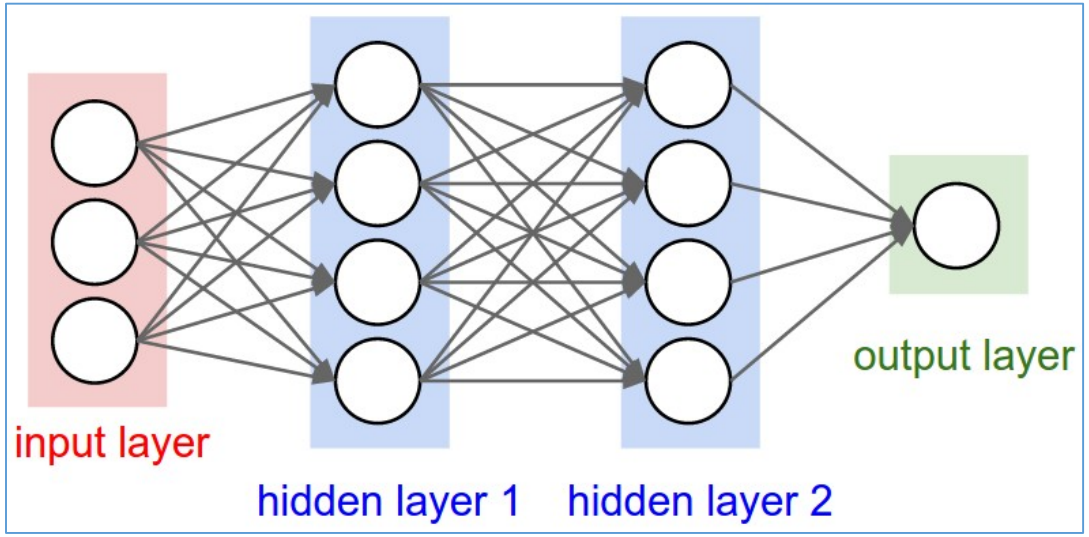
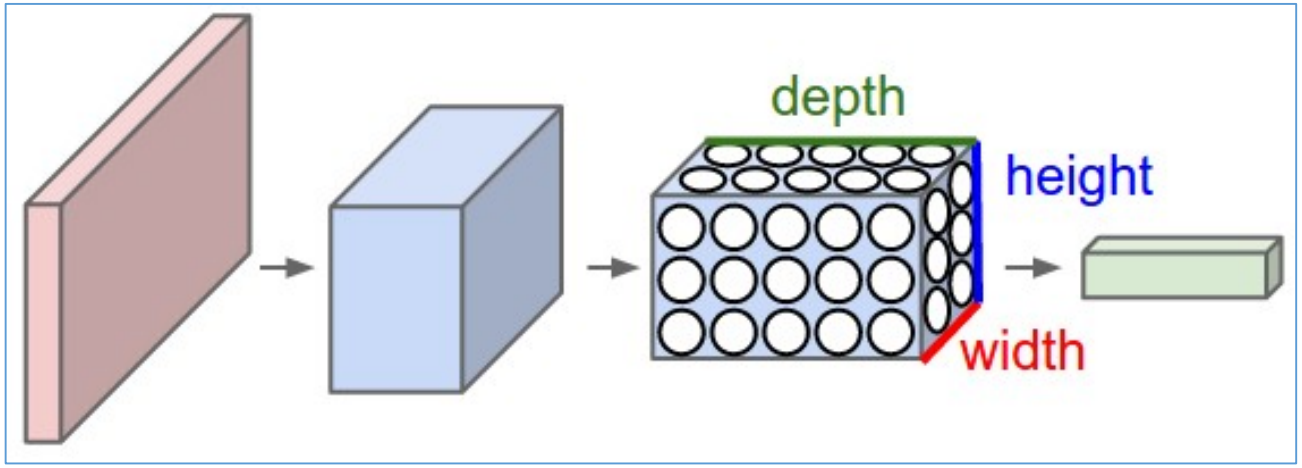
Deep Learning and Convolutional Neural Network (42028)

Convolutional Neural Network (CNN) - 1

CNN Basics

Deep-NNs Vs CNNs





ANN with 3-layers

- Fully connected
- Not great for images as input
- May lead to overfitting
- Too much of full connectivity

CNN with 3-layers

- Well suited for images with 3 dimensions
- CNNs have neurons arranged in 3D
- It is a sequence of layers which transforms input 3D volume to 3D outputs volume

Image Source and reference to read : http://cs231n.github.io/convolutional-networks/

Convolutional Neural Networks (CNN)

CNNs are the foundations of modern state-of-the-art deep learning based computer vision.

Layers in a CNN:

Three main type of layers used to build a CNN architecture

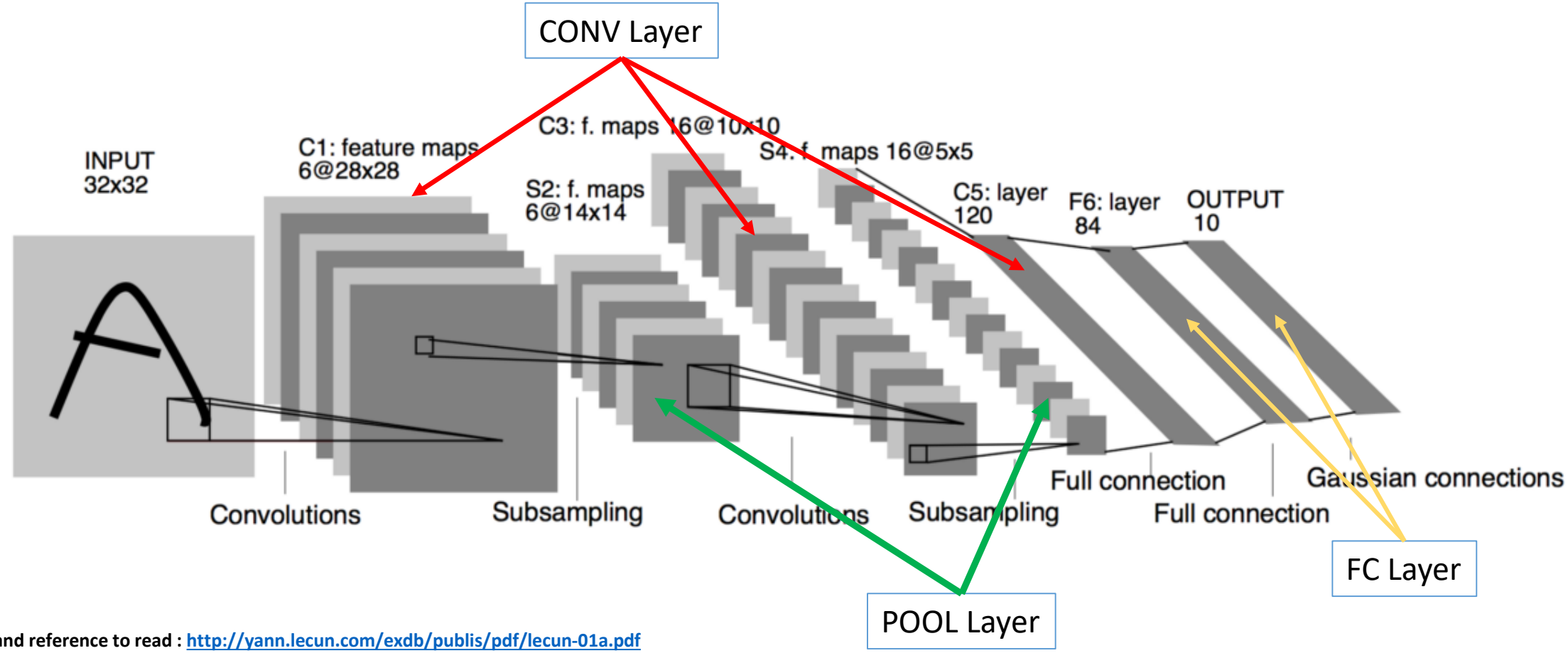
1. Convolutional Layer (**CONV**)
2. Pooling Layer (**POOL**)
3. Fully Connected layer (**FC**)

These three types of layers are stacked together to form a CNN architecture!

Source and reference to read : http://cs231n.github.io/convolutional-networks/
https://medium.com/@pechyonkin/key-deep-learning-architectures-lenet-5-6fc3c59e6f4

Convolutional Neural Networks (CNN)

Sample CNN architecture (LENET-5):



Source and reference to read : http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

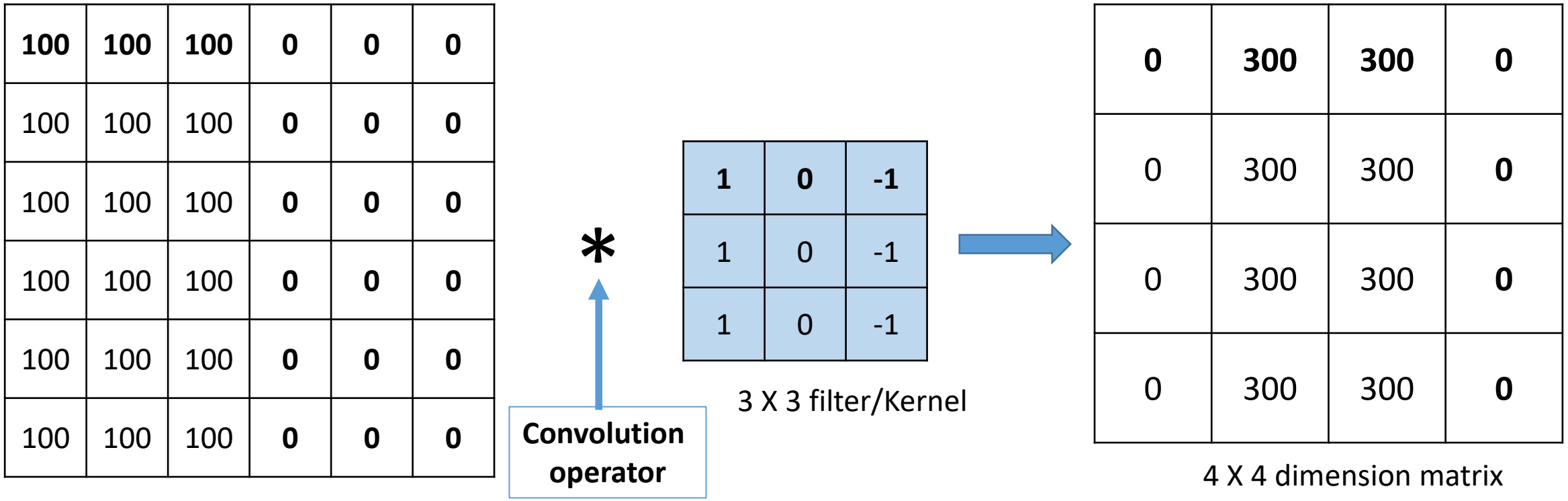
Convolution Layer (CONV)

- **CONV**olution is the first layer to extract features from an input image
- Core building block of a CNN
- Convolutions are basic operation in this layer
- A number of filters (e.g. edge detectors) are applied to the input image

Source and reference to read : http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Convolution Layer (CONV)

Convolution Operation



Convolution Layer (CONV)

Convolution Operation

(100 X 1 + 100 X 1 + 100 X 1) +
(100 X 0 + 100 X 0 + 100 X 0) +
(100 X -1 + 100 X -1 + 100 X -1)

1	0	-1			
100	100	100	0	0	0
100	100	100	0	0	0
100	100	100	0	0	0
100	100	100	0	0	0
100	100	100	0	0	0
100	100	100	0	0	0

6 X 6 dimension image

*

Convolution operator

1	0	-1
1	0	-1
1	0	-1

3 X 3 filter/Kernel

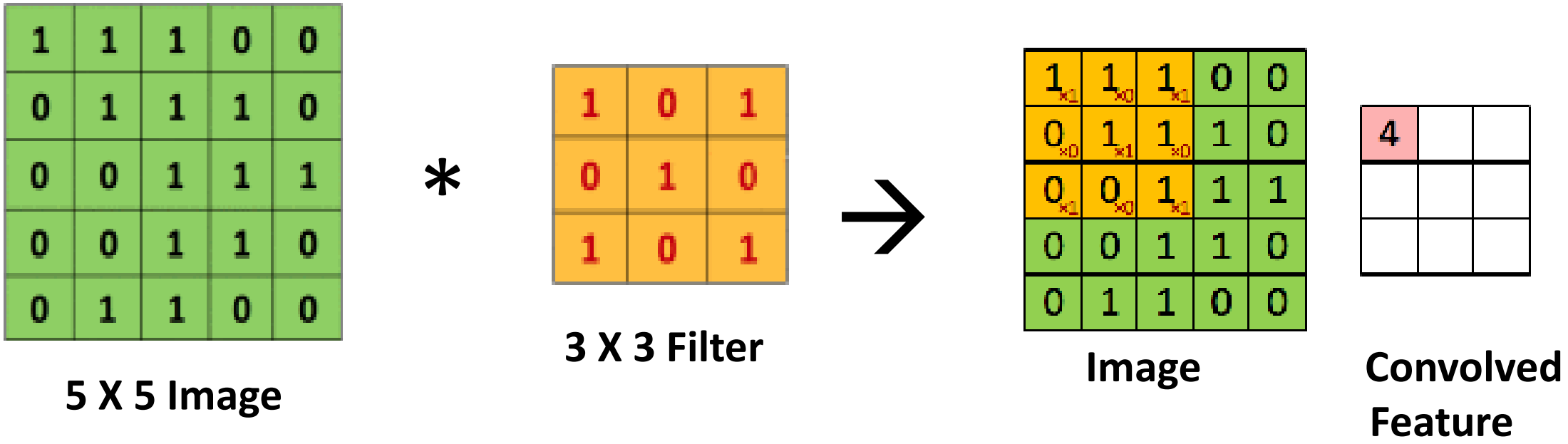
0	300	300	0
0	300	300	0
0	300	300	0
0	300	300	0

4 X 4 dimension matrix

Vertical Edge detector

Convolution Layer (CONV)

Convolution Operation



5 X 5 Image

3 X 3 Filter

Image

Convolved Feature

Image source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Convolution Layer (CONV)

Padding

100	**100**	**100**	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**
100	100	100	**0**	**0**	**0**

6 X 6 dimension image without padding

Padding (p) = 1

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	0	0	**0**	**0**	**0**	**0**	**0**

8 X 8 dimension matrix with padding

Convolution Layer (CONV)

Padding

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	100	100	100	**0**	**0**	**0**	**0**
0	0	0	**0**	**0**	**0**	**0**	**0**

8 X 8 dimension matrix

*

Convolution operator

1	**0**	**-1**
1	0	-1
1	0	-1

3 X 3 filter/Kernel

-200	**0**	**200**	**200**	**0**	**0**
-300	0	300	300	0	**0**
-300	0	300	300	0	**0**
-300	0	300	300	0	**0**
-300	0	300	300	0	**0**
-200	**0**	**200**	**200**	**0**	**0**

6 X 6 dimension matrix
== Input Matrix dimension

Convolution Layer (CONV)

Padding

Input Matrix Dimension : $n\ x\ n$

Filter size: $f\ x\ f$

Padding (p) : 1

So, $(\boldsymbol{n\ X\ n}) * (\boldsymbol{f\ X\ f})$ will produce $(\boldsymbol{n + 2p - f + 1})\ \boldsymbol{X}\ (\boldsymbol{n + 2p - f + 1})$

Input Matrix

Output Matrix

e.g.: $(6\ X\ 6)\ * (3\ X\ 3) \rightarrow\ \ 6\ X\ 6$ Output matrix

Reference and Source: http://cs230.stanford.edu/files/C4M1.pdf

Convolution Layer (CONV)

Padding

Given: **Input Matrix Dimension** : $n\ x\ n$

Filter size: $f\ x\ f$

Required Output Size = $(\boldsymbol{n + 2p - f + 1})\boldsymbol{X}\ (\boldsymbol{n + 2p - f + 1})$

Question: What is pad size ($\boldsymbol{p}$) so that the input and output matrix are of same sizes?

So, $(n + 2p - f + 1) = n$

$$p = \frac{(f - 1)}{2}$$

Reference and Source: http://cs230.stanford.edu/files/C4M1.pdf

Convolution Layer (CONV)

Padding (Same and Valid)

Valid Padding: ≈ *No Padding* (Padding $\boldsymbol{p}$ = 0)

So, Output size will be → $(\boldsymbol{n} - \boldsymbol{f} + \mathbf{1})\boldsymbol{X}\ (\boldsymbol{n} - \boldsymbol{f} + \mathbf{1})$

Same Padding: ≈ *Output size and input size is same*, this requires appropriate padding. Hence use $p = \frac{(f-1)}{2}$, for calculate the required padding.

Convolution Layer (CONV)

Stride

It is the number of pixels by which we slide the filter over the input matrix

Example:

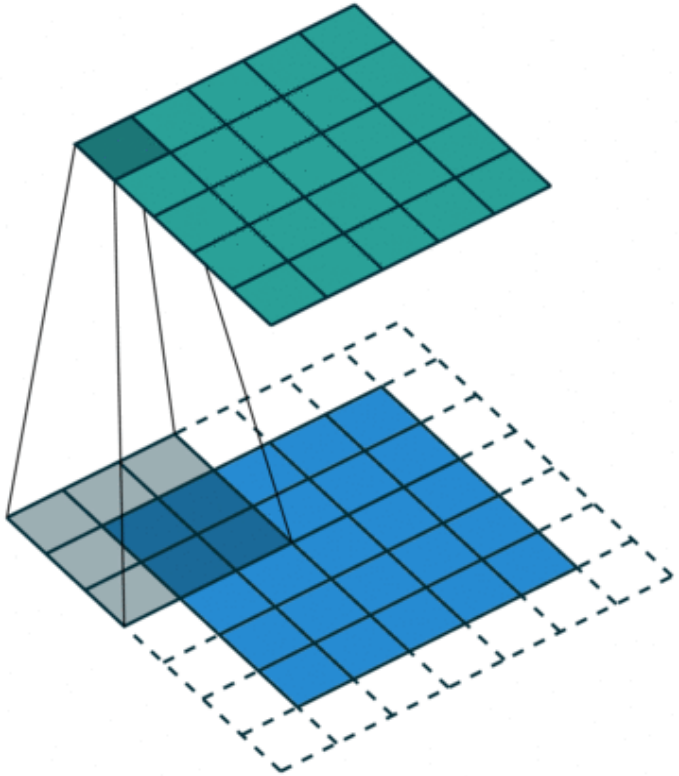
1. Stride(s) = 1: Move the filter by one pixel horizontally and vertically
2. Stride(s) = 2: Move the filter by two pixels horizontally and vertically

Source and Reference: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

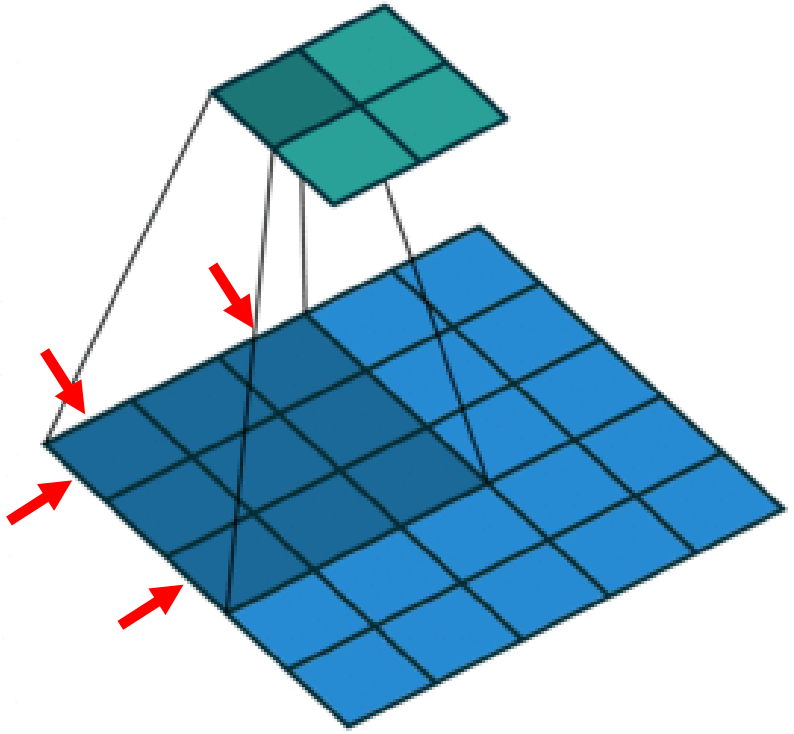
Convolution Layer (CONV)

Stride and Padding illustration

Convolution with stride (s) =1
padding (p) = 1



Convolution with stride (s) =2
padding (p) = 0



Convolution with stride (s)=2,
padding (p) = 1

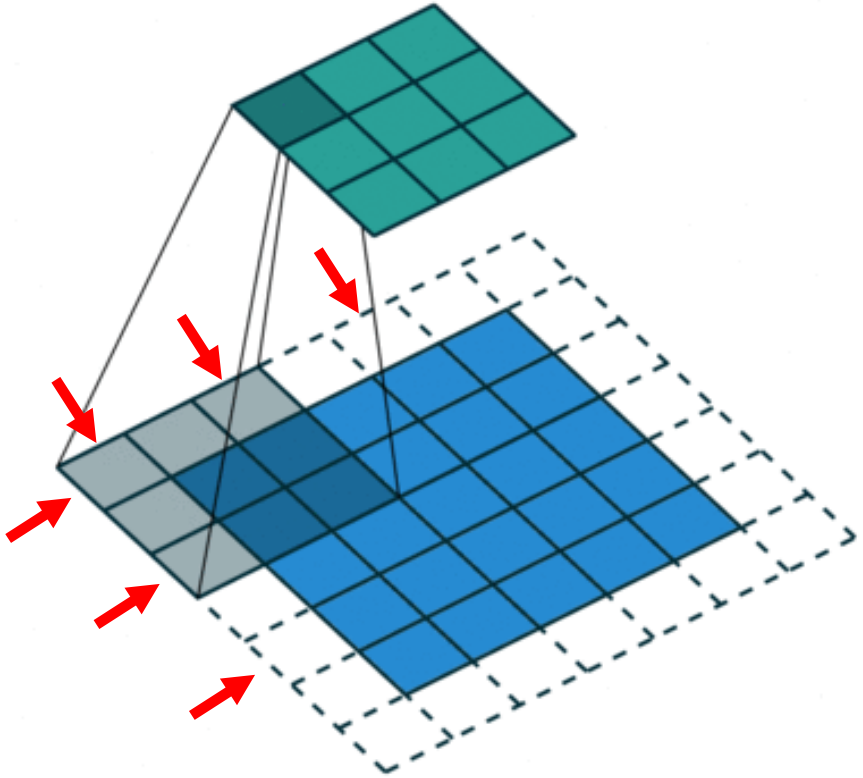


Image source: https://github.com/vdumoulin/conv_arithmetic

Convolution Layer (CONV)

Output size with Stride and padding

Given:

Input Matrix Dimension : $n \; x \; n$

Filter size: $f \; x \; f$

Padding: p

Stride : s

$$\textbf{Output Size} = \left(\frac{n + 2p - f}{s} + 1 \right) \boldsymbol{X} \left(\frac{n + 2p - f}{s} + 1 \right)$$

Example:
Input Matrix Dimension : $7 \; x \; 7$, **Filter size**: $3 \; x \; 3$
Padding: 0, Stride : 2

$$\textit{Output Size} = 3 \; X \; 3$$

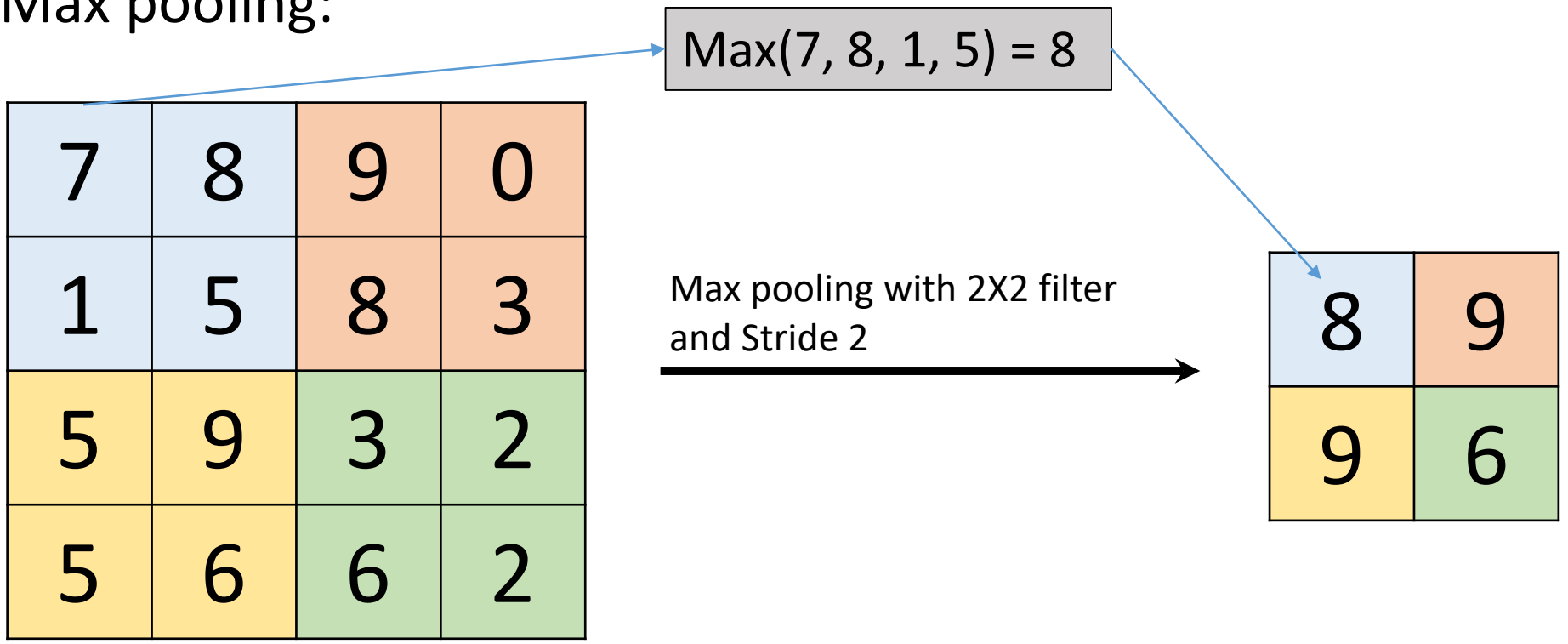
Reference and Source: http://cs230.stanford.edu/files/C4M1.pdf

Pooling Layer (POOL)

- Pooling layer is a down sampling operation which reduces the dimensionality of a matrix.
- In other words, it reduces the number of parameters for large image, but retain the valuable information.
- 3 types:
  - Max pooling
  - Average pooling
  - Sum pooling

Pooling Layer (POOL)

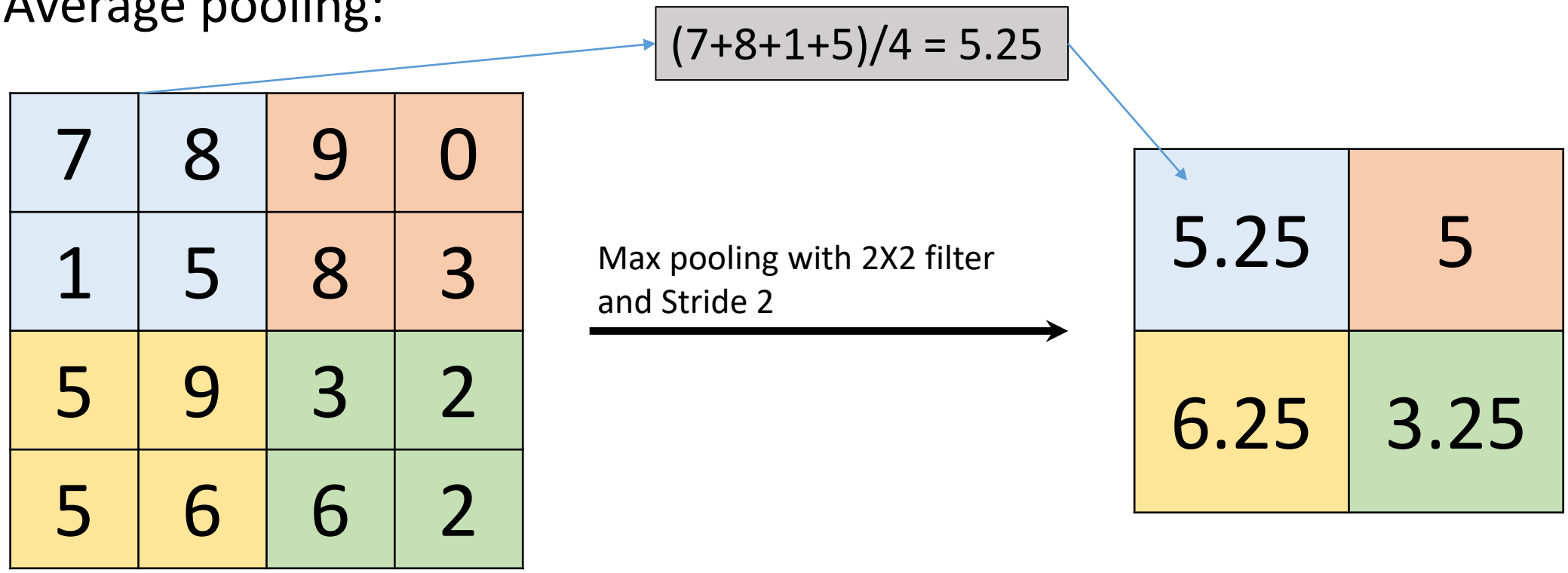
- Max pooling:



Source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Pooling Layer (POOL)

- Average pooling:



Source: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Fully Connected Layer (FC)

- In FC layer, the output matrix after convolution layer is flattened and feed into a fully connected layer similar to ANN

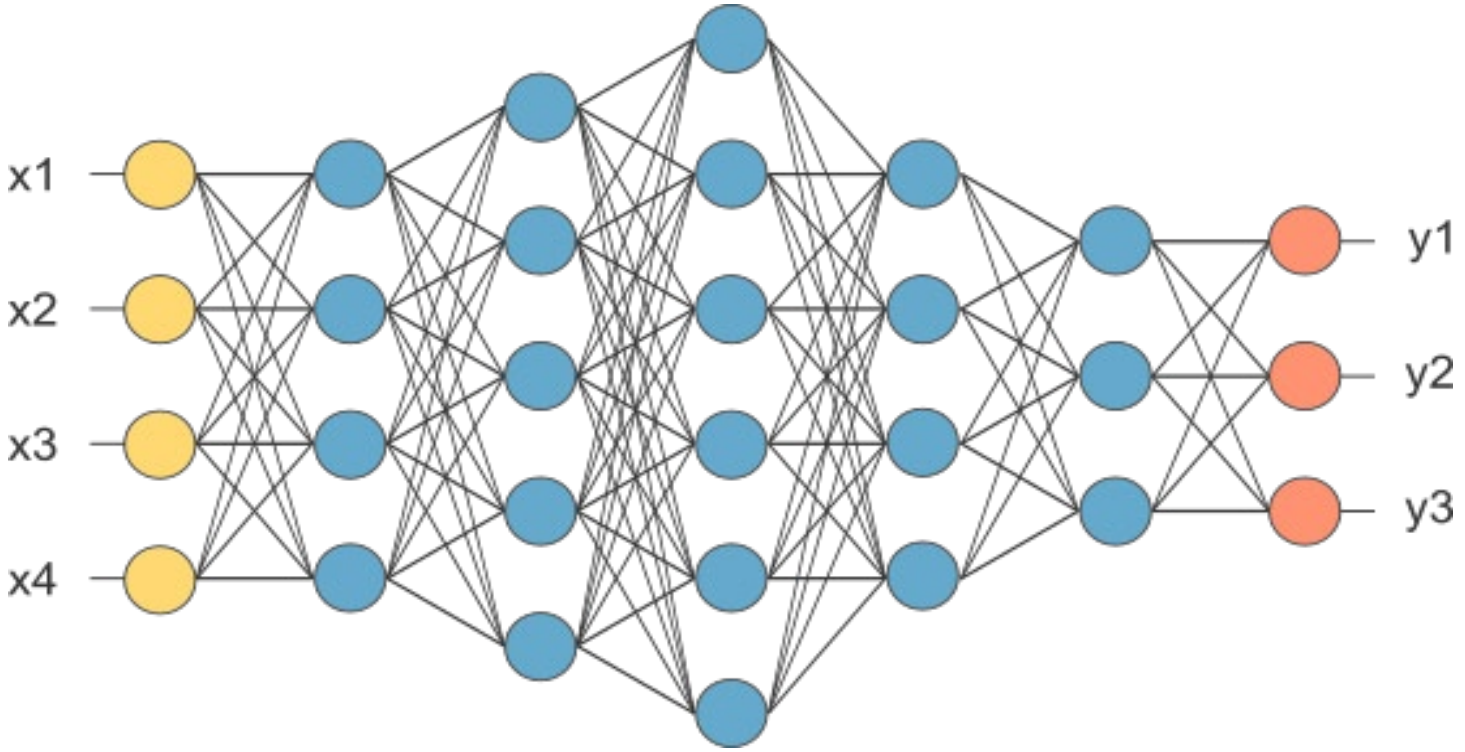


Image source ad reference: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Fully Connected Layer (FC)

- It is a traditional Multi-layer Perception/Neural Network
- For multi-class classification, usually Softmax activation is used.
- Softmax ensures the output
- Output of the CONV and POOL layers represent a high level feature of the Input image
- FC layer uses this feature to classify the input image into the desired class.

Reference: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

CNN layers visualization and intuition

Example: Face recognition using CNNs

Uses simple shapes to form higher level features like facial shapes!

Uses edges to detect simple shapes

Low level feature like edges from raw pixels

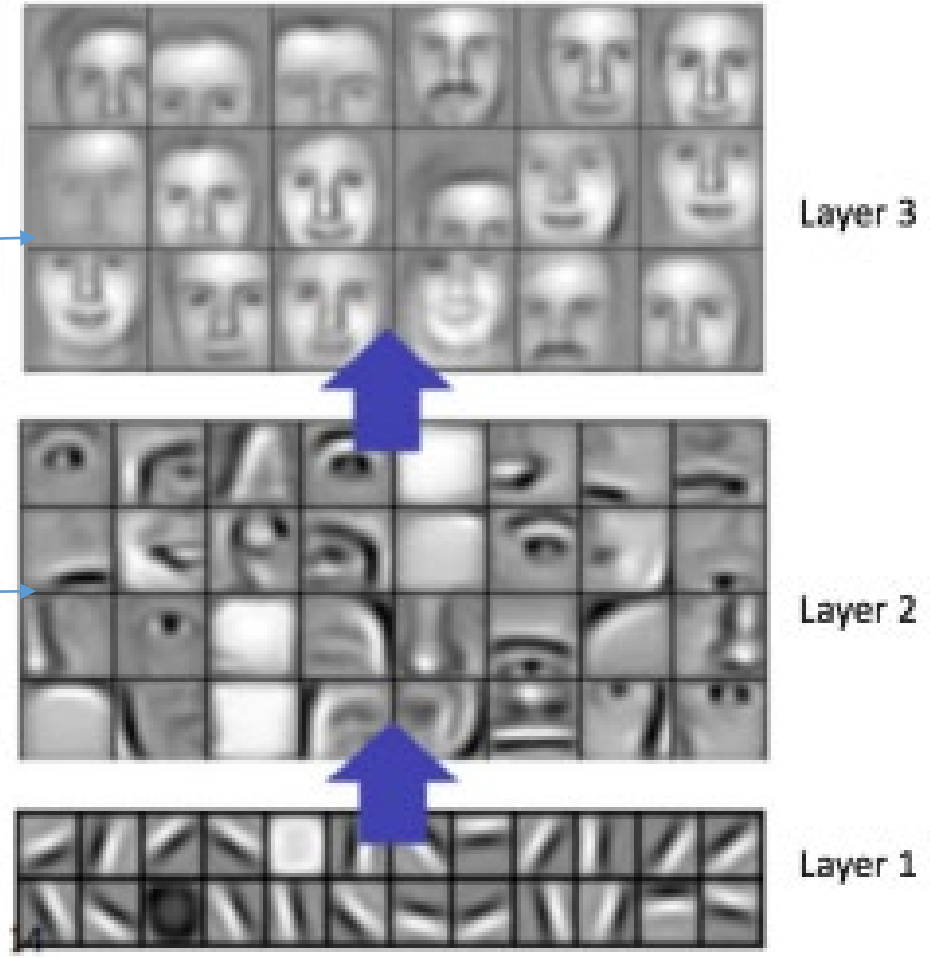


Image Source: http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf