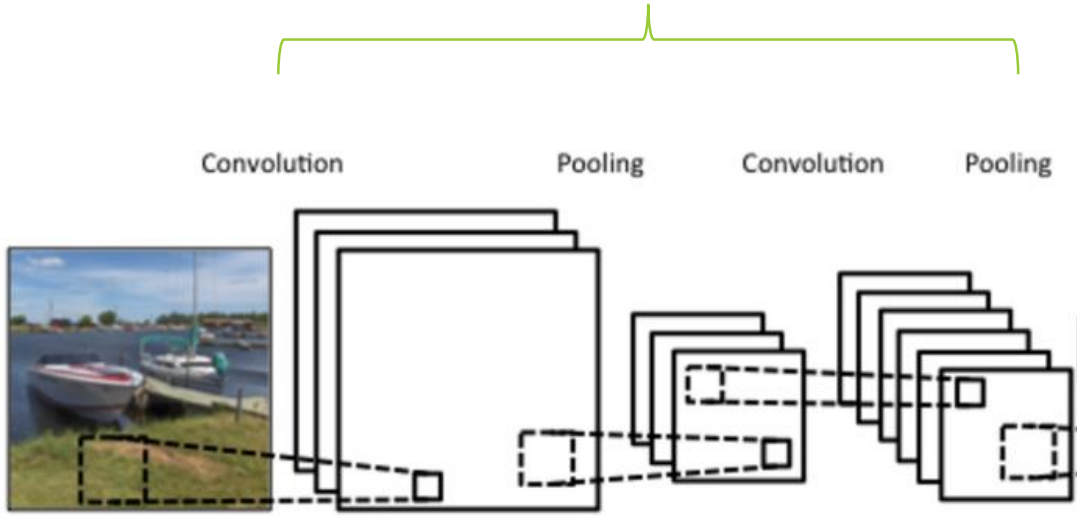
機器學習於材料資訊的應用
Machine Learning on Material Informatics

陳南佑(NAN-YOW CHEN)

楊安正(AN-CHENG YANG)

卷積神經網路
Convolutional Neural Network (CNN)

- CNN = Convolutional Filters



- It’s a category of Neural Networks that have proven very effective in areas such as image recognition and classification.

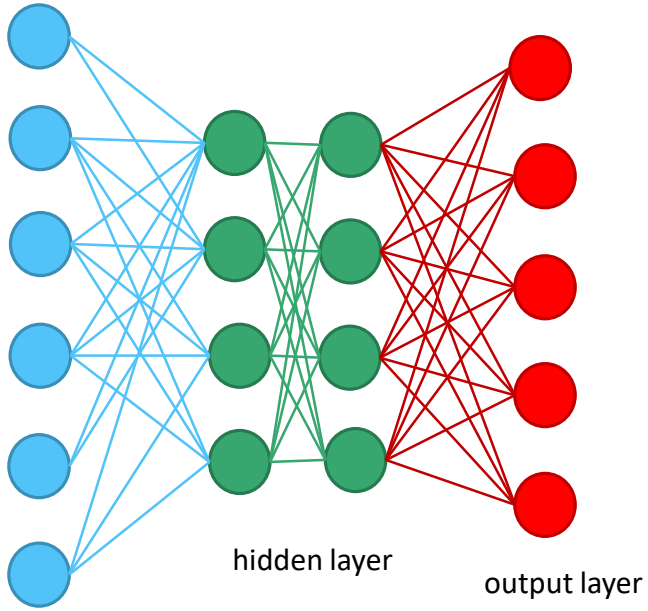
Ref. An Intuitive Explanation of Convolutional Neural Networks

Why CNN for image

- Reduce the training parameters to improve efficiency.

Size : 1654 x 930 = 1538220

1538220



input layer

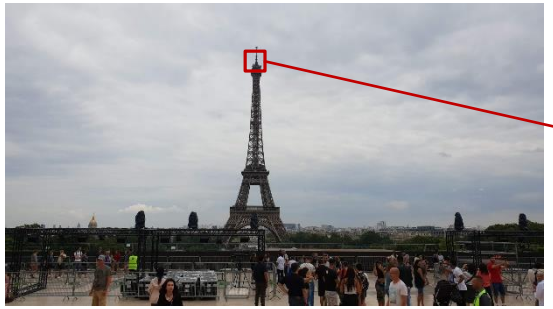
Why CNN for image

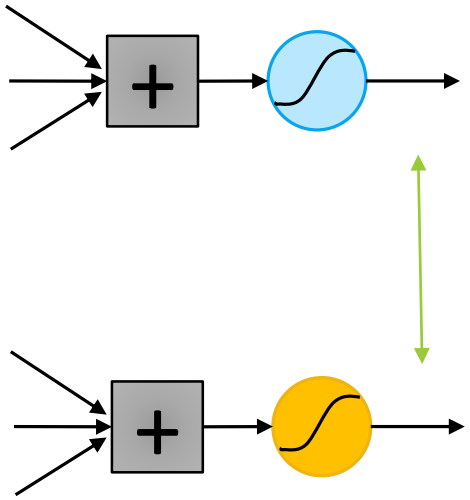
- Some patterns are much smaller than the whole image.

- A neuron does not have to see the whole image to discover the pattern.

Why CNN for image

- The same patterns appear in different regions.





They can use the same set of parameters!

“ Tip ” detector

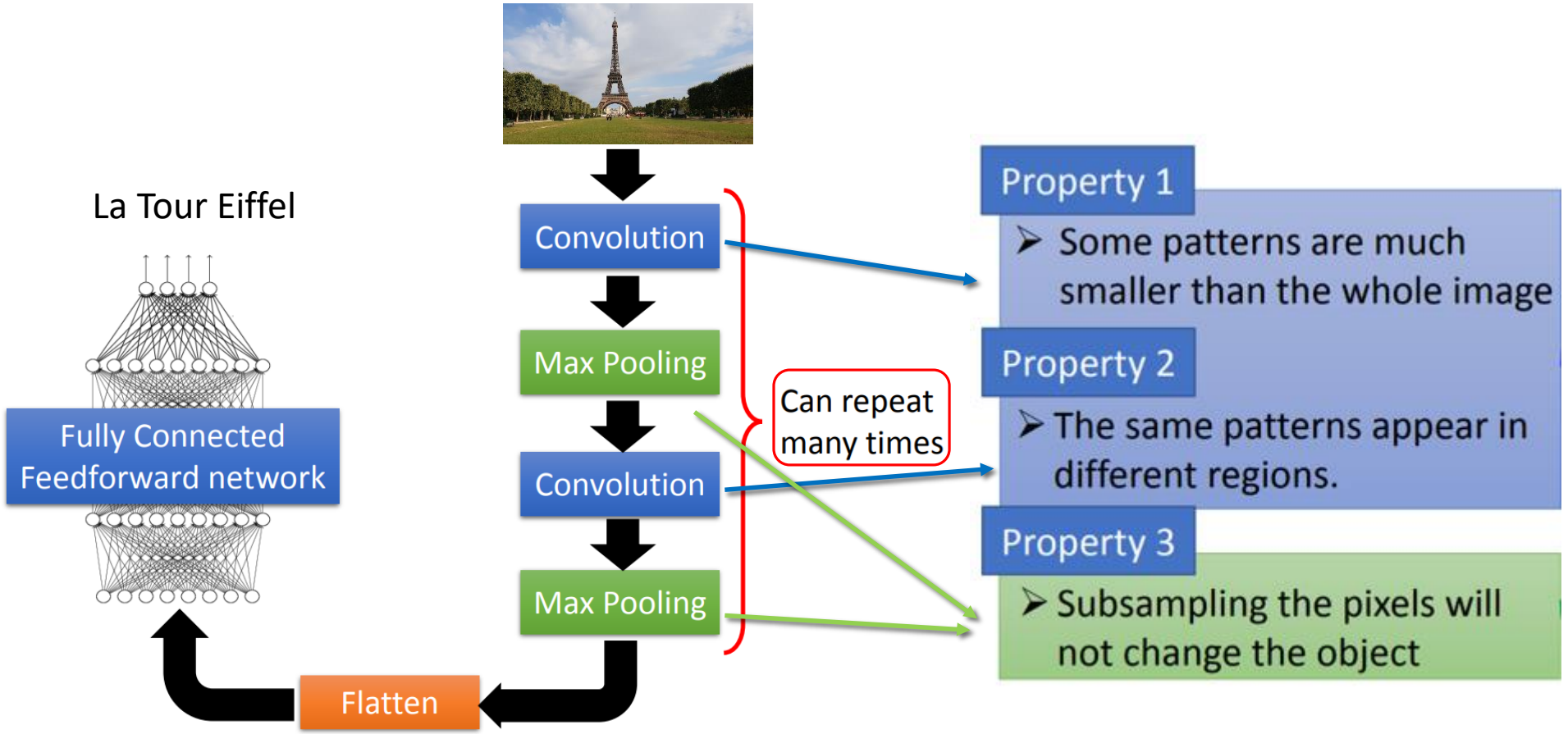
Why CNN for image

- Subsampling the pixels will not change the object.

subsampling

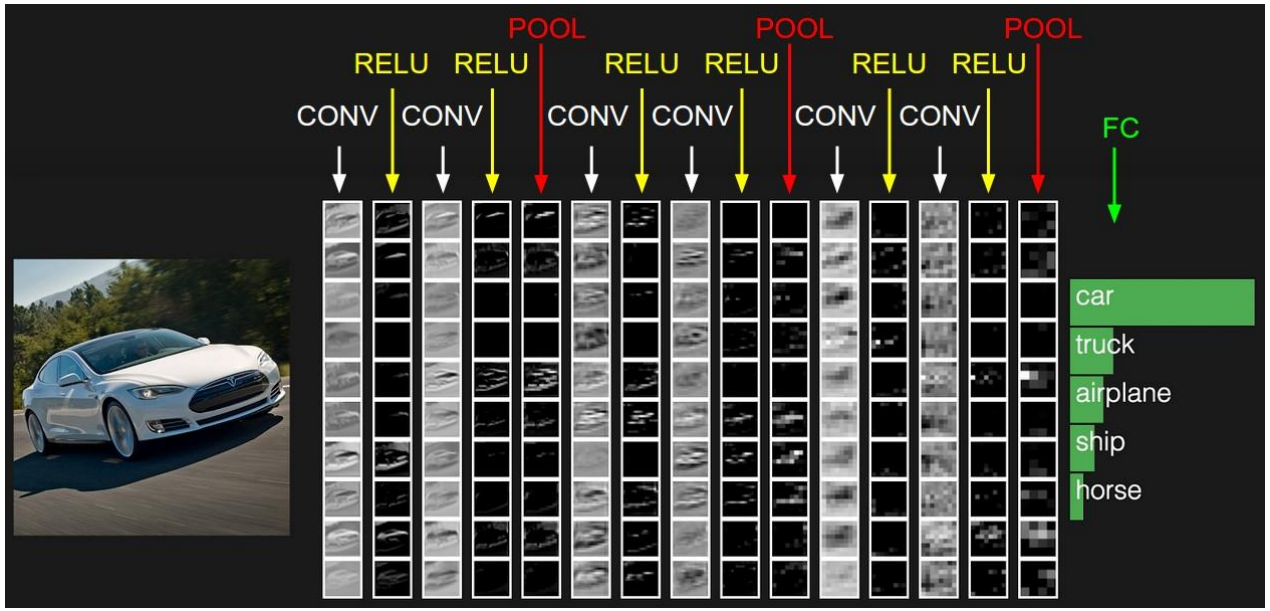
- Less parameters for the network to process the image.

CNN for image



Convolutional Neural Network

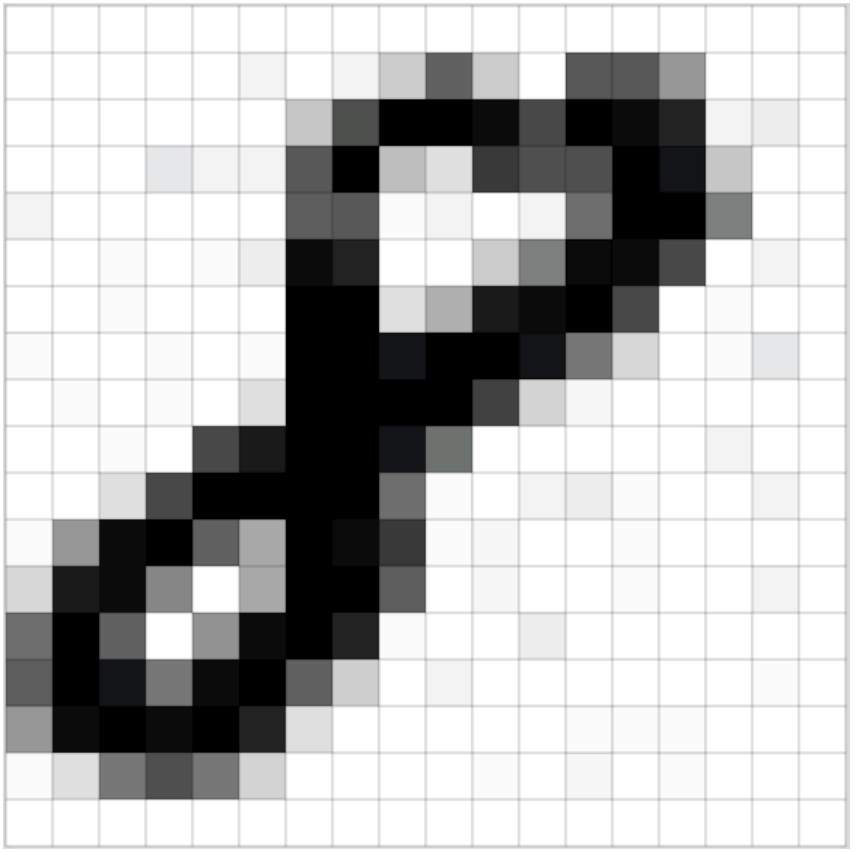
- There are four main operations in the CNN:
  - Convolution
  - Activation Function : Non Linearity (ReLU)
  - Pooling or Sub Sampling
  - Classification (Fully Connected Layer)



Ref. An Intuitive Explanation of Convolutional Neural Networks & http://cs231n.github.io/convolutional-networks/

Convolution

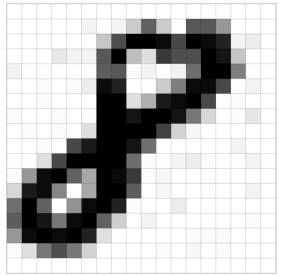
Number : 8
Size : 18 x 18 pixels
Gray scale : 0- 255



https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

Convolution - Filter (Kernel)

- Each filter detects a small pattern (3 x 3).



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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

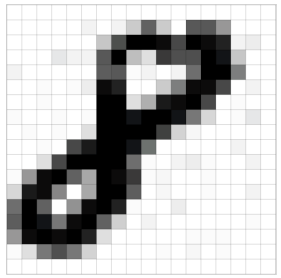
Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

Convolution - Filter (Kernel)



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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Matrix

3

= 1x1 + 0x(-1) + 0x(-1)
+ 0x(-1) + 1x1 + 0x(-1)
+ 0x(-1) + 0x(-1) + 1x1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

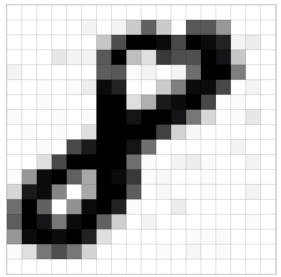
-1

= 1x(-1) + 0x1 + 0x(-1)
+ 0x(-1) + 1x1 + 0x(-1)
+ 0x(-1) + 0x1 + 1x(-1)

Those are the network parameters to be learned!

Convolution - Stride

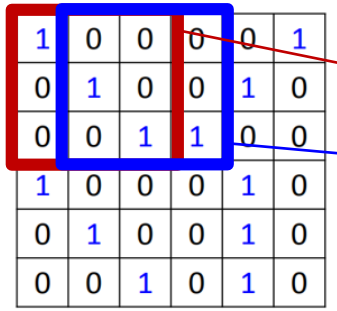
- Stride settings detect different patterns



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

6 x 6 image

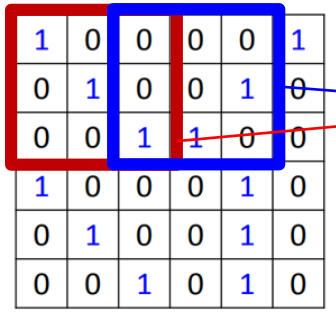
Stride = 1



1	-1	-1
-1	1	-1
-1	-1	1

3 -1

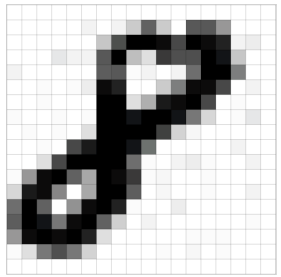
Stride = 2



3 -3

Convolution - Padding

- Padding settings detect different size patterns .



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

6 x 6 image

Valid padding

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

Same padding

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

Stride = 1

1	-1	-1
-1	1	-1
-1	-1	1

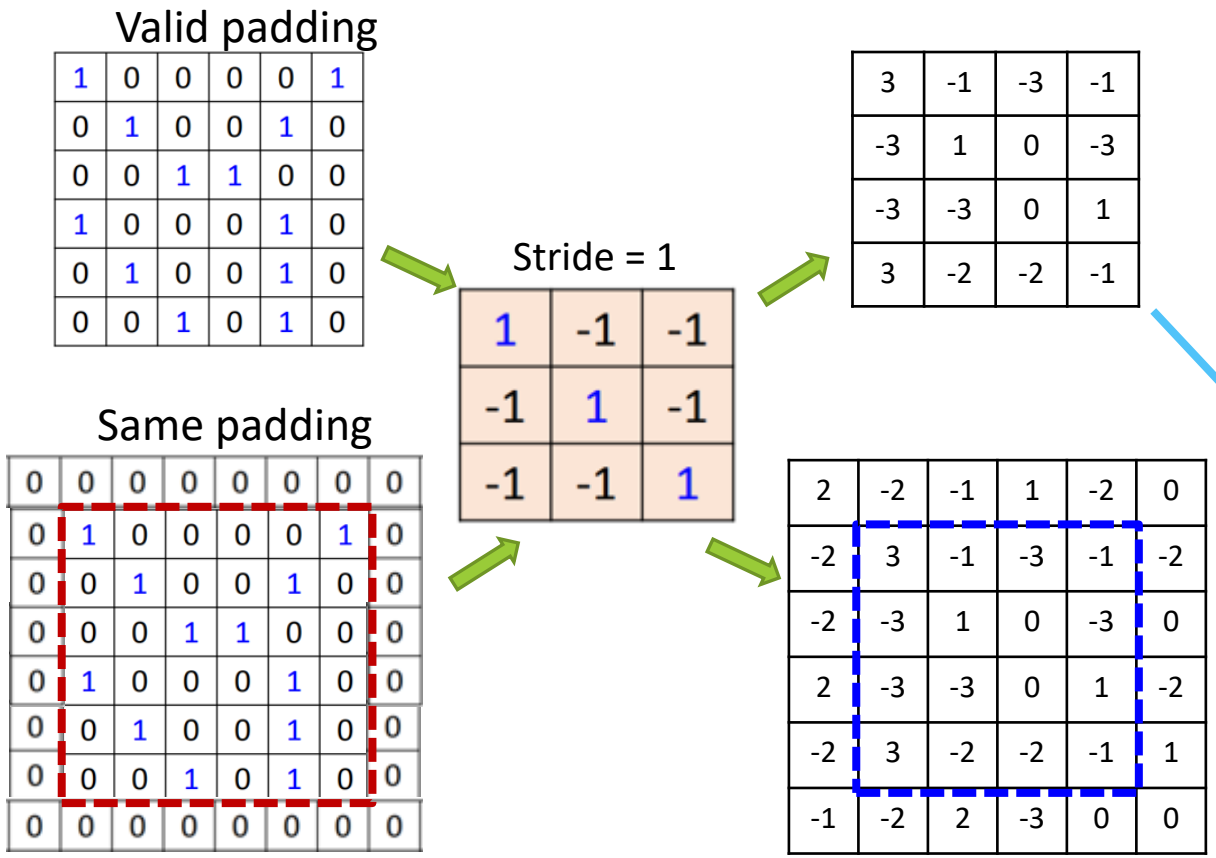
4 x 4

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

6 x 6

2	-2	-1	1	-2	0
-2	3	-1	-3	-1	-2
-2	-3	1	0	-3	0
2	-3	-3	0	1	-2
-2	3	-2	-2	-1	1
-1	-2	2	-3	0	0

Convolution - Padding



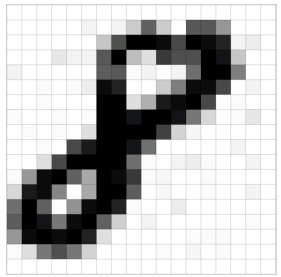
- S : stride, W : input size, F : filter size, P : padding size, ⌈ ⌉: ceil function
  - New height size = new width size = ⌈(W – F + 2 x P) / S + 1⌉

S = 1, W = 6, F = 3, P = 0,
New size = (6 – 3 + 2 x 0) / 1 + 1 = 4

S = 1, W = 6, F = 3, P = 1,
New size = (6 – 3 + 2 x 1) / 1 + 1 = 6

Convolution - Feature map

- Do the same process for every filter.



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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

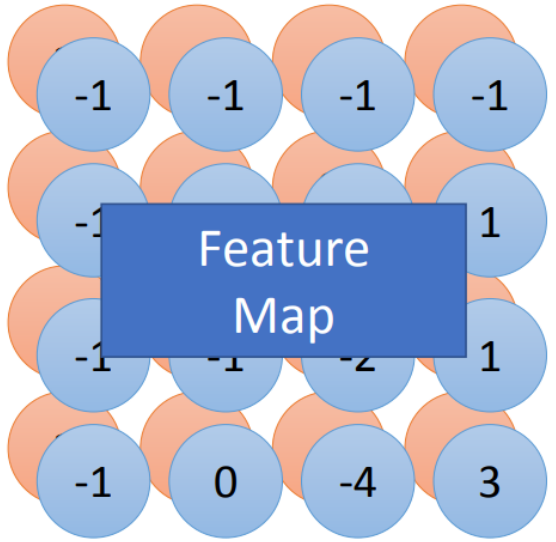
Filter 1

Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

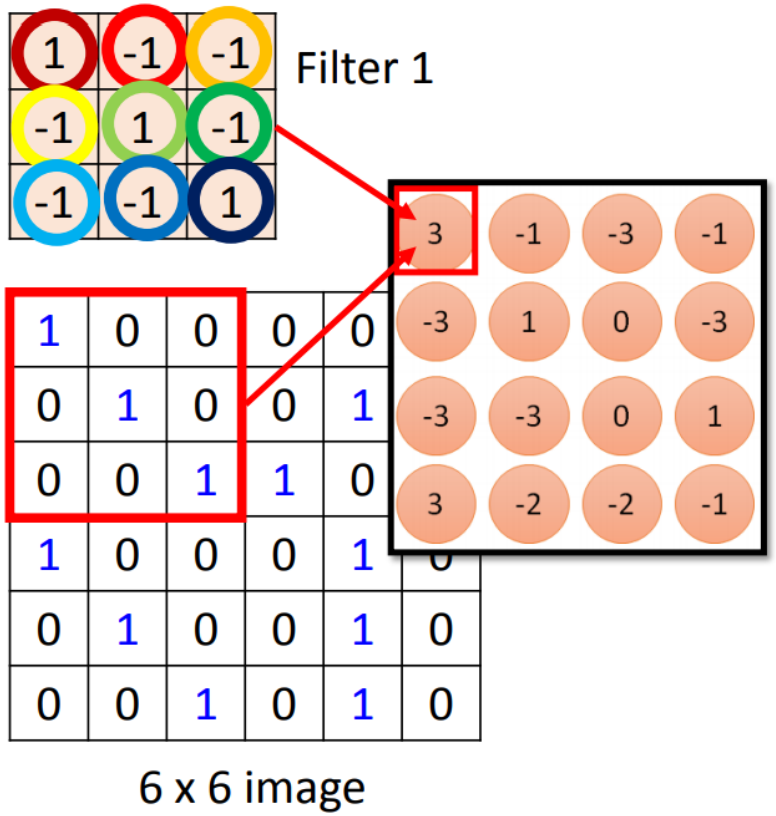
Matrix

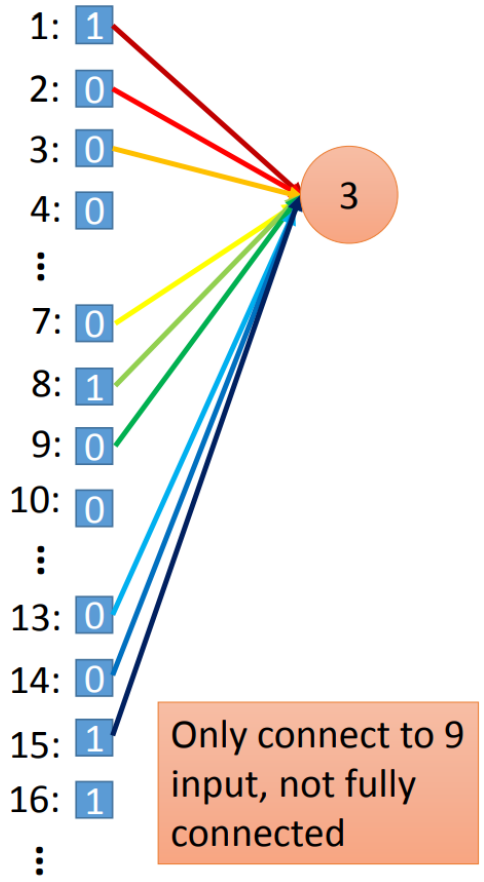


4 x 4 image

Convolution - Not fully connected

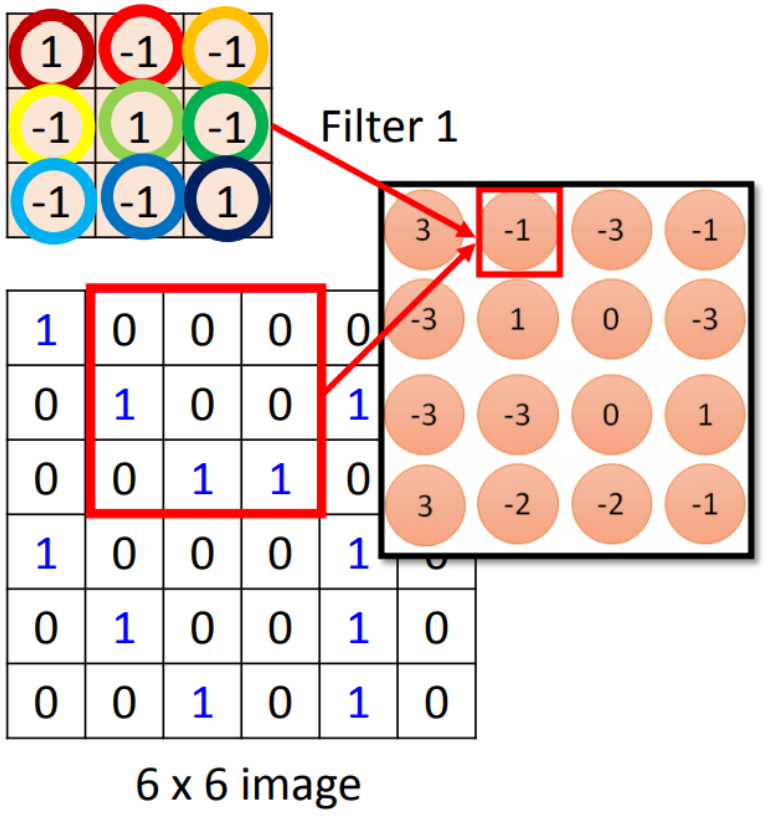
- Less parameters!

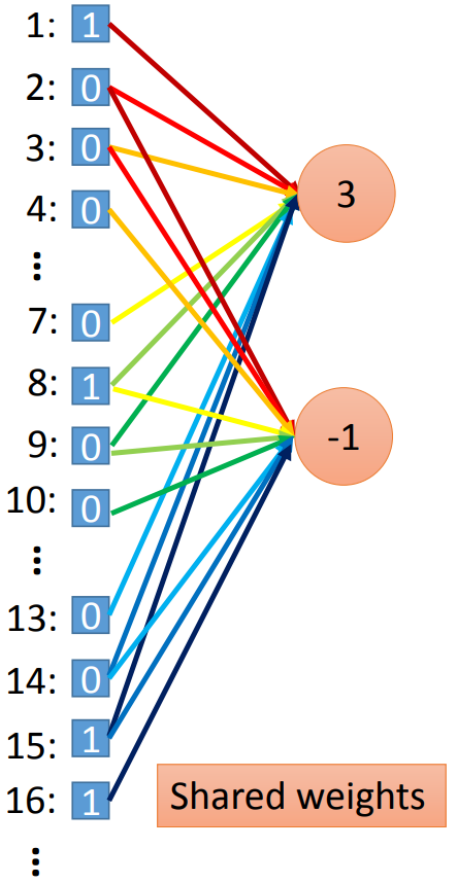




Convolution - Shared weights

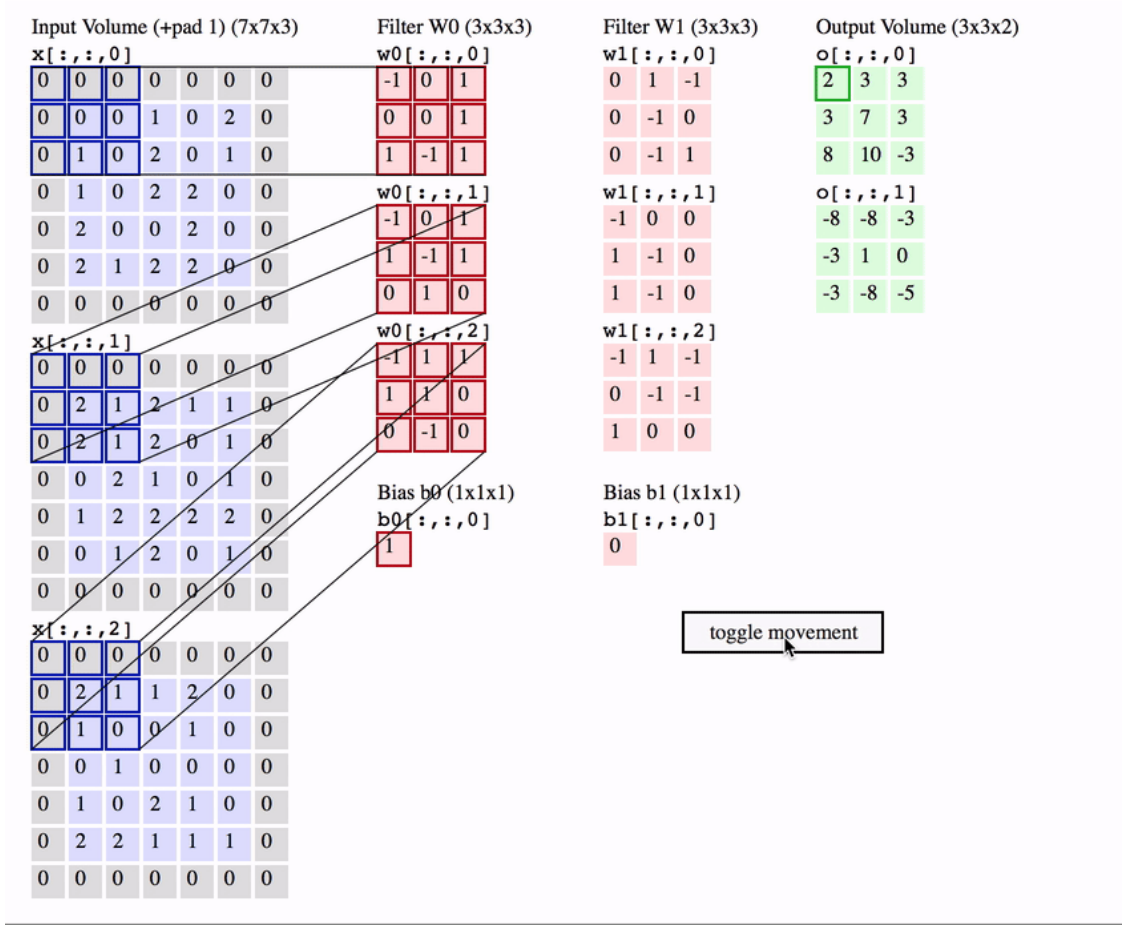
- Even less parameters !



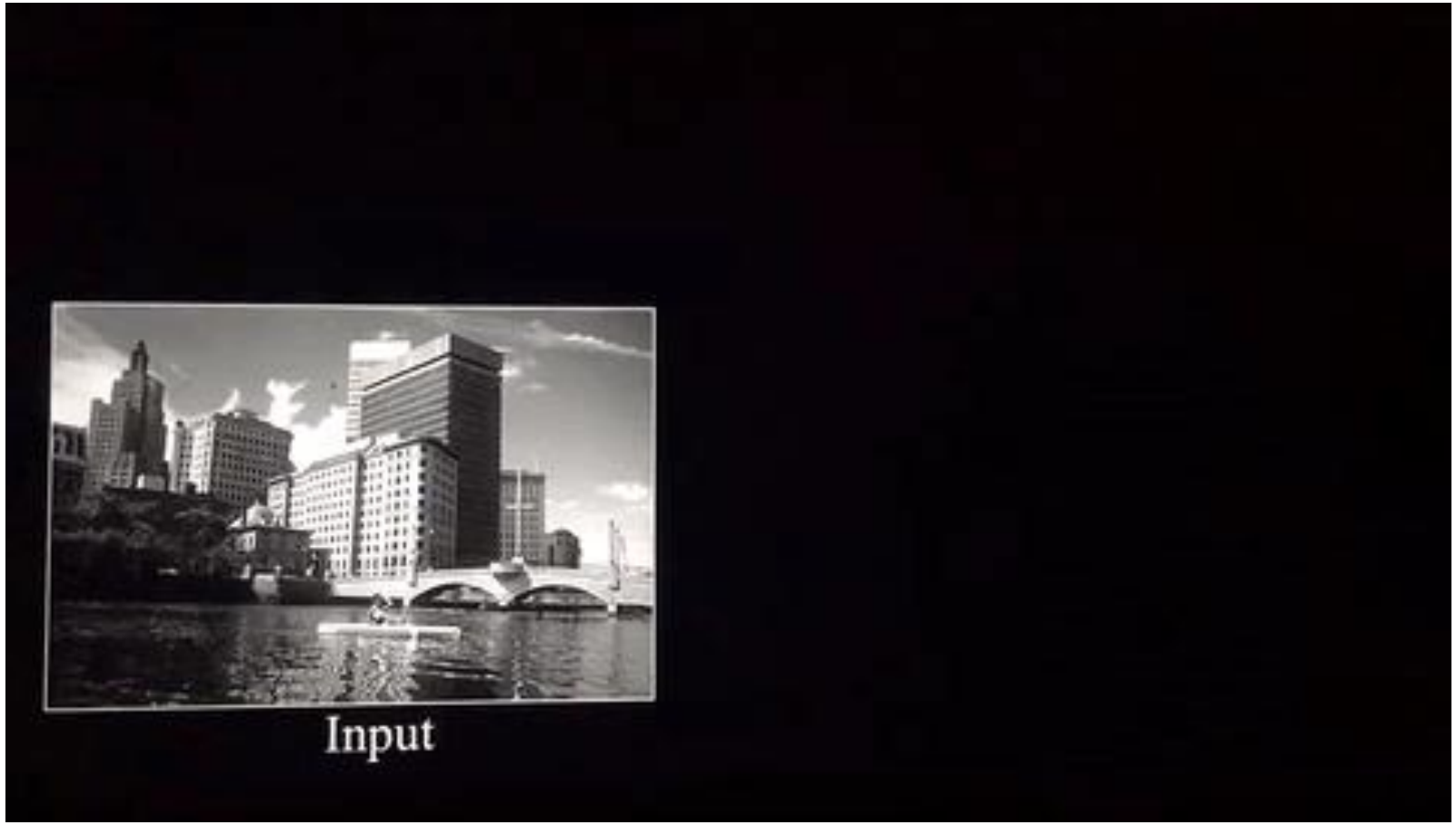


Convolution - Example

- K = 2 filters F = 3 , stride S = 2, and input padding P = 1



Convolution - Example

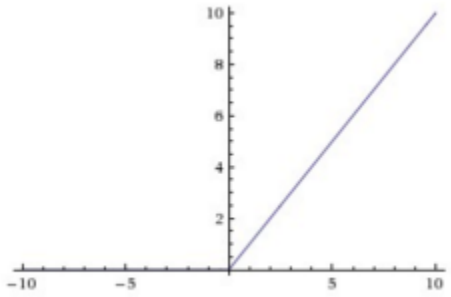


Ref. An Intuitive Explanation of Convolutional Neural Networks & https://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

Activation function - ReLu

- Rectified Linear Unit, ReLU

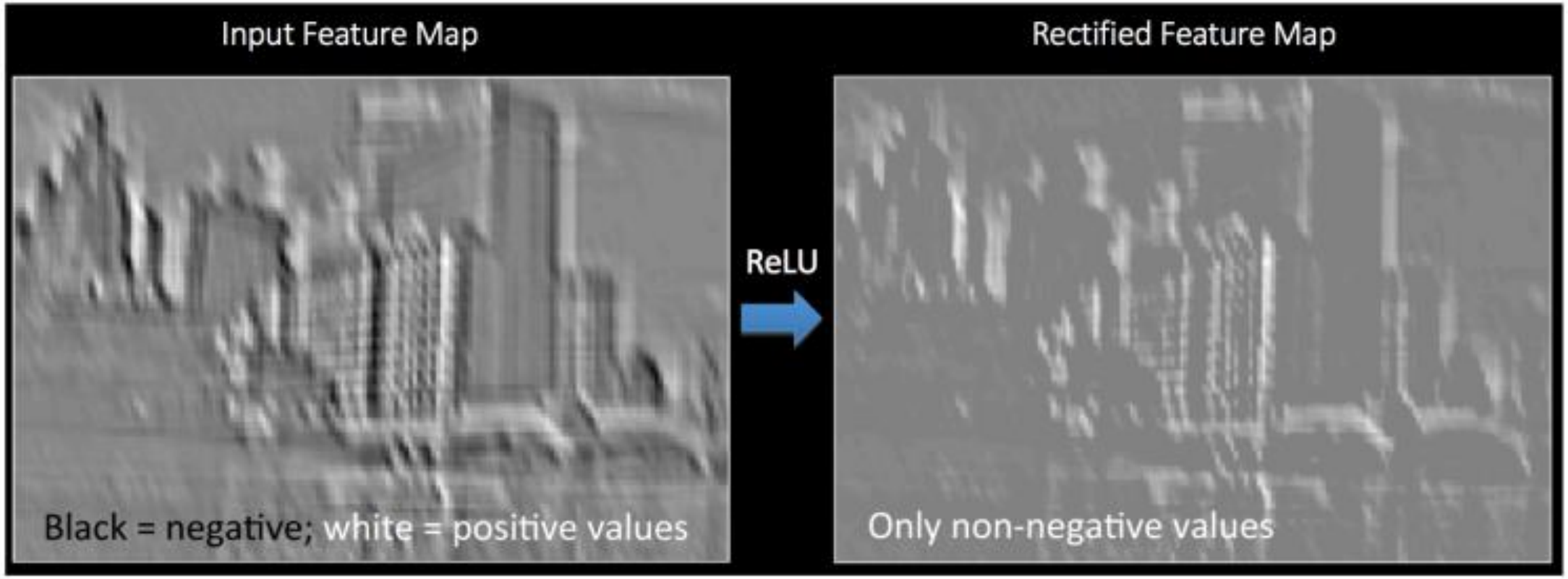
Output = Max(zero, Input)



- ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.
- Other non linear functions such as tanh or sigmoid can also be used instead of ReLU, but ReLU has been found to perform better in most situations.

Ref. An Intuitive Explanation of Convolutional Neural Networks

Activation function - ReLu

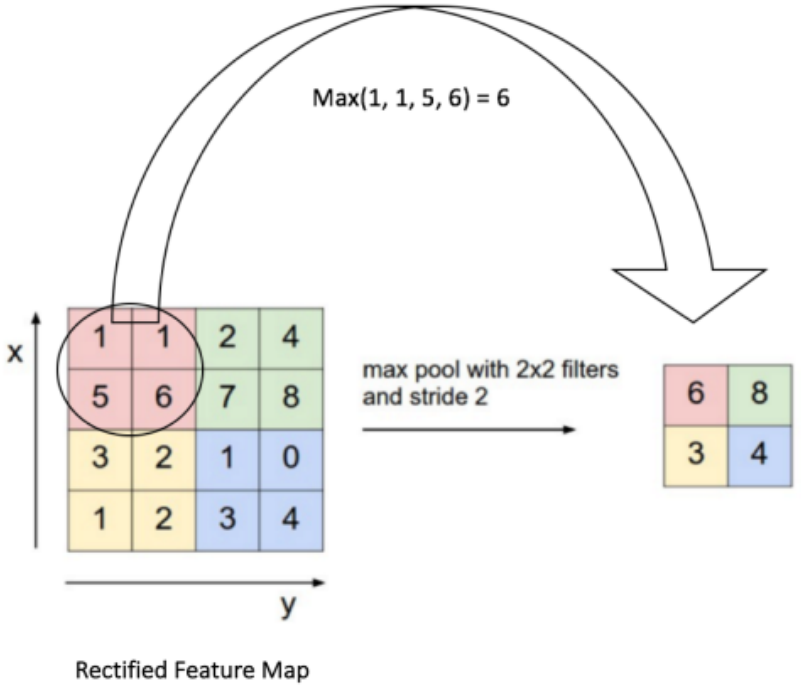


Ref. An Intuitive Explanation of Convolutional Neural Networks

Pooling

- Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information.
- Spatial Pooling can be of different types: Max, Average, Sum etc.

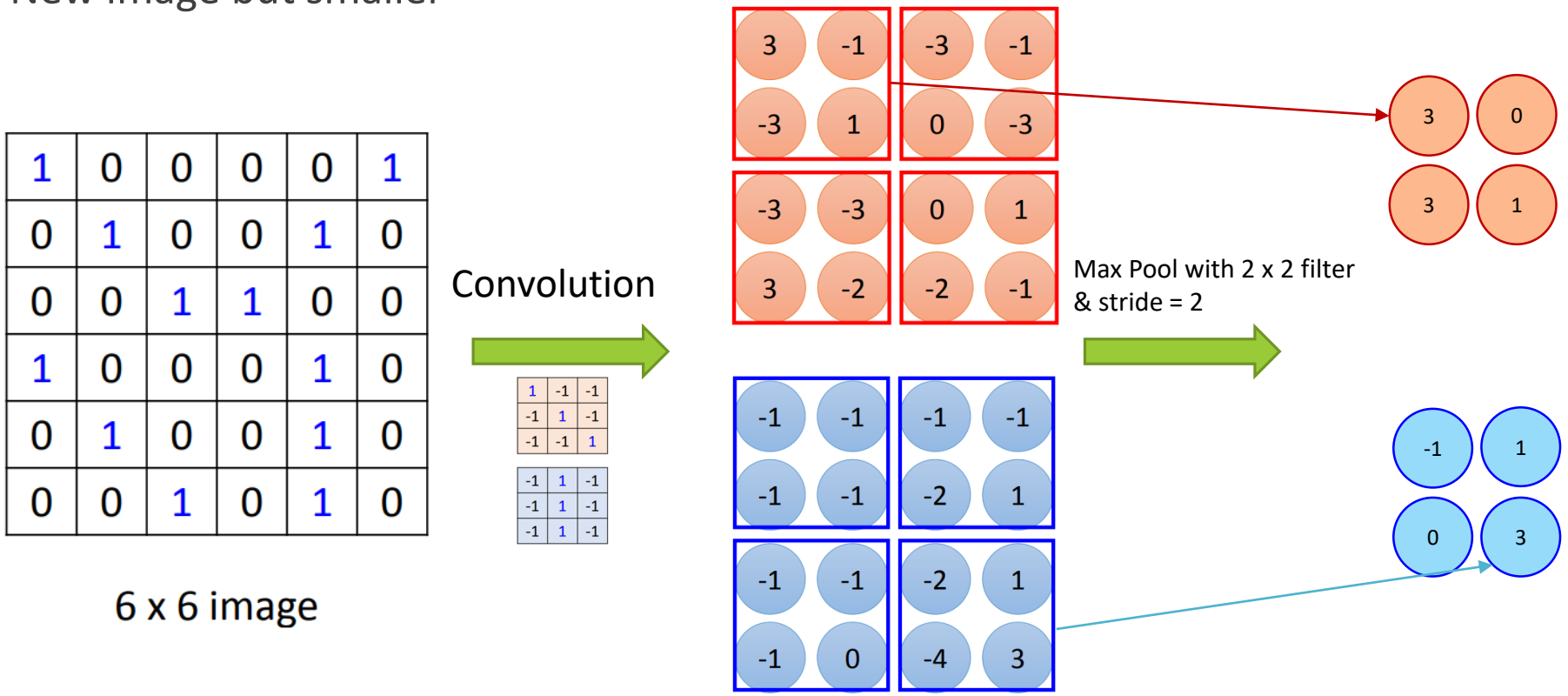
Max Pooling



Ref. An Intuitive Explanation of Convolutional Neural Networks & http://cs231n.github.io/convolutional-networks/

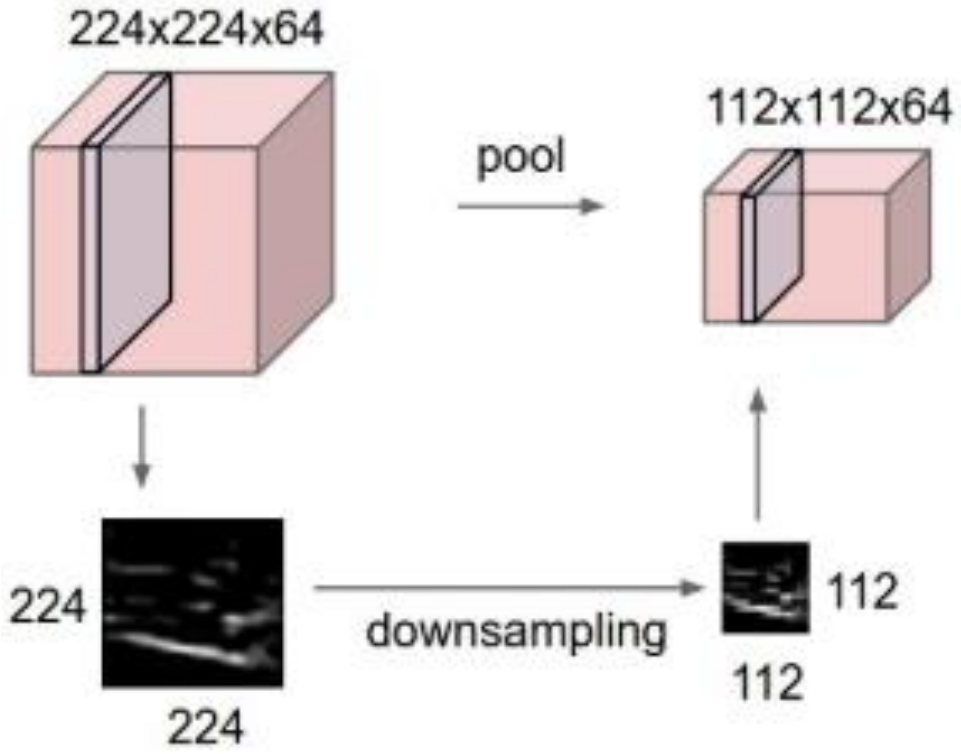
Pooling - Max pooling

- New image but smaller



Pooling - Downsample

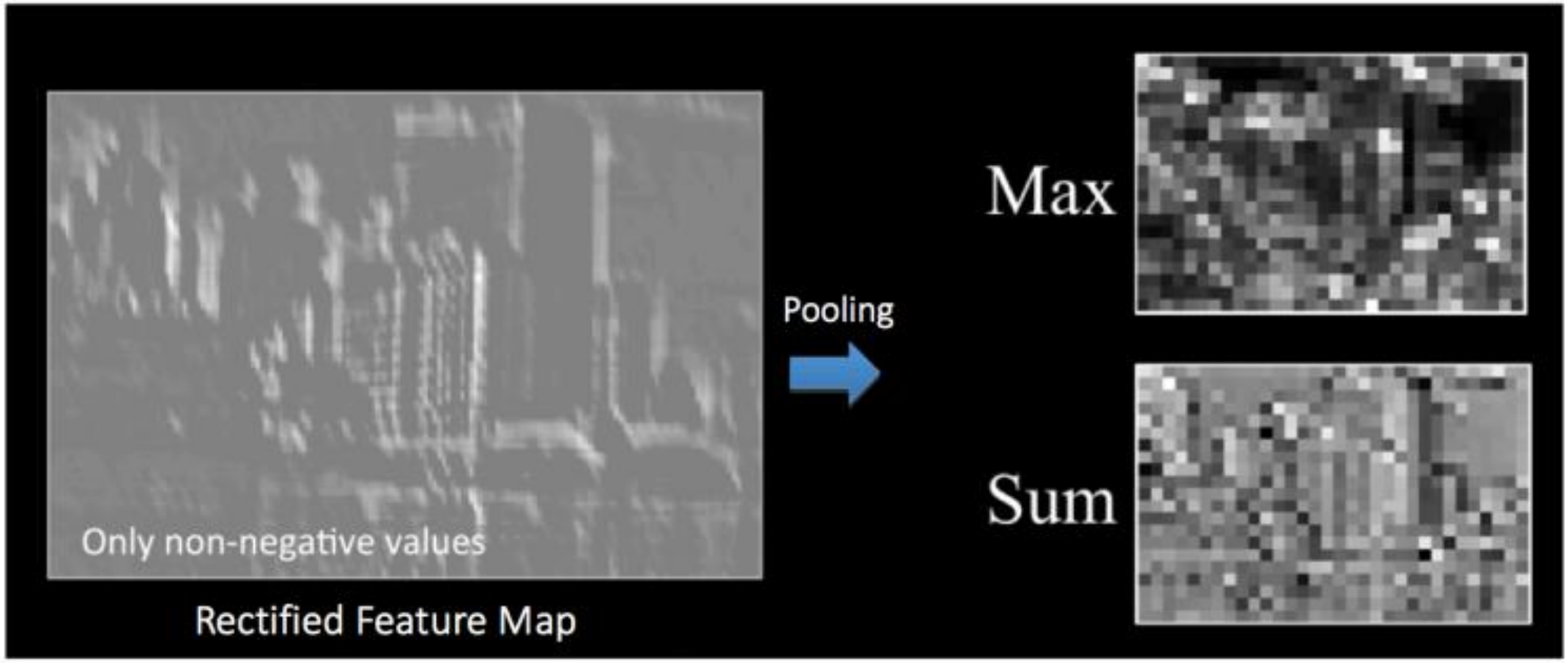
- Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume.



- S : stride, W : input size, F : filter size, P : padding size
- New height size = new width size = [(W – F) / S + 1]

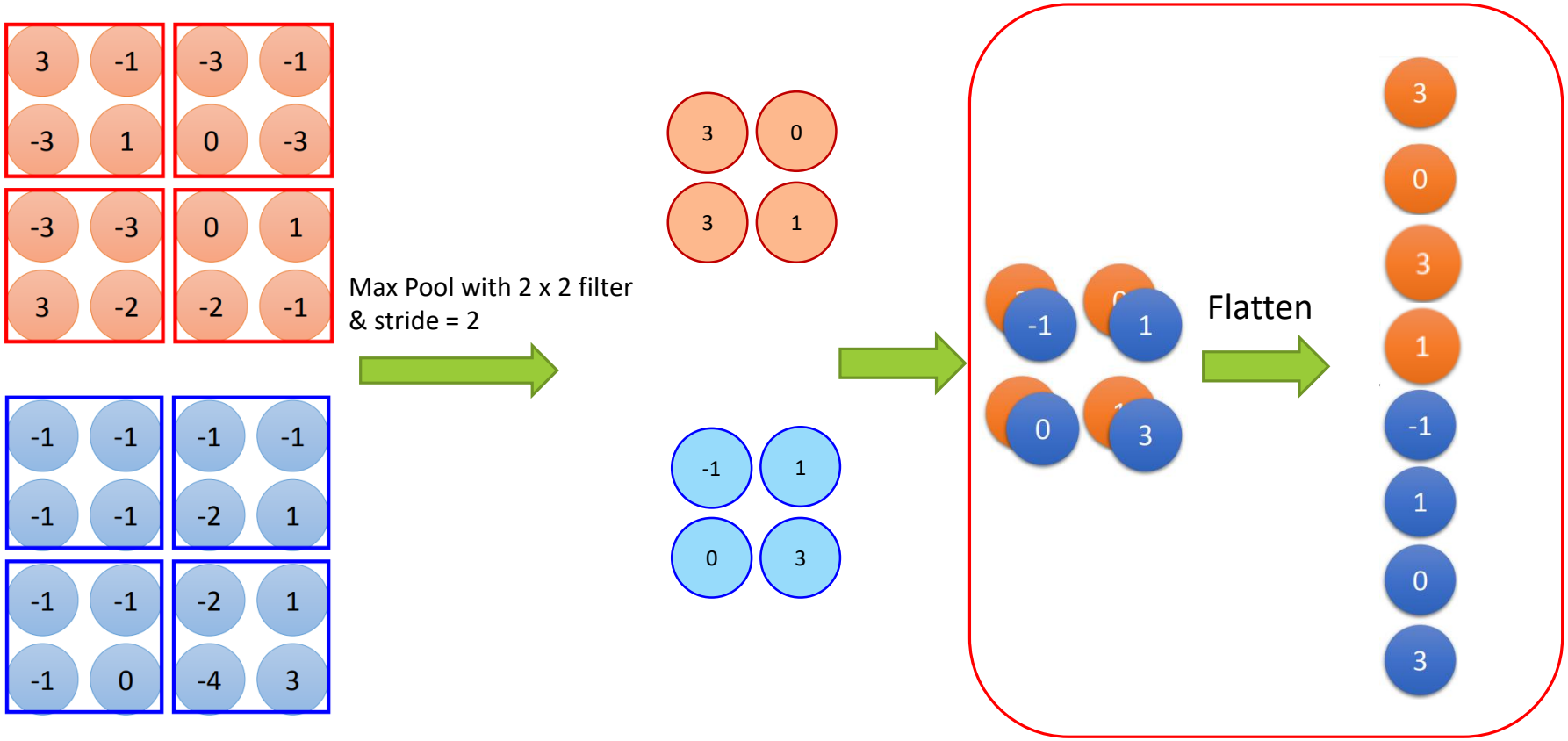
Ref. An Intuitive Explanation of Convolutional Neural Networks & http://cs231n.github.io/convolutional-networks/

Pooling - Example



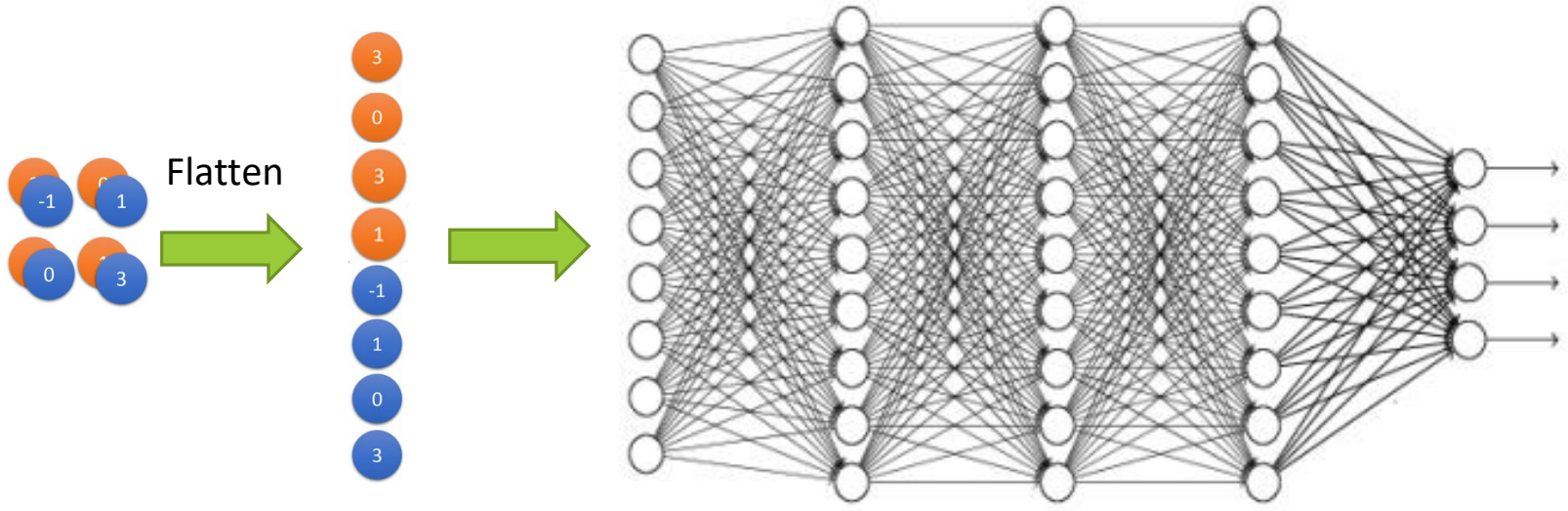
Ref. An Intuitive Explanation of Convolutional Neural Networks & http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Flatten


3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
-1 -1 -1 -1
-1 -1 -2 1
-1 -1 -2 1
-1 0 -4 3
Max Pool with 2 x 2 filter
& stride = 2
3 0
3 1
-1 1
0 3
-1 1
0 3
Flatten
3
0
3
1
-1
1
0
3

Fully connected layer

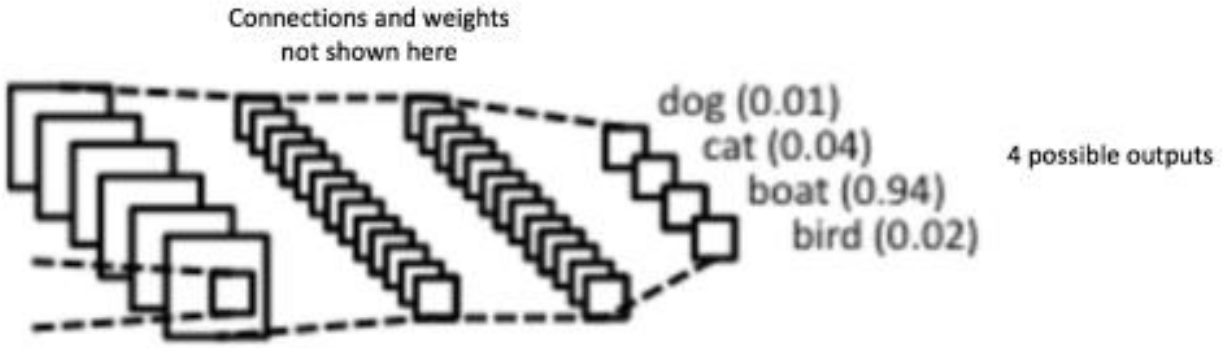
- The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer (other classifiers like SVM can also be used, but will stick to softmax in this post).
- The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer.



Ref. An Intuitive Explanation of Convolutional Neural Networks & http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML17_2.html

Classification

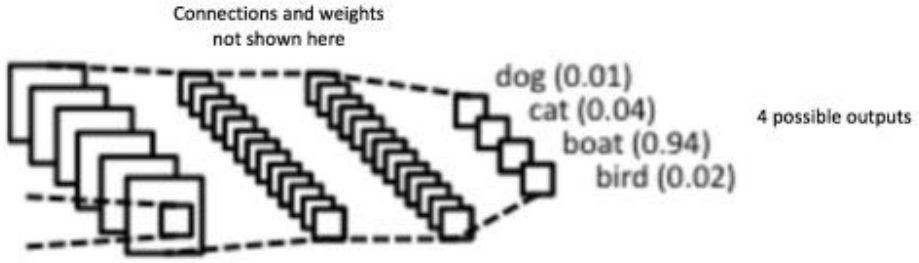
- The output from the convolutional and pooling layers represent high-level features of the input image.
- The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.
- For example, the image classification task we set out to perform has four possible outputs as shown below.



Ref. An Intuitive Explanation of Convolutional Neural Networks

Classification - Softmax

- Apart from classification, adding a fully-connected layer is also a (usually) cheap way of learning non-linear combinations of these features.
- Most of the features from convolutional and pooling layers may be good for the classification task, but combinations of those features might be even better.
- The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the Softmax as the activation function in the output layer of the Fully Connected Layer.
- The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one.



The standard (unit) softmax function $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ is defined by the formula

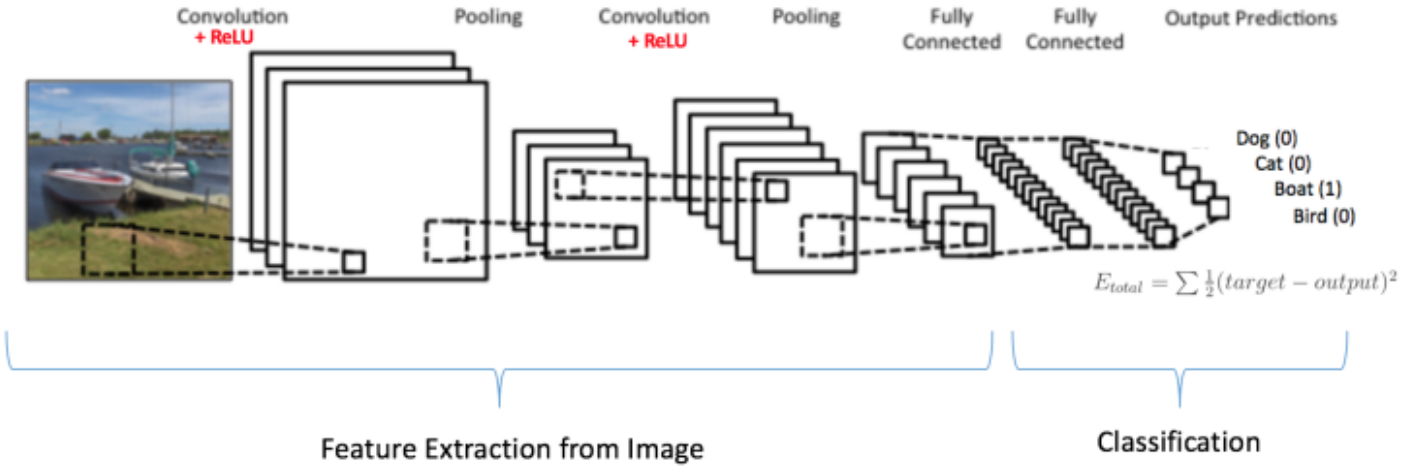
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

Ref. An Intuitive Explanation of Convolutional Neural Networks & https://en.wikipedia.org/wiki/Softmax_function

Training using backpropagation

- The Convolution + Pooling layers act as Feature Extractors from the input image while Fully Connected layer acts as a classifier.
- Since the input image is a boat, the target probability is 1 for Boat class and 0 for other three classes, i.e.

•Input Image = Boat
•Target Vector = [0, 0, 1, 0]



Ref. An Intuitive Explanation of Convolutional Neural Networks

Training using backpropagation

- **Step1:** We initialize all filters and parameters / weights with random values

- **Step2:** The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class.
  - Lets say the output probabilities for the boat image above are [0.2, 0.4, 0.1, 0.3]
  - Since weights are randomly assigned for the first training example, output probabilities are also random.

Ref. An Intuitive Explanation of Convolutional Neural Networks

Training using backpropagation

- **Step3:** Calculate the total loss at the output layer (summation over all 4 classes)
  - **Total Loss = ∑ ½ (target probability – output probability) ²**

- **Step4:** Use Backpropagation to calculate the gradients of the loss with respect to all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output loss.
  - This means that the network has learnt to classify this particular image correctly by adjusting its weights / filters such that the output loss is reduced.
  - Parameters like number of filters, filter sizes, architecture of the network etc. have all been fixed before Step 1 and do not change during training process – only the values of the filter matrix and connection weights get updated.

- **Step5:** Repeat steps 2-4 with all images in the training set.

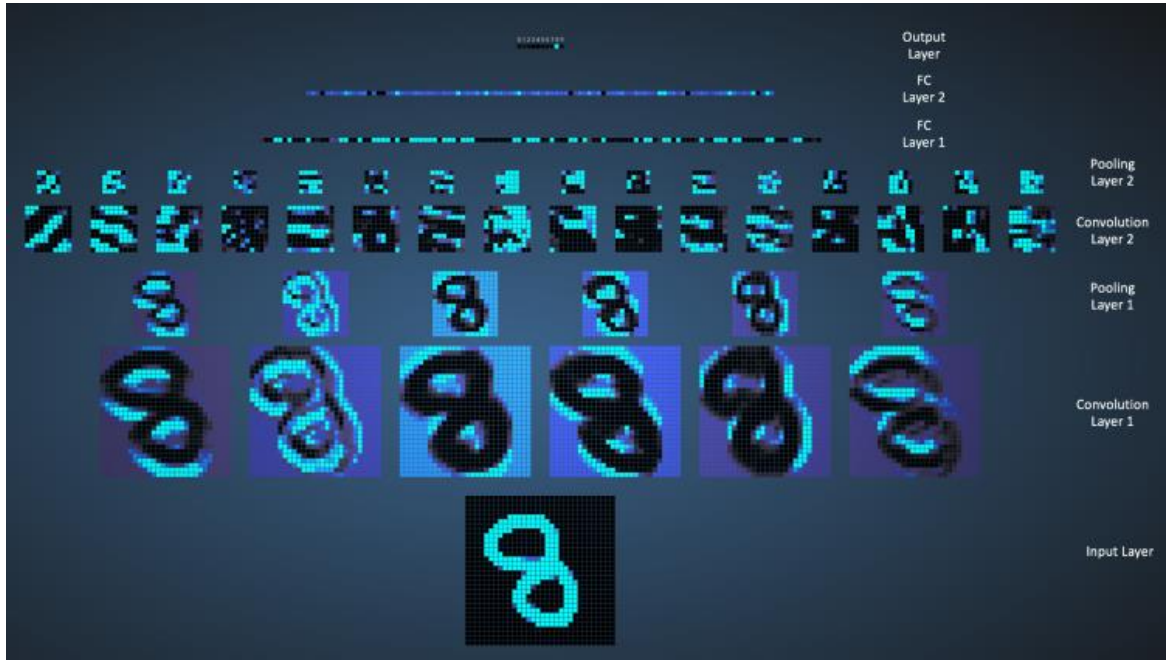
MNIST database

- The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples.
- It is a subset of a larger set available from NIST(National Institute of Standards and Technology). The digits have been size-normalized and centered in a fixed-size image.
- It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

http://yann.lecun.com/exdb/mnist/ & https://en.wikipedia.org/wiki/MNIST_database

MNIST

- We will see below how the network works for an input ‘8’.
- The input image contains 1024 pixels (32 x 32 image) and the first Convolution layer (Convolution Layer 1) is formed by convolution of six unique 5×5 (stride 1) filters with the input image. As seen, using six different filters produces a feature map of depth six.



Ref. An Intuitive Explanation of Convolutional Neural Networks & A. W. Harley in ISVC, pages 867-877, 2015

MNIST

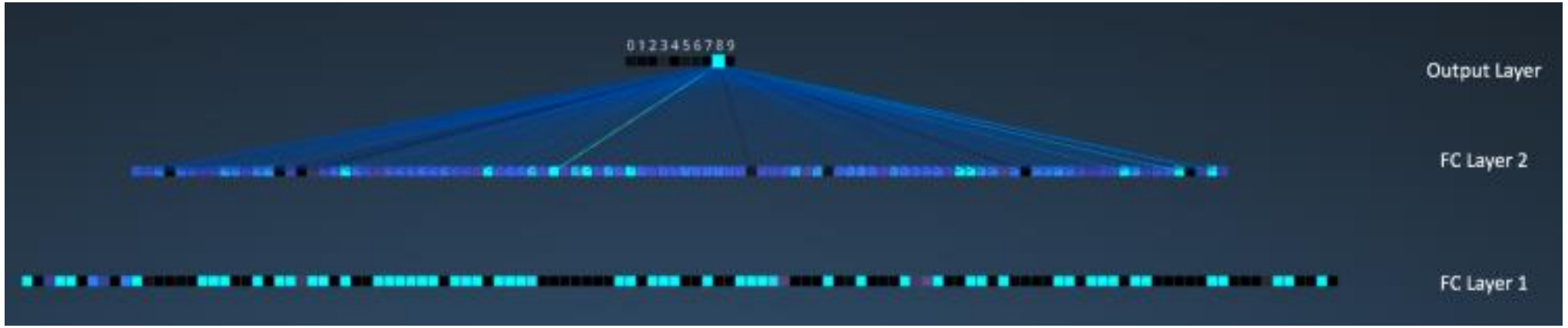
- Convolutional Layer 1 is followed by Pooling Layer 1 that does 2 × 2 max pooling (with stride 2) separately over the six feature maps in Convolution Layer 1.

- Pooling Layer 1 is followed by sixteen 5 × 5 (stride 1) convolutional filters that perform the convolution operation. This is followed by Pooling Layer 2 that does 2 × 2 max pooling (with stride 2).

Ref. An Intuitive Explanation of Convolutional Neural Networks & A. W. Harley in ISVC, pages 867-877, 2015

MNIST

- We have three fully-connected (FC) layers. There are:
  - 120 neurons in the first FC layer
  - 100 neurons in the second FC layer
  - 10 neurons in the third FC layer corresponding to the 10 digits (called the Output layer)



- Each of the 10 nodes in the output layer are connected to all 100 nodes in the 2nd Fully Connected layer.
- Note how the only bright node in the Output Layer corresponds to '8' – this means that the network correctly classifies our handwritten digit (brighter node denotes that the output from it is higher, i.e. 8 has the highest probability among all other digits).

Other ConvNet architectures

- Convolutional Neural Networks have been around since early 1990s. We discussed the LeNet above which was one of the very first convolutional neural networks. Some other influential architectures are listed below:
  - LeNet (1990s).
  - 1990s to 2012: In the years from late 1990s to early 2010s convolutional neural network were in incubation. As more and more data and computing power became available, tasks that convolutional neural networks could tackle became more and more interesting.
  - AlexNet (2012) – In 2012, Alex Krizhevsky (and others) released AlexNet which was a deeper and much wider version of the LeNet and won by a large margin the difficult ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. It was a significant breakthrough with respect to the previous approaches and the current widespread application of CNNs can be attributed to this work.
  - ZF Net (2013) – The ILSVRC 2013 winner was a Convolutional Network from Matthew Zeiler and Rob Fergus. It became known as the ZFNet (short for Zeiler & Fergus Net). It was an improvement on AlexNet by tweaking the architecture hyperparameters.

Ref. An Intuitive Explanation of Convolutional Neural Networks

Other ConvNet architectures

- GoogLeNet (2014) – The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al. from Google. Its main contribution was the development of an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M).
- VGGNet (2014) – The runner-up in ILSVRC 2014 was the network that became known as the VGGNet. Its main contribution was in showing that the depth of the network (number of layers) is a critical component for good performance.
- ResNets (2015) – Residual Network developed by Kaiming He (and others) was the winner of ILSVRC 2015. ResNets are currently by far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice (as of May 2016).
- DenseNet (August 2016) – Recently published by Gao Huang (and others), the Densely Connected Convolutional Network has each layer directly connected to every other layer in a feed-forward fashion. The DenseNet has been shown to obtain significant improvements over previous state-of-the-art architectures on five highly competitive object recognition benchmark tasks. Check out the Torch implementation here.
- ……

Ref. An Intuitive Explanation of Convolutional Neural Networks

Next class

- Cross validation
- Overfitting
- Batch normalization
- ……