DATA ANALYSIS AND VISUALIZATION

INSTRUCTOR: UMME AMMARAH

CONVOLUTIONAL NEURAL NETWORK (CNN)

COMPUTER VISION PROBLEMS

Image Classification

Cat? (0/1)

64x64

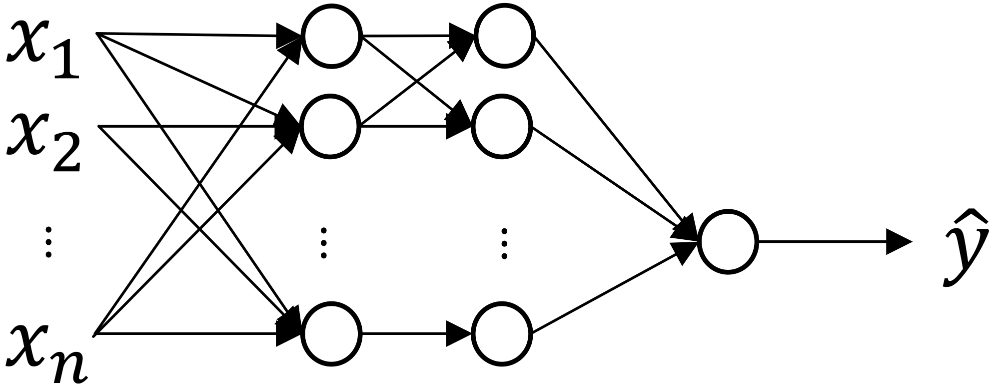
Neural Style Transfer

Object detection

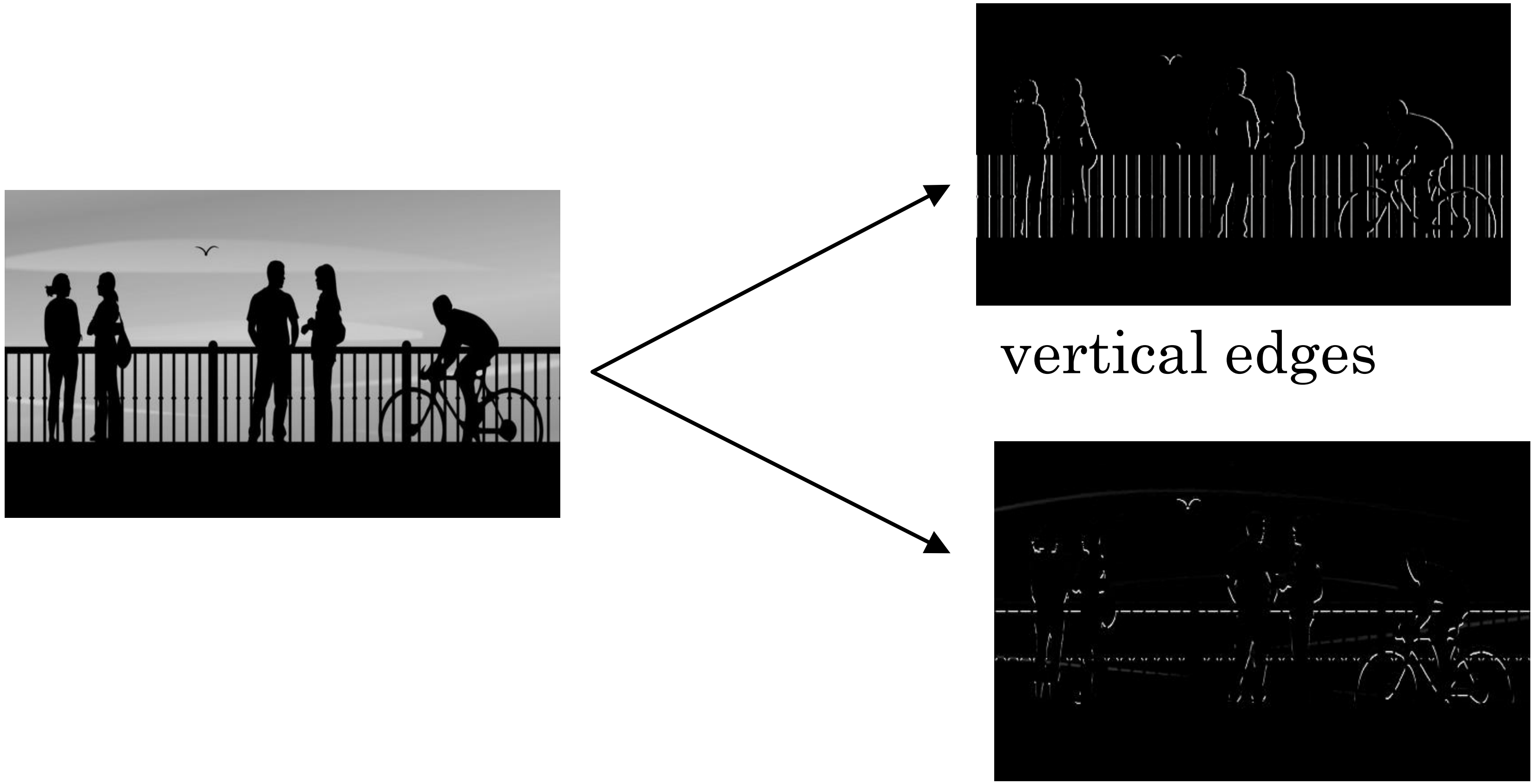
DEEP LEARNING ON LARGE IMAGES

64x64

→ Cat? (0/1)



COMPUTER VISION PROBLEM



vertical edges

horizontal edges

VERTICAL EDGE DETECTION

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

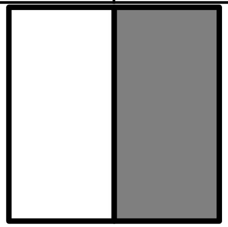
*

=

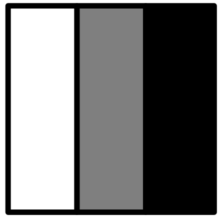
0	-2	-4	-7
-3	-2	-3	-16

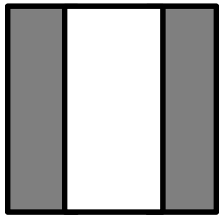
VERTICAL EDGE DETECTION

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$



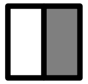
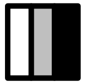
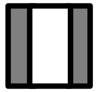
*



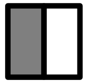
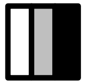
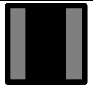


VERTICAL EDGE DETECTION EXAMPLES

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \end{bmatrix}$$

VERTICAL AND HORIZONTAL EDGE DETECTION

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

Vertical

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

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

$*$

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

$=$

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

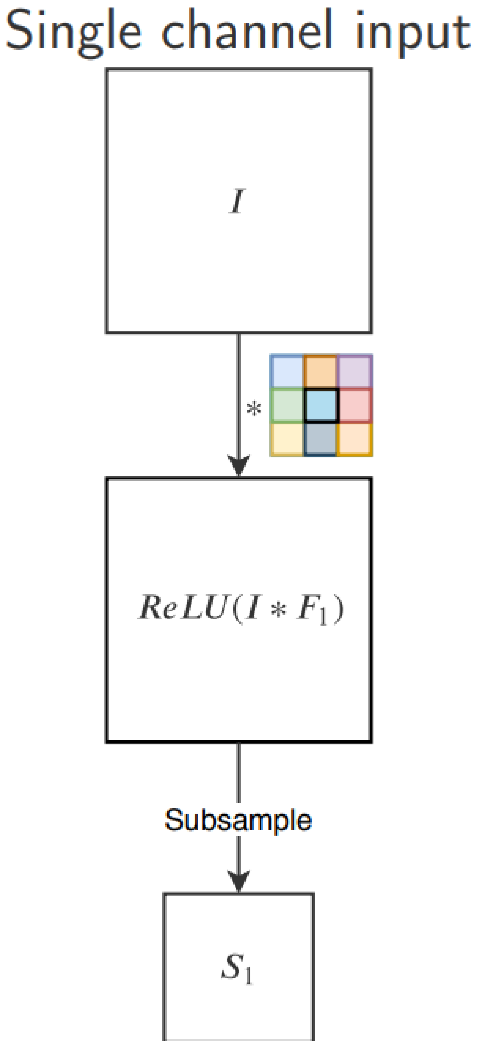
LEARNING TO DETECT EDGES

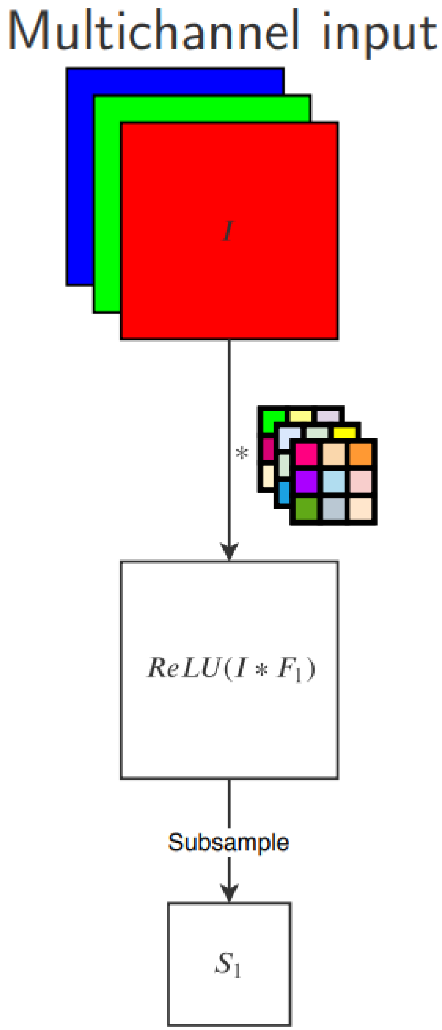
1	0	-1
1	0	-1
1	0	-1

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

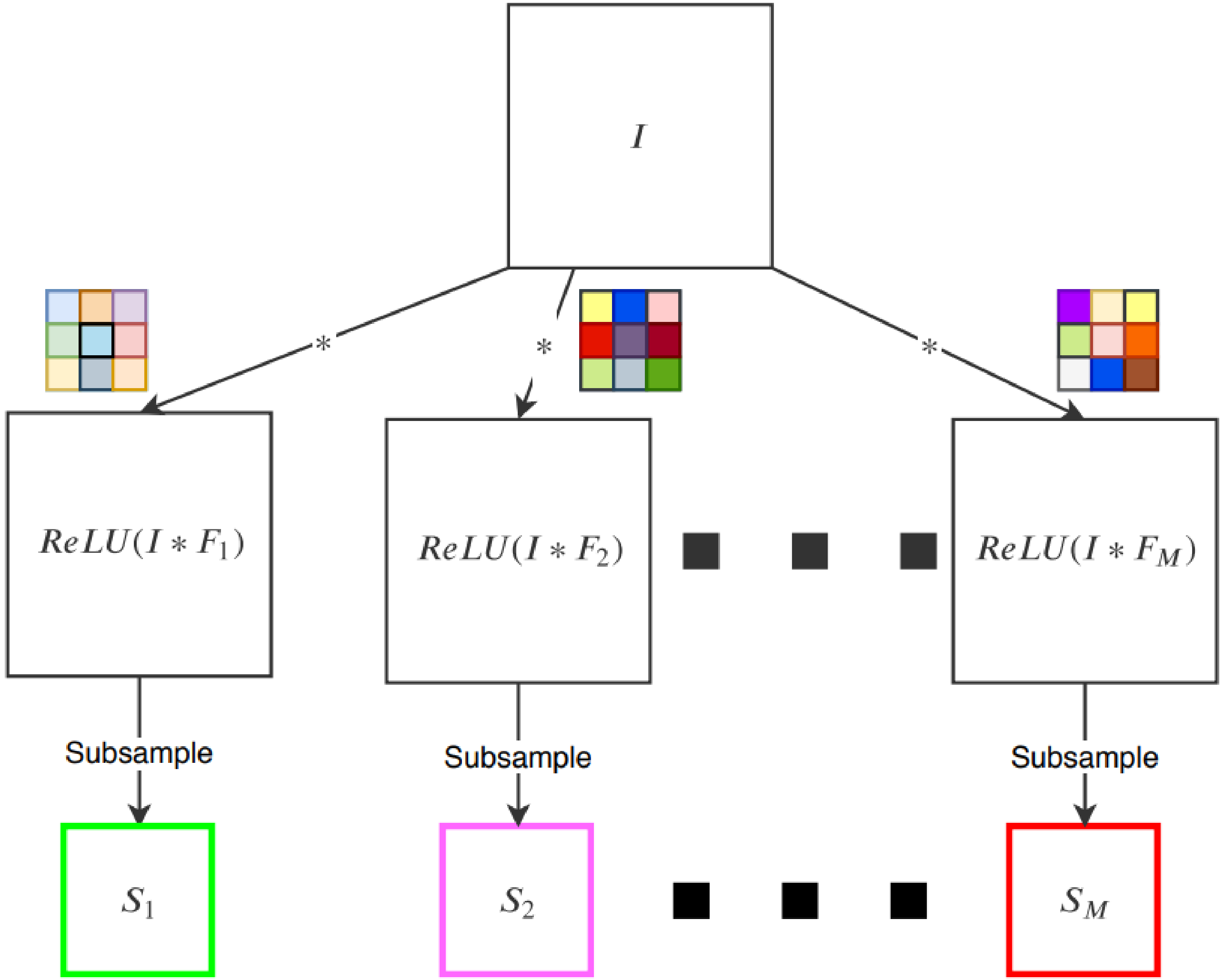
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

BUILDING BLOCKS OF CNNS

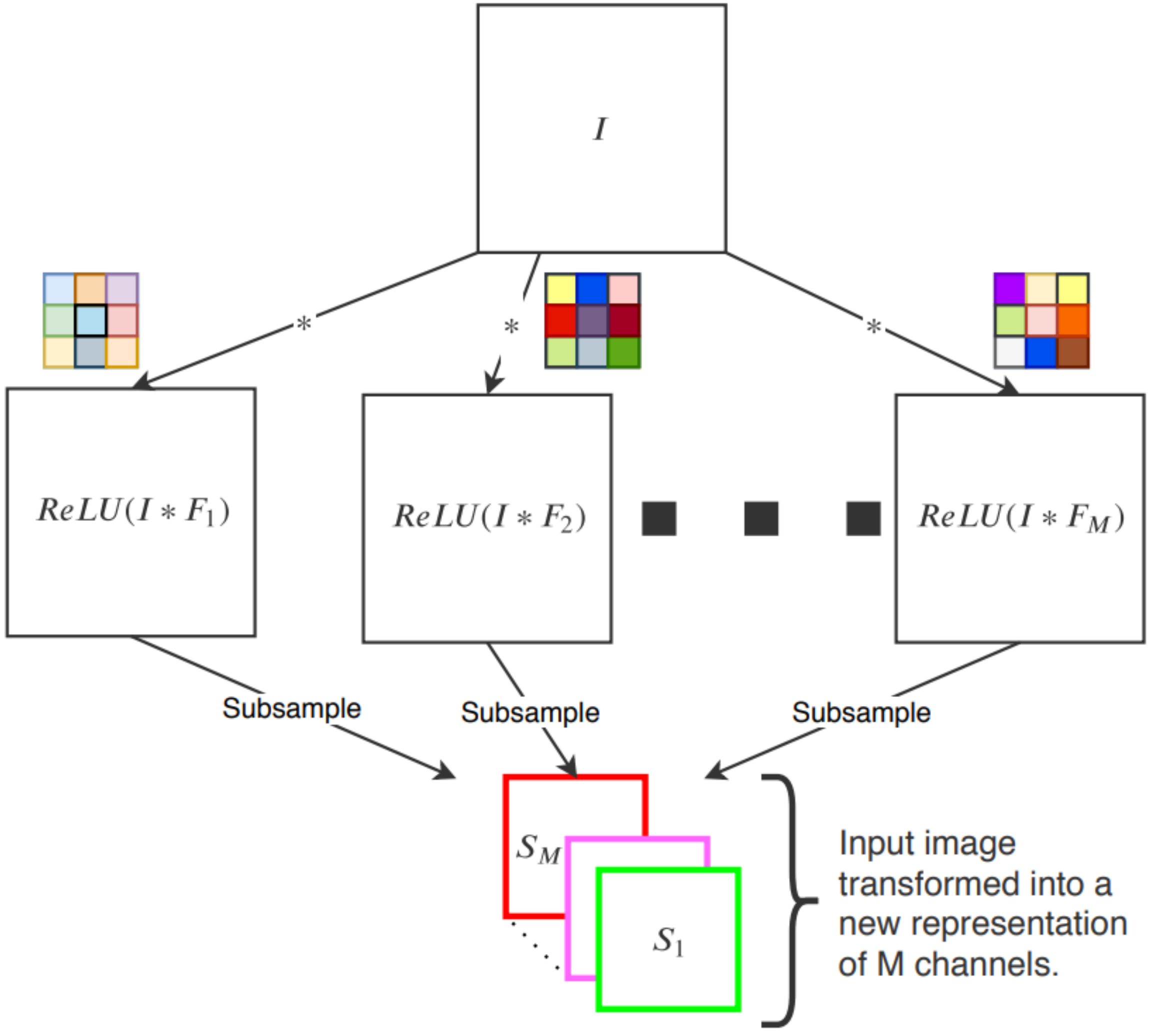




BUILDING BLOCKS OF CNN



BUILDING BLOCKS OF CNN

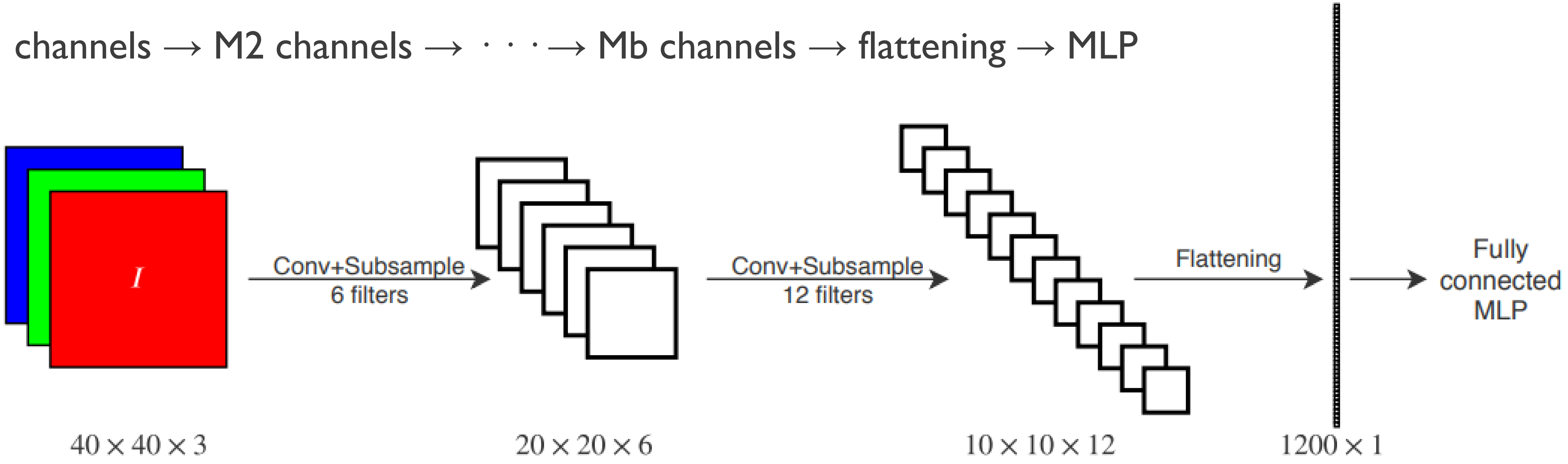


TYPES OF LAYER IN A CONVOLUTIONAL NETWORK:

- Convolution
- Pooling
- Fully connected

CNN

- Convolution by M filters produces M channels.
- They represent an M-channel transformation of the input image I.
- This M-channel image can now be transformed further via additional convolution filters.
- Convolution-subsampling block can be repeated multiple times.
- I $\rightarrow$ M1 channels $\rightarrow$ M2 channels $\rightarrow \cdots \rightarrow$ Mb channels $\rightarrow$ flattening $\rightarrow$ MLP



NN VS CNN

- Global receptive fields due to being fully connected.
- Separate weights for each neuron.

- *Local receptive fields* due to being sparsely connected.
- *Shared weights* among different neurons.
- *Subsampling* of each layer's outputs.

- Receptive field of a neuron consists of previous layer neurons that it is connected to (or looking at).

CONVOLUTIONAL LAYER

- Consists of multiple arrays of neurons. Each such array is called a slice or more accurately feature map.
- Each neuron in a feature map
  - is connected to only few neurons in the previous layer, but
  - uses the same weight values as all other neurons in that feature map.
- So within a feature map, we have both
  - local receptive fields, and
  - shared weights.

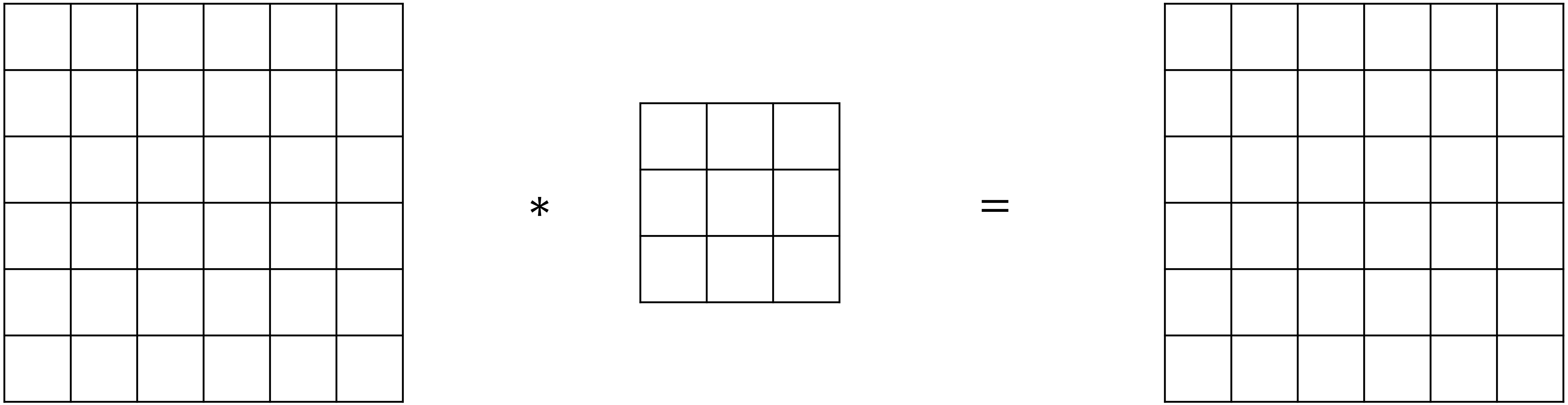
CONVOLUTIONAL LAYER

- Example: A feature map may have
  - 100 neurons placed in a 10×10 array, with
  - each neuron getting input from a 5×5 patch of neurons in the previous layer (receptive field), and
  - the same $26 (= 5 \times 5 + 1)$ weights shared between these 100 neurons.
- Viewed as detectors, all 100 neurons detect the same 5×5 pattern but at different locations of the previous layer.
- Different feature maps will learn to detect different kinds of patterns.
  - For example, one feature map might learn to detect horizontal edges while others might learn to detect vertical or diagonal edges and so on.

CONVOLUTIONAL LAYER

- To compute activations of the 100 neurons, a dot-product is computed between the same shared weights and different 5×5 patches of previous layer neurons.
- This is equivalent to **sliding a window of weights over the previous layer and computing the dot-product at each location of the window**.
- Therefore, activations of the feature map neurons are computed via *convolution* of the previous layer with a *kernel* comprising the shared weights. Hence the name of this layer.

PADDING

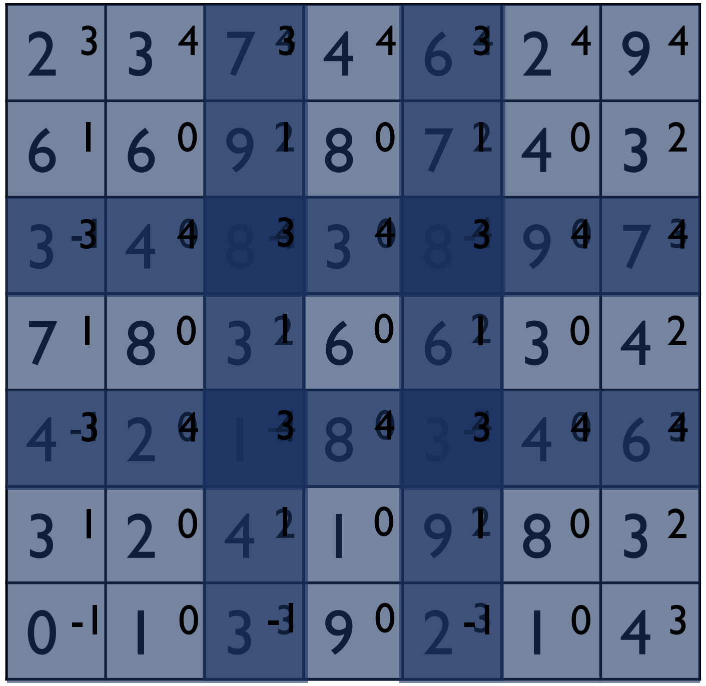


VALID AND SAME CONVOLUTIONS

"Valid":

"Same": Pad so that output size is the same as the input size.

STRIDED CONVOLUTION



*

3	4	4
1	0	2
-1	0	3

=

SUMMARY OF CONVOLUTIONS

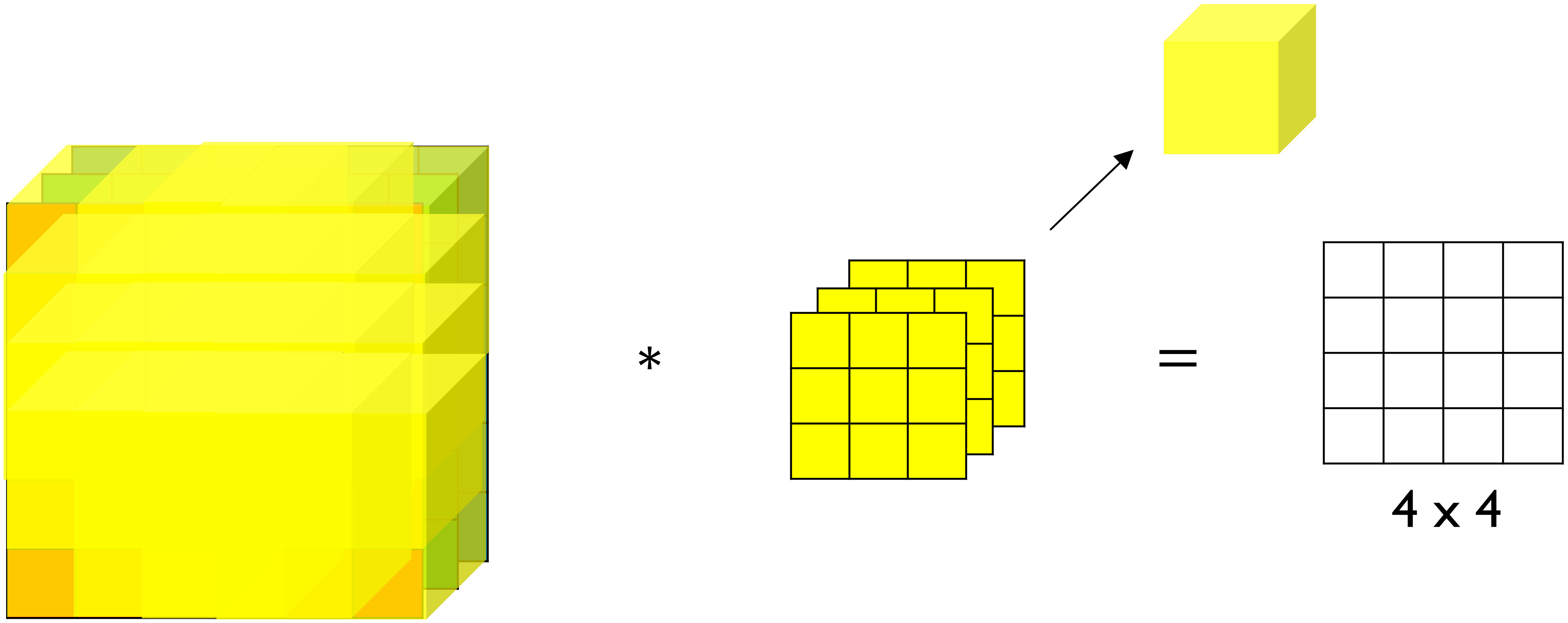
$n \times n$ image $\quad f \times f$ filter

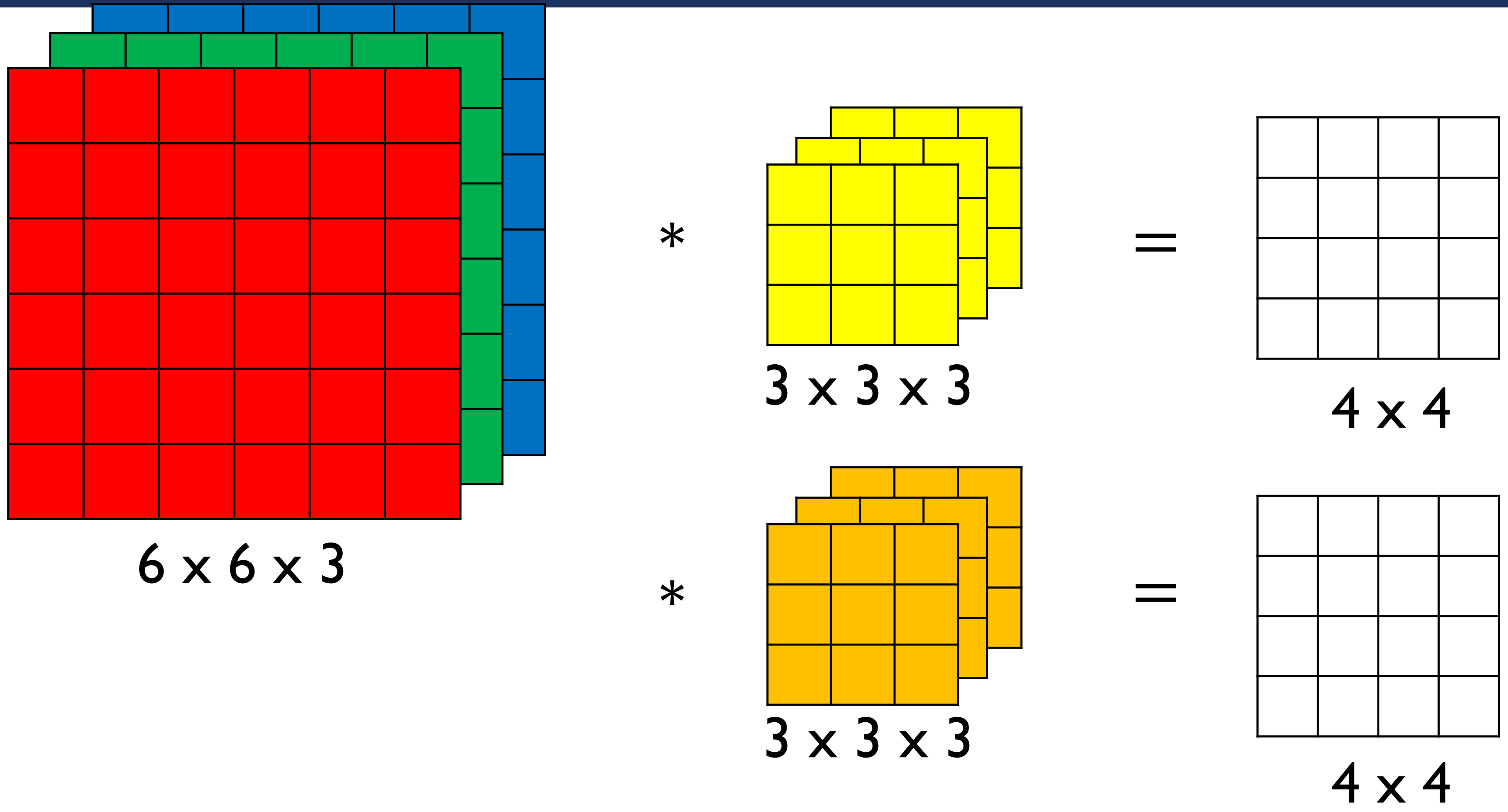
padding p $\quad$ stride s

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

CONVOLUTIONS ON RGB IMAGES

CONVOLUTIONS ON RGB IMAGE





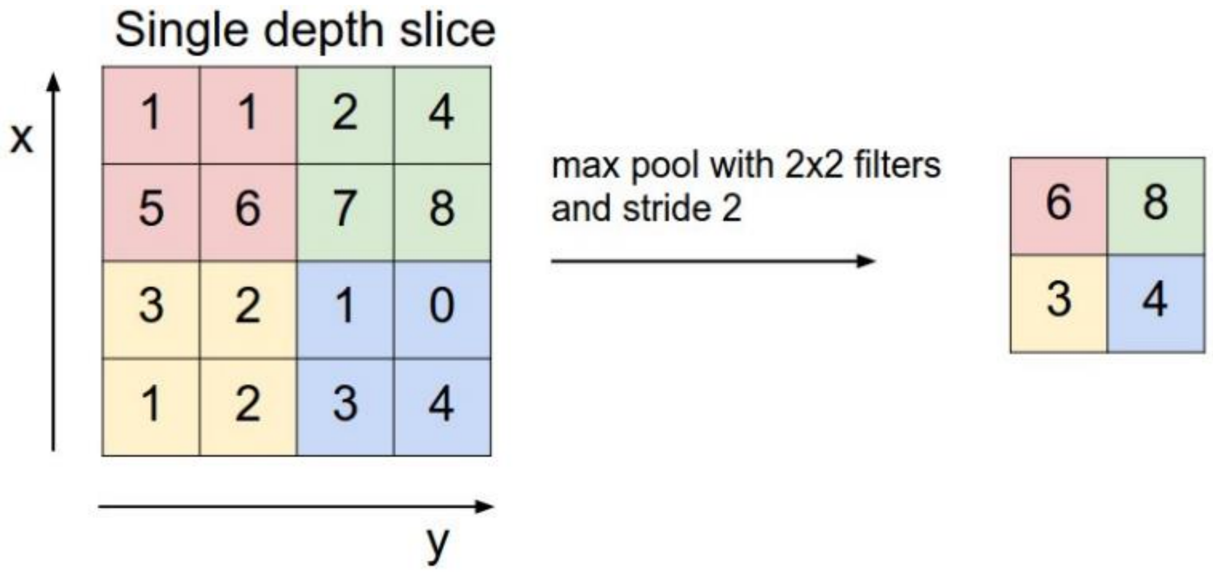
6 x 6 x 3
*
3 x 3 x 3
=
4 x 4
*
3 x 3 x 3
=
4 x 4

SUBSAMPLING LAYER

- Reduces the spatial dimensions of the previous layer by downsampling. Also called ***pooling layer***.
- No adjustable weights. Just a fixed down sampling procedure.
- Reduces computations in subsequent layers.
- Reduces number of weights in subsequent layers. This reduces overfitting
- Note that pooling with larger receptive fields discards too much data.
- Subsampling layer can be skipped if convolution layers uses stride>1 since that also produces a subsampled output.

POOLING

- Options: From non-overlapping 2×2 patches
  - pick top-left (standard downsampling by factor 2)
  - pick average (*mean-pooling*)
  - pick maximum (*max-pooling*)
  - pick randomly (*stochastic-pooling*)

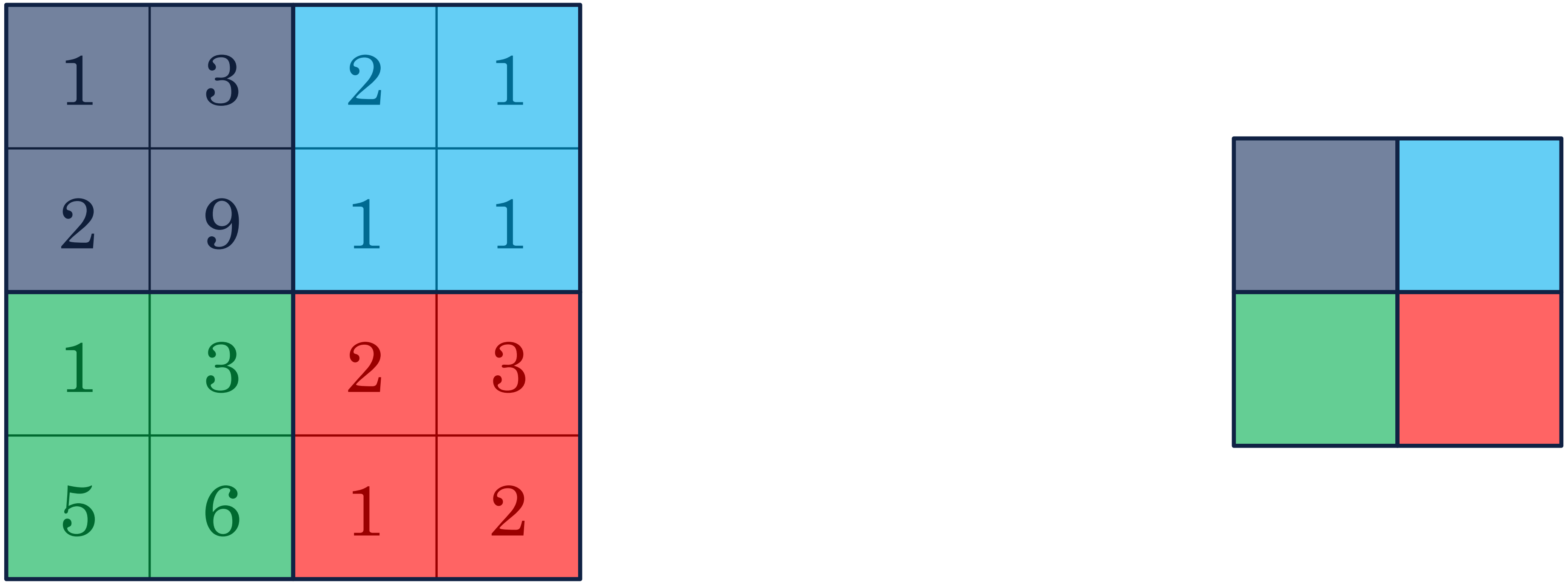


POOLING LAYER

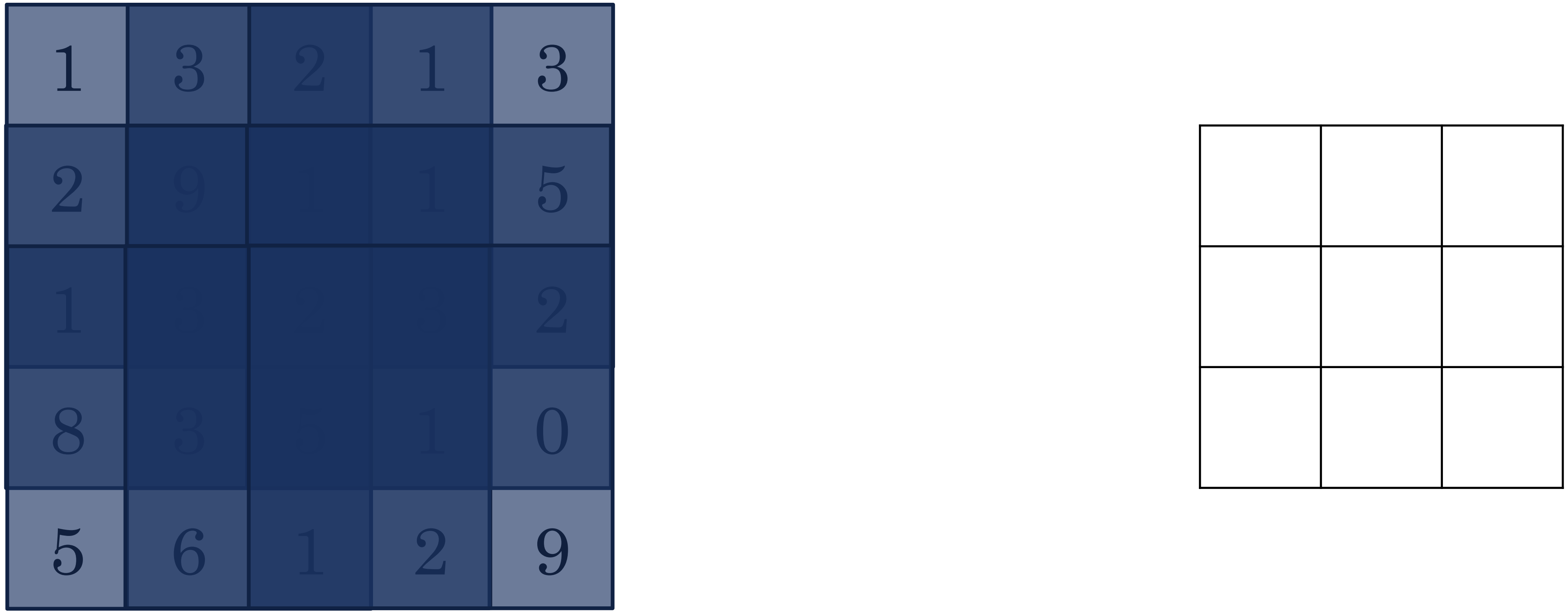
A pooling layer

- with $F \times F$ receptive field and stride S,
- "looking at" a $W_1 \times H_1 \times D_1$ input volume,
- produces a $W_2 \times H_2 \times D_2$ output volume, where
  - $W_2 = \frac{W_1 - F}{S} + 1$
  - $H_2 = \frac{H_1 - F}{S} + 1$
  - $D_2 = D_1$.

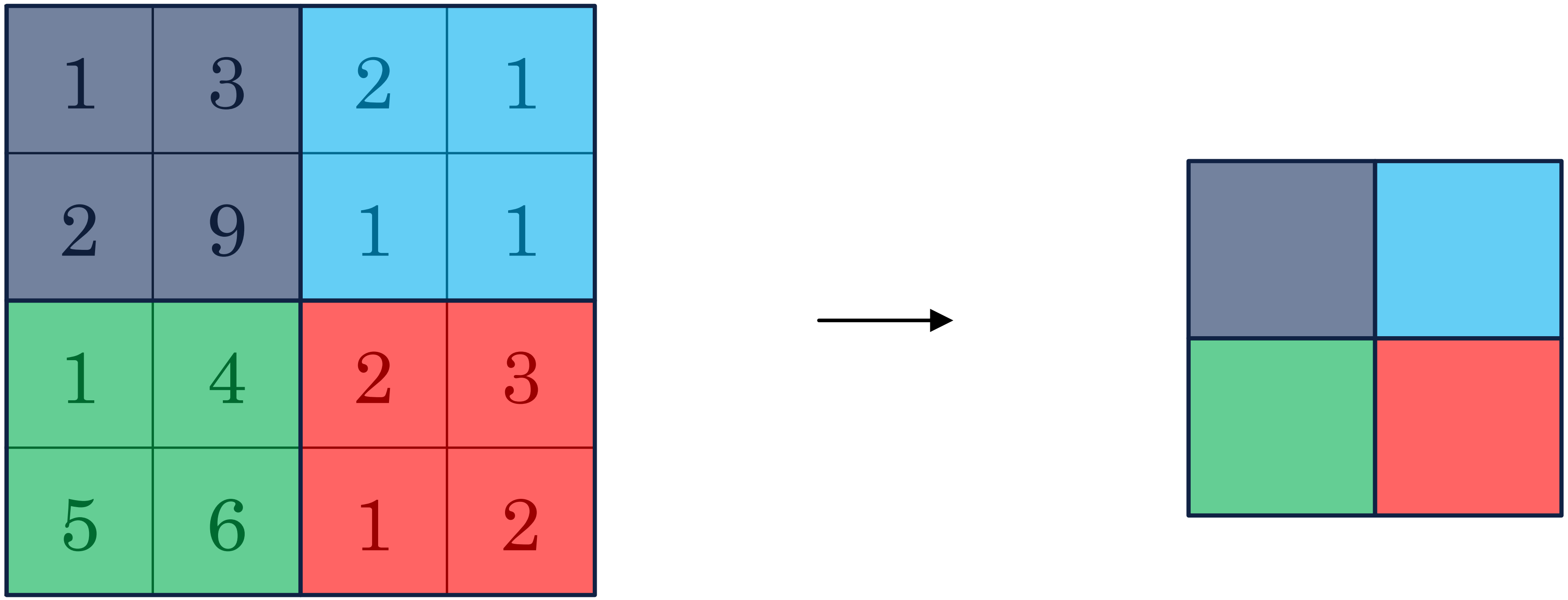
POOLING LAYER: MAX POOLING



POOLING LAYER: MAX POOLING



POOLING LAYER: AVERAGE POOLING



SUMMARY OF POOLING

Hyperparameters:

f : filter size

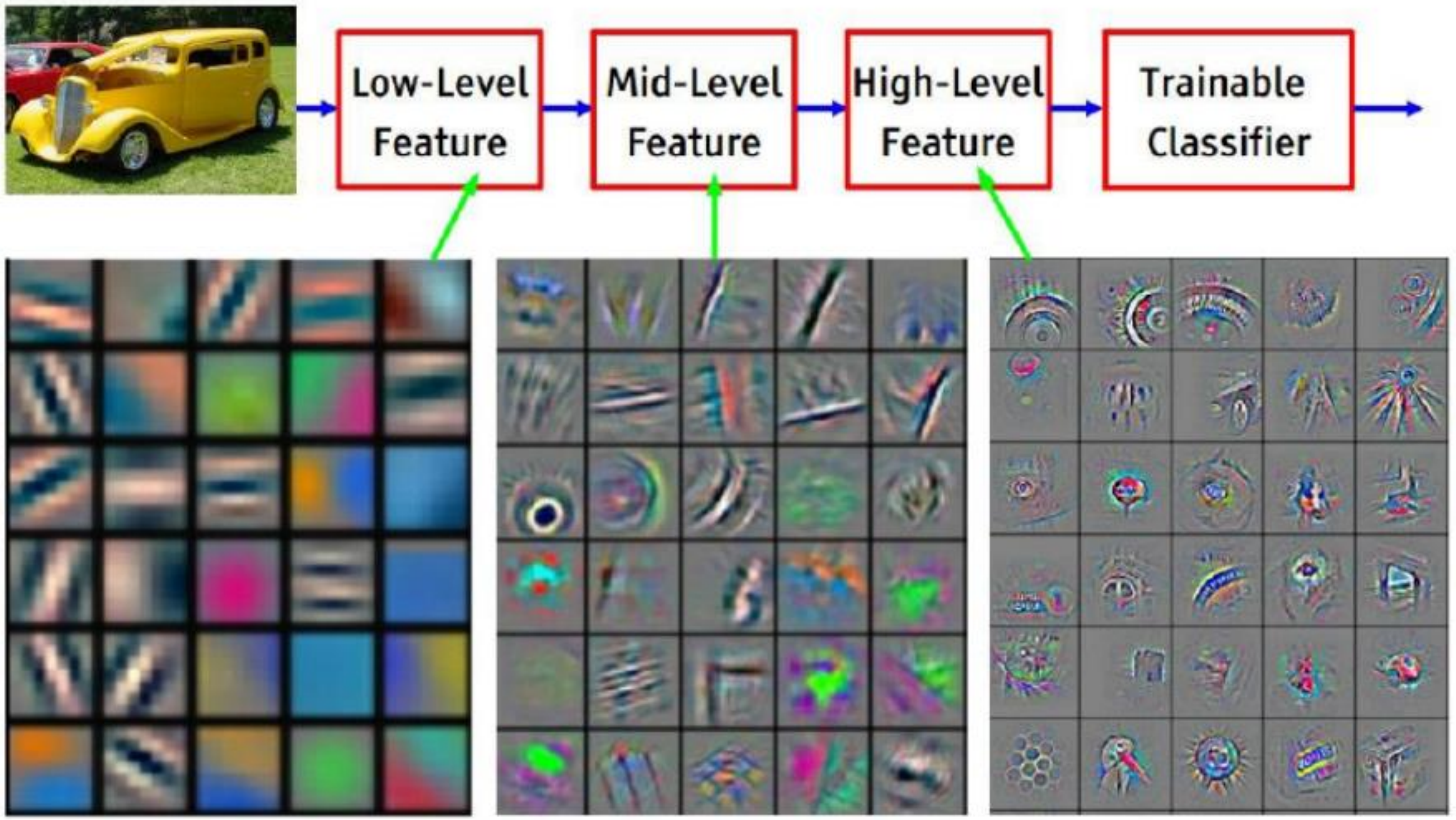
s : stride

Max or average pooling

FULLY CONNECTED LAYERS

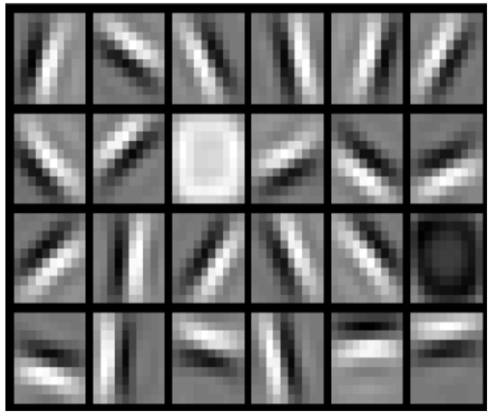
- After flattening, a fully connected MLP can be used.
- The last layer has
  - neurons equal to the desired output size, and
  - activation functions based on the problem to be solved.
- The flattened layer can therefore be viewed as a transformation $\phi(\mathbf{x})$ that is fed into an MLP.
- Similarly, outputs of earlier layers are *intermediate representations* of the input.

INTERMEDIATE REPRESENTATION

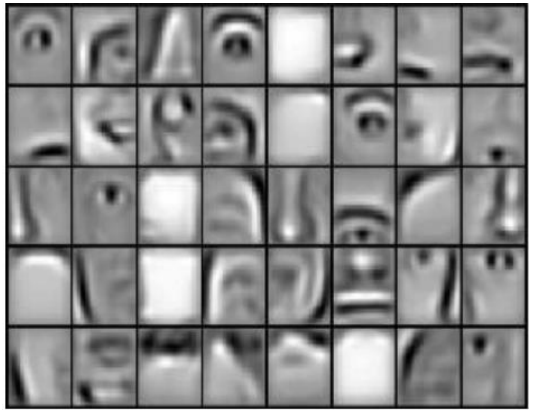


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

INTERMEDIATE REPRESENTATION



Layer 1

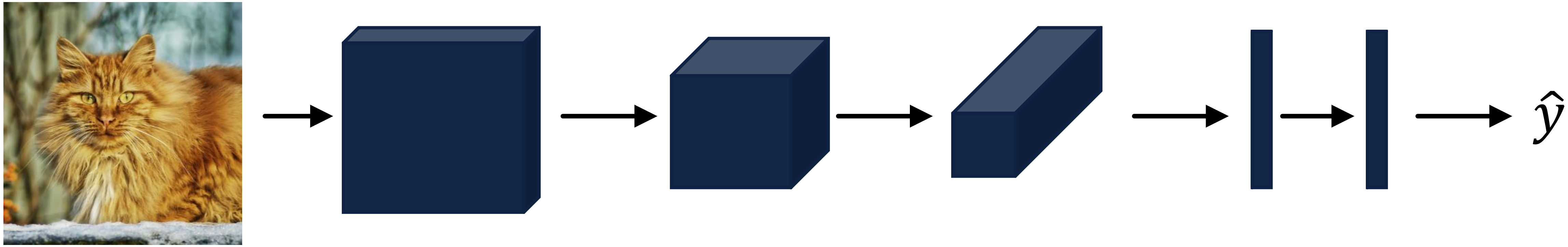


Layer 2

Layer 3

PUTTING IT TOGETHER

Training set $(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})$.



Cost $J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Use gradient descent to optimize parameters to reduce J