SUPPORT VECTOR MACHINES & NEURAL NETWORKS

LECTURE 9 – ARTIFICIAL NEURAL NETWORKS

A. Basic structure of neural networks
   - Neuron, activation function, perceptron, feed-forward NN
B. Backward propagation and learning
   - Loss/reward function, online vs. batch, leaning algorithm
C. Multi-layer neural networks and deep leaning
   - Scale, feature and computation, ReLU and SGD
D. Radial basis function neural network (RBFN)
E. Convolutional neural network (CNN)

Convolutional neural networks (CNN)

- Convolutional neural network (CNN, or ConvNet) is a specially constructed artificial neural network that mimics the organization of animal visual cortex in its connectivity pattern between neurons to analyze visual imagery.

- CNN uses a mathematical operation called convolution in at least one of its layers to explore the hierarchical feature/pattern in data and assemble patterns of increasing complexity using simpler/smaller patterns embedded in filters.

- Commonly seen applications of CNN include face/image recognition, image classification, video recognition, medical image analysis, and natural language processing.

Convolutional neural networks (CNN)

- Development: (from Wikipedia)
- Yann André LeCun (born 1960, PhD 1987) is a French computer scientist working primarily in the fields of machine learning, computer vision, mobile robotics, and computational neuroscience. He worked at Bell Labs (1988-1996) and he is currently the Silver Professor of the Courant Institute of Mathematical Sciences at New York University, and Vice President, Chief AI Scientist at Meta.

- LeCun is well known for his work on optical character recognition and computer vision using convolutional neural networks (CNN), and is a founding father of convolutional nets.

- LeCun received the 2018 Turing Award together with Yoshua Bengio and Geoffrey Hinton, for their work on deep learning. The three are sometimes referred to as the "Godfathers of AI" and "Godfathers of Deep Learning"

Convolutional neural networks (CNN)

References:

LeCun, Yann; Bengio, Yoshua (1995). "Convolutional networks for images, speech, and time series". In Arbib, Michael A. (ed.). The handbook of brain theory and neural networks (Second ed.). The MIT press. pp. 276–278.

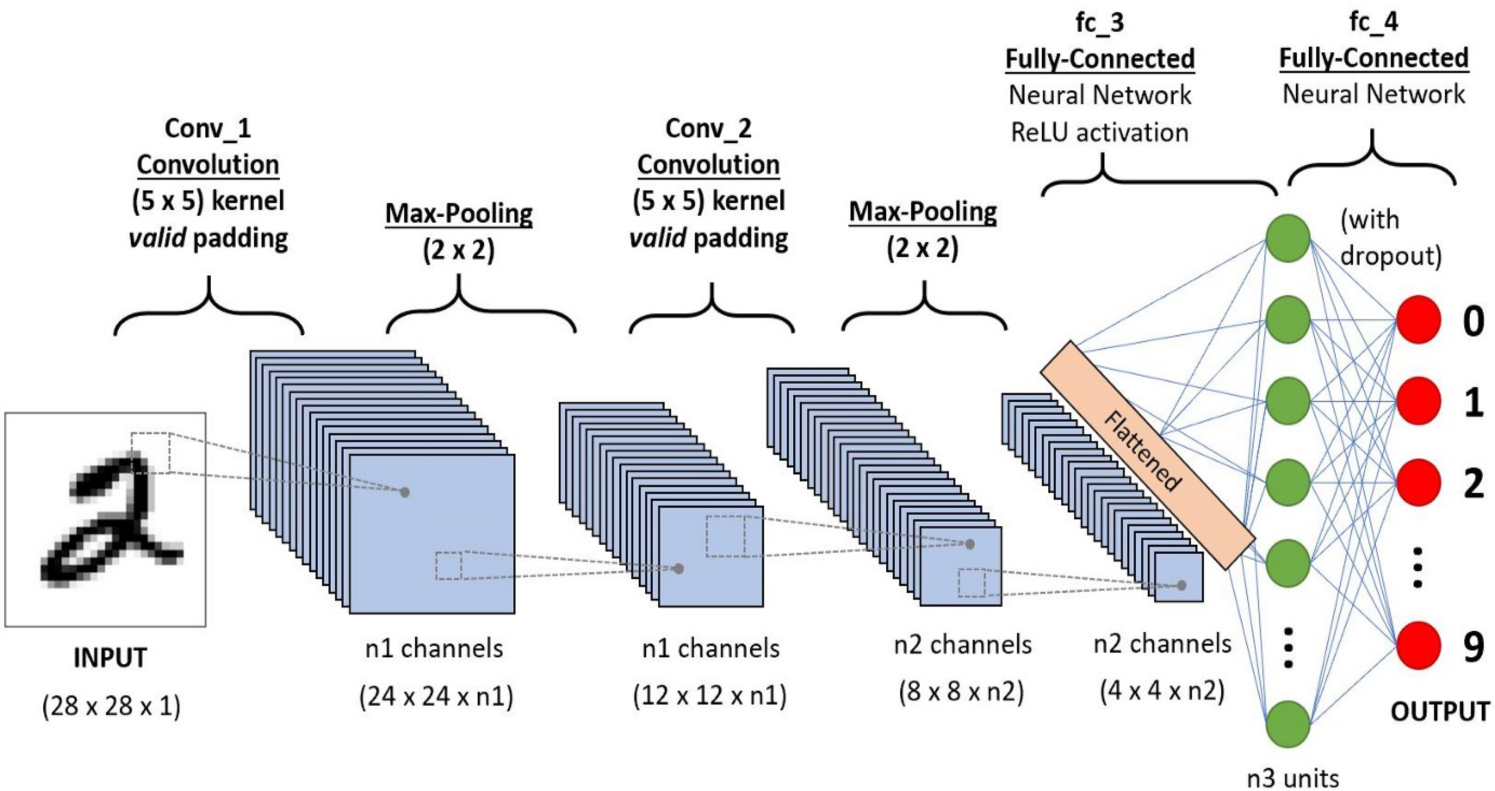
LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep learning". Nature. 521 (7553): 436–444. Bibcode:2015Natur.521..436L. doi:10.1038/nature14539. PMID 26017442. S2CID 3074096.

- LeNet-5, a pioneering 7-level convolutional network by LeCun et al. in 1998, that classifies digits, was applied by several banks to recognize hand-written numbers on checks digitized in 32x32 pixel images.
- Using Graphic Processing Units (GPU) - the breakthrough of CNN in the 2000s required fast implementations on graphics processing units.

  K. S. Oh and K. Jung (2004) showed that standard neural networks can be greatly accelerated on GPUs. Their implementation was 20 times faster than an equivalent implementation on CPU.

Image of a convolutional neural network

- Classify handwritten digits <image from Toward Data Science>



Examples of image data

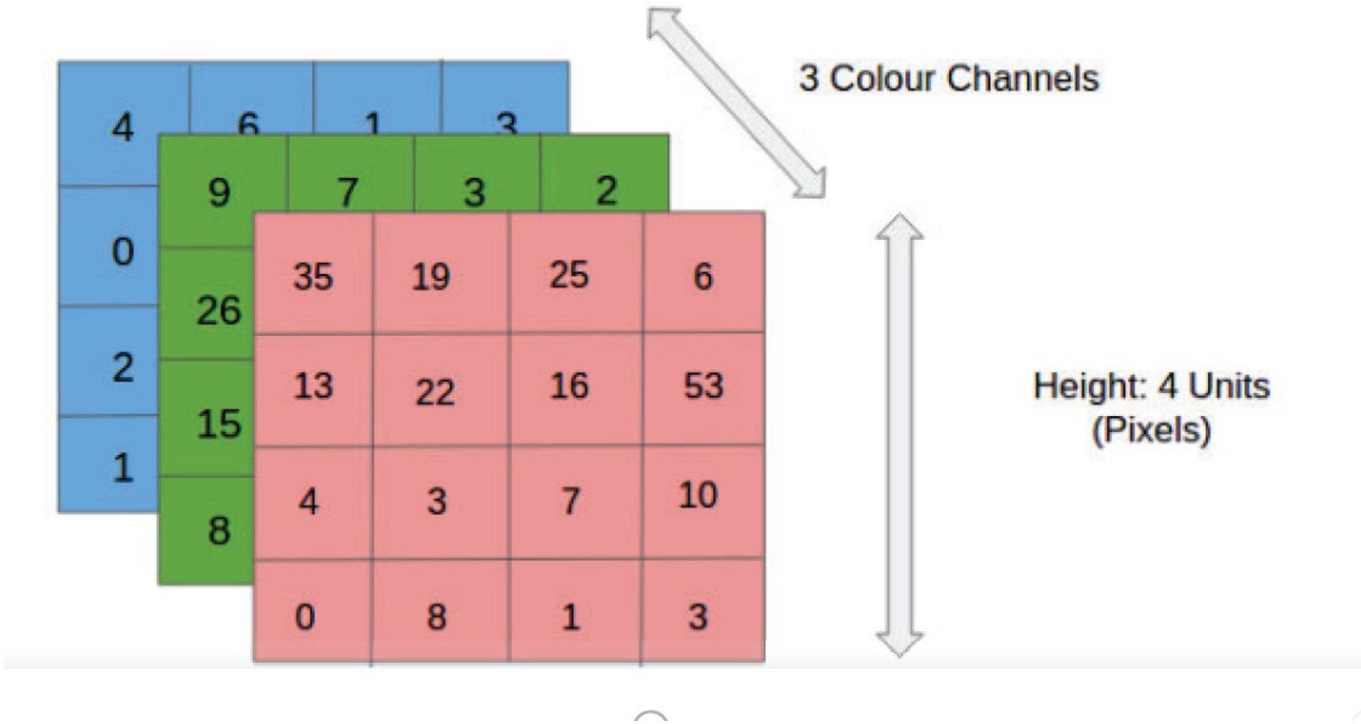
- Size in megapixels

This is an excerpt from *Post Capture Pocket Guide*.

Sensor Resolution (megapixels)	Typical Image Resolution (pixels)	Maximum Print Size	Print Resolution	Maximum Output Size
2.16	1800 x 1200	6 x 4 inch	300 dpi	Snapshot prints
3.9	2272 x 1704	7.6 x 5.7 inch	300 dpi	'Jumbo' snapshot prints
5.0	2592 x 1944	8.6 x 6.5 inch	300 dpi	8 x 6 inch enlargements
7.1	3072 x 2304	10.2 x 7.7 inch	300 dpi	A4 sized prints
8.0	3264 x 2448	13.6 x 10.2 inch	240 dpi	A4 sized prints
10.0	3648 x 2736	18.2 x 13.7 inch	200 dpi	A3 sized prints
12.1	4000 x 3000	20 x 15 inch	200 dpi	A3+ sized prints
14.7	4416 x 3312	22.1 x 16.6 inch	200 dpi	A2 sized prints
21.0	5616 x 3744	31.2 x 20.8 inch	180 dpi	A1 sized prints

Basic operations involved <image from Towards Data Science>

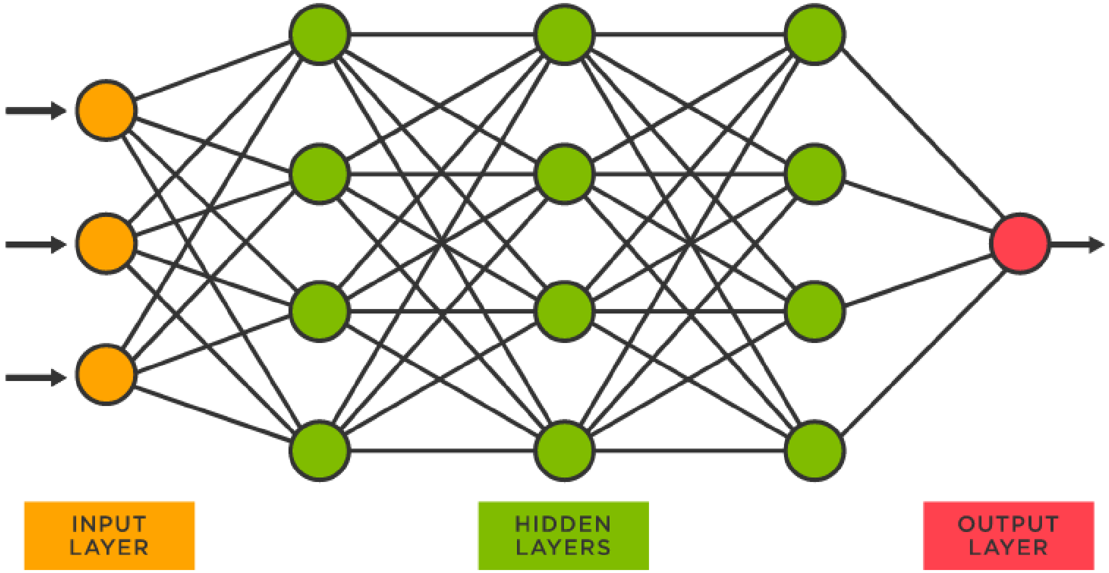
- Input image



- Tensor of 4x4x3

Regular MLP neural networks

- A fully connected network structure < image from tibco.com >



Regular MLP neural networks

- Fact: Regular MLP networks have a fully connected network structure.
- Concerns:

  - A large number of links in the network

    $\Rightarrow$ large scale network ?

  - Too many weights to be trained

    $\Rightarrow$ long computing time? Even with SGD and ReLU

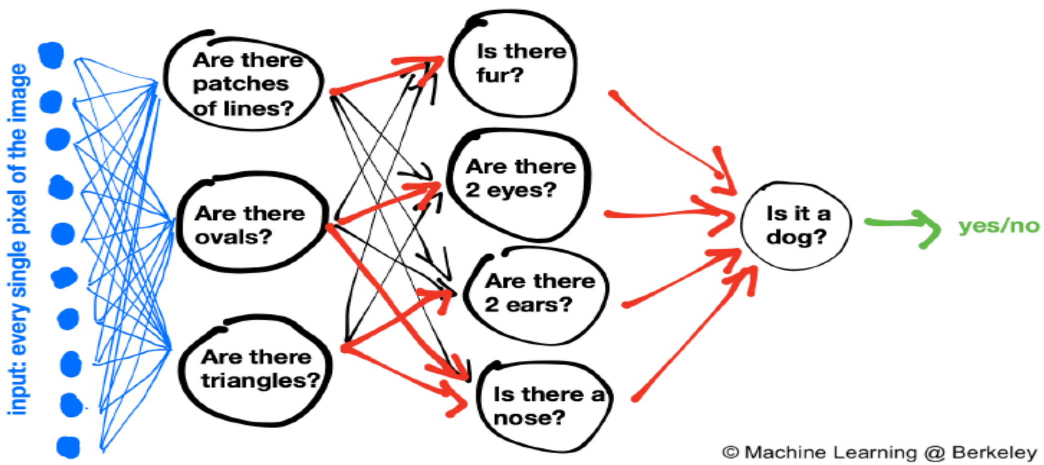
  - Each input is "impartially" treated

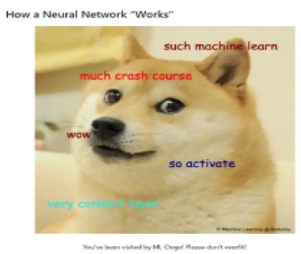
    $\Rightarrow$ losing some spacially local structure information?

Feature identification

- Basic idea: Can we relate groups of nearby inputs to identify meaningful "features/characteristics" to reduce the scale of the network before employing a fully connected MLP network?

  - Identify a dog in a photo (Machine Learning Crash Course: Part 3 - ML@B Blog)

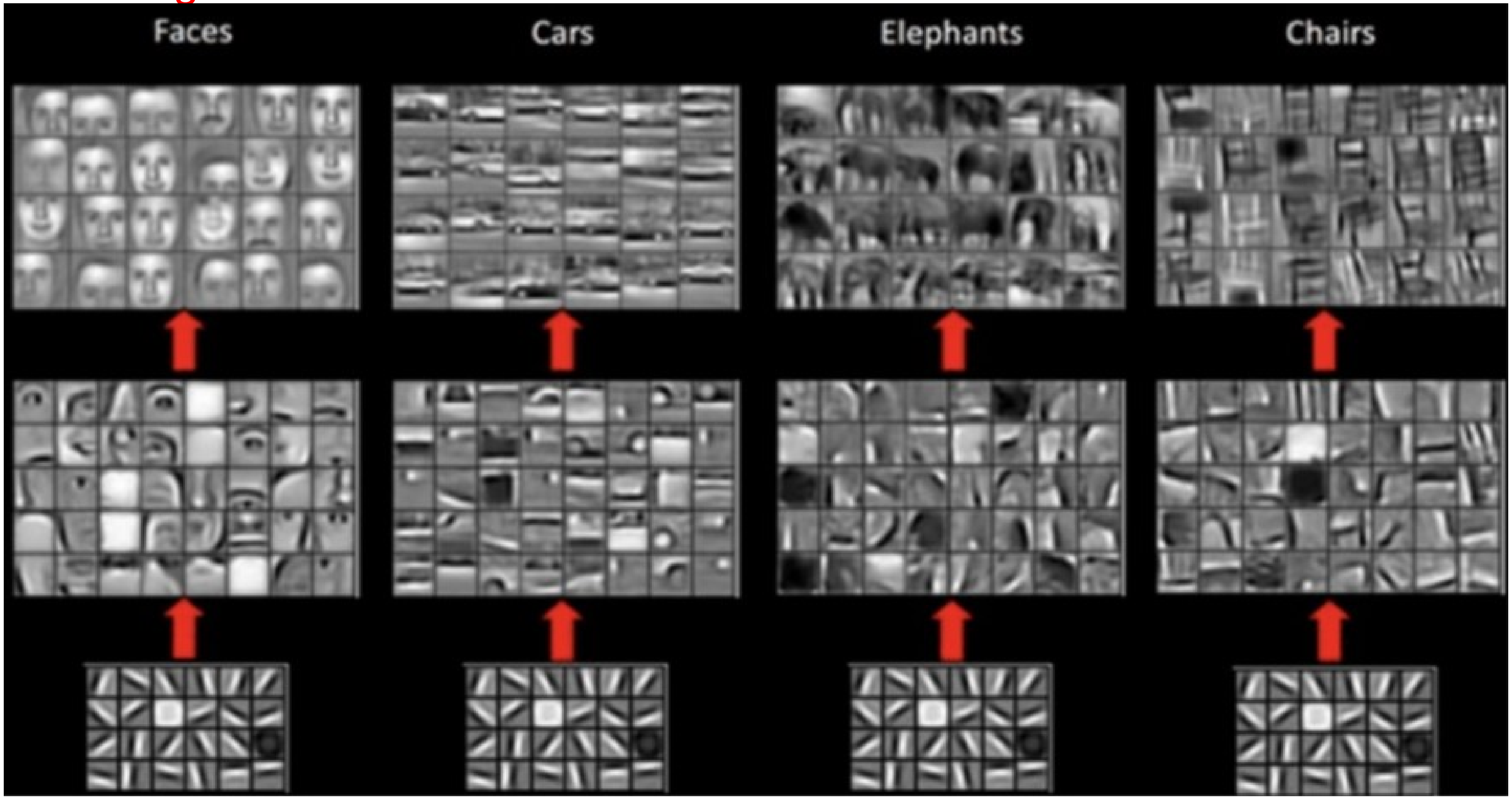




  - Pixels line segments distinct features judgement

Layer-by-layer feature identification

- <Image from TOPBOTS>



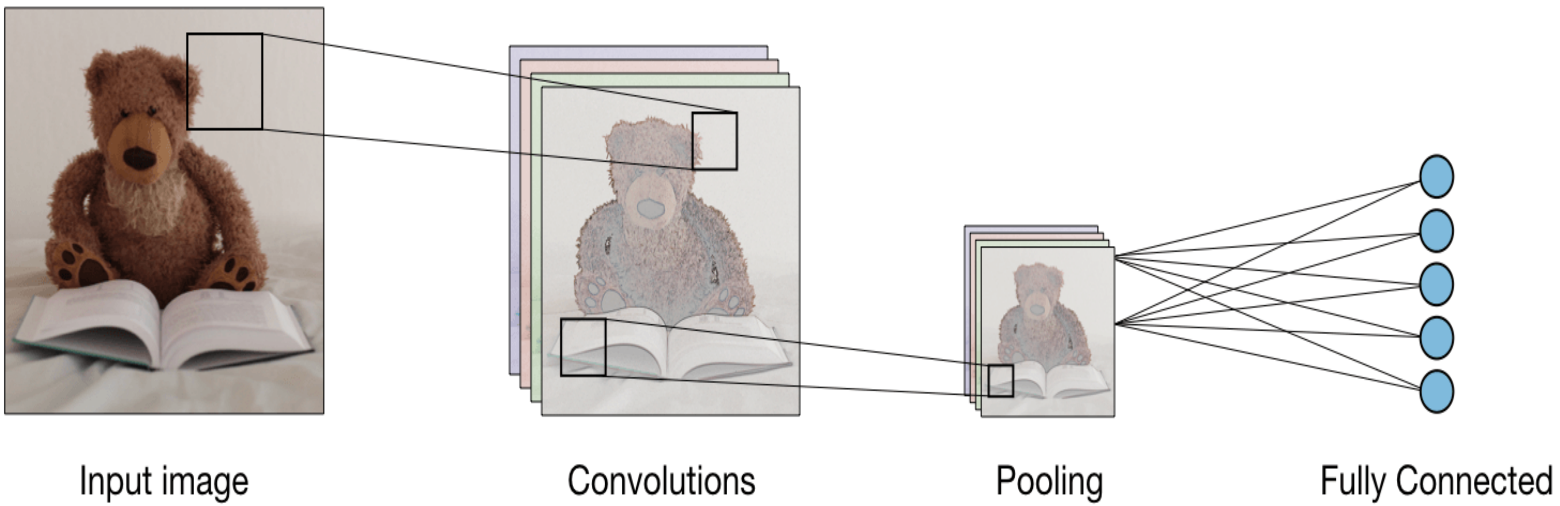
- Using special structure of inputs for identifying:

  “line segments” ⇒ “nose, ear, eye, …” ⇒ faces

  “line segments” ⇒ “headlight, wheel, …” ⇒ cars

Architecture of CNN

- CNN first creates representations of small parts of the input, then from them assembles representations of larger areas.



卷积 汇集

<image from CS230 https://Stanford.edu/~shervine>

Basic ideas: filter/kernel for feature identification

- Identify "\" and "/" from a 2x2x1 black-white picture

picture	"\" filter	dot pro.	ReLU	"/" filter	dot pro.	ReLU
1 1 1 1	1 -1 -1 1	0	0	-1 1 1 -1	0	0
1 -1 1 -1		0	0		0	0
1 -1 -1 1		4 (> t=3)	**1**		-4	0
-1 -1 1 -1		-2	0		2	0
-1 1 1 -1		-4	0		4 (> t=3)	**1**

* 4 input neurons with weights = 1, -1, -1, 1 and threshold = 3 for backslash "\"

4 input neurons with weights = -1, 1, 1, -1 and threshold = 3 for slash "/"

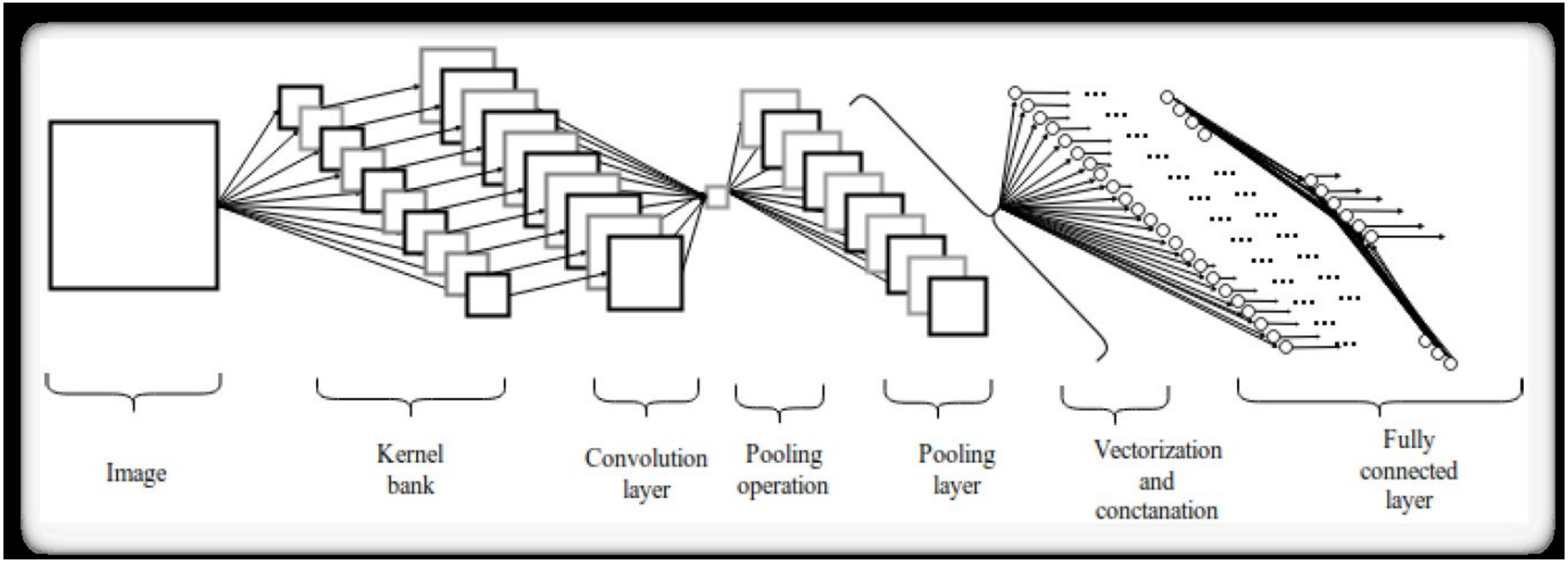
Basic ideas – convolution for spacially local structure

- Identify “X” and “O” from a 3x3x1 black-white picture

picture	“\” filter	convolution	feature-1	“/” filter	convolution	feature-2	feature	X/O
1 -1 1	1 -1	4 -4	\ 0	-1 1	-4 4	0 /	\/	**X**
-1 1 -1	-1 1	-4 4	0 \	1 -1	4 -4	/ 0	/\	
1 -1 1								
-1 1 -1		-4 4	0 \		4 -4	/ 0	/\	**O**
1 -1 1		4 -4	\ 0		-4 4	0 /	\/	
-1 1 -1								
1 1 1		0 0	0 0		0 0	0 0	No Found	
1 1 1		0 0	0 0		0 0	0 0		
1 1 1								
1 -1 -1		4 -2	\ 0		-4 2	0 0	\	\ ?
-1 1 -1		-2 4	0 \		2 -4	0 0	\	
-1 -1 1								

Basic architecture of CNN

- A basic CNN architecture (from CS395T(51800) bajaj@cs.utexa.edu)

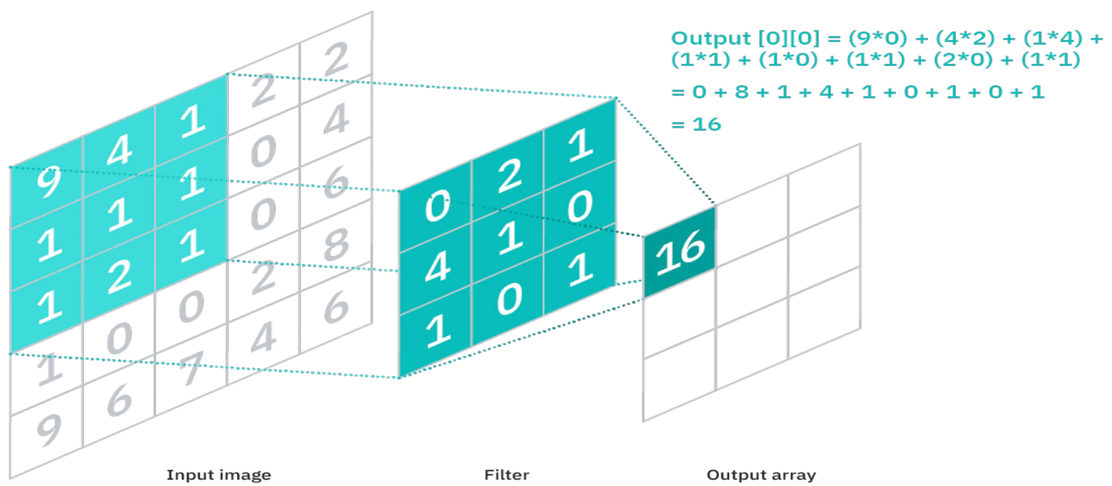


- input image $\rightarrow$ $convoltion$ $\rightarrow$ $pooling$ $\rightarrow$ fully connected NN

Convolutional layers

- CNN has at least a hidden layer to perform convolutions. Typically, it performs a tensor/matrix dot product of the convolution filter/kernel with the layer's input tensor/matrix. This product usually takes the Frobenius inner product, and uses the ReLU activation function. As the convolution filter/kernel slides along the input tensor for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

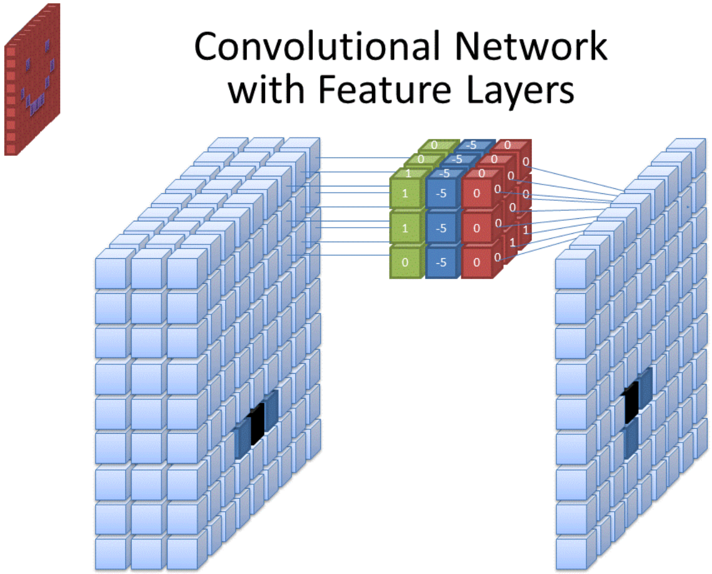
<image from IBM>



Convolutional layers

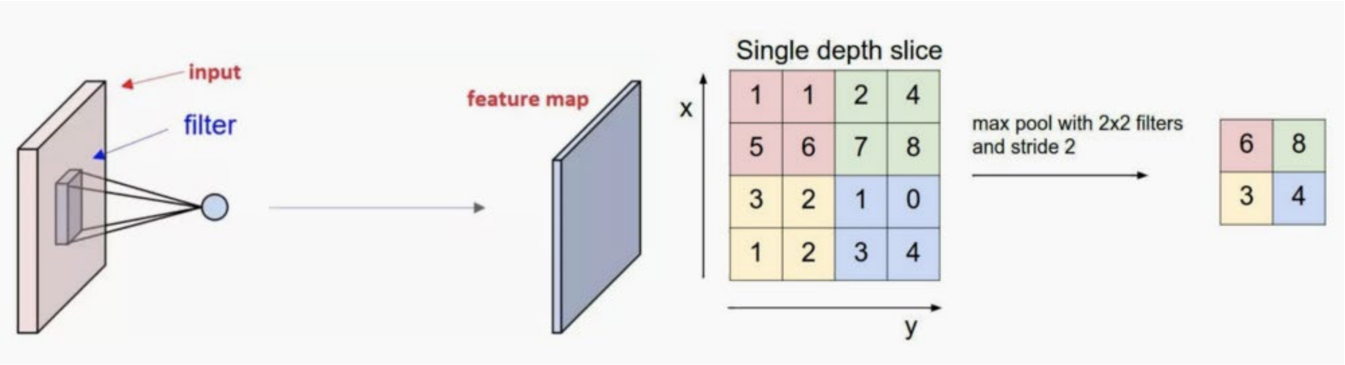
- The input of CNN is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input depth). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) x (feature map height) x (feature map width) x (feature map depth).

- <image from Wikimedia Commons>



Convolution and pooling

- Learned "filters/weights" produce the strongest response to a spatially local input pattern - "feature".
- Feature maps are divided into rectangular sub-regions, and the features in each rectangle are independently down-sampled to a single value, commonly by taking their maximum or average value – "pooling"



<image from TOPBOTS>

Pooling layers

- Convolutional networks may include local and/or global pooling layers along with traditional convolutional layers.
- Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer.
- Local pooling combines small clusters, tiling sizes such as 2 x 2 x 1 are commonly used. Global pooling acts on all the neurons of the feature map.
- There are two common types of pooling in popular use: max and average. *Max pooling* uses the maximum value of each local cluster of neurons in the feature map, while *average pooling* takes the average value.
- It is common to periodically insert a pooling layer between successive convolutional layers (each one typically followed by an activation function, such as a ReLU layer) in a CNN architecture.

Shared use of weights

- Each neuron in a neural network computes an output value by applying a specific function to the input values received from the receptive field in the previous layer. The function that is applied to the input values is determined by a group of weights and a bias. Learning consists of iteratively adjusting these biases and weights.

- The vectors of weights and biases are called *filters* and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces the memory footprint because a single bias and a single vector of weights are used across all receptive fields that share that filter, as opposed to each receptive field having its own bias and vector weighting.

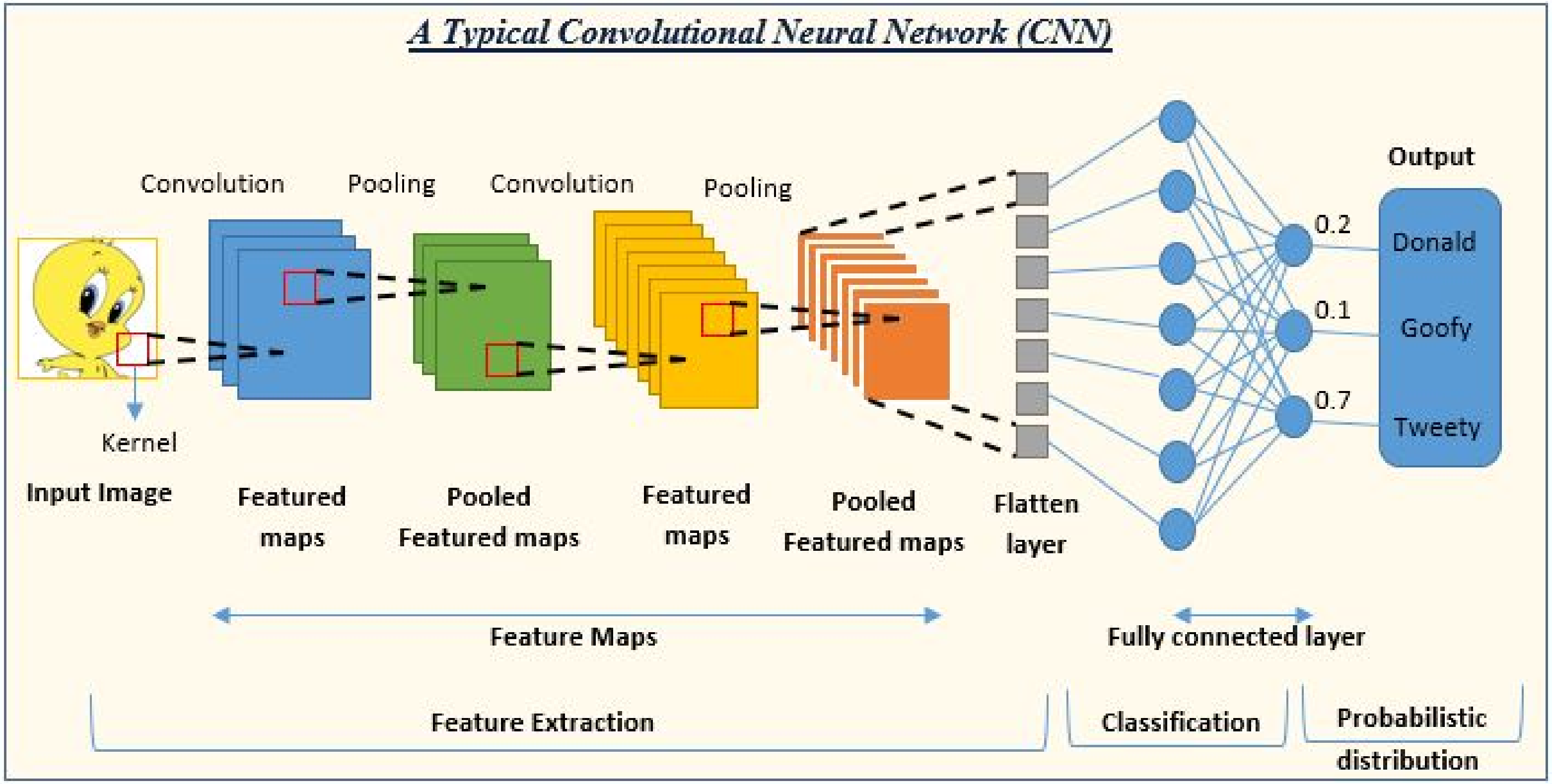
Softmax activation function

- The softmax function takes an input of a vector of scores $\boldsymbol{x} \in \mathbb{R}^n$ and outputs a vector of output probability $\boldsymbol{p} \in \mathbb{R}^n$, usually used at the end of the CNN architecture, such that

$$p_{\mathrm{j}} = g(x_j) = e^{x_j} / \sum_{i=1}^{n} e^{x_i}, \text{ for } j = 1, \ldots, n.$$

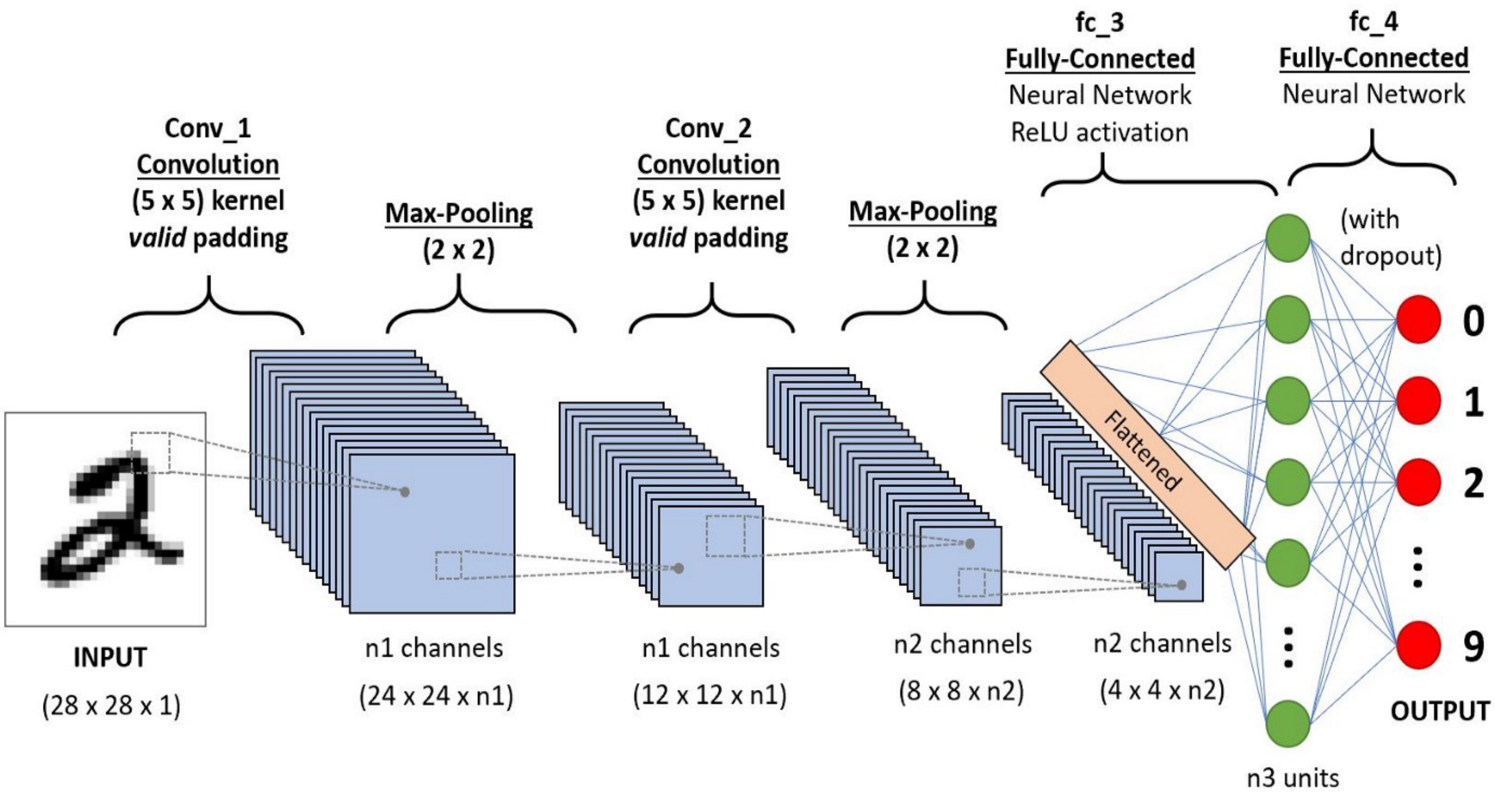
CNN classification

- <image from Analytics Vidhya>



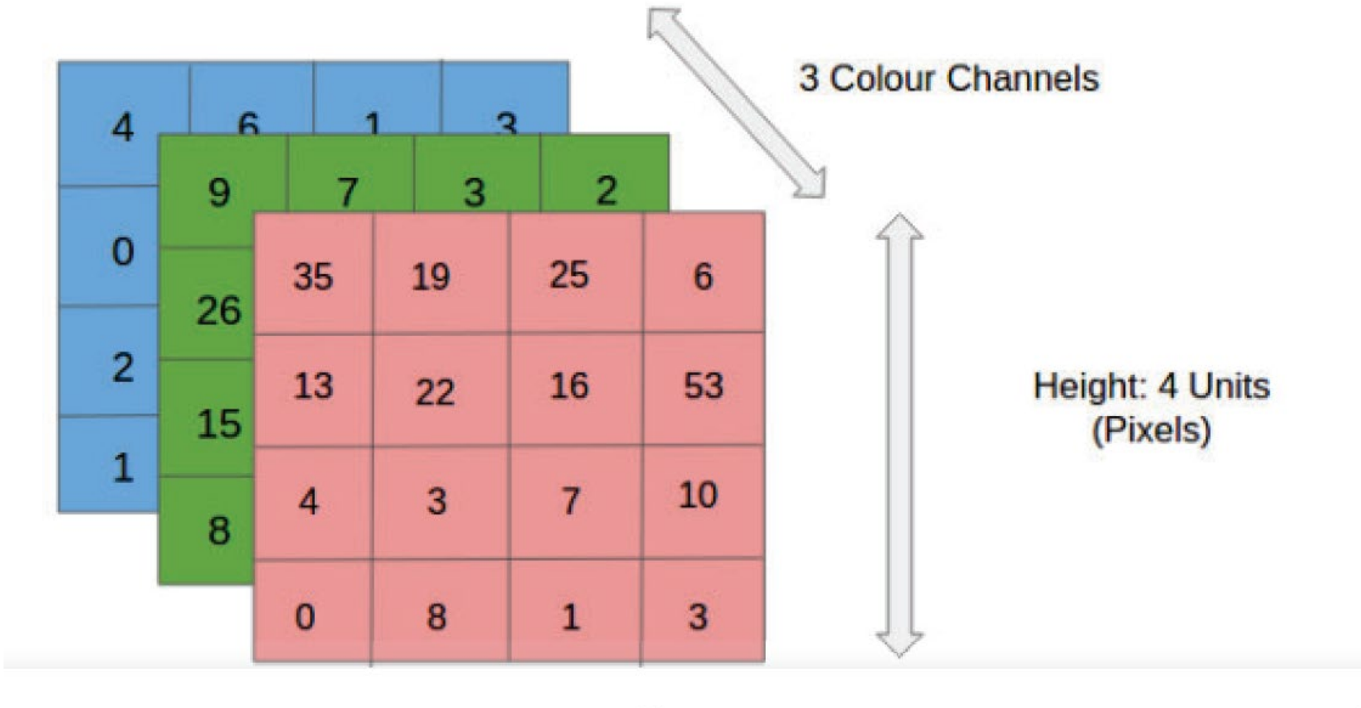
Example of a convolutional neural network

- Classify handwritten digits <image from Toward Data Science>



Basic operations involved <image from Towards Data Science>

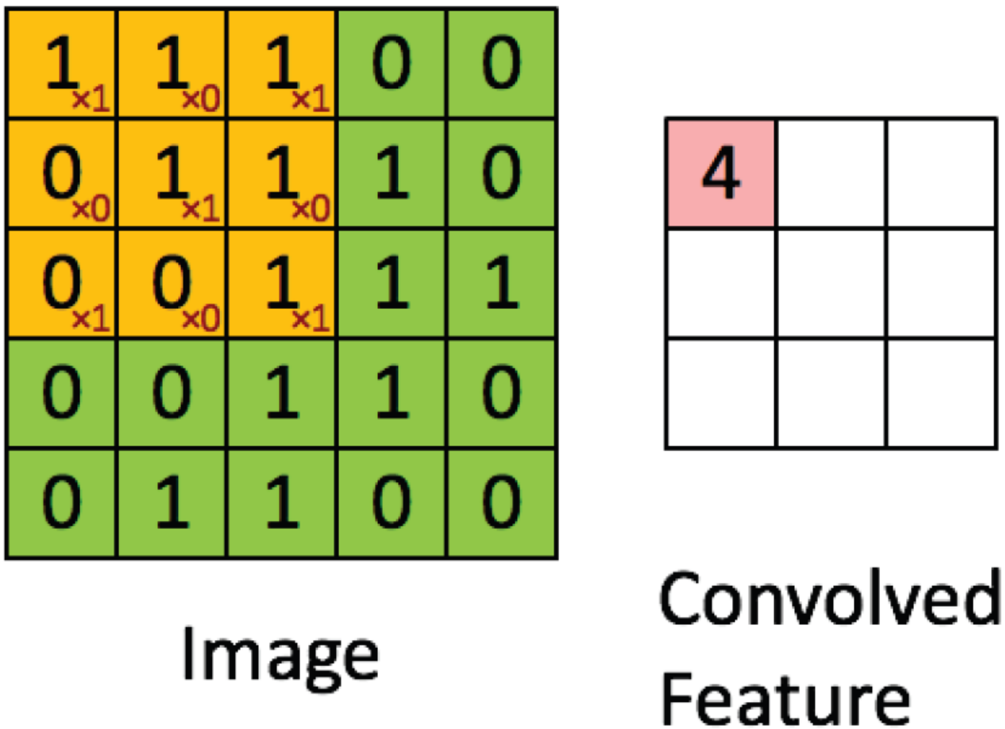
- Input image



- Tensor of 4x4x3

Basic operations involved <image from Towards Data Science>

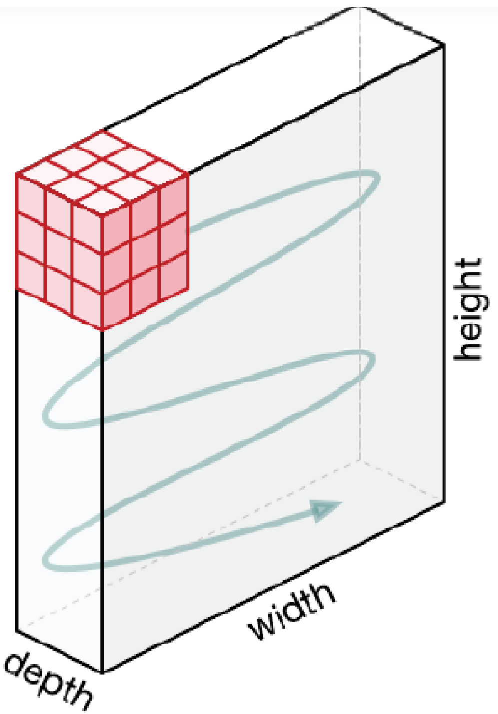
- Convolution



Convoluting a 5×5×1 image with a 3×3×1 kernel to get a 3×3×1 convolved feature

Basic operations involved <image from Towards Data Science>

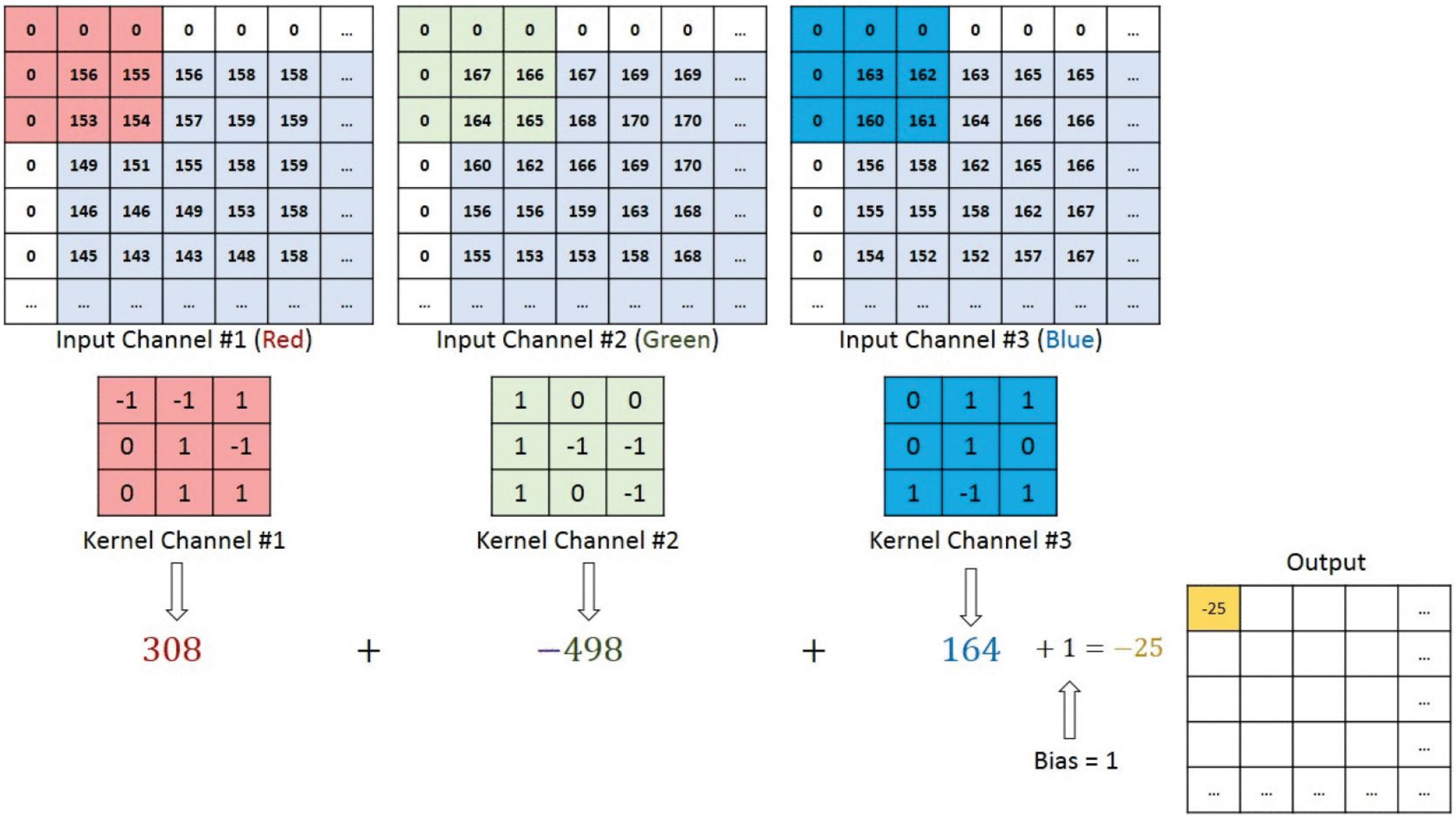
- Movement of a filter



Movement of the Kernel

Basic operations involved <image from Towards Data Science>

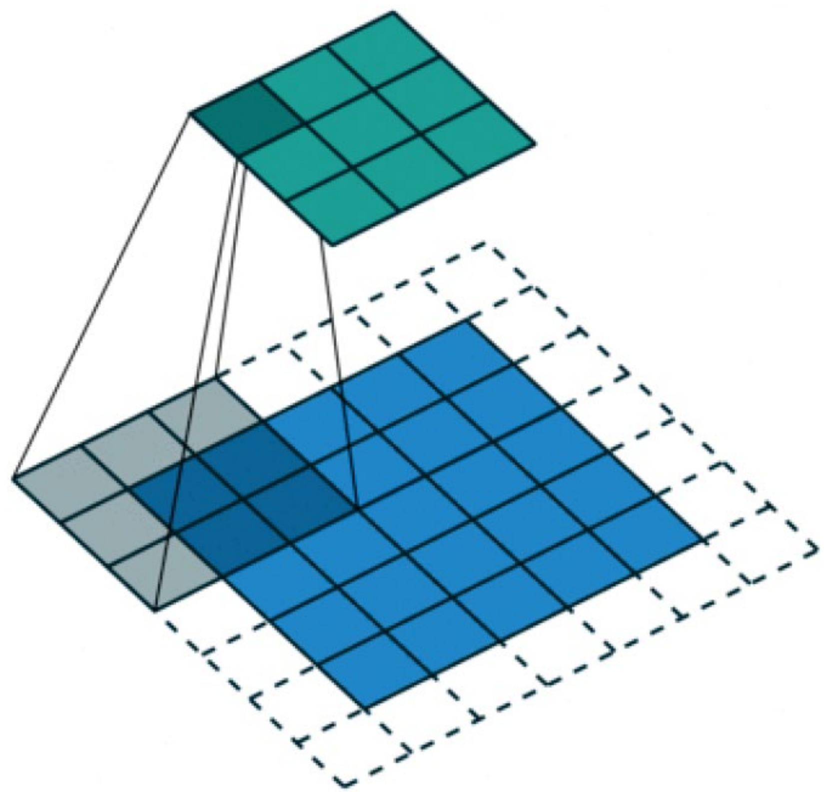
- Feature maps



Convolution operation on a MxNx3 image matrix with a 3×3×3 Kernel

Basic operations involved <image from Towards Data Science>

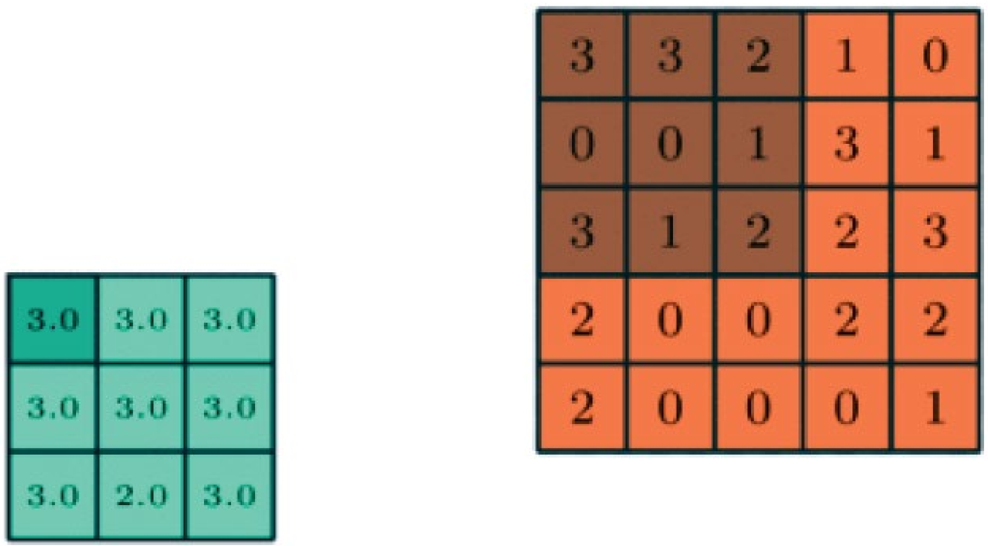
- Stride and padding



Convolution Operation with Stride Length = 2

Basic operations involved <image from Towards Data Science>

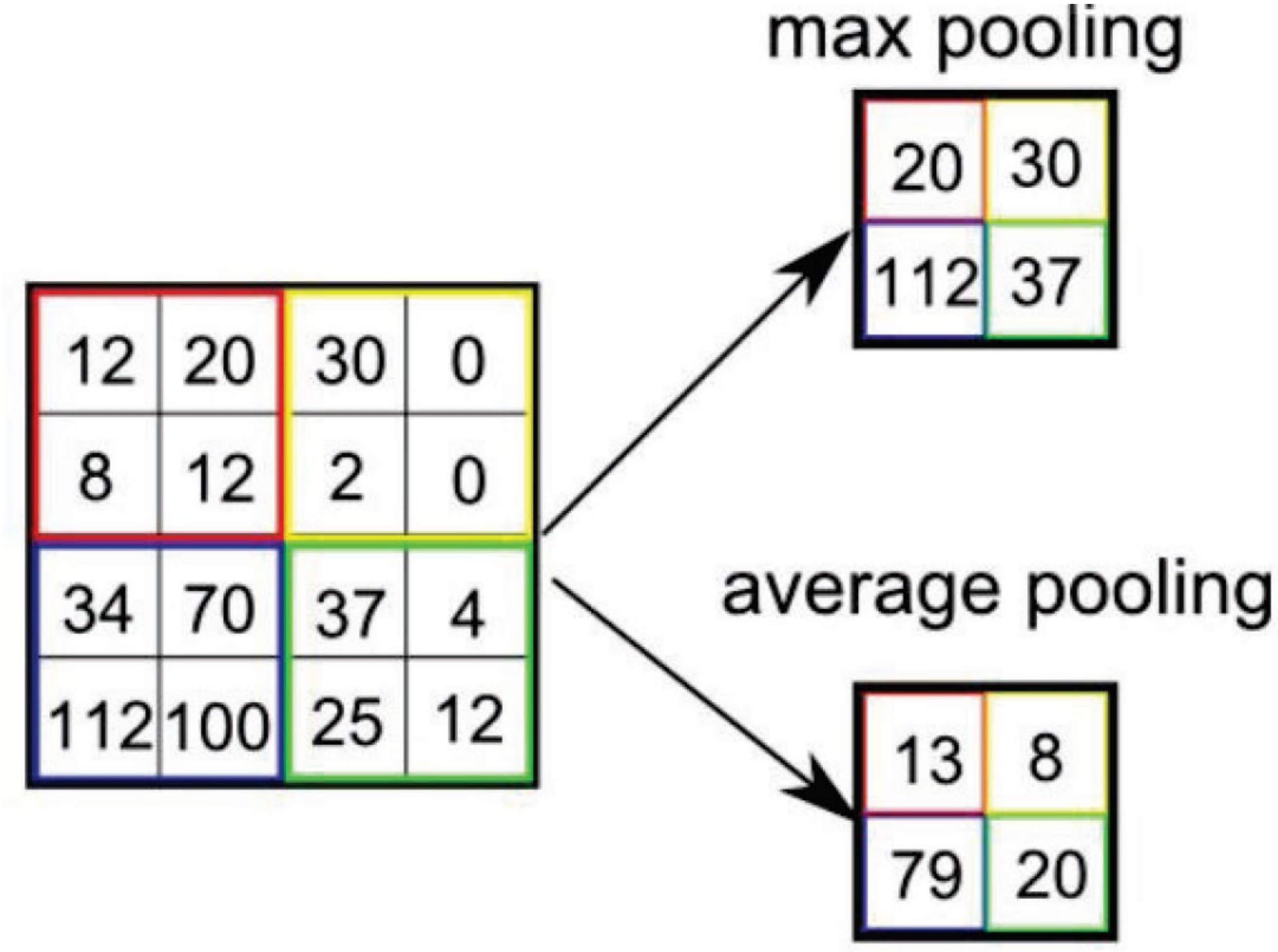
- Pooling layer



3×3 pooling over 5×5 convolved feature

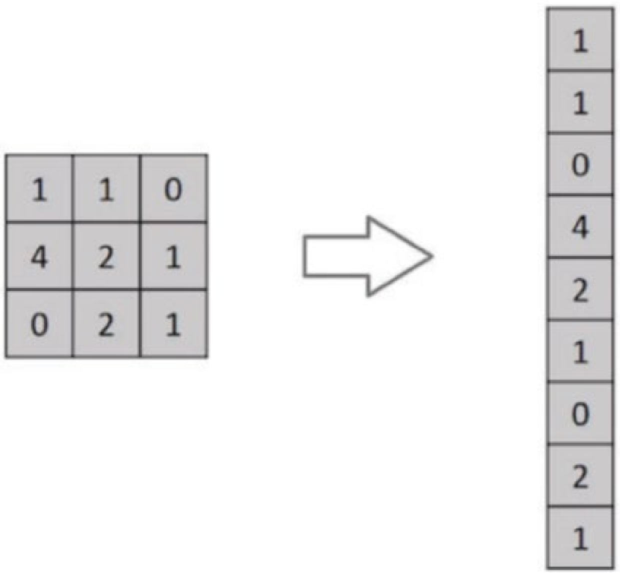
Basic operations involved <image from Towards Data Science>

- Pooling operators



Basic operations involved <image from Towards Data Science>

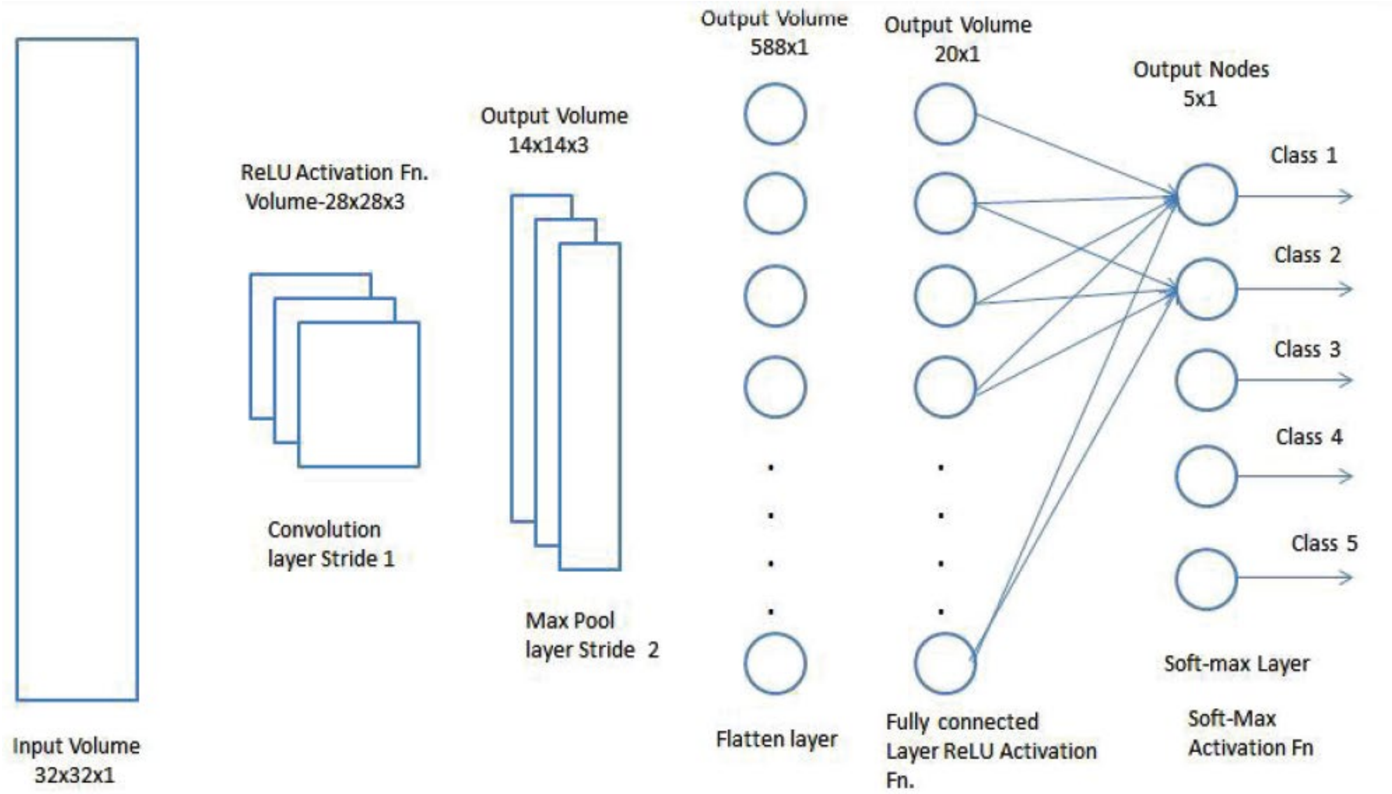
- Flattening a matrix to a vector



Flattening of a 3×3 image matrix into a 9×1 vector

Basic operations involved <image from Towards Data Science>

- CNN classifier



Interesting video on CNN

- Helpful to watch

"A friendly introduction to Convolutional Neural Networks and Image Recognition"

<https://www.youtube.com/watch?v=2-Ol7ZB0MmU&t=222s&ab_channel=Serrano.Academy>