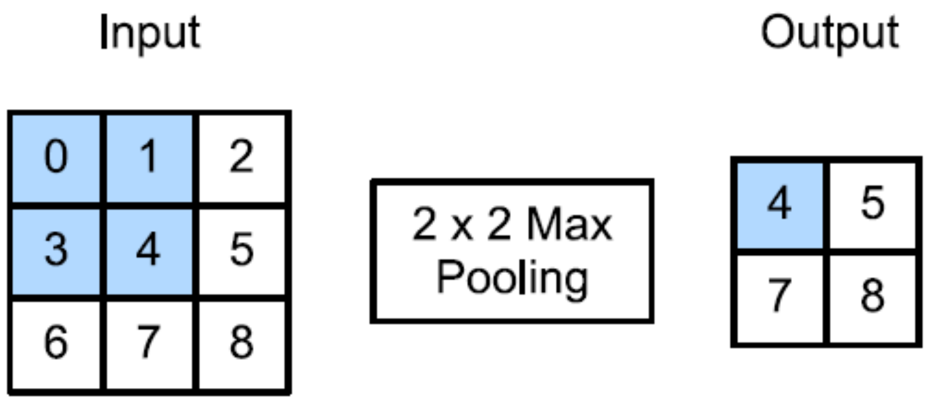
- At first, a 11 convolution, i.e., $k_h = k_w = 1$, does not seem to make much sense.
- After all, a convolution correlates adjacent pixels. A 1 *1 convolution obviously does not. Nonetheless, they are popular operations that are sometimes included in the designs of complex deep



- Figure shows the cross-correlation computation using the 1 *1 convolution kernel with 3 input channels and 2 output channels.
- Note that the inputs and outputs have the same height and width.
- Each element in the output is derived from a linear combination of elements *at the same position* in the input image.

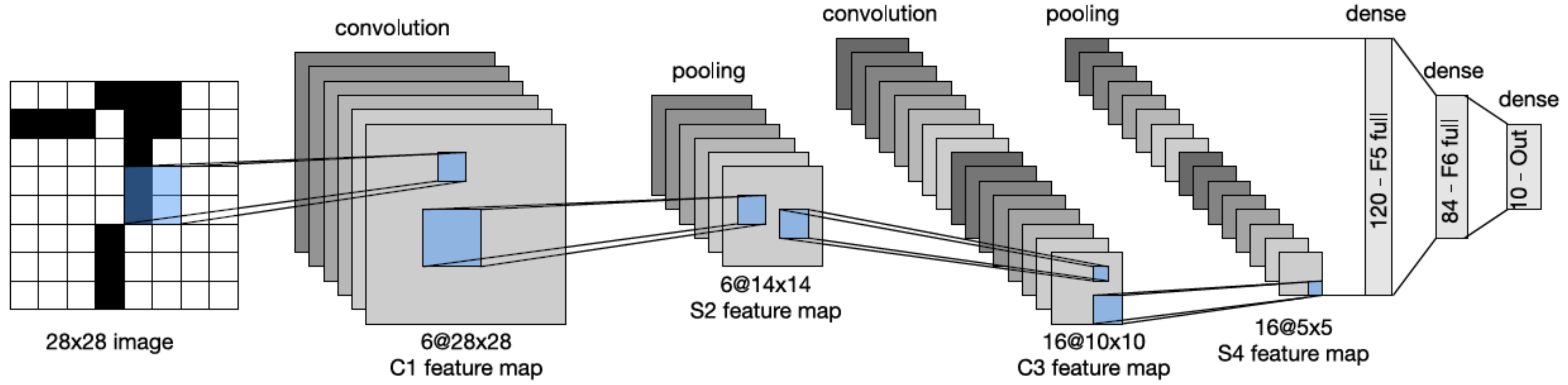
Pooling

- *Average pooling* is essentially as old as CNNs.
- The idea is akin to downsampling an image.
- Rather than just taking the value of every second (or third) pixel for the lower resolution image, we can average over adjacent pixels to obtain an image with better signal to noise ratio since we are combining the information from multiple adjacent pixels.
- *Max-pooling* was introduced in (Riesenhuber and Poggio, 1999) in the context of cognitive neuroscience to describe how information aggregation might be aggregated hierarchically for the purpose ofobject recognition



$$\max(0,1,3,4)=4,$$
$$\max(1,2,4,5)=5,$$
$$\max(3,4,6,7)=7,$$
$$\max(4,5,7,8)=8.$$

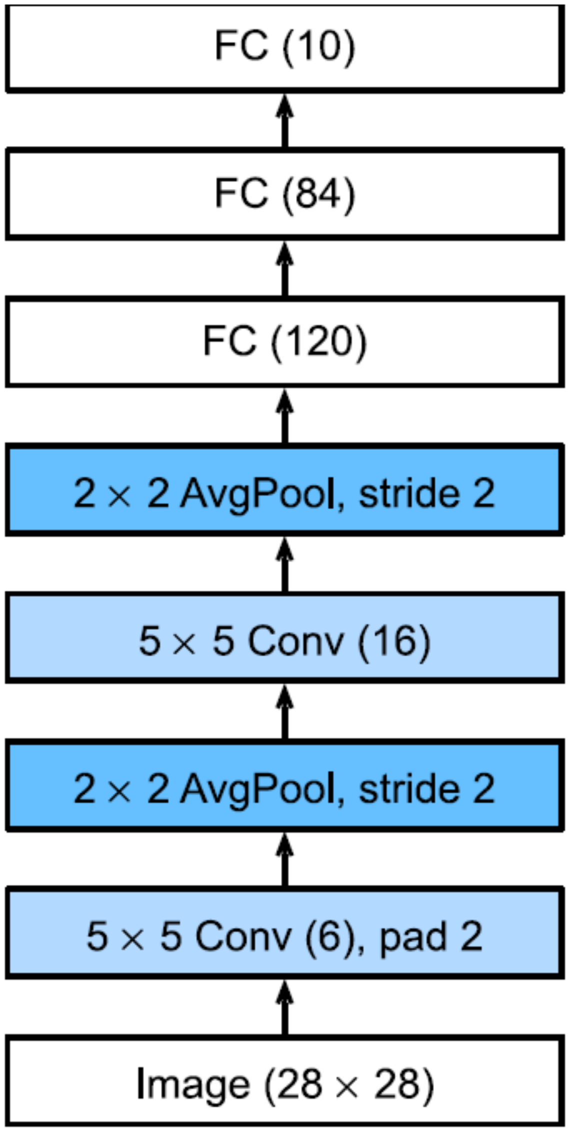
Convolutional Neural Networks (LeNet)



- The basic units in each convolutional block are a convolutional layer, a sigmoid activation function, and a subsequent average pooling operation.
- Note that while ReLUs and maxpooling work better, these discoveries had not yet been made at the time.
- Each convolutional layer uses a 5 *5 kernel and a sigmoid activation function.
- These layers map spatially arranged inputs to a number of two-dimensional feature maps, typically increasing the number of channels.
- The first convolutional layer has 6 output channels, while the second has 16. Each 2*2 pooling operation (stride 2) reduces dimensionality by a factor of 4 via spatial down sampling.
- The convolutional block emits an output with shape given by (batch

LeNet-5

- Let’s see what happens inside the network.
- By passing a single-channel (black and white) 2828 image through the network and printing the output shape at each layer
- we can inspect the model to make sure that its operations line up with what we expect



UNC CHARLOTTE

General Trends

- Note that the height and width of the representation at each layer throughout the convolutional block is reduced (compared with the previous layer).
- The first convolutional layer uses 2 pixels of padding to compensate for the reduction in height and width that would otherwise result from using a 5*5 kernel.
- As we go up the stack of layers, the number of channels increases layer-over-layer from 1 in the input to 6 after the first convolutional layer and 16 after the second convolutional layer.
- Finally, each fully connected layer reduces dimensionality, finally emitting an output whose dimension matches the number of classes.

Deep Convolutional Neural Networks

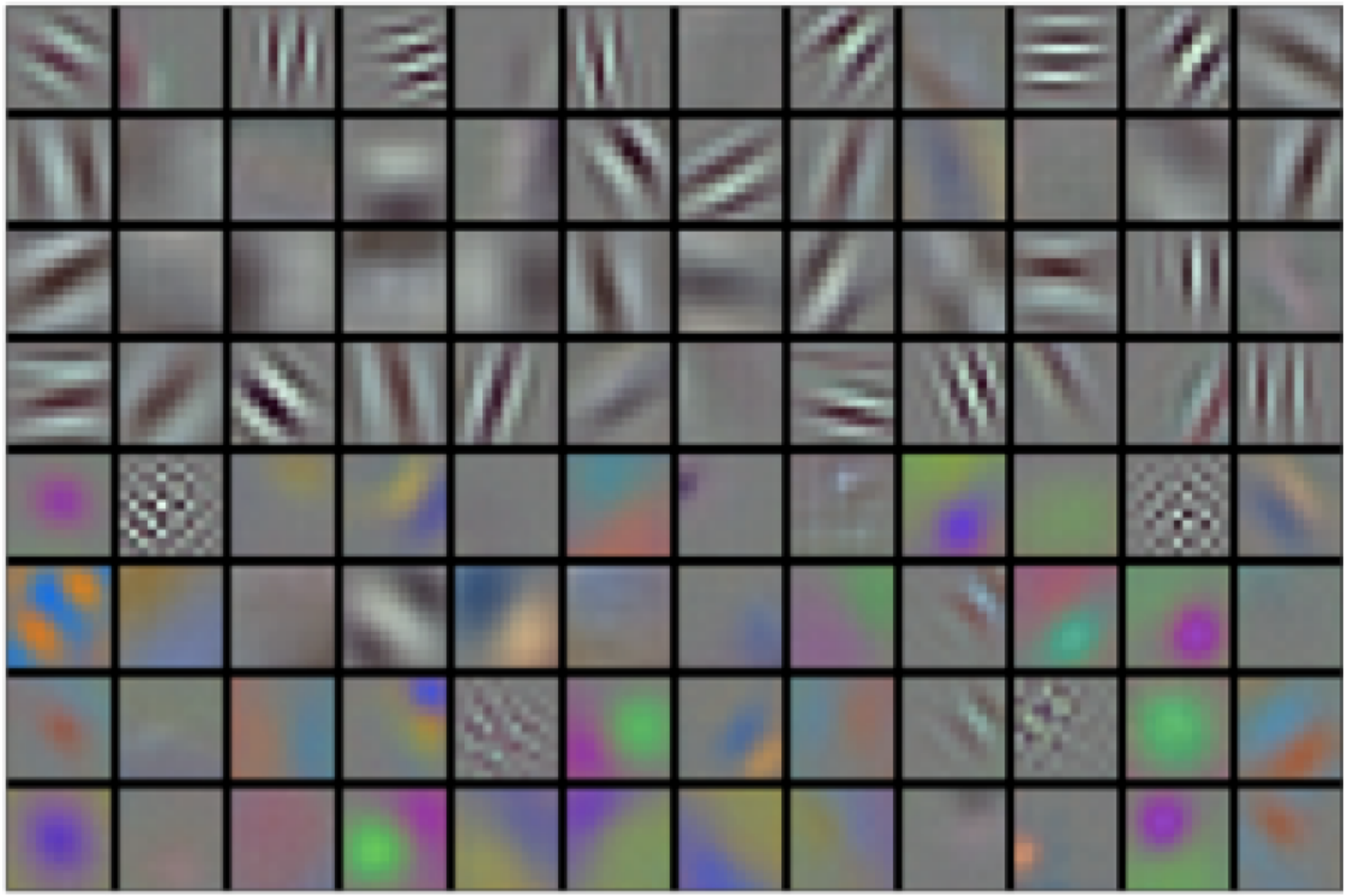
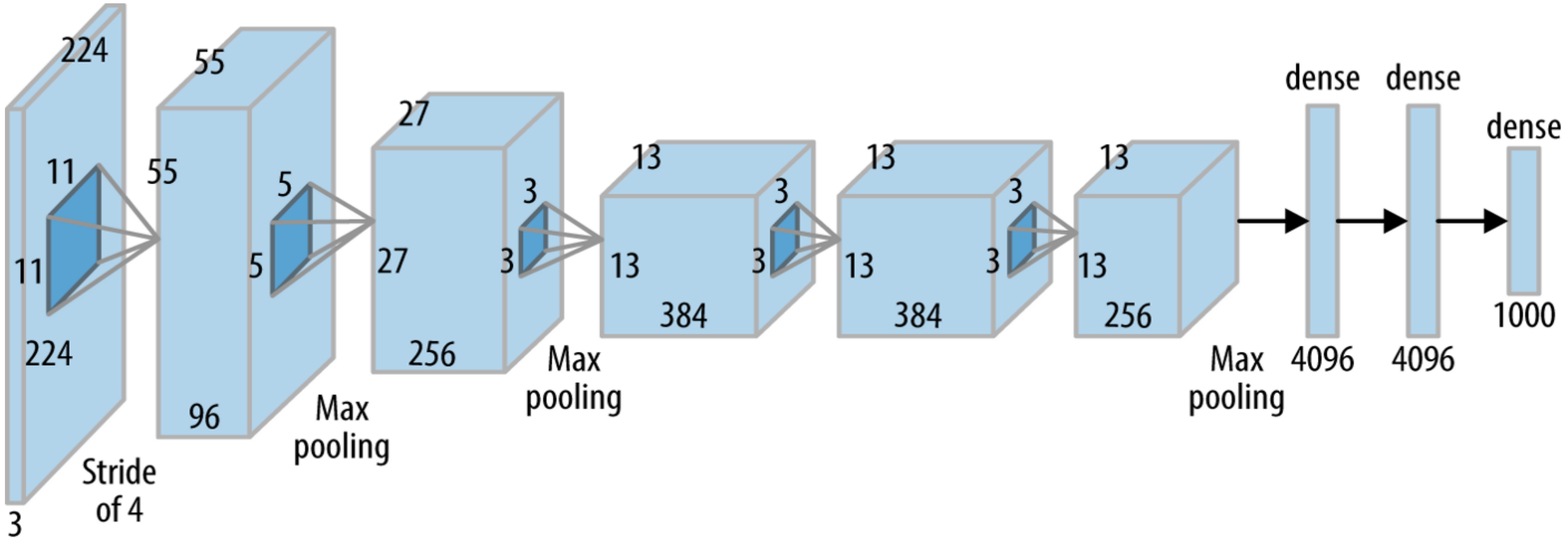


Image filters learned by the first layer of AlexNet.

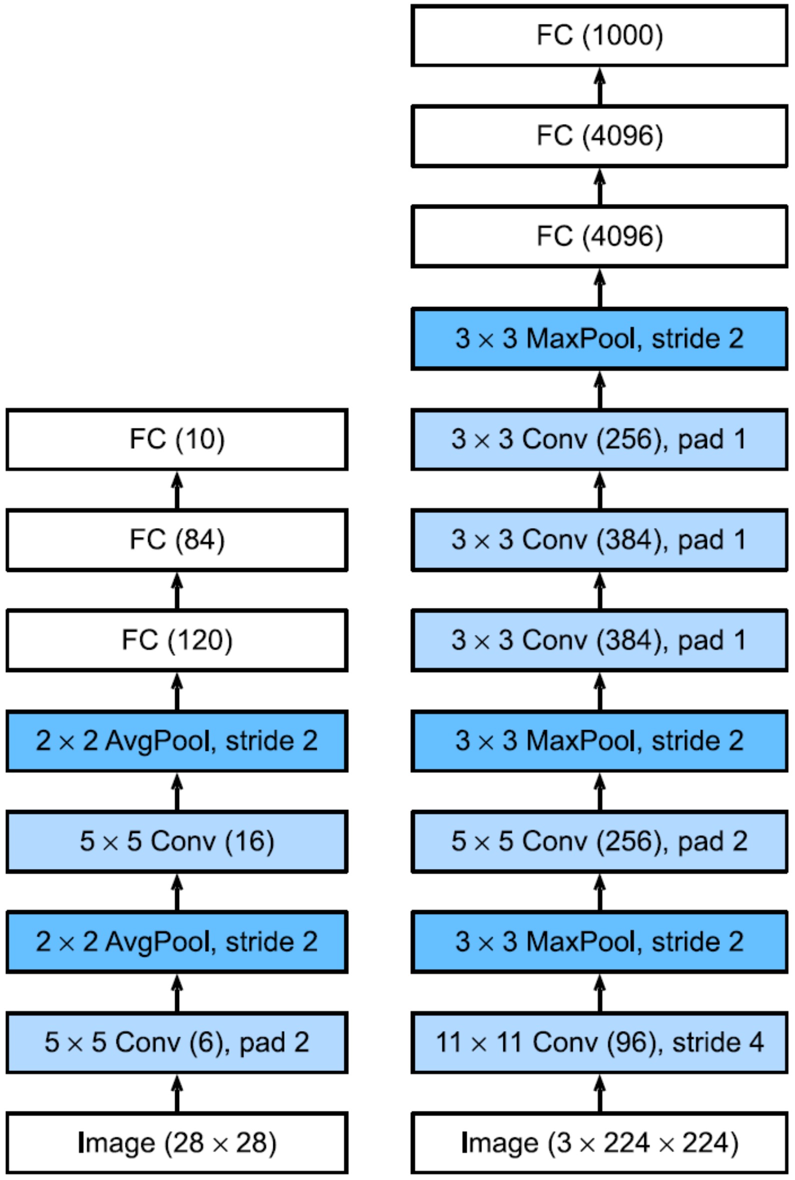
AlexNet

- AlexNet, which employed an 8-layer CNN, won the ImageNet Large Scale Visual Recognition Challenge 2012 by a large margin (Russakovsky *et al.*, 2013).
- This network showed, for the first time, that the features obtained by learning can transcend manually-designed features, breaking the previous paradigm in computer vision.
- The architectures of AlexNet and LeNet are strikingly similar, as Fig.8.1.2 illustrates.



UNC CHARLOTTE

AlexNet vs LeNet



- There are also significant differences between AlexNet and LeNet.
- First, AlexNet is much deeper than the comparatively small LeNet5. AlexNetv consists of eight layers: five convolutional layers, two fully connected hidden layers, and one fully connected output layer.
- Second, AlexNet used the ReLU instead of the sigmoid as its activation function. Let’s delve into the details below.
- Moreover, AlexNet has ten times more convolution channels than LeNet.

UNC CHARLOTTE

AlexNet Architecture

- In AlexNet’s first layer, the convolution window shape is 11 *11. Since the images in ImageNet are eight times higher and wider than the MNIST images, objects in ImageNet data tend to occupy more pixels with more visual detail.
- Consequently, a larger convolution window is needed to capture the object.
- After the last convolutional layer, there are two huge fully connected layers with 4096 outputs.
- Besides, AlexNet changed the sigmoid activation function to a simpler ReLU activation function.
- The gradient of the ReLU activation function in the positive interval is always 1. Therefore, if the model parameters are not properly initialized, the sigmoid function may obtain a gradient of almost 0 in the positive interval, so that the model cannot be effectively trained.
- AlexNet controls the model complexity of the fully connected layer by dropout while LeNet only uses weight decay.

AlexNet: Discussion

- AlexNet’s structure bears a striking resemblance to LeNet, with a number of critical improvements, both for accuracy (dropout) and for ease of training (ReLU).
- What is equally striking is the amount of progress that has been made in terms of deep learning tooling.
- What was several months of work in 2012 can now be accomplished in a dozen lines of code using any modern framework.
- Reviewing the architecture, we see that AlexNet has an Achilles heel when it comes to efficiency: the last two hidden layers require matrices of size 6400 4096 and 4096 4096, respectively. This corresponds to 164 MB of memory and 81 MFLOPs of computation, both of which are a nontrivial outlay, especially on smaller devices, such as mobile phones.
- This is one of the reasons why AlexNet has been surpassed by much more effective architectures that we will cover in the following sections.
- Nonetheless, it is a key step from shallow to deep networks that are used nowadays.
- Note that even though the number of parameters by far exceeds the amount of training data in our experiments (the last two layers have more than 40 million parameters, trained on a datasets of 60 thousand images), there is hardly any overfitting: training and validation loss are virtually identical throughout training. This is due to the improved regularization, such as Dropout, inherent in modern deep network designs.

Networks Using Blocks

- The design of neural network architectures has grown progressively more abstract, with researchers moving from thinking in terms of individual neurons to whole layers, and now to blocks, repeating patterns of layers.
- A decade later, this has now progressed to researchers using entire trained models to repurpose them for different, albeit related, tasks.
- Such large pretrained models are typically called *foundation models* (Bommasani *et al.*, 2021). Back to network design.
- The idea of using blocks first emerged from the Visual Geometry Group (VGG) at Oxford University, in their eponymously-named *VGG* network (Simonyan and Zisserman, 2014).
- It is easy to implement these repeated structures in code with any modern deep learning framework by using loops and subroutines.

Other Problems with Traditional CNNs

The basic building block of CNNs is a sequence of the following:
(i) a convolutional layer with padding to maintain the resolution
(ii) a nonlinearity such as a ReLU,
(iii) a pooling layer such as max-pooling to reduce the resolution.

One of the problems with this approach is that the spatial resolution decreases quite rapidly. For instance, in the case of ImageNet, it would be impossible to have more than 8 convolutional layers in this way.