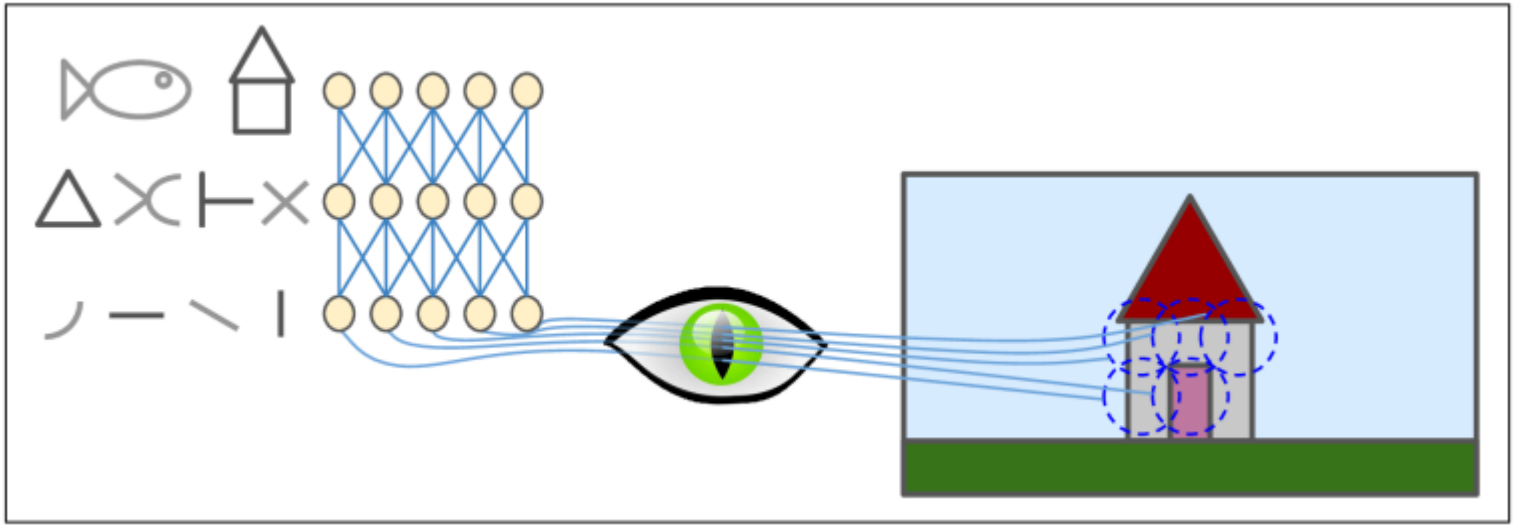
DL in Applied Mathematics

Lecture 10: Convolutional Neural Networks:
Part 2

Outline

- Brief history of CNN
- Basic structure of CNN
- CNN architecture
- CNN architecture examples

Visual Cortex



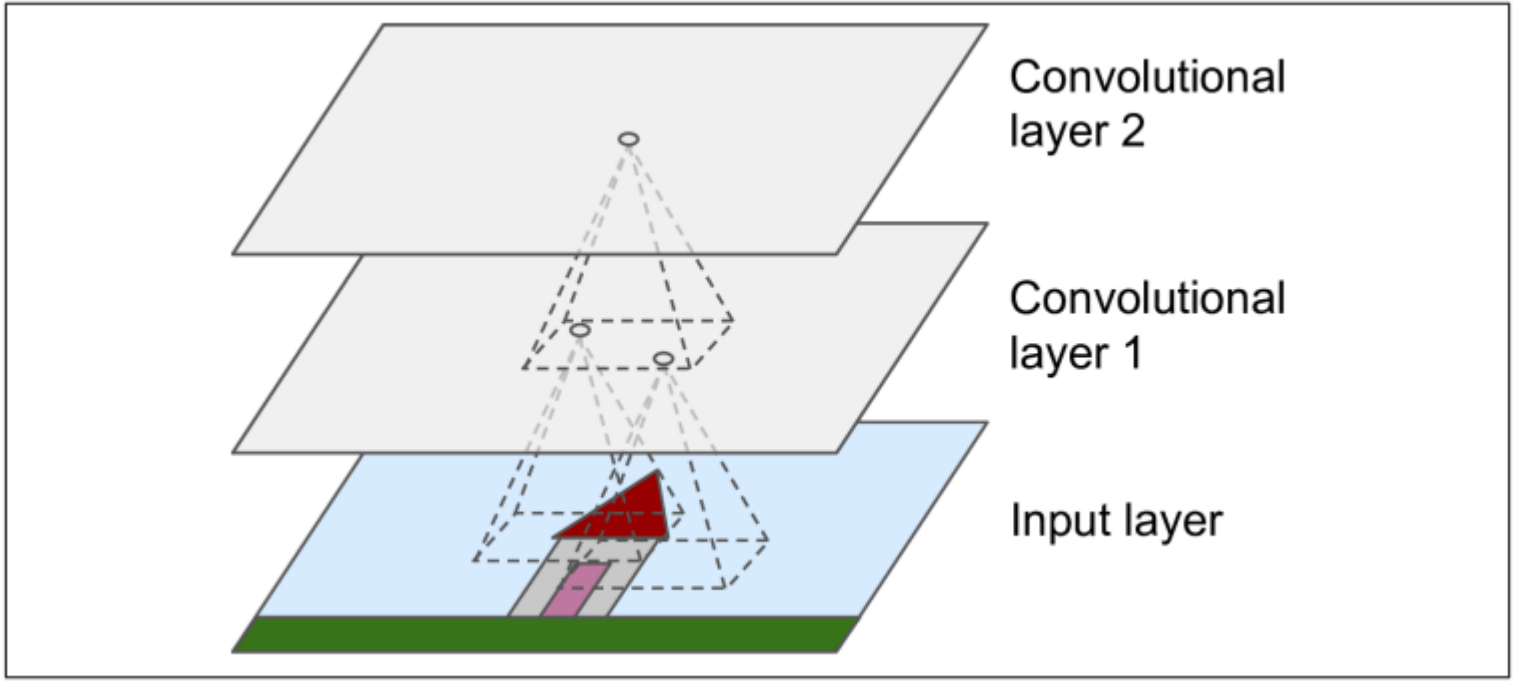
Neurons have small local receptive field and large receptive fields:
For example: two neurons same receptive field but vary orientations (David H. Hubel and Torsten Wiesel)

History of CNN

- The study of visual cortex inspired neocognitron (Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position)
- LeNet-5 in 1998 paper by Yann LeCun and etc.

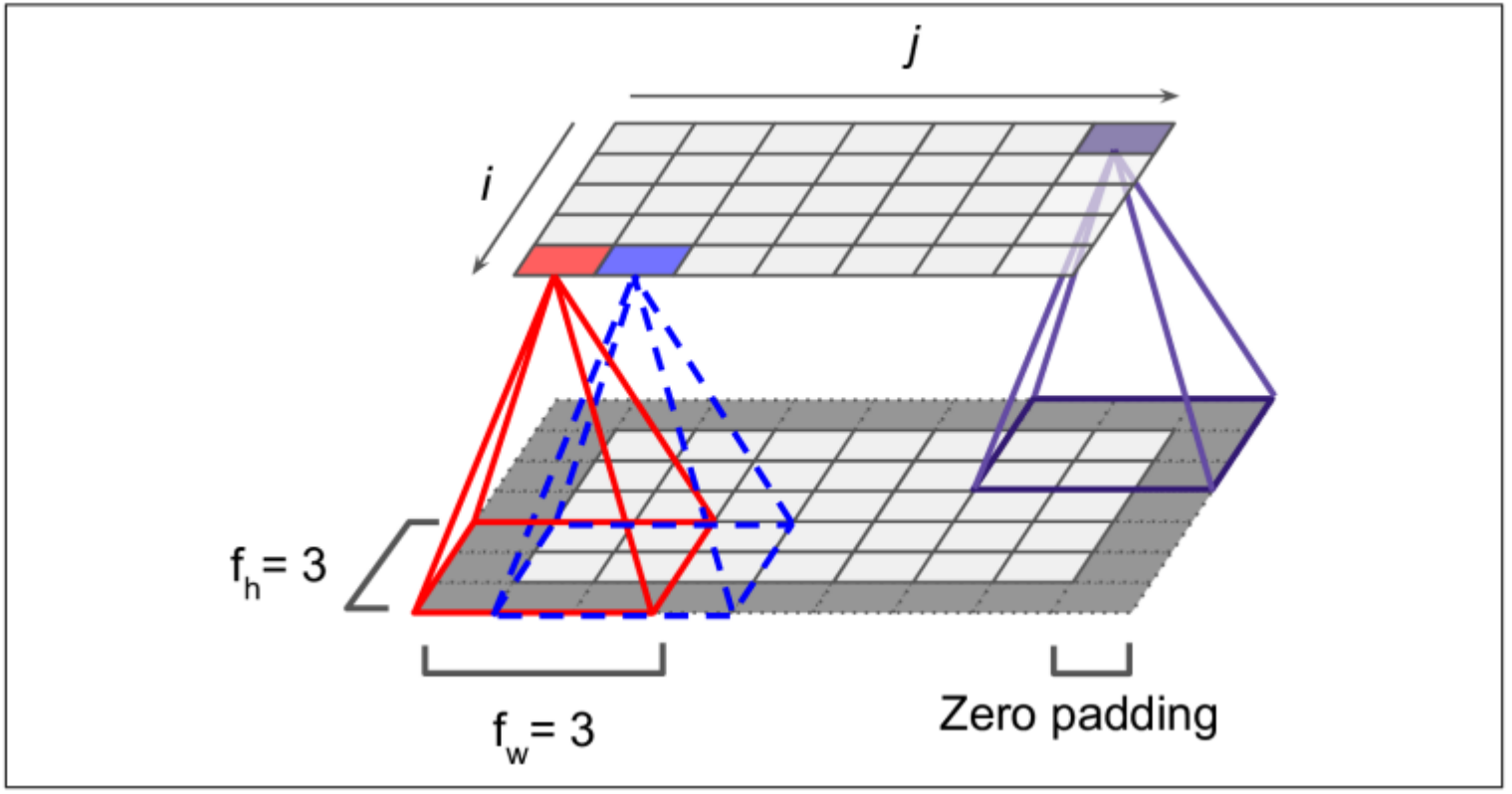
• Basic structure of CNN

Convolutional layer



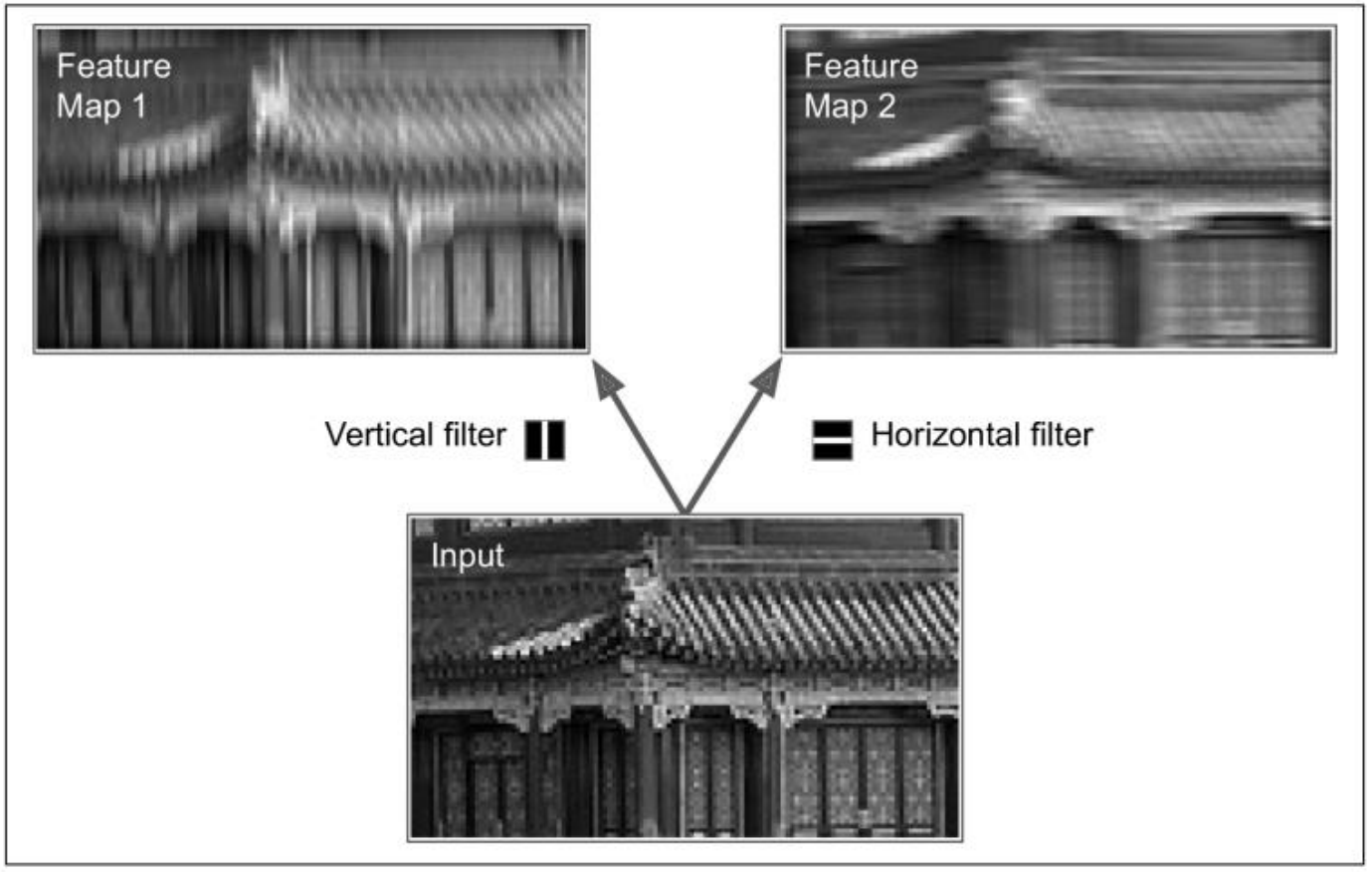
• Basic structure of CNN

Convolutional layer



• Basic structure of CNN

Filters

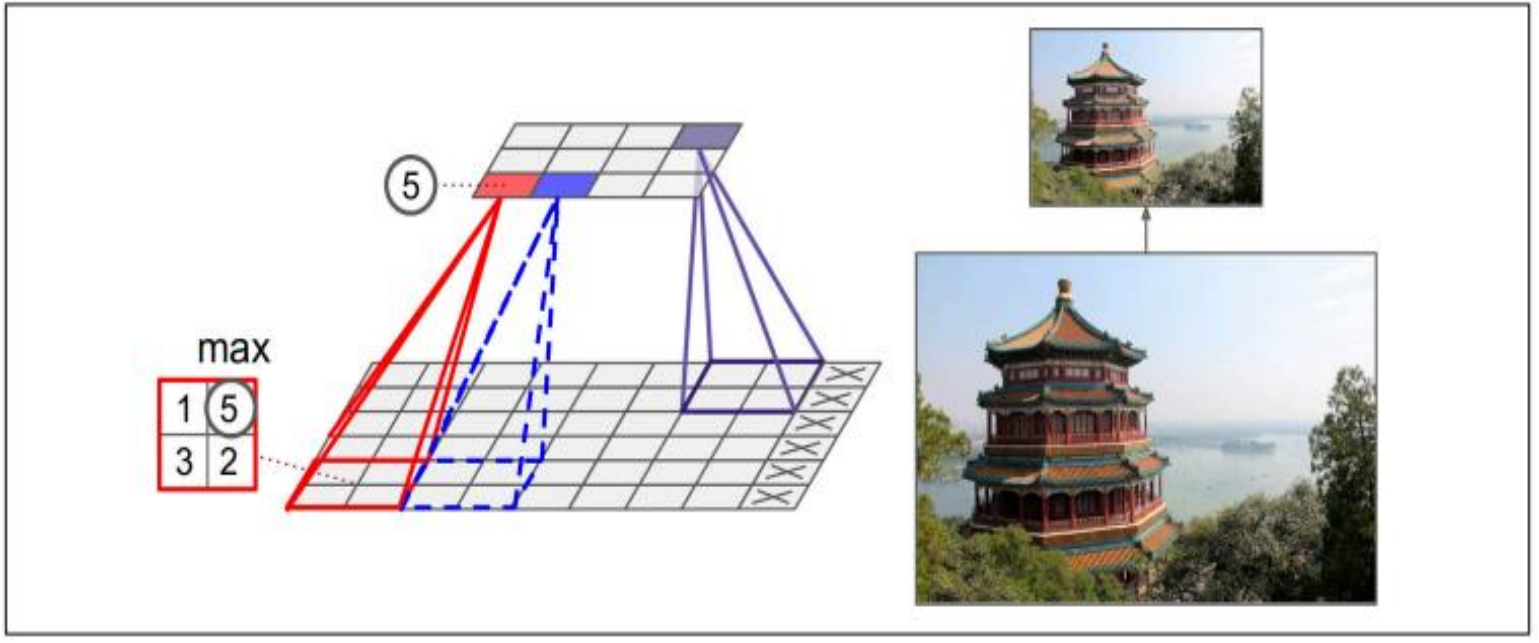


How to decide the number of convolutional layers and filters?

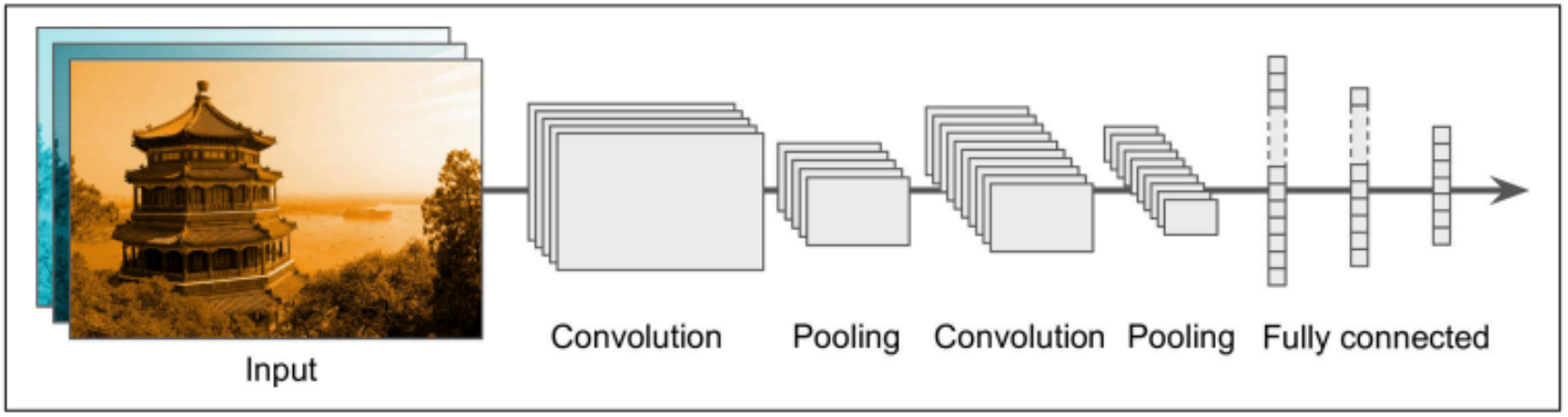
- Should be used initially fewer filters and gradually increase and monitor the error rate to see how it is varying
- Very small filter sizes will capture very fine details of the image
- Very big filter sizes will leave out minute details from the image

• Basic structure of CNN

Pooling layer



- CNN architecture



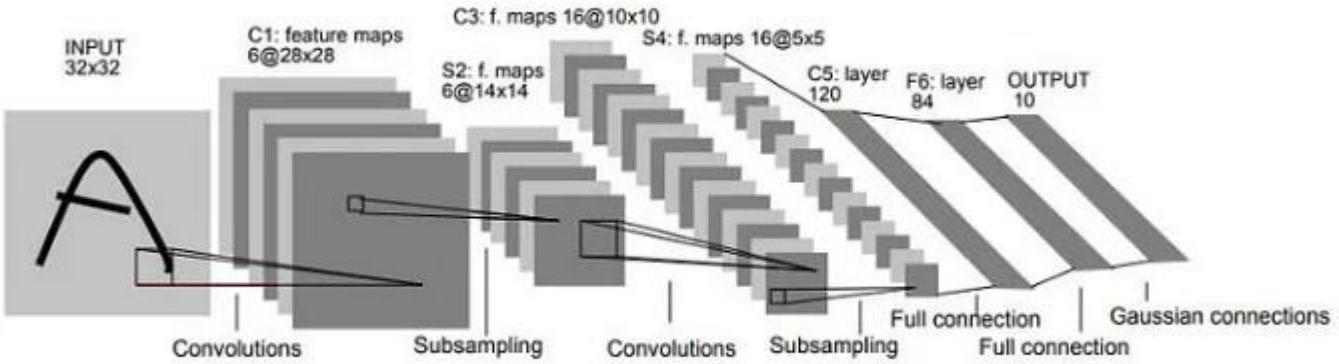
Type of CNN

- Image classification
- Object detection
- Object tracking
- Semantic segmentation and so on

Image classification examples

- LeNet – 5 (1998)
- AlexNet (2012)
- GoogLeNet (2014)
- ResNet (2015)

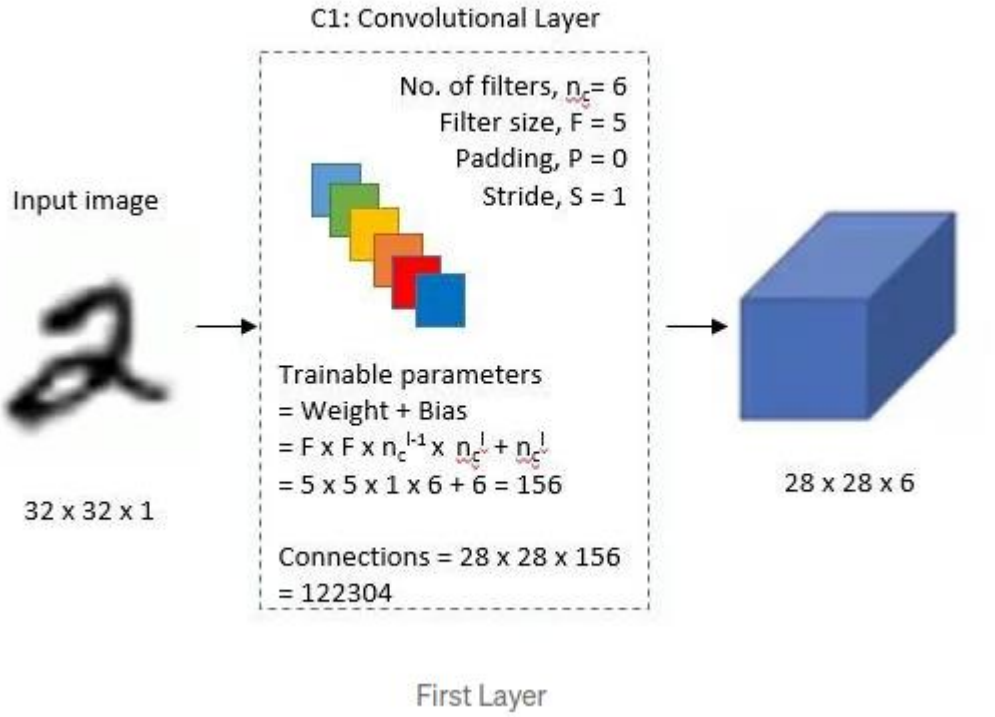
LeNet-5 is the “Hello World” in the domain CNN



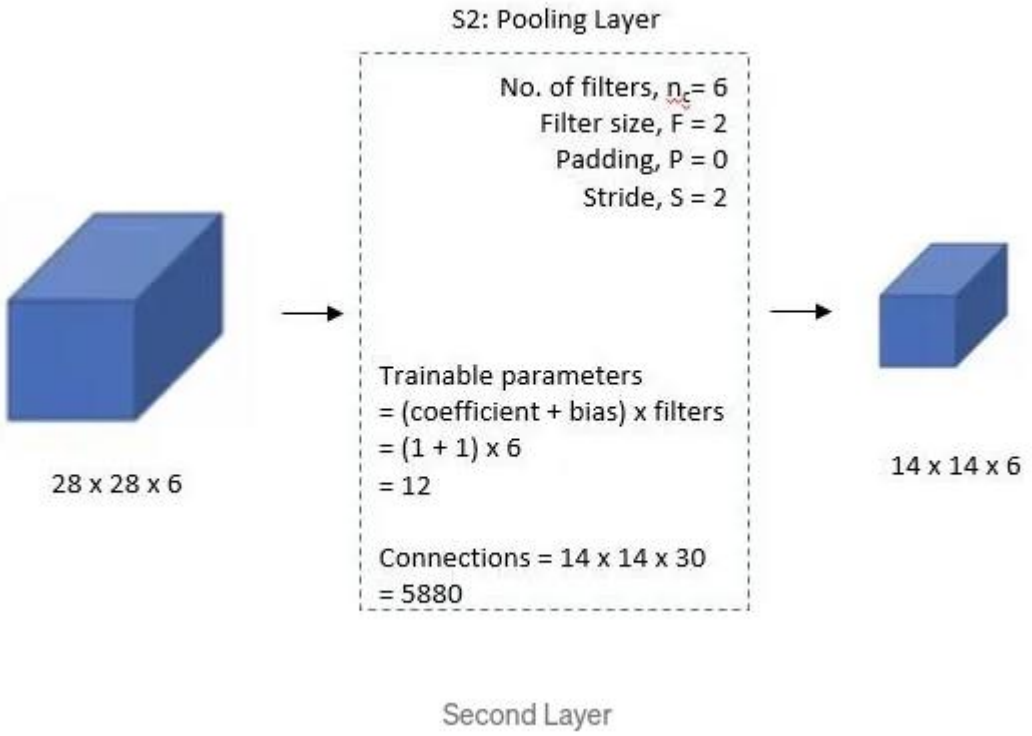
LeNet-5 Architecture

Three convolutional layers, two subsampling layers, and two fully linked layers make up the layer composition

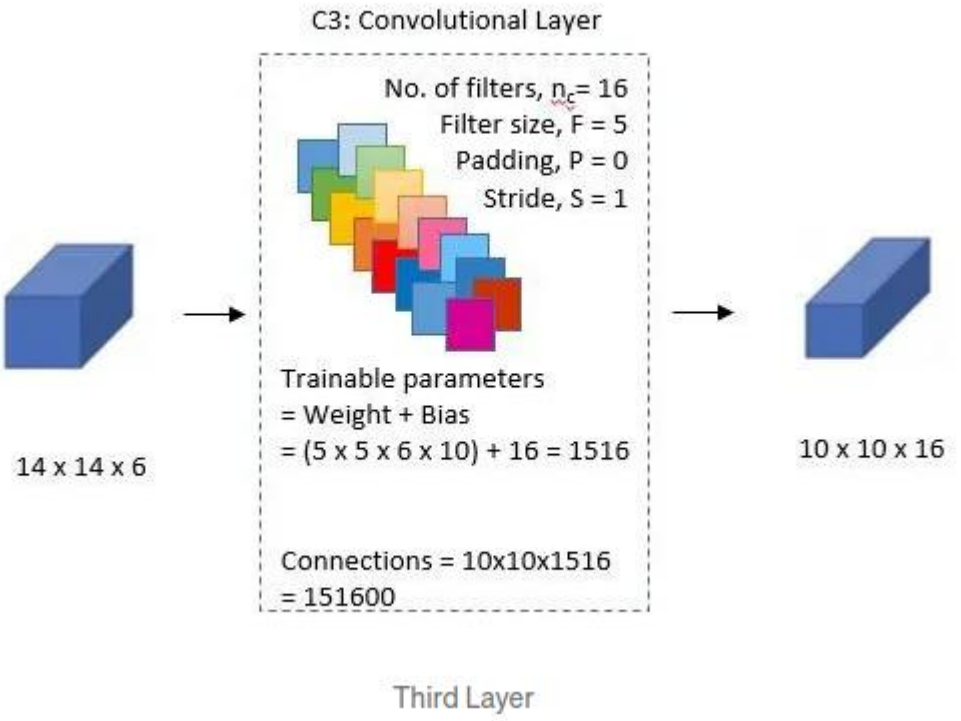
LeNet -5


C1: Convolutional Layer
No. of filters, n_c= 6
Filter size, F = 5
Padding, P = 0
Stride, S = 1
Input image
Trainable parameters
= Weight + Bias
= F x F x n_c^{l-1} x n_c^l + n_c^l
= 5 x 5 x 1 x 6 + 6 = 156
Connections = 28 x 28 x 156
= 122304
32 x 32 x 1
28 x 28 x 6
First Layer

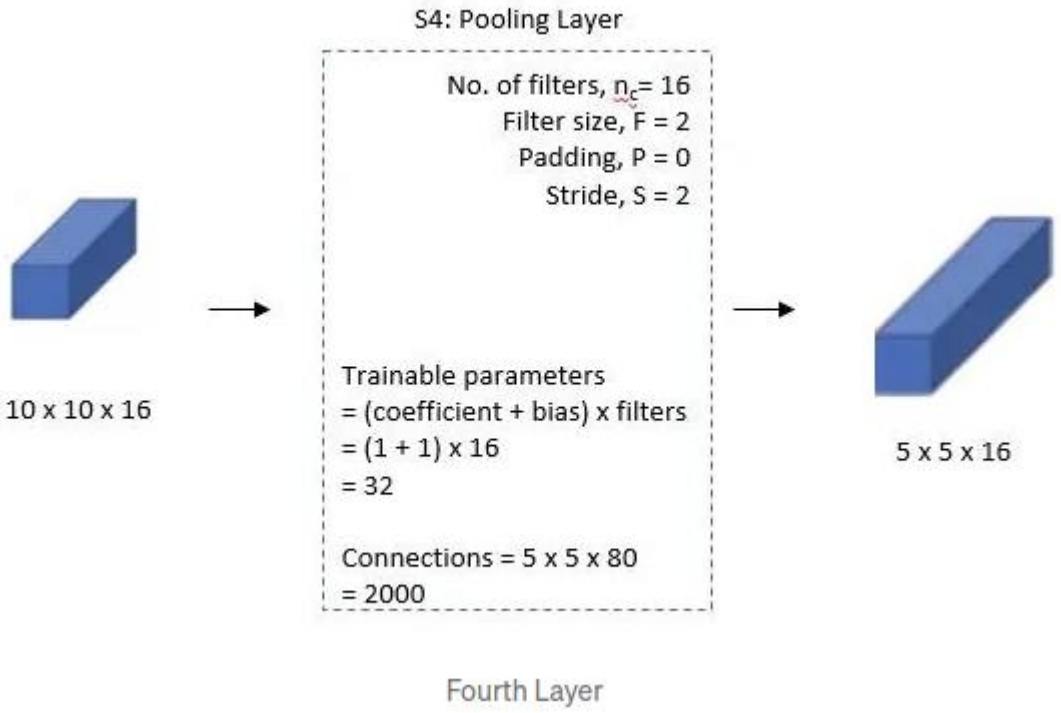
LeNet -5


S2: Pooling Layer
No. of filters, n_c= 6
Filter size, F = 2
Padding, P = 0
Stride, S = 2
Trainable parameters
= (coefficient + bias) x filters
= (1 + 1) x 6
= 12
Connections = 14 x 14 x 30
= 5880
28 x 28 x 6
14 x 14 x 6
Second Layer

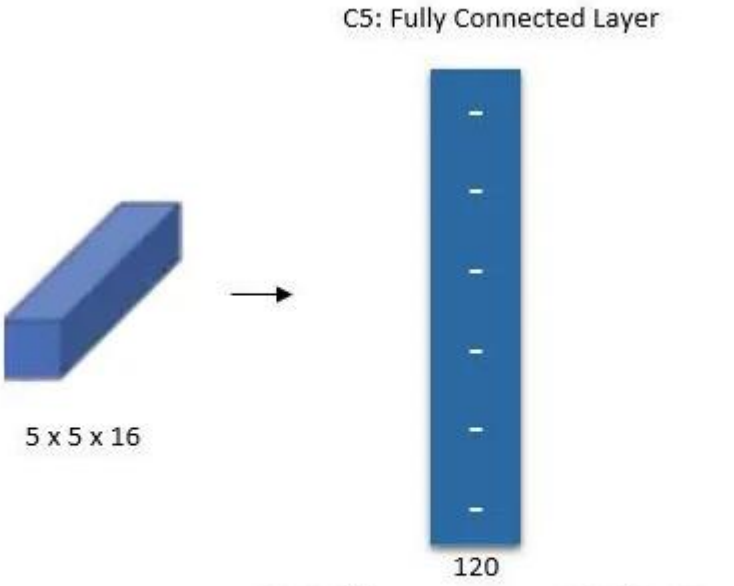
LeNet -5


C3: Convolutional Layer
No. of filters, n_c= 16
Filter size, F = 5
Padding, P = 0
Stride, S = 1
Trainable parameters
= Weight + Bias
= (5 x 5 x 6 x 10) + 16 = 1516
Connections = 10x10x1516
= 151600
14 x 14 x 6
10 x 10 x 16
Third Layer

LeNet -5

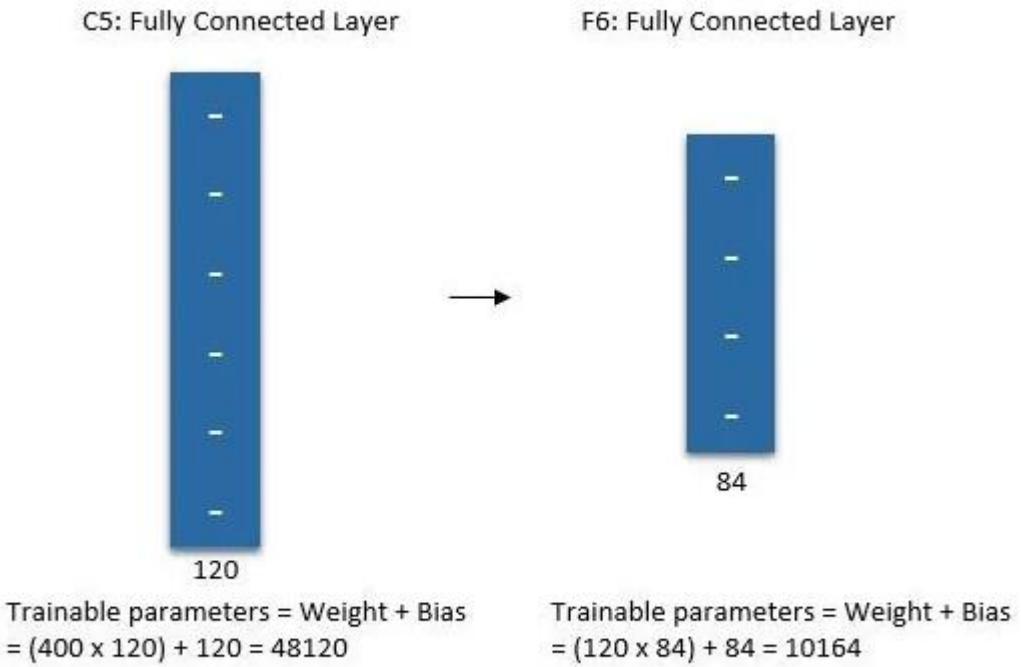

S4: Pooling Layer
No. of filters, n_c= 16
Filter size, F = 2
Padding, P = 0
Stride, S = 2
10 x 10 x 16
Trainable parameters
= (coefficient + bias) x filters
= (1 + 1) x 16
= 32
Connections = 5 x 5 x 80
= 2000
5 x 5 x 16
Fourth Layer

LeNet -5

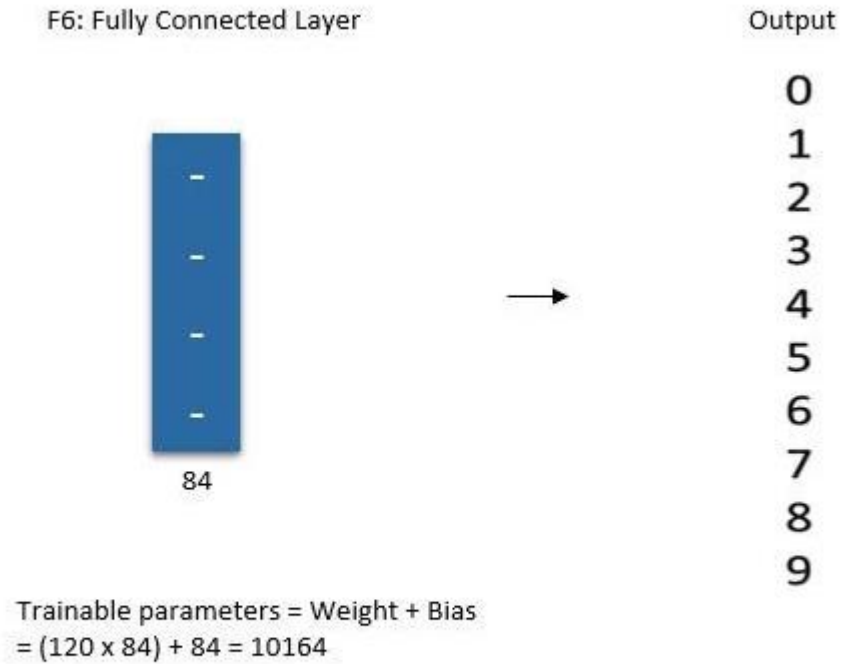


Fifth Layer

LeNet -5


C5: Fully Connected Layer
F6: Fully Connected Layer
120
84
Trainable parameters = Weight + Bias
= (400 x 120) + 120 = 48120
Trainable parameters = Weight + Bias
= (120 x 84) + 84 = 10164
Sixth Layer

LeNet -5


F6: Fully Connected Layer
Output
84
0
1
2
3
4
5
6
7
8
9
Trainable parameters = Weight + Bias
= (120 x 84) + 84 = 10164
Output Layer

LeNet -5

```
model = keras.Sequential()

model.add(layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu',
input_shape=(32,32,1)))
model.add(layers.AveragePooling2D())

model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model.add(layers.AveragePooling2D())

model.add(layers.Flatten())

model.add(layers.Dense(units=120, activation='relu'))

model.add(layers.Dense(units=84, activation='relu'))

model.add(layers.Dense(units=10, activation = 'softmax'))
```

The s**oftmax activation** function transforms the raw outputs of the neural network into a vector of probabilities

ReLU function increases the complexity of the neural network by introducing non-linearity, which allows the network to learn more complex representations of the data. The ReLU function is defined as f(x) = max(0, x), which sets all negative values to zero

Object detection

- R CNN
- Fast R CNN
- Faster R CNN
- YOLO (now version 9)