18-819F: Introduction to Quantum Computing
47-779/785: Quantum Integer Programming & Quantum Machine Learning

Convolutional Neural Networks (CNN)

Lecture 05

2022.09.19.

Agenda

- Images and matrices
  - Unsuitability of conventional feed-forward DNN for image analysis
  - Imitating biology
- Convolutional neural networks
  - Filters (kernels) for feature identification
  - Image padding, pooling, and striding
  - Back propagation in CNN
- Example architecture of CNN for digit classification

Dealing with Images

- Images taken by modern digital cameras are large arrays of numbers, each corresponding to the brightness of a pixel at a specific location.
- Digitally scanned images produce much of the same arrays of numbers.
- Color images have three sets of arrays, each corresponding to a color channel for red, green, and blue.
- Each color array can store intensity values from 0 to 255; the 256 values can fit into 1 byte of information.
- The range of values correspond to the sensitivity of the human eye.

Image Intensity Data

- The image of the character **7** in a black background can be represented as an intensity pattern of numbers. A map of such a representation is a matrix that stores the values of the intensities as illustrated on the graphic to the right.

- The white regions have the highest intensity with values of 255, while the black background has the lowest intensity, with values of 0.

- A digital representation of 7 can therefore be thought of as an $8 \times 8 -$ pixel array (matrix) of numbers ranging from 0 to 255. This data can be store in 1 byte of information.

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	0	255	255	255	255	0	0
0	0	0	0	0	255	0	0
0	0	0	0	0	255	0	0
0	0	0	0	0	255	0	0
0	0	0	0	0	255	0	0
0	0	0	0	0	255	0	0
0	0	0	0	0	0	0	0

Parameters for the Image of the Character 7

- If the image of 7 which is now an 8×8 array (matrix) of numbers, were to be processed using ordinary, fully-connected deep neural network (DNN), the number of parameters that would have to be learned would be huge.

- To provide concrete context, the 8×8 matrix of numbers would require 64 input neurons (nodes).

- If the DNN has 2 hidden layers, each with 8 neurons (nodes), and an output layer with two neurons, the number of parameters for the fully-connected network would be:
- $(64\times8) + (8\times8) + (8\times2) = 592.$

- If each layer included a bias, we would need an additional $1 + 8 + 8 + 1 = 18$ parameters, bringing the total to $592 + 18 = 610.$

- The number of parameters explodes to a large number if we insist on using ordinary feed-forward neural networks.

Convolution

- Convolution is a mathematical operation that can be used to preprocess an image to reduce its size. Convolution in this context is slightly different from that used in signal processing.

- In signal processing, convolution is defined as an operation between two functions $f(x)$ and $g(x)$ such that

$$(f * g)(x) = \int f(u)g(x-u)du = \int_0^\infty f(u)g(x-u)du \quad \text{Eqn. (5.1).}$$

- Given $f(x) = x^3 - x$ and $g(x) = x$, then

$$(f * g)(x) = \int_0^x (u^3 - u)(x-u)\,du = \int_0^x (xu^3 - u^4 - xu + u^2)du = \left(\frac{xu^4}{4} - \frac{u^5}{5} - \frac{xu^2}{2} + \frac{u^3}{3}\Big|_0^x\right);$$

$$\Longrightarrow (f * g)(x) = \frac{x^5}{20} - \frac{x^3}{6}.$$

- This type of convolution is different in detail but similar in spirit to the convolution process used in neural network image processing.

Image Convolution

- Convolution in imaging is defined as an operation on two arrays of numbers, one large and the other small. The smaller array is slid over the large array sequentially across elements of the large array and then downward until all elements of the large array have been overlapped with those of the smaller one. At each overlap step, a result is obtained by multiplying the overlapping elements.

- The result of the operation between the two arrays is another array whose elements are called the feature map.

- The smaller array is called the filter (or kernel), whose elements transform the larger arrays.
- For an image array I and a filter array F, the convolution operation can be written as

$$G[m,n] = (I * F)[m,] = \sum_j \sum_k I[j,k]F[m-j,n-k] \quad \text{Eqn. (5.2).}$$

Example of an Image Array Convolution

- We illustrate the process of image convolution by taking an image of 6×6 pixels whose elements are numbers and a filter with a 3×3 array of numbers. See illustration on right.

- First pixel value of feature map is obtained by overlapping the kernel with the 3×3 area of the image in the top left corner. Performing the element-by-element multiplication and then adding the results gives the first pixel of the feature map in the top left corner. Thus

$$G[1,1] = (10\times1) + (10\times1) + (10\times1) + 0 + 0 + 0 + (10\times(-1)) + (10\times(-1)) + (10\times(-1)) = 0.$$

- The rest of the elements of the feature map can be obtained in a similar manner. The filter is slid one step to the right until the end and then one step down from the beginning until the bottom of the image is reached.

10	10	10	10	10	10
5	5	5	5	5	5
10	10	10	10	10	10
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Image

*

1	1	1
0	0	0
−1	−1	−1

Kernel

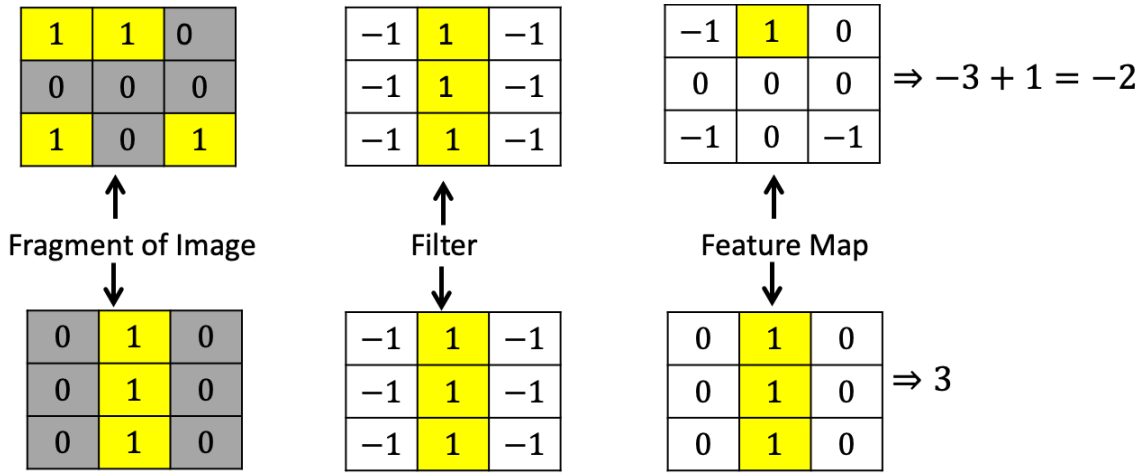
=

0	0	0	0
15	15	15	15
10	10	10	10
0	0	0	0

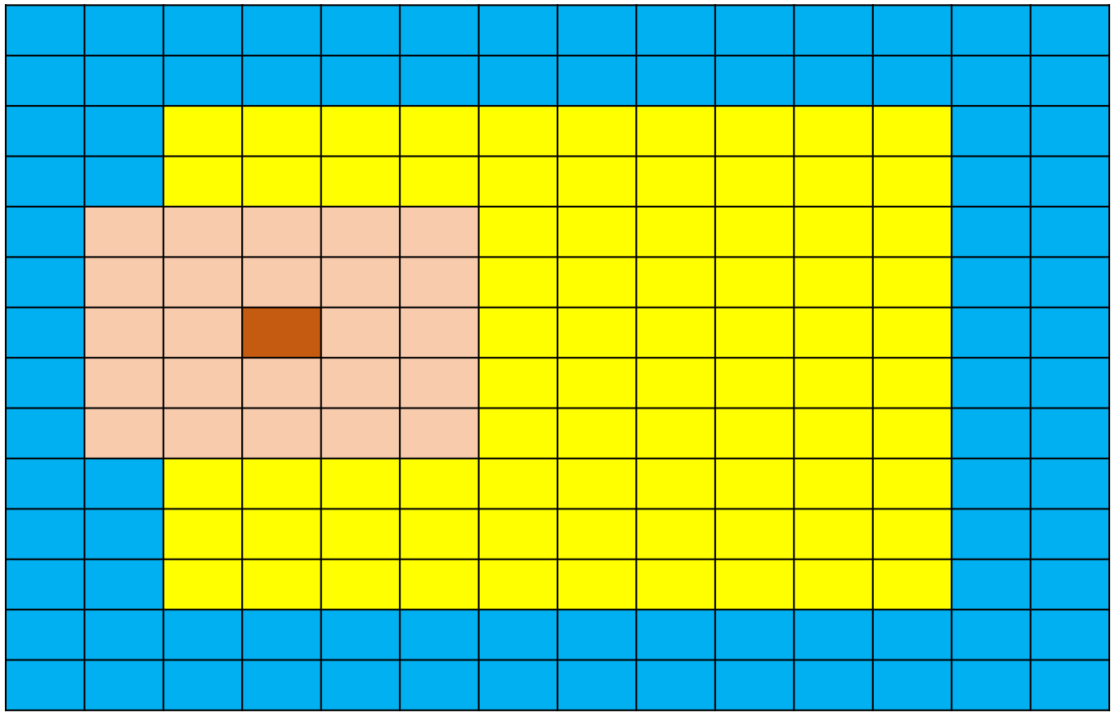
Feature Map

Filters and Features

- Filters are usually designed to identity features in images and are sometimes called feature detectors.
- They can be designed to look for diagonal lines, horizontal lines, vertical lines or some other feature that can be designed into a filter.
- A filter looking for vertical lines in fragments of images is shown on the right. In the first 3×3 fragment of the image, there is no vertical line, and the result of the convolution process is negative. In the second 3×3, there is a vertical line, and the feature map values indicate this by a positive number.



Padding



- As a filter is stepped across and down an image, loss of image data can occur if care is not taken to properly prepare the image for convolution.
- Padding is a technique used to prevent loss of information at the edges of the image. It is common to apply zero value pixels (zero-padding) all around the 10×10 yellow image area as shown by the blue color.
- The 5×5 filter (rust color) can now be applied without loss of any information as it is stepped across and down image.

Feature Map Relationship to Size of Filter and Image

- A feature map size is related to the image and filter (kernel) size. For an $n \times n$ image and a filter size of $f \times f$, the feature map size is given by

$$Output = (n - f + 1) \times (n - f + 1) \quad \text{Eqn. (5.3).}$$

- If the image is padded, then the feature output map size is given by

$$Output = (n + 2p - f + 1) \times (n + 2p - f + 1) \quad \text{Eqn. (5.4).}$$

- For an unpadded image (known as valid), the feature map size is given by Eqn. (5.3) above.

- If the image is padded and we would like the feature map size to be exactly the same size as the original image (known as same), then the padding must be determined from

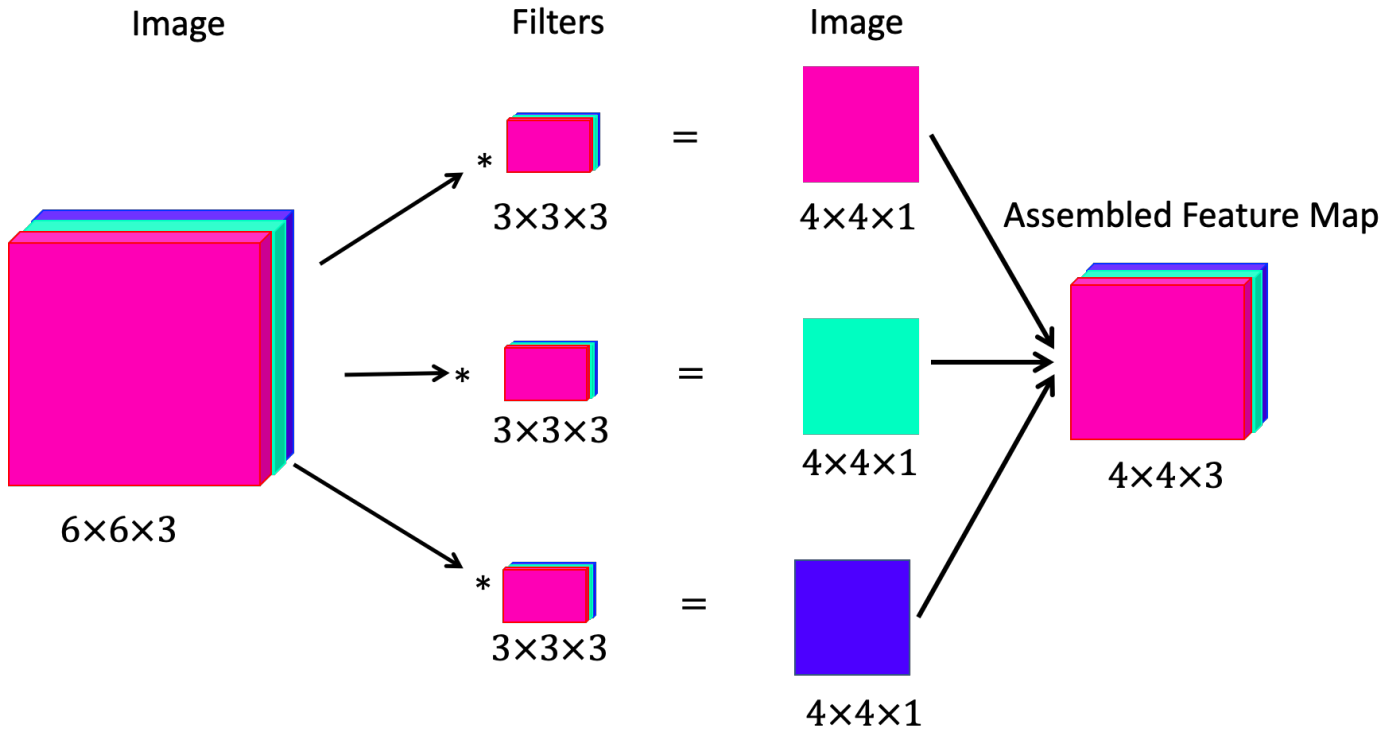
$$n + 2p - f + 1 = n \Longrightarrow p = \frac{f-1}{2} \quad \text{Eqn. (5.5)}$$

Striding the Filter During Convolution

- A filter can be stepped across and down an image during the convolution process one pixel at a time. However, it is sometimes advantageous to slide the filter in step sizes of 2 or more pixels per step.
- Striding leads to a faster convolution operation.
- Although it can be performed at different horizonal and vertical steps across the image, it is more usual to use the same stride step size, s, horizontally and vertically.
- For color images, the same stride size must be used for all channels.
- If the image size is $n \times n$, with a padding size of p, and a filter size $f \times f$, one can determine the size off the output feature array for stride size s from

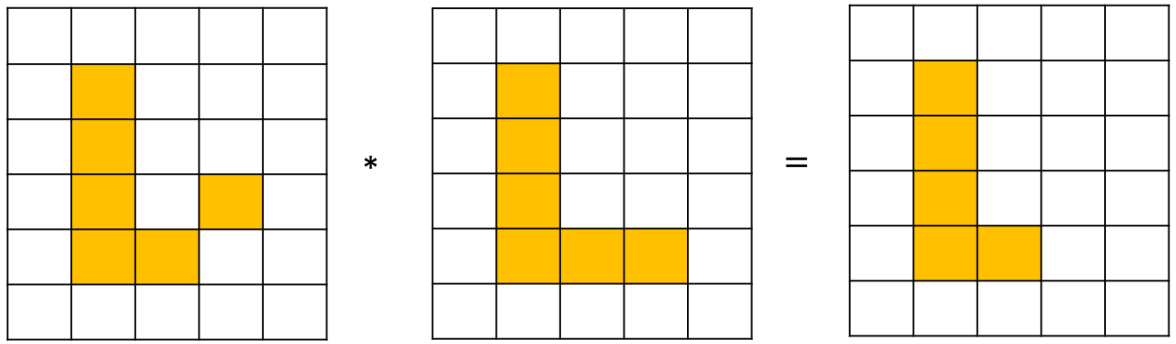
$$Output = \left[\frac{(n+2p-f)}{s} + 1\right] \times \left[\frac{(n+2p-f)}{s} + 1\right] \qquad \text{Eqn. (5.6).}$$

Multidimension Convolution with Multiple Filters



- If an image has color, convolution is performed over 3D (with a tensor). The 2D image is essentially distributed over 3 channels of the colors red, green, and blue (which is also a tensor). The filter for the convolution must have the three channels for the colors. The graphic above illustrates the process for convolution with 3 filters, each with 3 channels.

Feature Error Identification



- Sometimes a filter designed to look for certain features may miss them because the image maybe slightly corrupted, for example, by pixels that are out of place by some mistake.
- Such a situation might occur during identification of the letter *L* in the image in the illustration. A pixel has inadvertently been moved to a wrong location. The filter will therefore report no match.
- If only the filter had a slightly wider field of view, it might have correctly identified the letter.

Pooling

- Pooling is technique that can fix the problem of slightly misplace pixels during convolution.
- The method subdivides the image into blocks and then either records the average pixel (average pooling) value for each block or it only records the highest value within the block. Recording the highest value is called max pooling.

- Max pooling is used more frequently because it has been found to be more efficient during the learning process.

- Pooling is most effective when multiple convolutions are applied in a network

Pooling in a Segment of an Image

1	2	−3	2
3	6	12	5
4	−1	4	6
−2	3	9	1

Input Image

1	2	−3	2
3	6	12	5
4	−1	4	6
−2	3	9	1

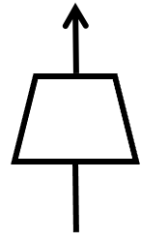
Subdivided Image

Max

6	12
4	9

Average

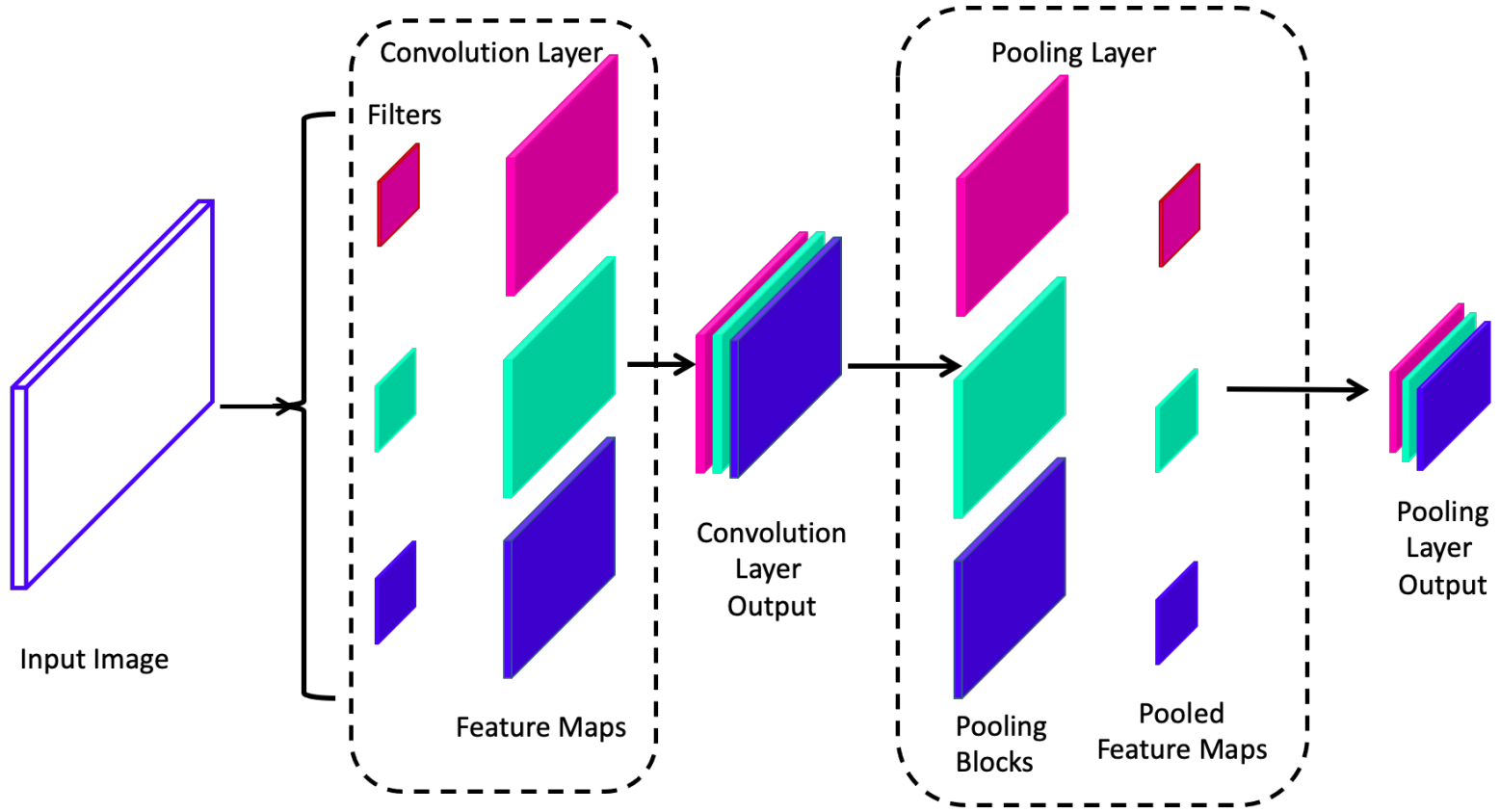
3	4
−1	5



Symbol for
Max Pooling

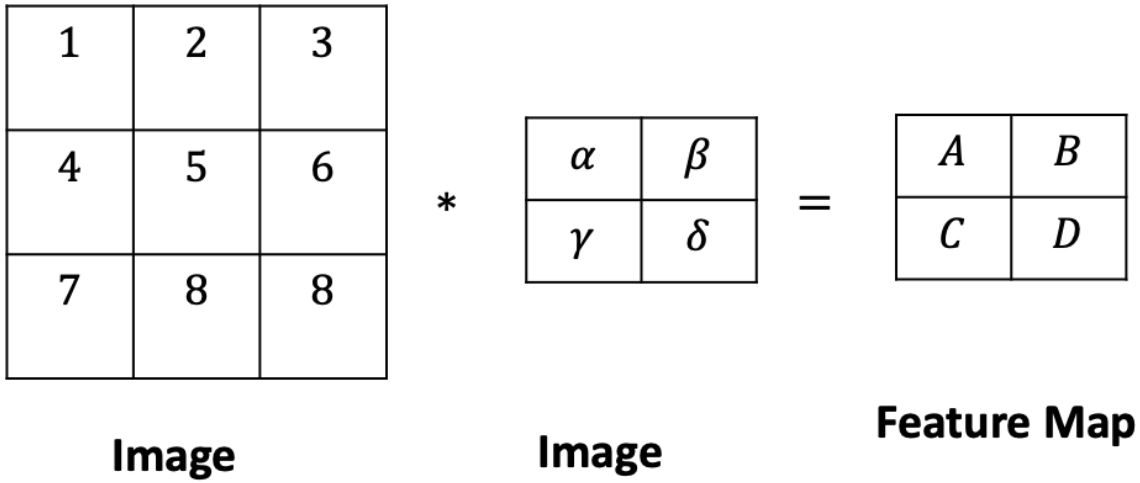
- The graphic above illustrates how pooling can be applied to a segment of an image. One can either use average pooling or max pooling. Both are acceptable but most practical implementations use max pooling.

Convolution and Sequential Pooling



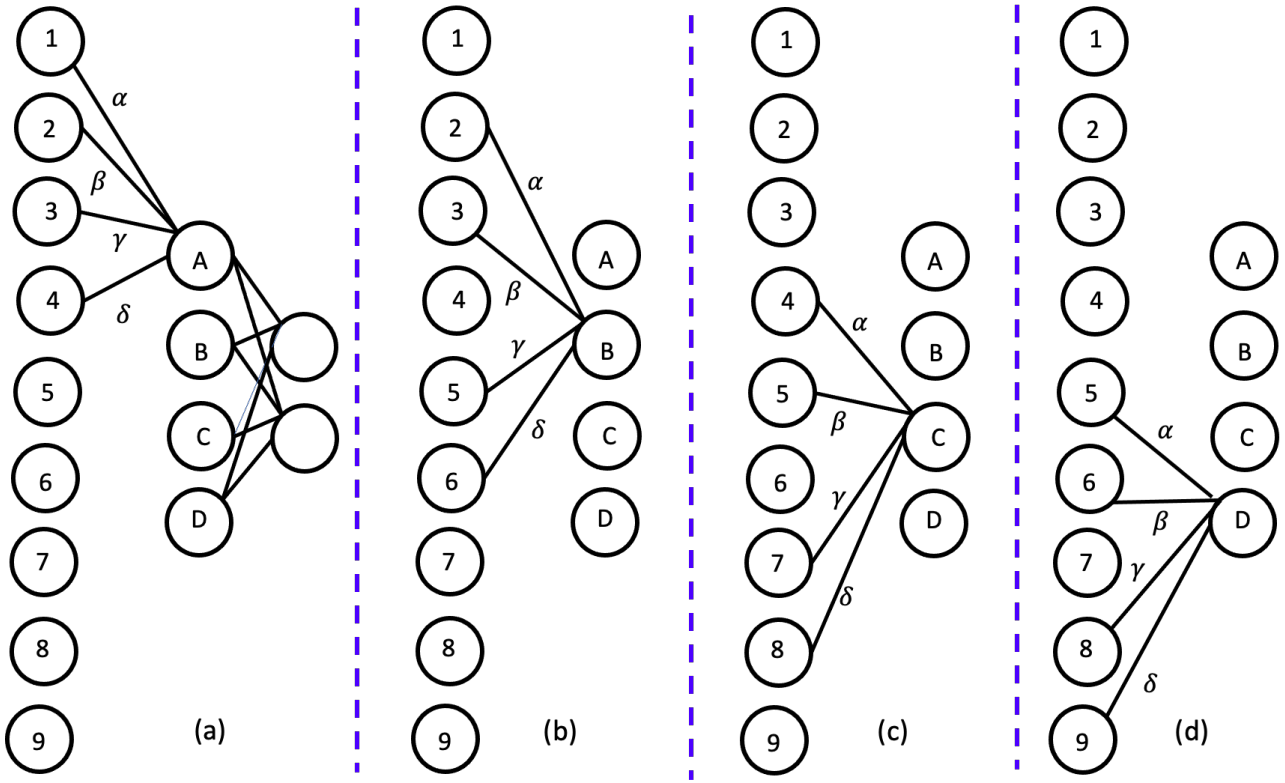
- It is often the case that convolution is followed by pooling to reduce the size of an image. Such a process is depicted in the graphic above.

Benefits of Convolution



- Beyond the fact that convolutional neural networks (CNN) are more suited for image transformation and processing, they are also not fully connected, which means the number of parameters that must be learned is not as large as it would be if image analysis was performed in fully connected DNNs.
- Furthermore, some nodes in convolutional neural networks share weights, which reduces storage requirements.
- These benefits can be gleaned from the image, filter, and feature arrays to the left, examined in conjunction with their network representations.

Simple Network for Convolution

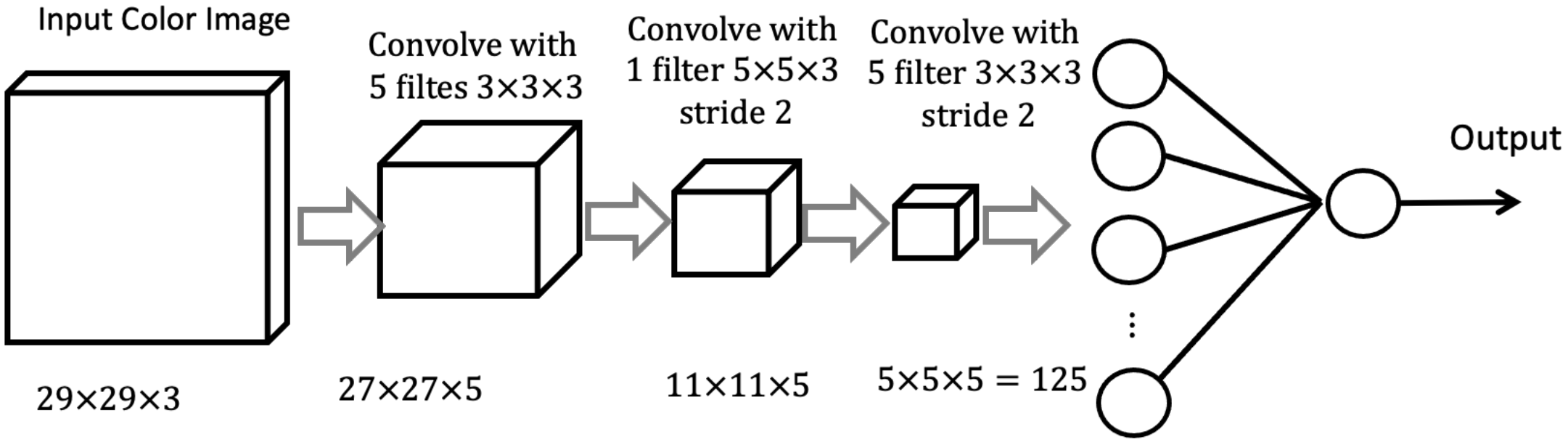


- The partial neural network for a convolutional layer (from previous slide) illustrating how some nodes such as 1, 2, 3, 4 and 5 share the same weights which reduces storage demands. Note also that network is not fully connected. Furthermore, a single value from the filter affects every element of the feature map.

Simple Convolution Neural Network

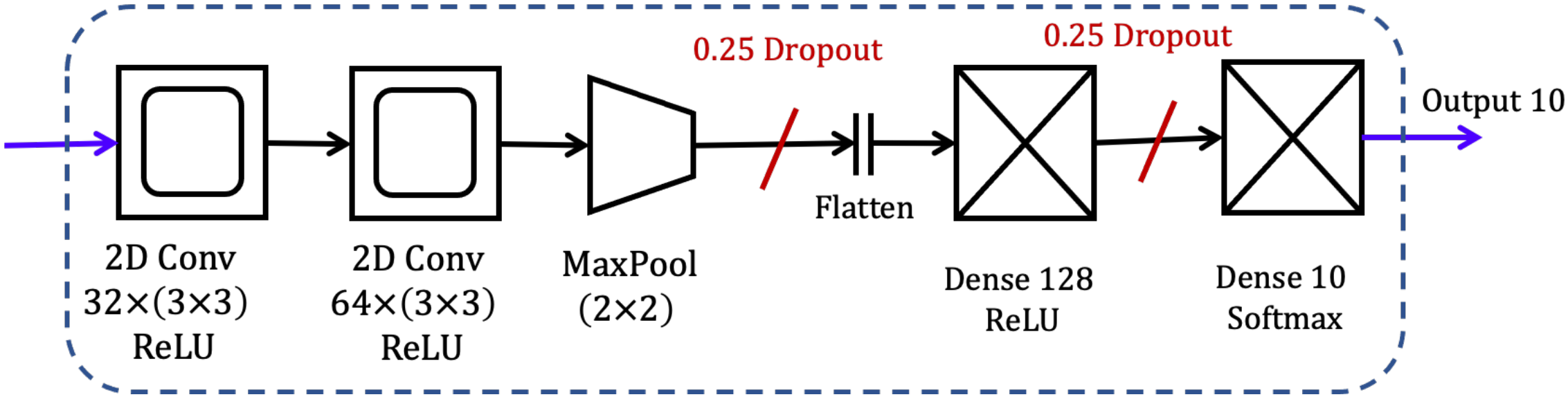
- Suppose we have a $29\times29\times3$ color image we want classify with a yes/no decision classifier. No padding is applied to the image. The following layers are applied to the image:
  - The image is first convolved with 5 filters of size $3\times3\times3$ in a single stride. The resulting feature map will be $27\times27\times5$.
  - Then it is convolved with a $5\times5\times5$ filter, with a stride step size of 2, the resulting feature map will $11\times11\times5$.
  - Further convolution with 5 filters each of dimension $3\times3\times3$ with stride step size of 2, results in $5\times5\times5$.
- Notice that each step reduces the size of the image. At the last step before the classifier the image is unrolled into a vector with $5\times5\times5 = 125$ features and then connected to the classifier. See graphical representation of the network on the next slide.

Simple Convolution Neural Network…



- The simple convolution neural network described in previous slide is graphically illustrated above. Note that no padding is applied to the image before convolution.

MNIST Digit Convolutional Neural Network

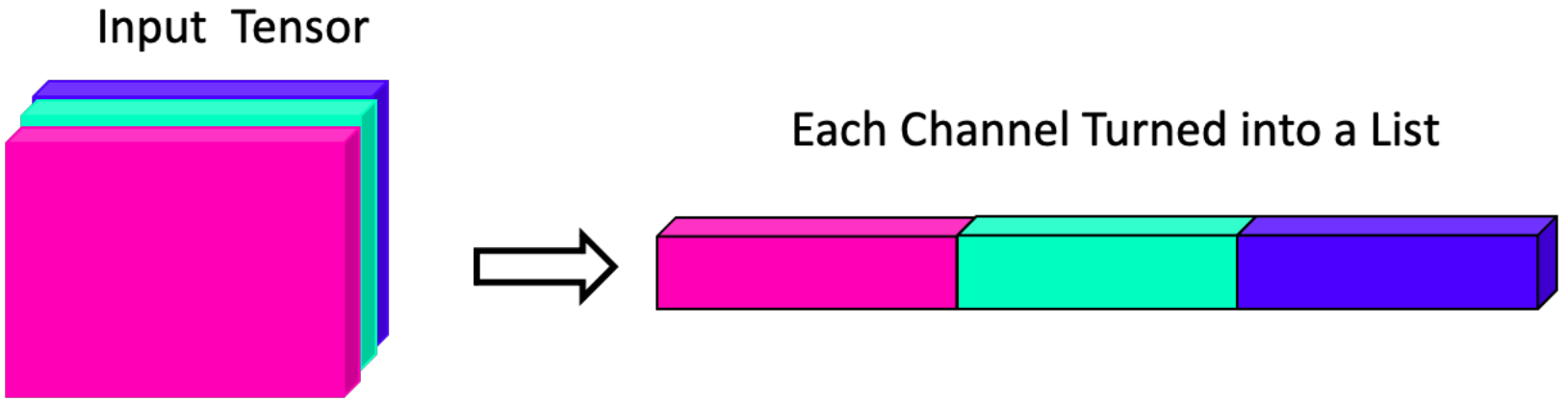


- One of the early success stories of convolutional neural networks was in handwritten digit recognition. There is a data base of handwritten digits kept by NIST called MNIST that is widely used as a test.

- The architecture of the network above (found in most Deep Learning frameworks*) is widely used for classifying handwritten digits.

*For example, Keras Machine Learning Library

Flattening Layer in a Convolutional Neural Network



- The action of the flattening layer in a convolutional neural network (such as in the digit classifying architecture) is to convert a tensor into a vector (list) before it is provided as input to fully-connected layers before classification.

- This action is illustrated in the graphic above, where each channel in the tensor is first separated from it and then vectorized.

Some Applications of Convolutional Neural Networks

- First success story of CNNs from 1990s was in reading checks at banks; document analysis in newspaper stories; and handwriting recognition.

- Today, facial recognition is probably the most widely used application: identifying faces in a picture by picking out unique features (CNN excel at visual classification of objects).

- Automakers keen on driverless cars are focusing on CNNs for autonomy since these networks can identify and avoid objects.

- Medical image analysis is another major area of application – identifying diseases

Summary

- Discussed natural suitability of images to convolutional neural networks
  - Structure of a CNN
  - Back propagation in CNN
  - Feature extraction filters, pooling and striding

- Example of simple CNN architectures for hand-written digit recognition