哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

Big data technology and Practice

李春山

2020年9月30日

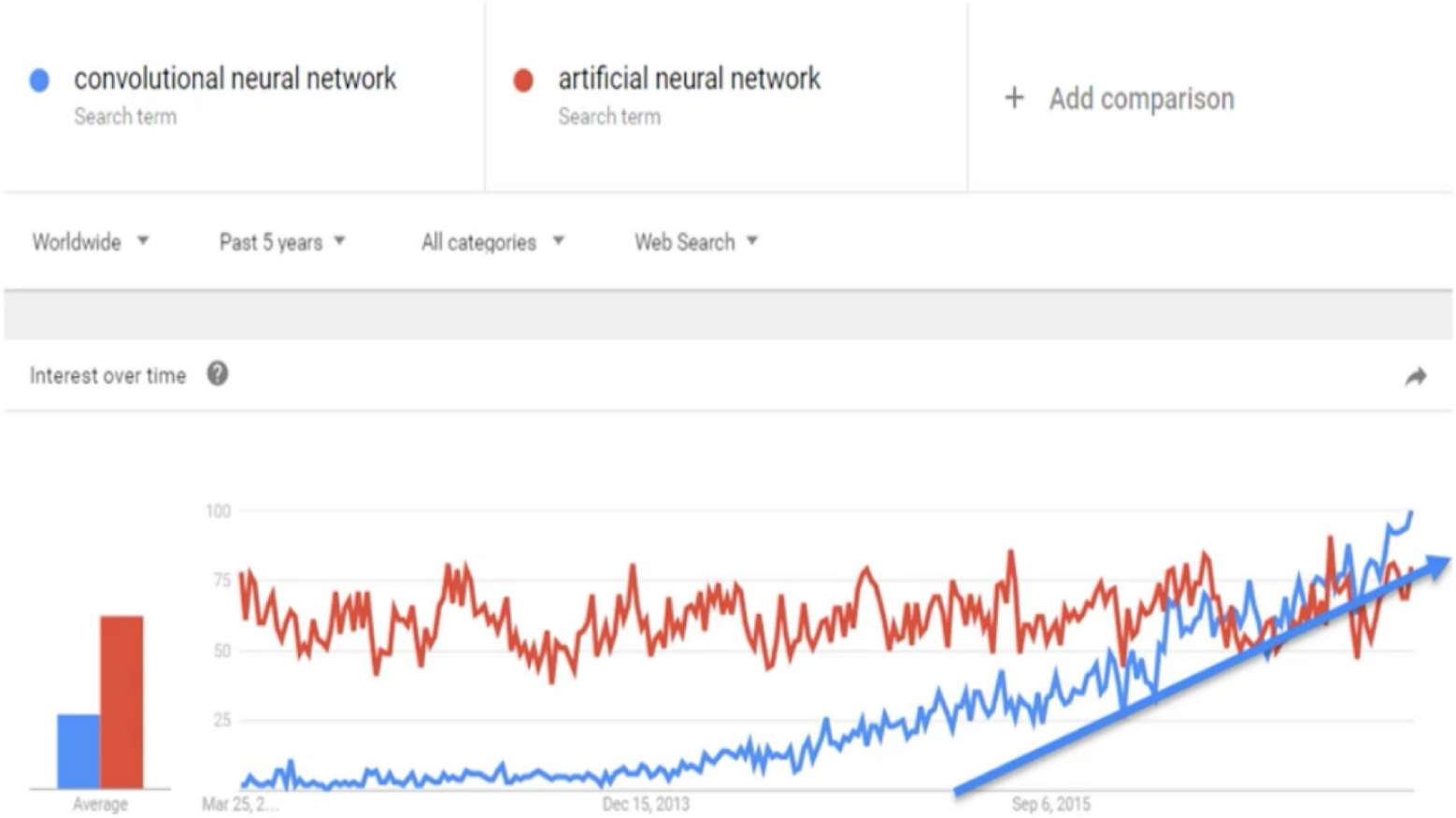
内容提要

Chapter 14: Convolutional Neural Networks (CNNs)

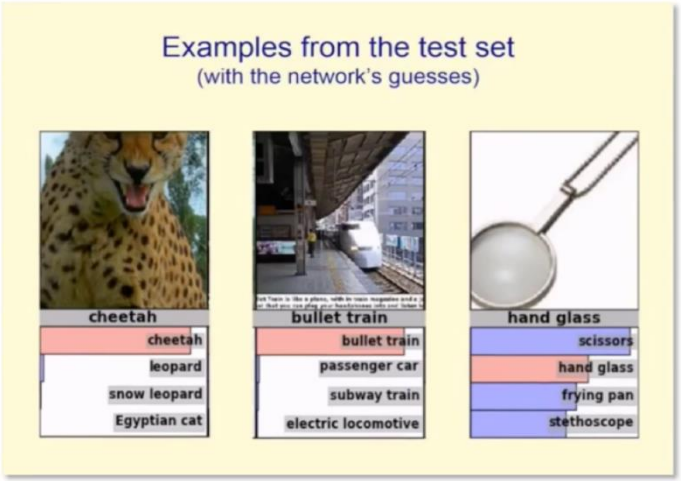
Chapter 14: Convolutional Neural Networks (CNNs)

- CNNs emerged from the study of the brain's visual cortex, used in image recognition since the 1980s.
- In the last few years, CNNs have managed to achieve superhuman performance on some complex visual tasks. Such as image search services, self-driving cars, automatic video classification systems, and more.
- Moreover, CNNs are not restricted to visual perception: they are also successful others, such as voice recognition or natural language processing (NLP);
- However, we will focus on visual applications for now.

Interest over time



Source: google trends



For each has a number of possibilities, how to recognize or decide it.

Image Source: a talk by Geoffrey Hinton

She is old lady or young girl?

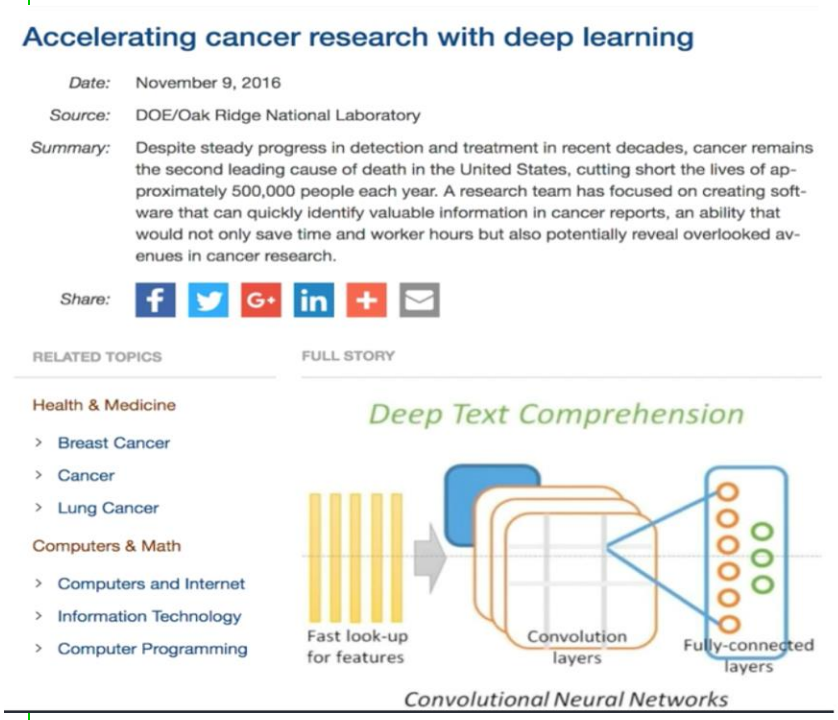
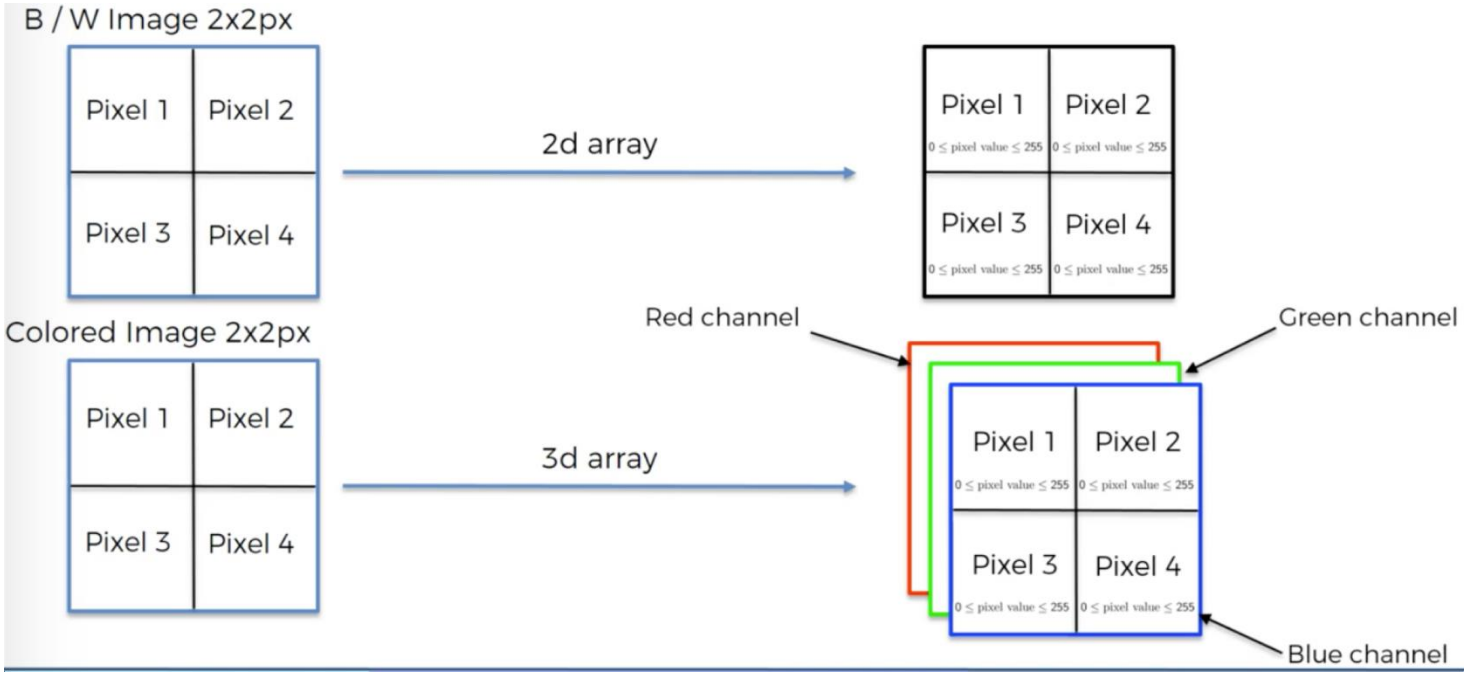


Image Source: Wikipedia

1. Concepts and Steps for Building CNNs

- **1). Image features**



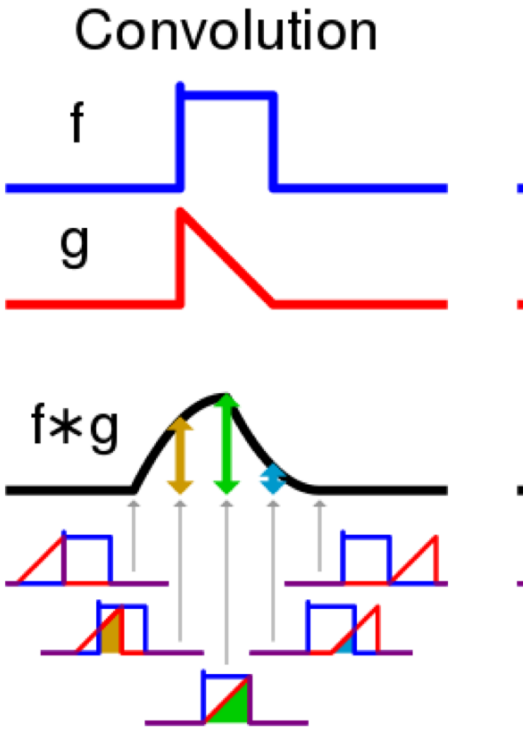
- An image, as input to computers, is represented as an array of pixel values. Depending on the resolution and size of the image, for instance, 32 x 32 x 3 array of numbers (The 3 refers to RGB values). Each of these numbers is given a value from 0 to 255 which describes the pixel intensity at that point.

2). Convolution

- Originally, a convolution function. It is a mathematical operation on two functions (f and g) to produce a third function that expresses how the shape of one is modified by the other. :

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

- Based on this idea, a number of concepts are developed for building a convolutional neural network.



3). Feature detector, also called kernel or filter or feature identifier

Definition:

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

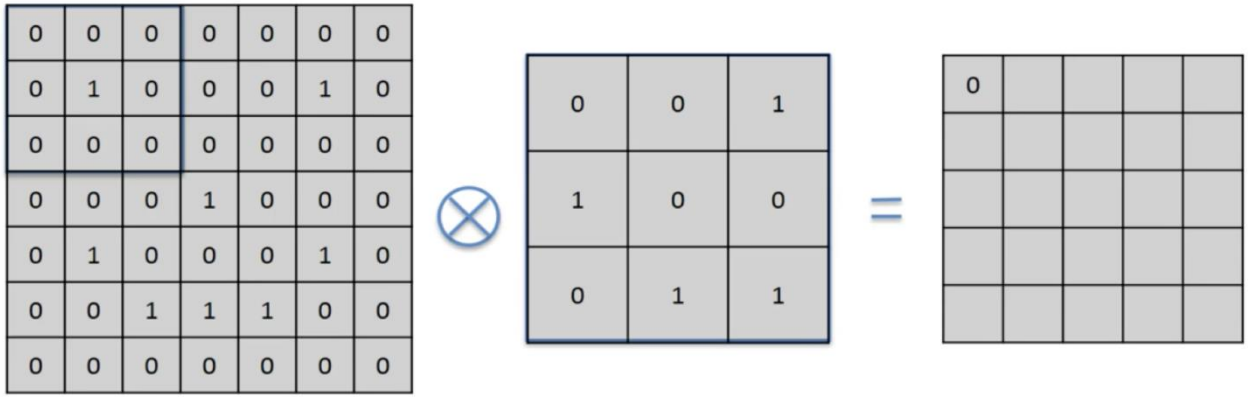
0	0	1
1	0	0
0	1	1

Feature Detector

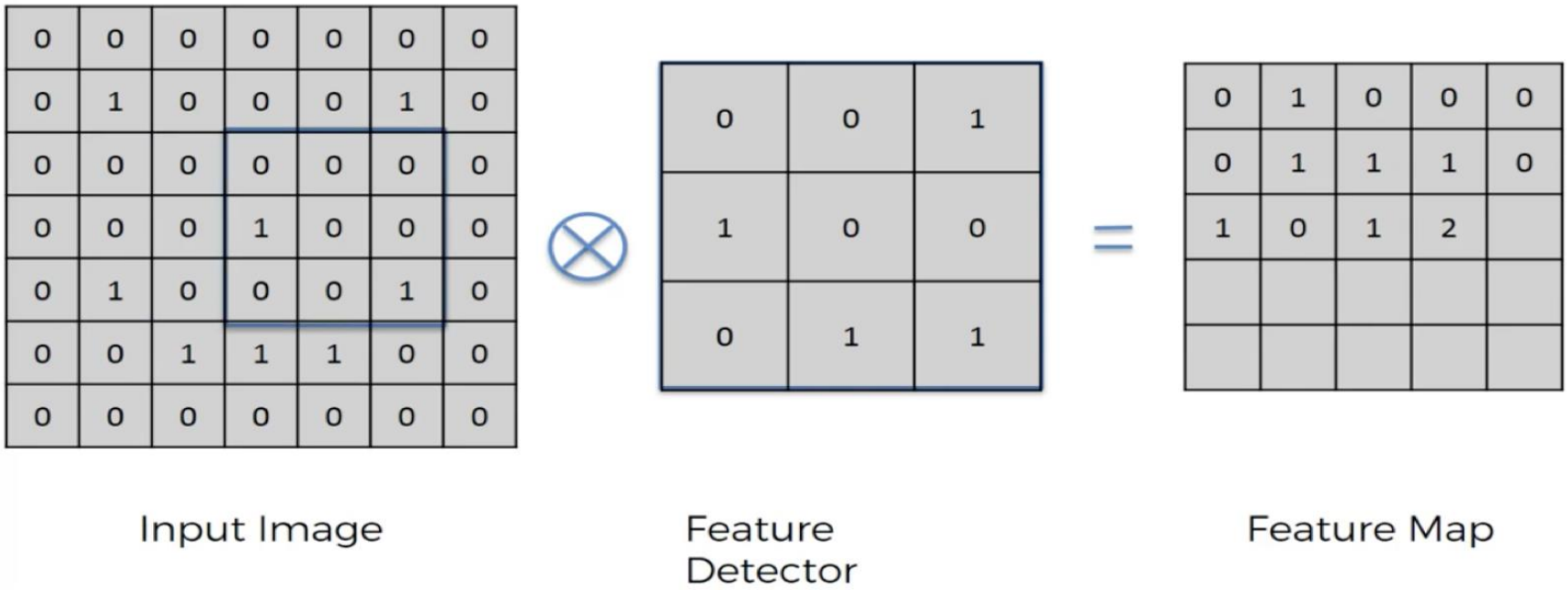
Here, Detector feature = 3x 3

- stride = 1 pixel
- Multiply the values of each two corresponding positions and then add them.
- Moving from left to right with stride = 1 here
- Then, move to the next row with stride =1, repeat as above.
- The region that it is filtered over is called the receptive field

important note: the depth of this filter has to be the same as the depth of the input, typical 3 (RBG)

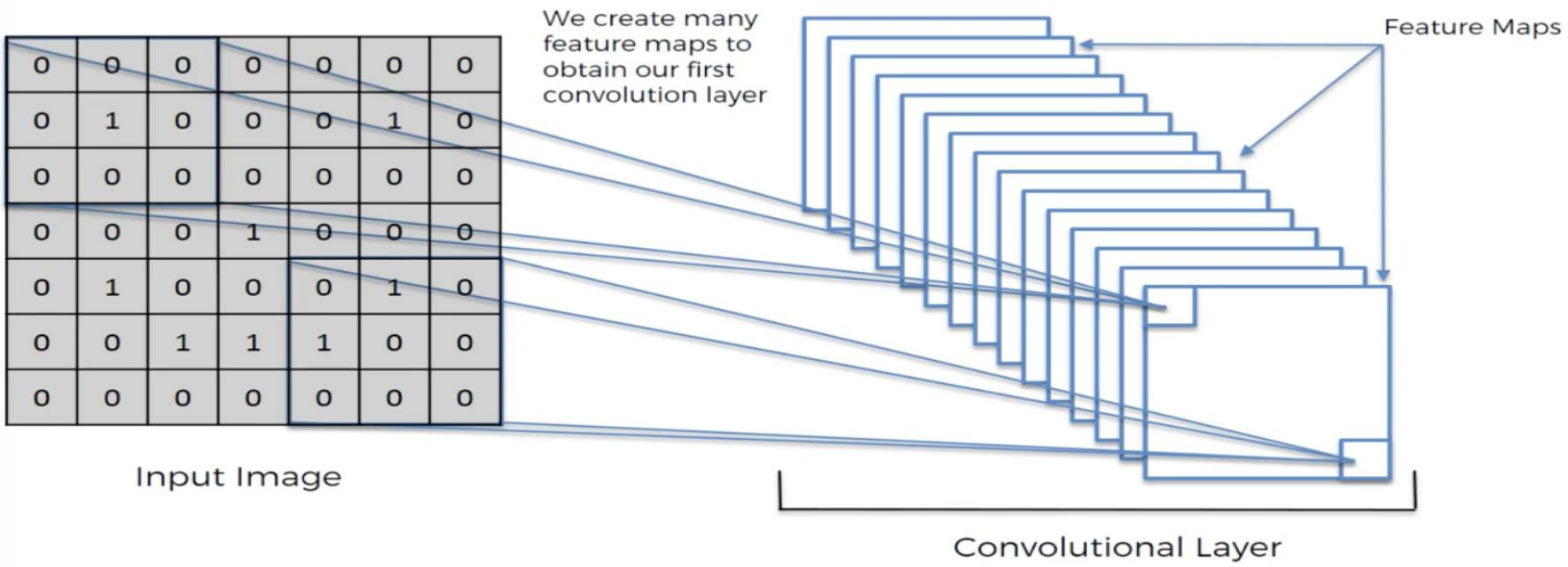


Feature detector



- The convolution of input image and feature detector produce "Feature Map" or "Activation Map"

Feature detector

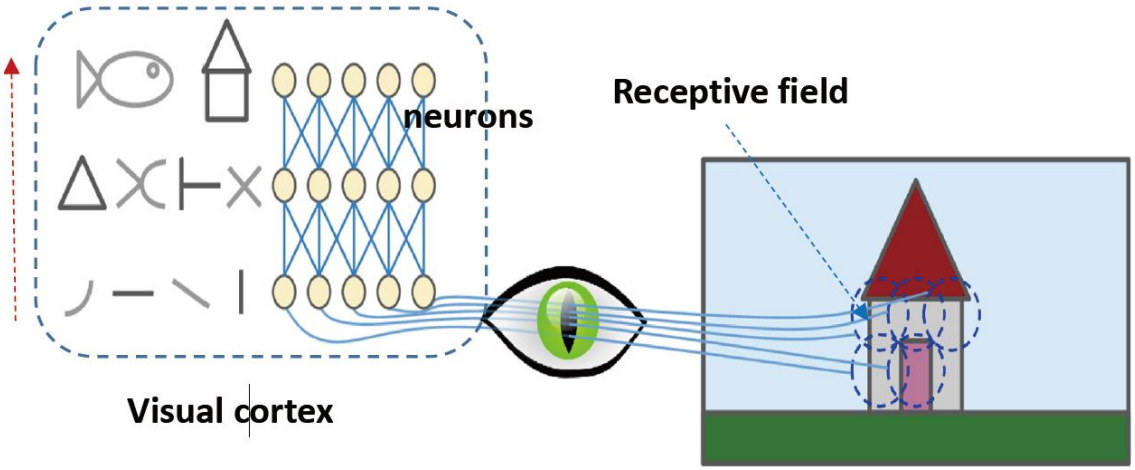


- Depending on the position, each feature map may have its own filter, so, there are different filters applied and produce these feature maps.
- We may create many feature maps to obtain our first convolutional layer.

The Visual Cortex and Convolution Layer

- David H. Hubel and Torsten Wiesel performed a series of experiments on cats in and later on monkeys, giving crucial insights on the structure of the visual cortex, and received the Nobel Prize in Physiology in 1981.

They showed that many neurons in the visual cortex have a small local receptive field, meaning they react only to visual stimuli located in a limited region of the visual field (See dashed circles).



- The receptive fields of different neurons may overlap, and together they tile the whole visual field.

The Visual Cortex and Convolution Layer

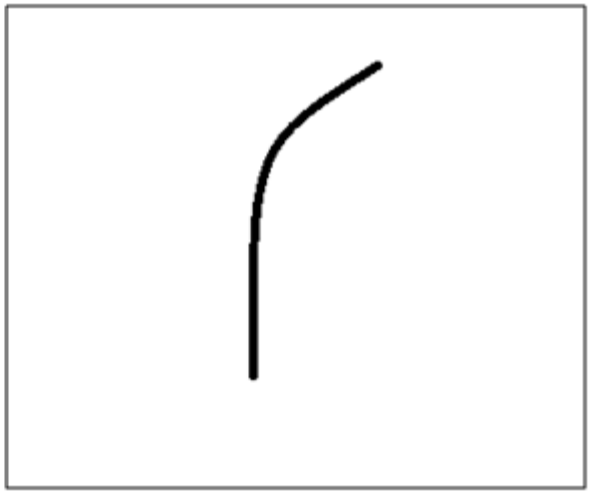
- Some neurons react only to images of horizontal lines, while others react only to lines with different orientations .
- Some neurons have larger receptive fields, and they react to more complex patterns that are combinations of the lower-level patterns.
- The observations led to the idea that the higher-level neurons are based on the outputs of neighboring lower-level neurons and each neuron is connected only to a few neurons from the previous layer.
- This powerful architecture is able to detect all sorts of complex patterns in any area of the visual field.
- These studies of the visual cortex inspired the neocognitron, introduced in 1980, now called convolutional neural networks.
- An important milestone was a 1998 paper by Yann LeCun, L.on Bottou, Yoshua Bengio, which introduced the famous LeNet-5 architecture, widely used to recognize handwritten check numbers.

Now, let's revisit the "Convolution"

- A **filters** can be thought of as a **feature identifier**.
- Straight edges, simple colors, and curves are some of **features** in an image, at low level.
- For instance, curve filter will have a pixel structure in which there will be higher numerical values along the area that is a shape of a curve.

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

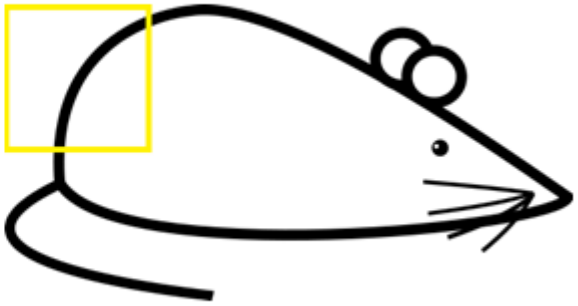


Visualization of a curve detector filter

Now, let's revisit the "Convolution"

- In the following sample, let's put our filter at the top left corner.

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

=

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

- Note: if there is a shape that generally resembles the curve that this filter is representing, then all of the multiplications summed together will result in a large value!

Now, let's revisit the "Convolution"

- Now let's see what happens when we move our filter.

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

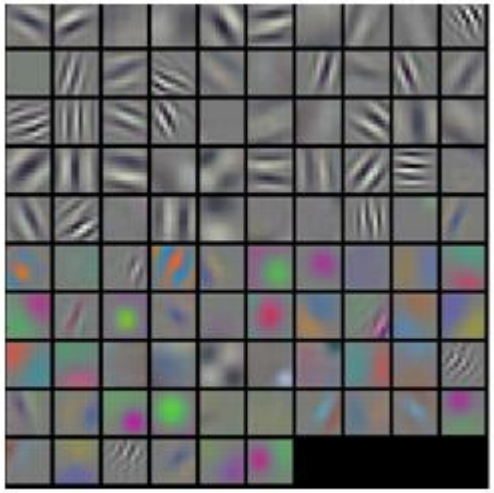
Pixel representation of receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

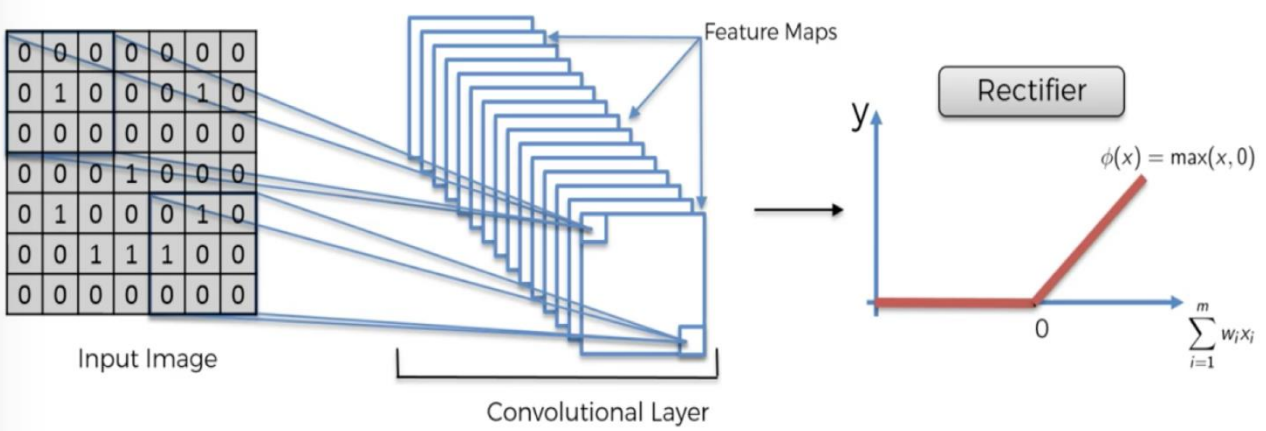
Multiplication and Summation = 0



Visualizations of filters

- The value is much lower, as there wasn't anything in the image section that responded to the curve detector filter.
- We can have other filters for lines. The more filters, the greater the depth of the feature map, and the more information we have about the input volume.
- In the picture as above, you'll see some examples of actual visualizations of the filters of the first convolutional layer of a trained network

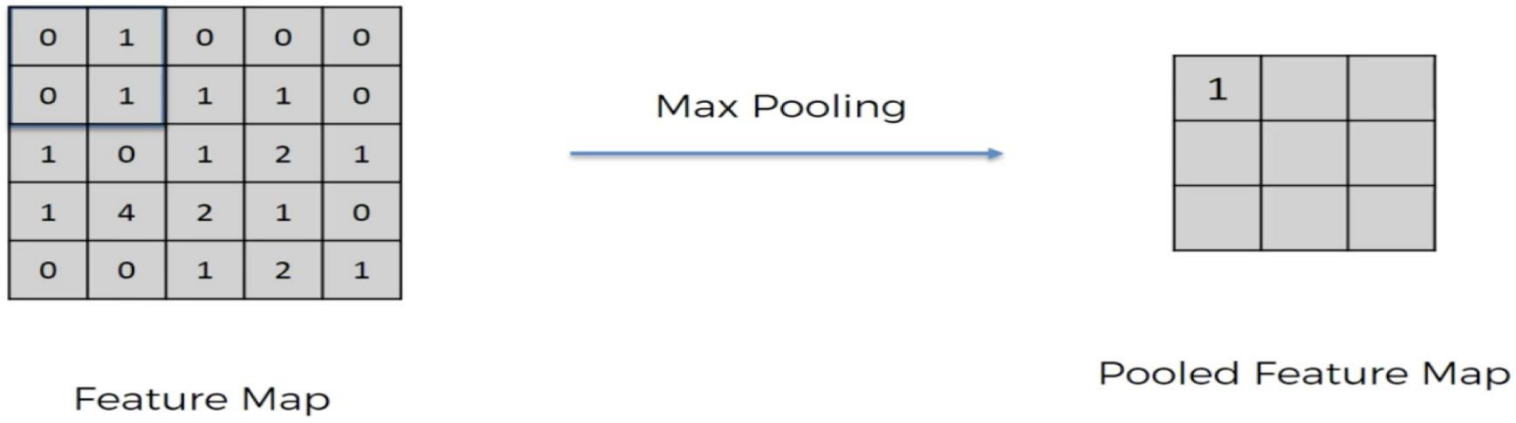
5). ReLU layer

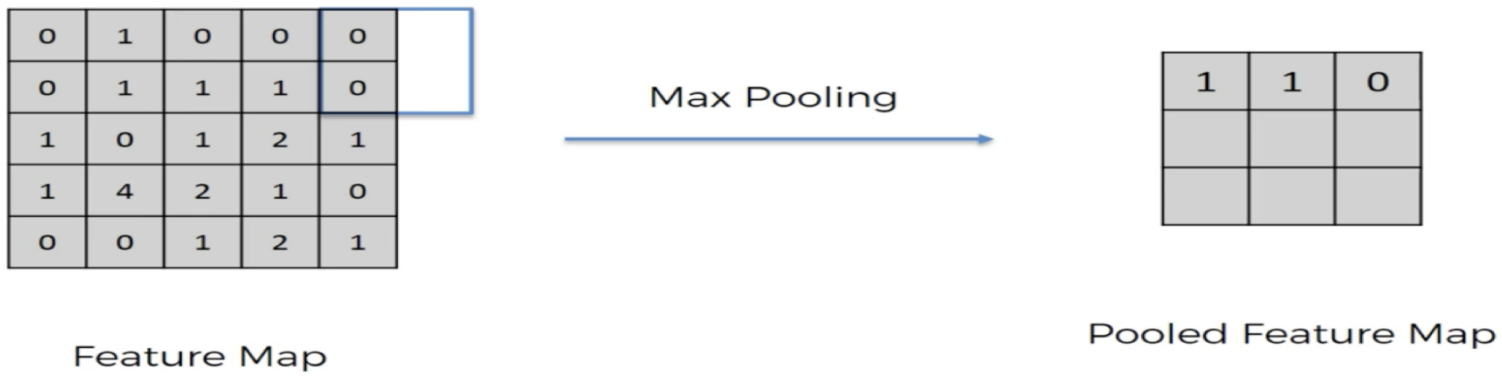


- Why needed?
- I. To increase non-linearity in our image or in neural network , as images themselves are highly non-linear . Especially, if you're recognizing different objects next to each other or just the transition between adjacent pixels.
- II. When applying convolution with feature maps, we risk of creating something linear and therefore we need to break up them.
- III. In the past, nonlinear functions like tanh and sigmoid were used, but researchers found out that ReLU layers work far better, as the network is able to train a lot faster without making a significant difference to the accuracy.

6). Pooling

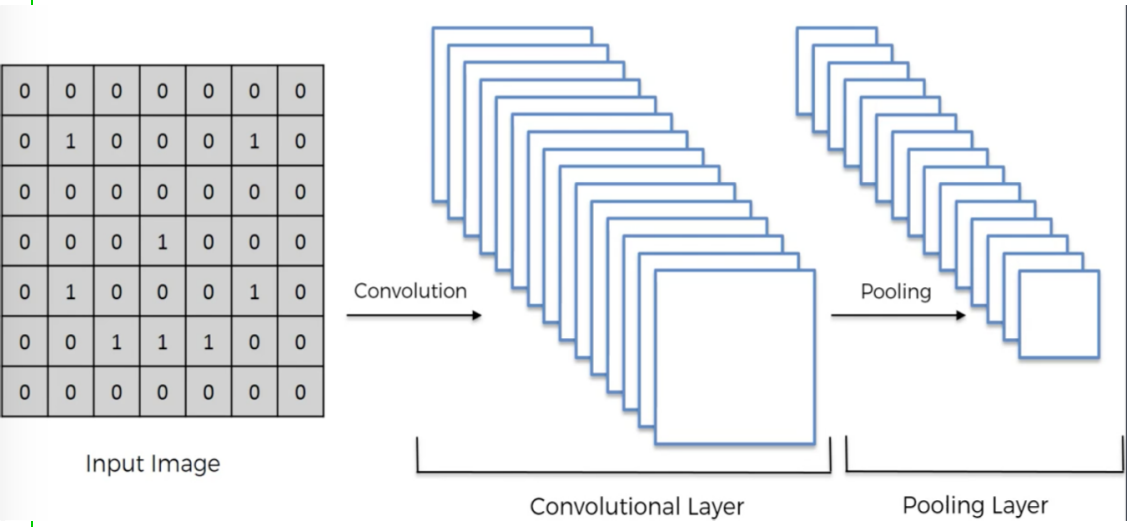
- There are different ways for pooling. For example, ”max pooling” is to find a maximum value in that box (filter), slides a stride ->





Why pooling?

- As the maximal numbers in a feature map represent where to found the closest similarity to a feature. So, it can preserve the features right.
- Alternatively, Mean pooling.
- Another benefit is to reduce the size of images (parameters), that are going to the final layer of a neural network for preventing overfitting.



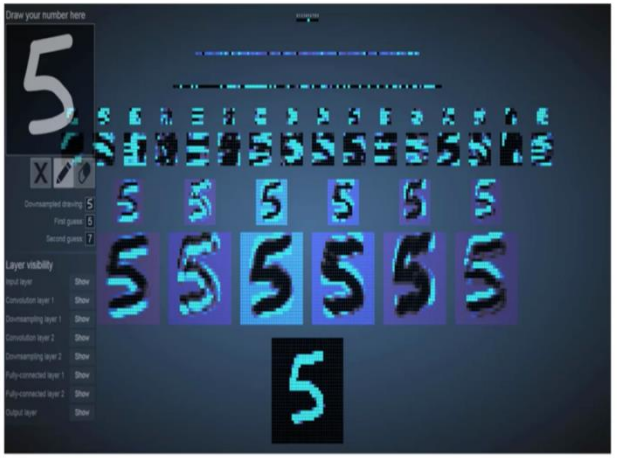
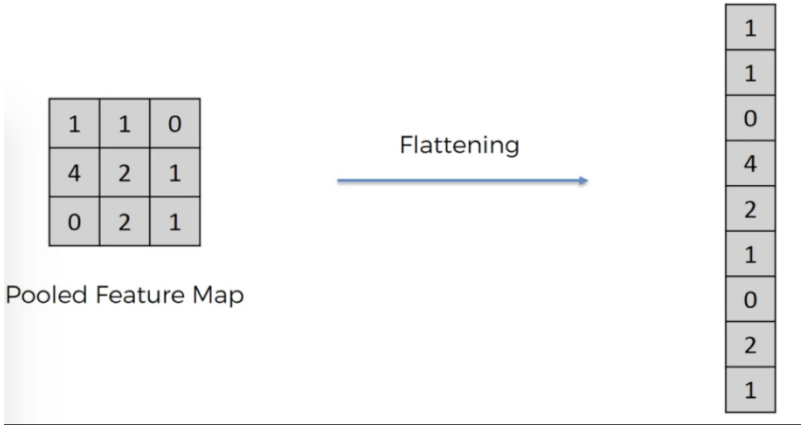
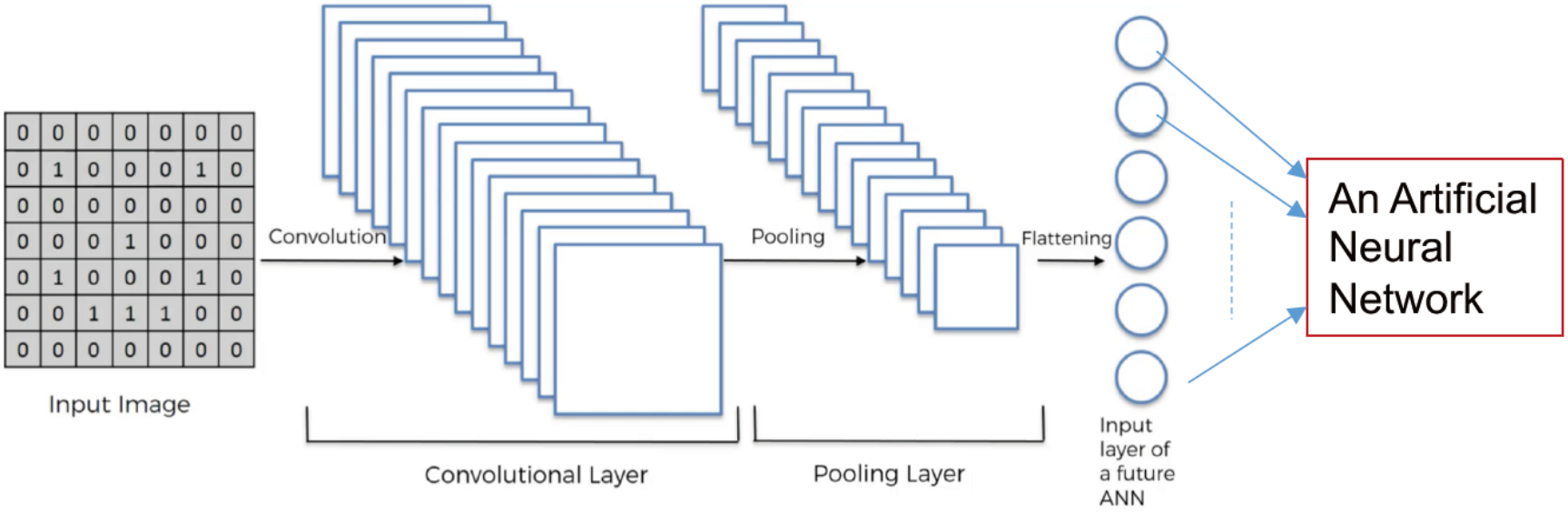


Image Source: scs.ryerson.ca/~aharley/vis/conv/flat.html

7). Flattening

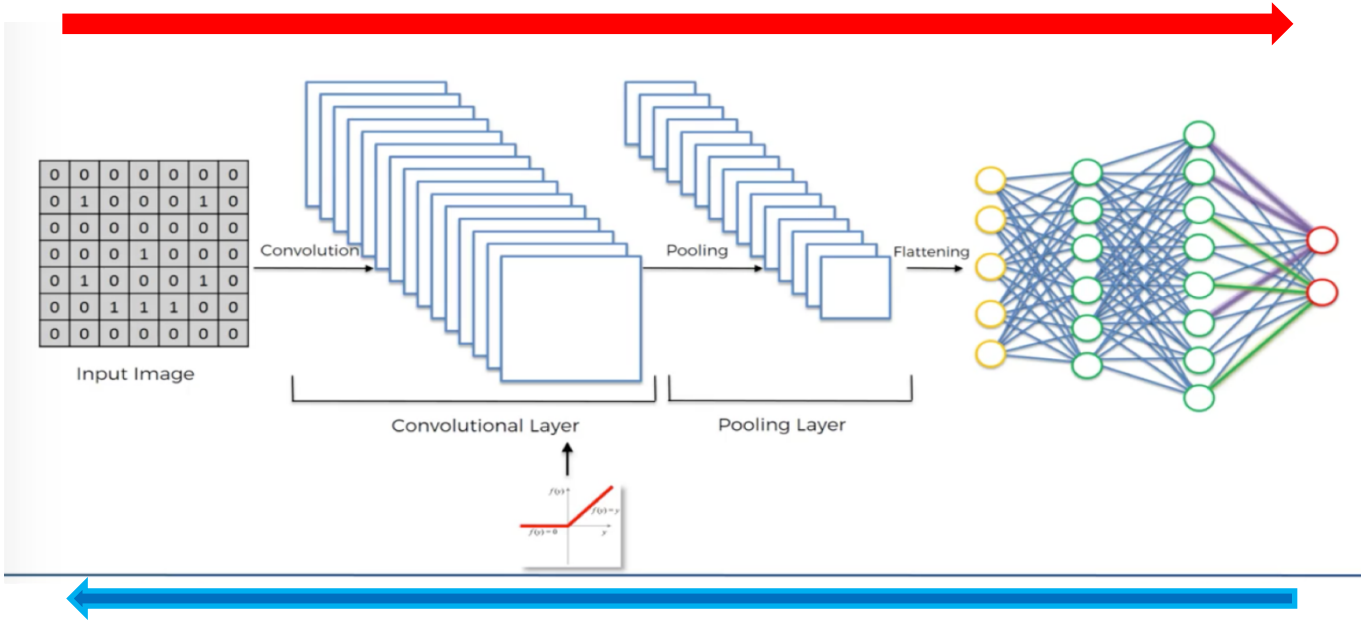
- All of the pooled images are flatten into one long vector (or column of all of these values), as input into an artificial neural network





8). Full connection

- It is a fully connected artificial neural network where all of these features are processed through the network.
- Then, the final fully connected layer performs the voting towards the classes.
- All of them, as a whole, is trained through a forward propagation and back propagation process.



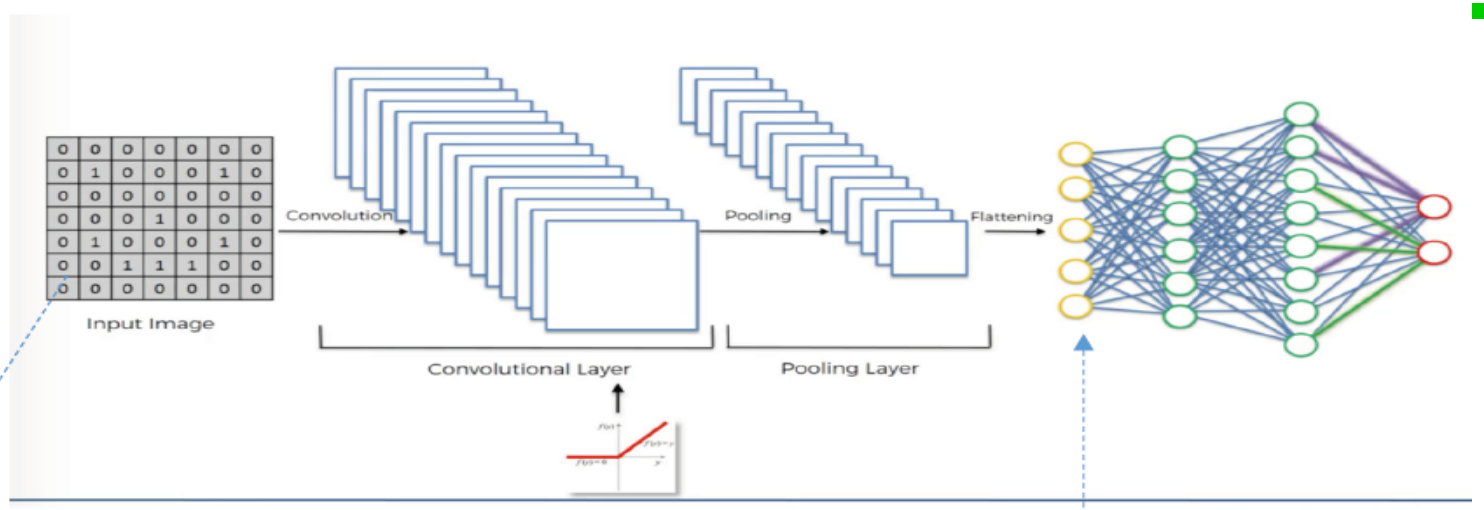
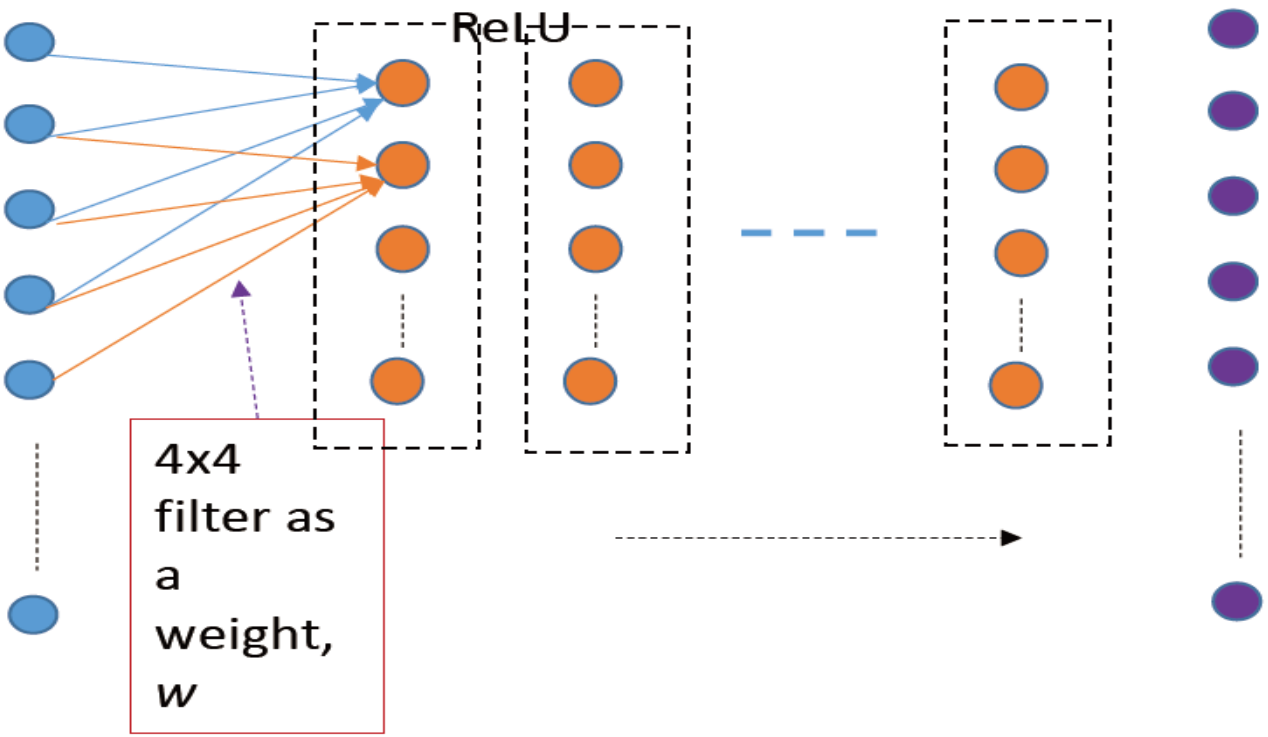


Image Input features vector

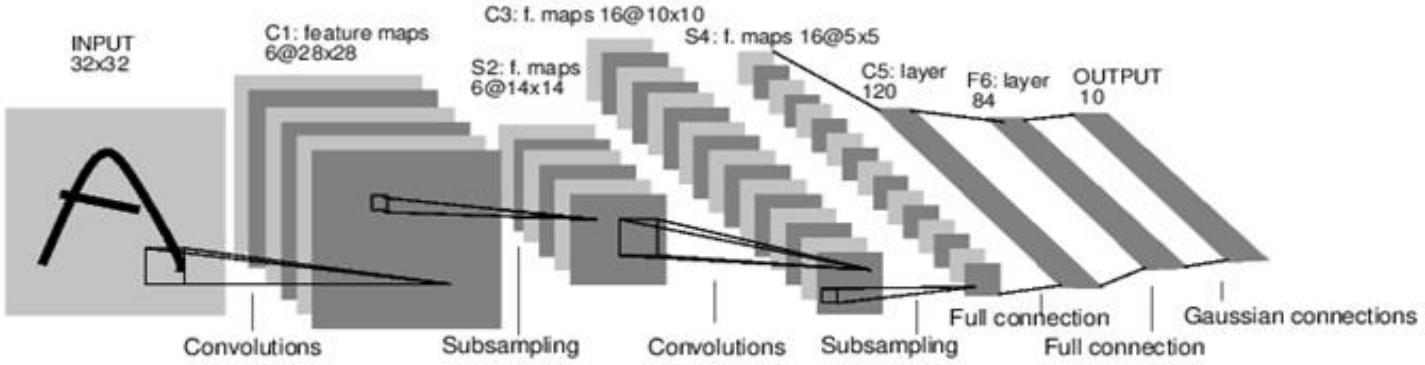


CNN example: LeNet-5

- The LeNet-5 architecture is perhaps the most widely known CNN architecture. It was created by Yann LeCun in 1998 and widely used for handwritten digit recognition (MNIST). It is composed of the layers shown as below:

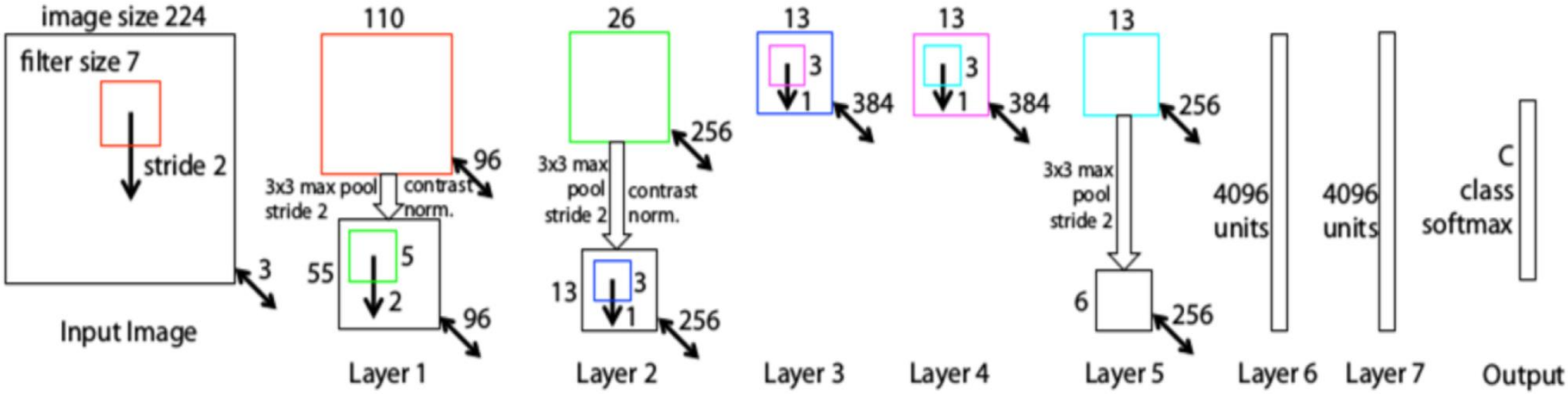
Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

- **I have edited a PDF file, CNN Architectures.pdf, given in …/Lectures/Session 7, where four well-known CNN architectures are presented**



A Full Convolutional Neural Network (LeNet)

LeNet



ZF Net Architecture

ZF Net

9). Training

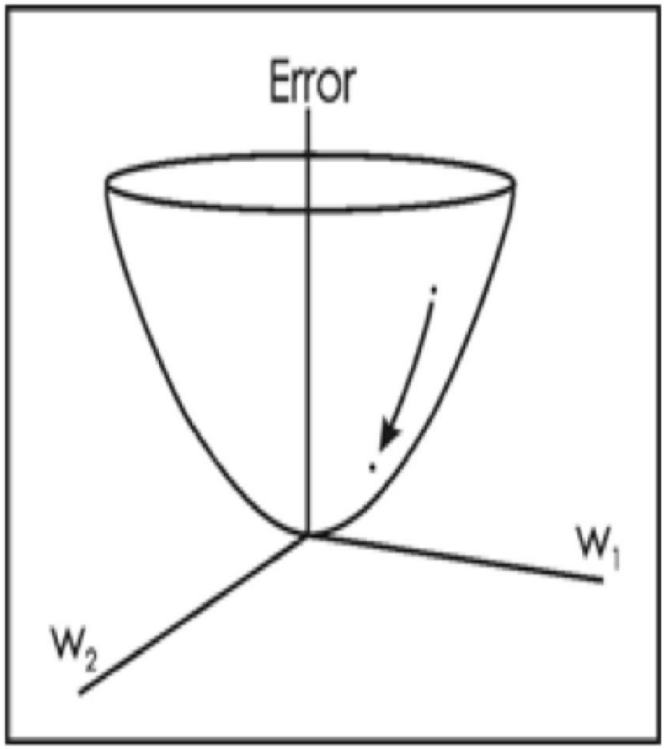
- Training is separated into 4 distinct sections, the forward pass, the loss function, the backward pass, and the weight update.
- For sample, we train the CNN to classify 10 hand-written digits.

- **Section 1: forward pass**
- Input a training-data, as an array of numbers, say 32 x 32 x 3, and pass it through the whole network. since all of the weights or filters values were randomly initialized, the output will probably be something like [.1 .1 .1 .1 .1 .1 .1 .1 .1 .1], basically an output that doesn't give preference to any number in particular

Training

- **Section 2: loss function**
- Remember that the training-data has both an image and a label. For example that the first training image inputted was a 3. The label for the image would be [0 0 0 1 0 0 0 0 0 0].
- A loss function can be defined in many different ways but a common one is MSE (mean squared error), which is 1⁄2 times (actual - predicted) squared.

Review – "Gradient Decent"



- It's a approach to minimize the amount of loss and to find out which inputs (weights in our case) most directly contributed to the loss (or error) of the network.
- The diagram shows 3-D weights. It adjusts the weights so that the loss decreases and eventually, to get the lowest point in our bowl shape.
- To do this we have to take a derivative of the loss with respect to the weights

$$w = w_i - \eta \frac{dL}{dW}$$

w = Weight
w_i = Initial Weight
η = Learning Rate

The learning rate is a parameter that is chosen by the programmer. A high learning rate means that bigger steps are taken in the weight updates and thus, it may take less time for the model to converge on an optimal set of weights.

L represents loss function or cost function

Section 3. backward pass

- It is determining which weights contributed most to the loss and finding ways to adjust them so that the loss decreases, by using “Gradient Decent”.
- Once we compute this derivative, we then go to the last step which is the weight update, as shown in the last slide. This is where we take all the weights of the filters and update them.
- **Section 4. Iteration**
- The process of forward pass, loss function, backward pass, and parameter update is one training iteration.
- The program will repeat this process for a fixed number of iterations for each set of training images (commonly called a batch). Hopefully, the network should be trained well enough, so that the weights of the layers are tuned correctly.

Remarks:

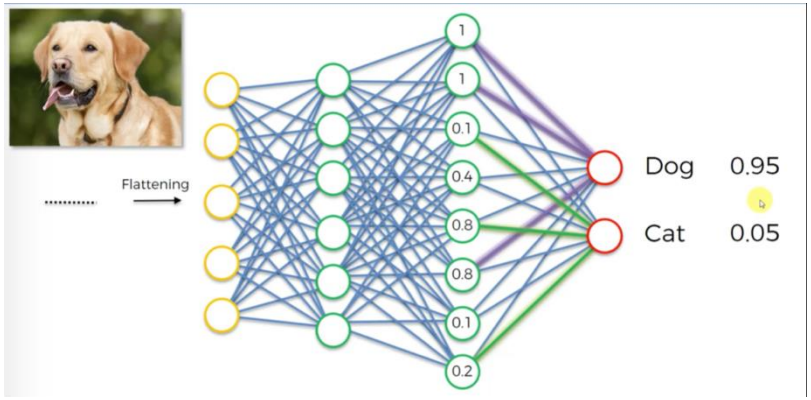
- An important thing is not only the weights are trained in artificial neural network part but also the filters are trained and adjusted by using gradient decent process , which allows us to come up with the best feature maps.

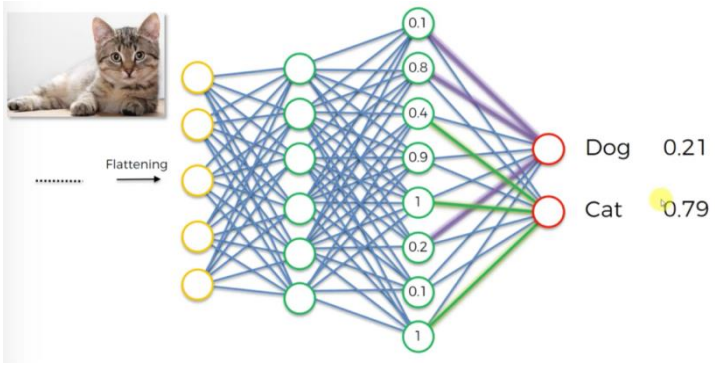
- At the end, we get a fully trained convolutional neural network which can recognize images and classify them.

- **10). Testing**
- To see whether or not our CNN works, we have a different set of images and labels and pass the images through the CNN. We compare the outputs to the ground truth and see if our network works!

3. Building CNNs with Keras

- By using a sample, we discuss this topic.
- In this sample, we have 8000 images of cats (and dogs, as well), then, we will train a convolutional neural network to predict if an image is a photo of a dog or cat.
- So, It is a supervised deep learning.





- It's implementation is presented in cnn.py, and its notebook in …/Lab7/CNN with Keras/cnn.ipynb, where I have put more comments for your study in the Lab class.

3. Building CNNs with Keras

- Let's have a look at them.
- We have dealt with most of modules or classes in ann.py, when we built an artificial neural network in the Last lab.
- But, "from keras.preprocessing.image import ImageDataGenerator" is a new topic

ImageDataGenerator

- **Why needed?**
- To achieve a higher accuracy, you need a larger dataset of images to train, however, unfortunately, you only have few images in your training data-set. How to get more images?
- One solution is to increase images artificially by using cv2 functions flips, shifts, rotations in Python. But, you have o write more codes.
- A generator function is a python function, but it behaves like an iterator.
- When the generator is called, it will return some value and save the state. Next time when we call the generator again, it will resume from the saved state, and return the next set of values just like an iterator.

ImageDataGenerator

- Thus, we can iterate over each (or batches of) image(s) in the large dataset and train our neural net quite easily.
- But, if you repeat in use of small number of images, you will have a new problem: the variations of images are low.
- For tackling this problem, we need to implement image augmentation techniques.
- Image Augmentations are methods of artificially increasing the variations of images in our dataset by using horizontal/vertical flips , rotations , variations in brightness of images, horizontal/vertical shifts etc.
- Solution 1: using cv2 library in python
- Solution 2: from keras.preprocessing.image import ImageDataGenerator

Advantages of using ImageDataGenerator

- Easy to write codes: simply call it and set values for parameters like horizontal_Fip, vertical_Fip, rescale, brightness_range, zoom_range, rotation_range etc.
- Easily combine ImageDataGenerator with a custom image generator.
- Fast — If you want to use multiple threads to load training data, It has a workers argument, which can be tuned

3. Building CNNs with TensorFlow

- A simple example (in lab time): The code loads two sample images, one of a Chinese temple, and the other of a flower.
- Then it creates two 7 x 7 filters (one with a vertical white line in the middle, and the other with a horizontal white line), and applies them to both images using a convolutional layer built, TensorFlow's conv2d() function (with zero padding and a stride of 2). By this function, a convolution layer with two feature maps, based on given filters or weights.
- Finally, it plots one of the resulting feature maps:

Two Issues:

- Convolutional layers have many hyperparameters: you must choose the number of filters, their height and width, the strides, and the padding type. So, to find the right hyperparameter values (training process) will be very time-consuming.

- As convolutional layers require a huge amount of RAM, especially during training, because the reverse pass of backpropagation requires all the intermediate values computed during the forward pass.

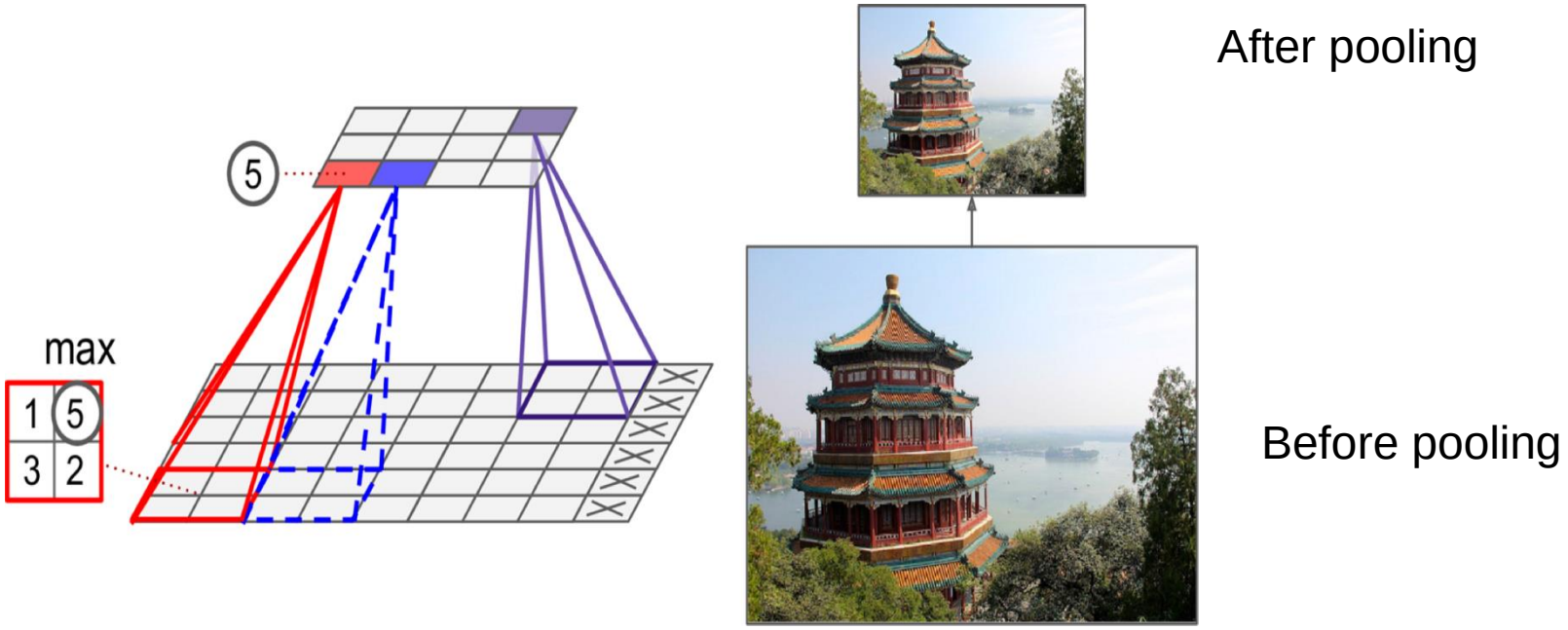
3. Pooling Layer

- Pooling Layer is to subsample (i.e., shrink) the input image, in order to reduce the computational load, the memory usage, and the number of parameters (thereby limiting the risk of overfitting).

- Reducing the input image size also makes the neural network tolerate a little bit of image shift (location invariance).

- Just like in convolutional layers, each neuron in a pooling layer is connected to the outputs of a limited number of neurons in the previous layer, located within a small rectangular receptive field.

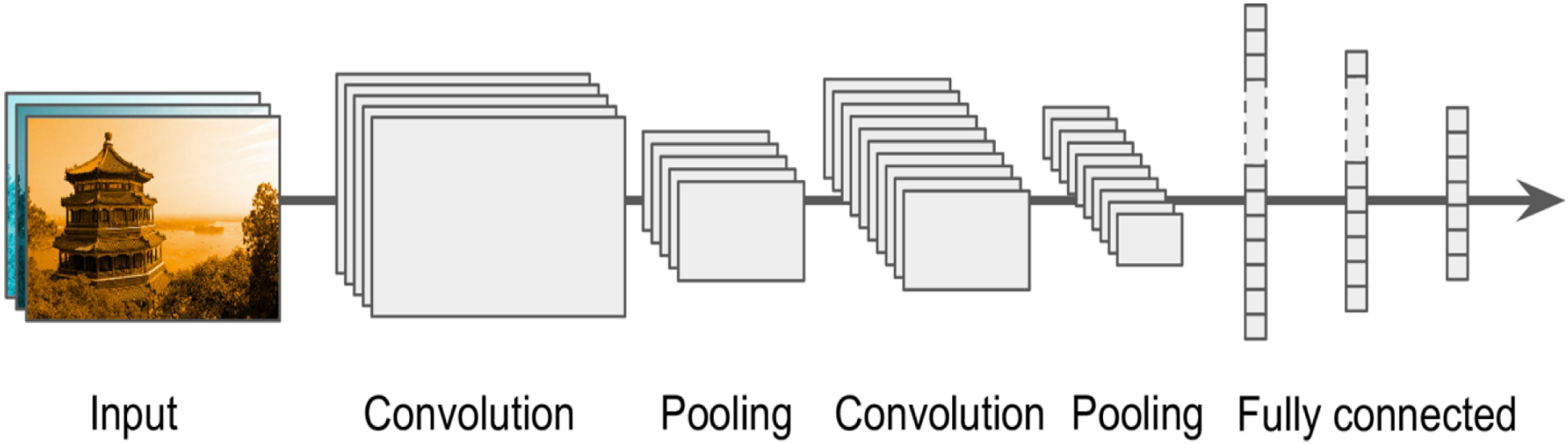
Pooling Layer



- You must define its size, the stride, and the padding type, just like before
- However, a pooling neuron has no weights; all it does is aggregate the inputs using an aggregation function such as the max or mean.

4. CNN Architecture

- Typical CNN architectures stack a few convolutional layers (each one generally followed by a ReLU layer, then a pooling layer, then another few convolutional layers (+ReLU), then another pooling layer, and so on.



- The image gets smaller and smaller as it progresses through the network, but it also typically gets deeper and deeper (i.e., with more feature maps).

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

结束

2020年9月30日