CS 4104
APPLIED MACHINE LEARNING

Dr. Hashim Yasin

National University of Computer and Emerging Sciences,

Faisalabad, Pakistan.

CONVOLUTIONAL NEURAL NETWORK

CNN

- Neural Networks that use convolution in place of general matrix multiplication in at least one layer

- There are three types of layers in the convolutional network,

    - **Convolution layer (Conv)**
    - **Pooling layer (Pool)**
    - **Fully connected layer (FC)**

CNN … Stride

$n \times n$ image, $f \times f$ filter

padding p, stride s

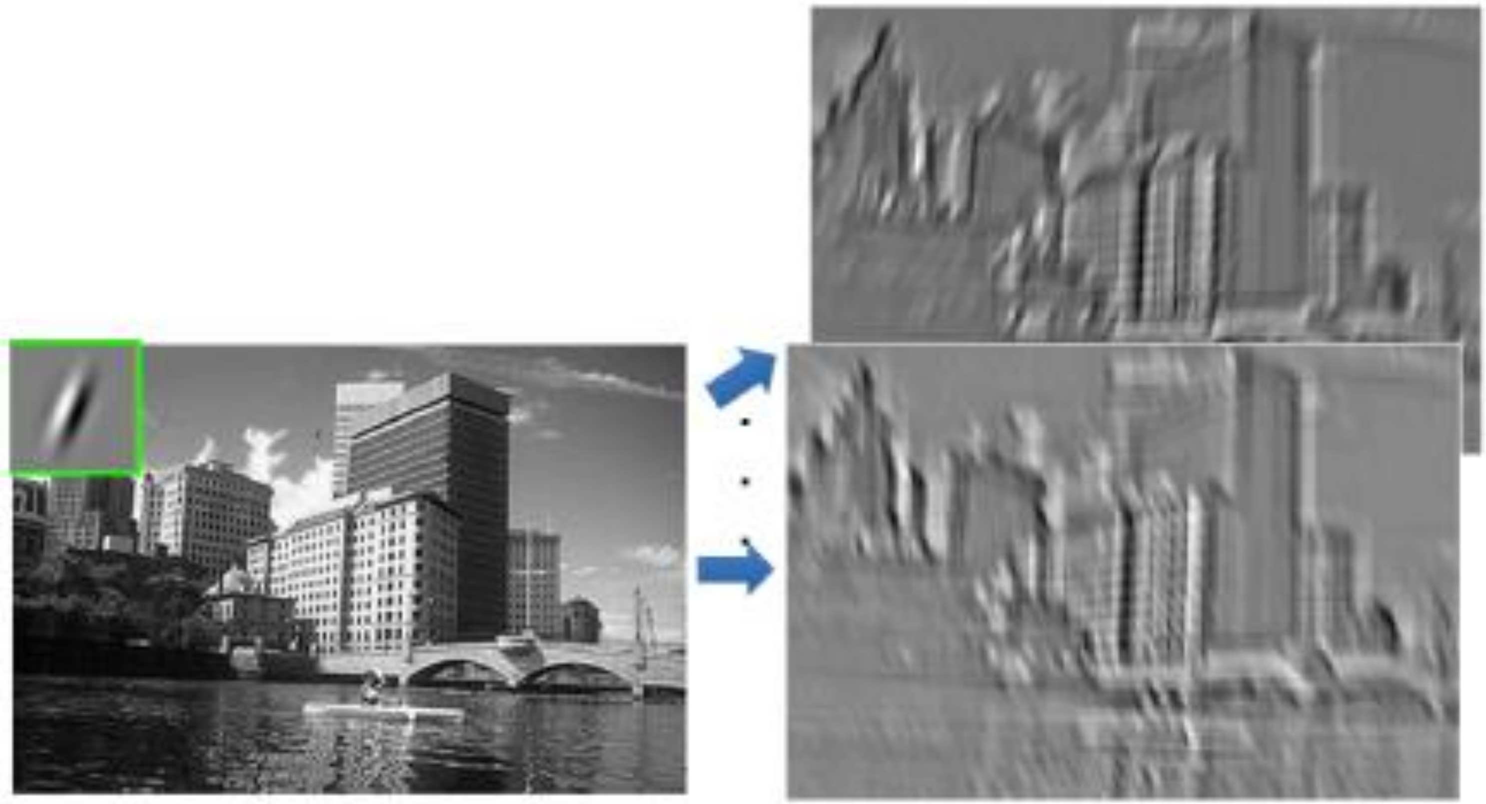
$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

$\lfloor z \rfloor = \text{floor}(z)$

If the fraction is not an integer, then we will get floor of the result.

CNN … Multiple Filters (Example)

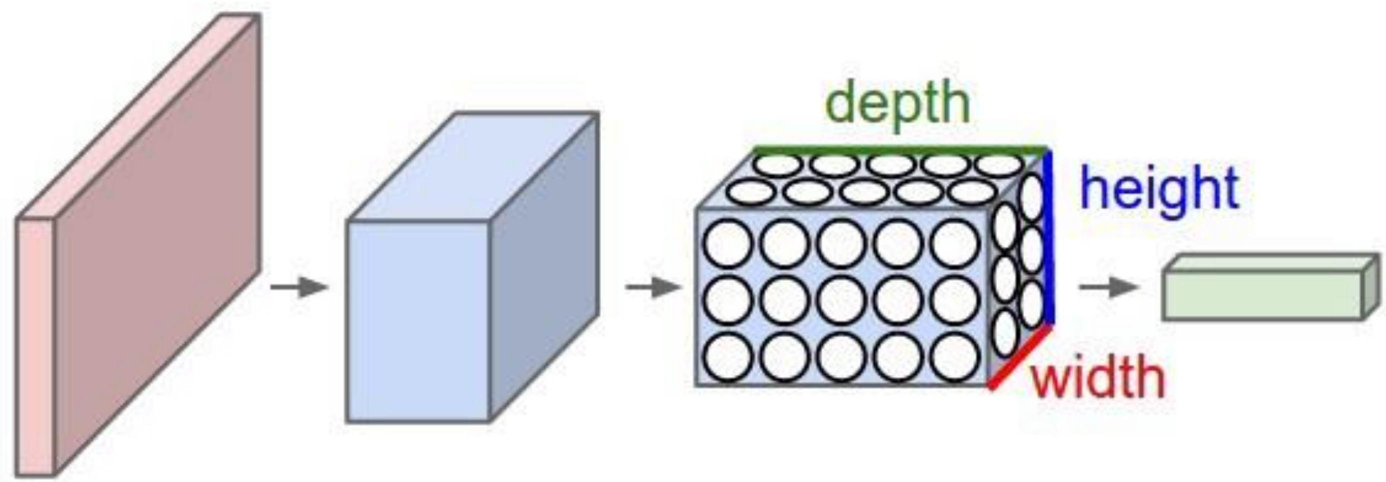
- If we use 100 filters, we get 100 feature maps



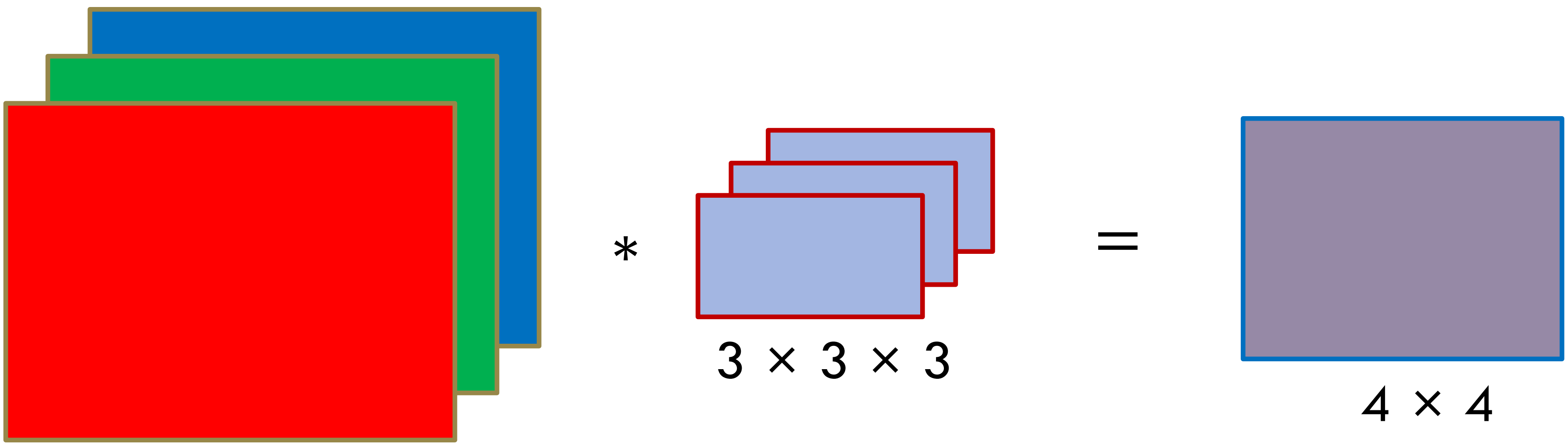
CONVOLUTION OVER VOLUME

RGB Images … Convolution

- We have only considered a 2-D image previously,
- We could operate on volumes as well, e.g.,
  - RGB Images have depth 3 input, a filter would have the same depth,



CNN … Over Volume



Height × Width × **Channel**

The channels/depths should be the same

CNN … Over Volume

$n \times n \times n_c$ image,

$f \times f \times f_c$ filter

$$\boxed{n - f + 1} \; \boxed{\times} \; \boxed{n - f + 1} \; \boxed{\times} \; \boxed{n_c'}$$

CNN … NOTATIONS

CNN … Notations

If layer l is the convolution layers,

- $f^{[l]} = \text{filter size}$
- $p^{[l]} = \text{padding}$
- $s^{[l]} = \text{stride}$

- Input image, $n^{[l-1]} \times n^{[l-1]} \times n_c^{[l-1]}$
- Input image with different height and width,

$$n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$$

CNN … Notations

- Output,

$$n^{[l]} \times n^{[l]} \times n_c^{[l]}$$

$$n^{[l]} = \left\lfloor \frac{n^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

- Output: when input image is with different height and width, $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$$n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

$$n_W^{[l]} = \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

CNN … Notations

- Output,

$$n^{[l]} \times n^{[l]} \times n_c^{[l]}$$

$$n^{[l]} = \left\lfloor \frac{n^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

- $n_c^{[l]}$ = number of filters,

- **Each filters,**

$$f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$$

CNN … Notations

Activations:

$$a^{[l]} = n^{[l]} \times n^{[l]} \times n_c^{[l]}$$

For **multiple activations** in case of batch gradient descent (vectorized implementation)

$$A^{[l]} = M \times n^{[l]} \times n^{[l]} \times n_c^{[l]}$$

CNN … Notations

Weights:

$$w^{[l]} = f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$$

Bias:

$$b^{[l]} = n_c^{[l]}$$

This bias can be represented as a matrix as,

$$b^{[l]} = (1, 1, 1, n_c^{[l]})$$

CNN … Example

$$z^{[1]} = (a^{[0]} \times w^{[1]}) + b^{[1]}$$

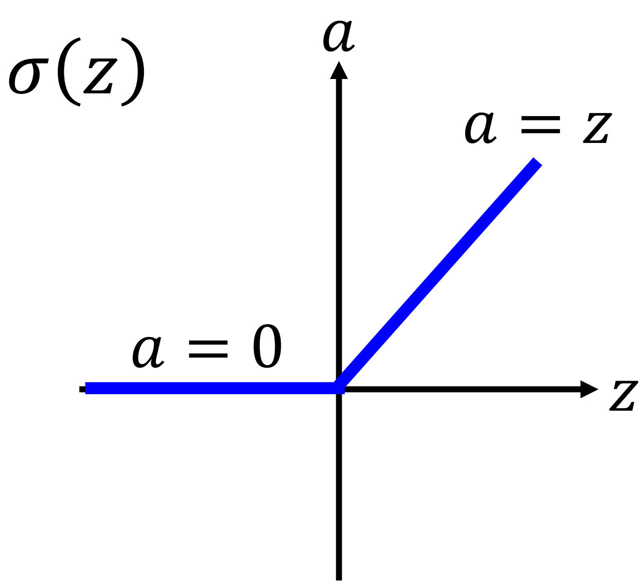
$$a^{[1]} = g(z^{[1]})$$

Where g is the **activation function** (apply non-linearity)

Here, in that case, we may apply **Rectified Linear Unit (ReLU)** as an activation function.

ReLU

- *Rectified Linear Unit* (ReLU) $f(x) = \max(0, x)$



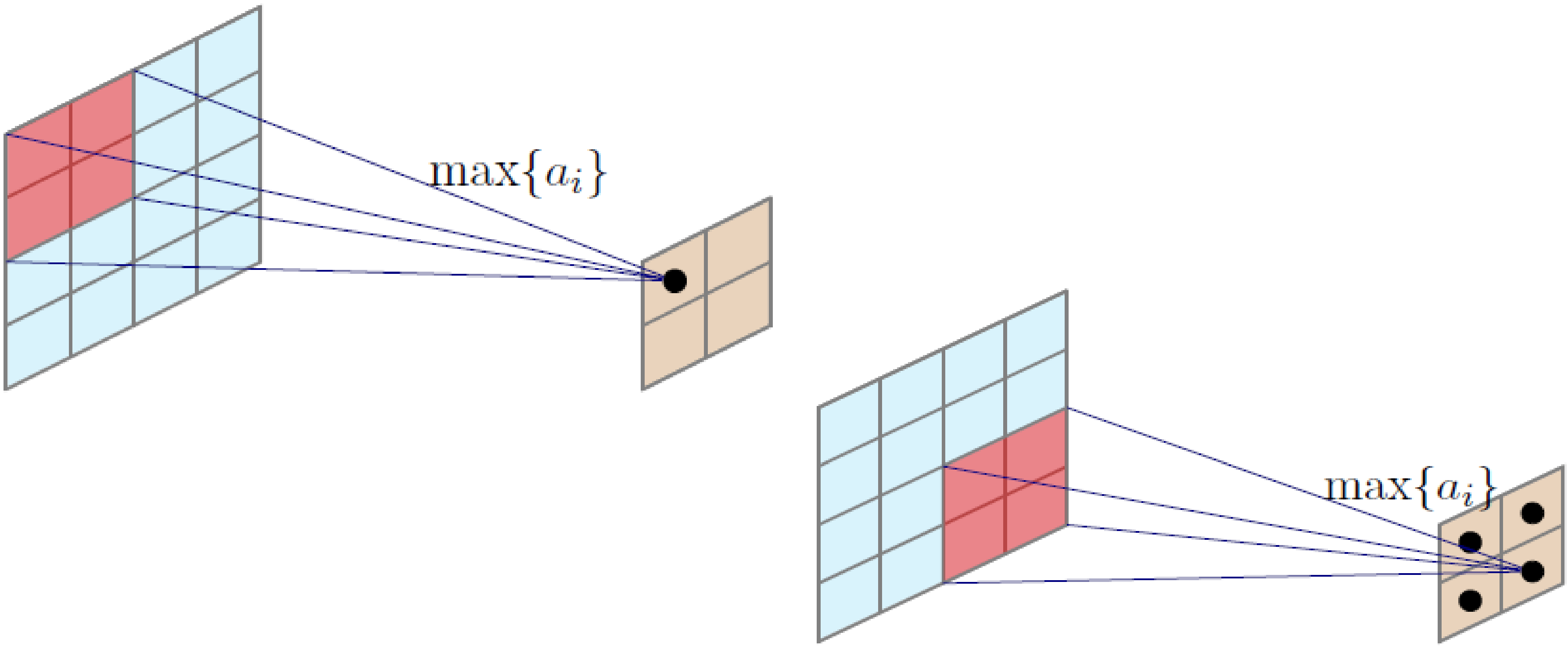
[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

1. Fast to compute
2. Biological reason, (one sided)
3. Efficient gradient propagation *(accelerate (e.g. a factor of 6) the convergence of stochastic gradient descent compared to the sigmoid/tanh functions.)*
4. Scale-invariant
5. Sparse activation

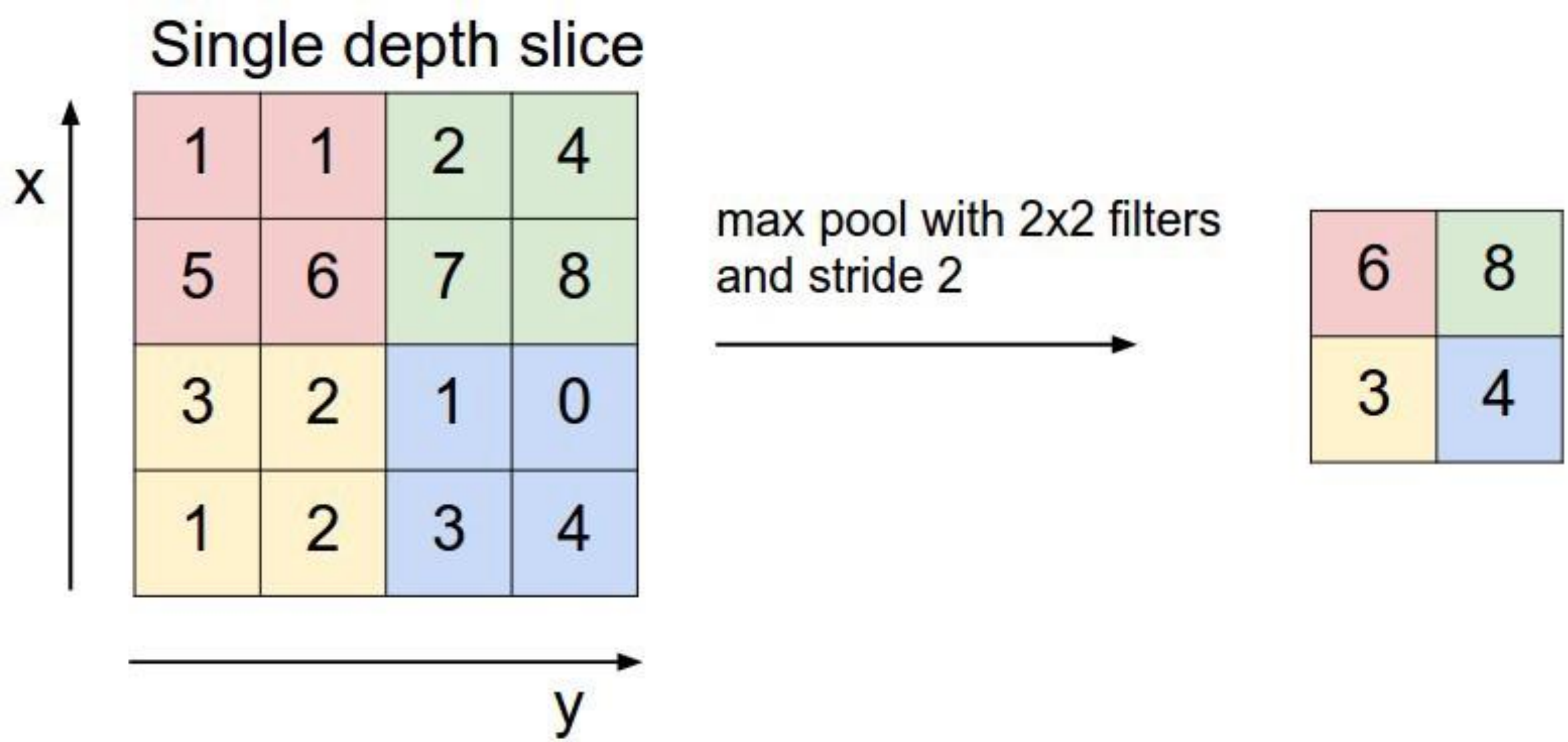
CNN … POOLING

Pooling



Other options: Average pooling, L2-norm pooling, random pooling

Pooling


Single depth slice
x
1
1
2
4
5
6
7
8
3
2
1
0
1
2
3
4
y
max pool with 2x2 filters
and stride 2
6
8
3
4

Max Pooling … Example

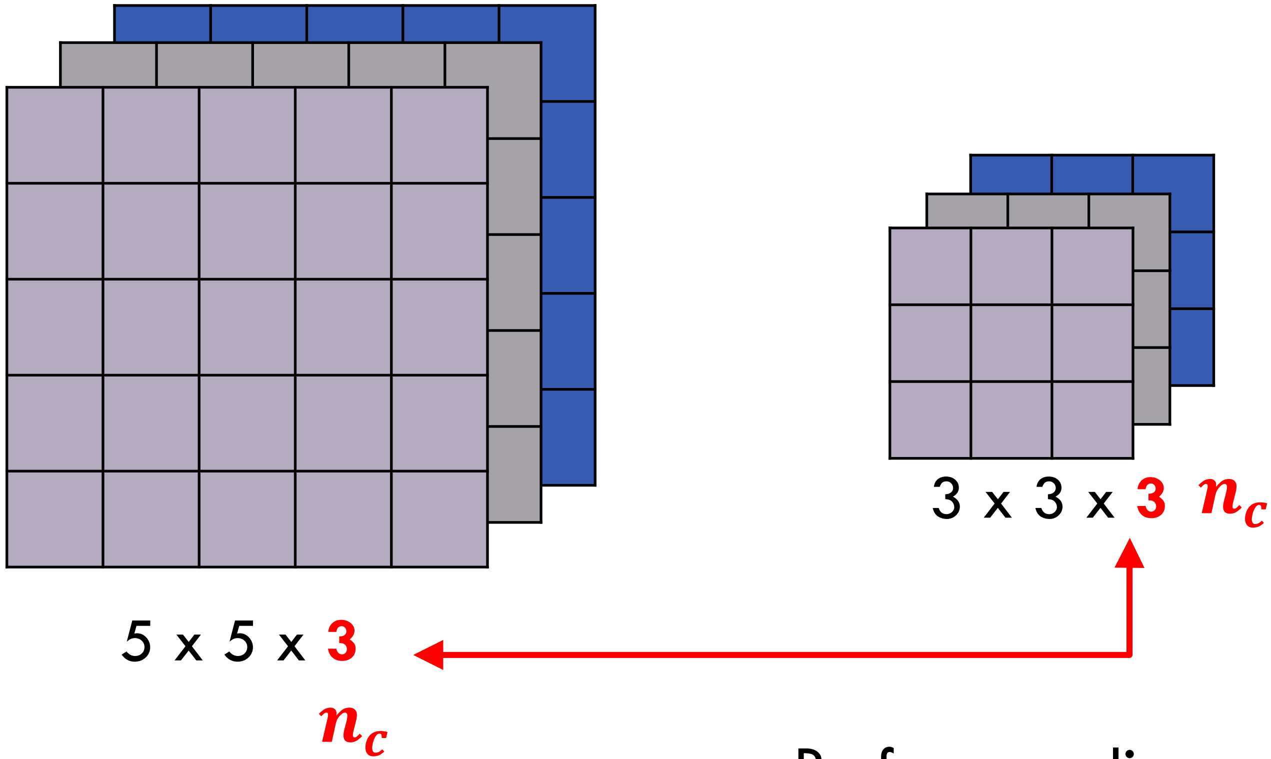
1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9

5 x 5

Filter: $f = 3$
Stride: $s = 1$

9	9	5
9	9	5
8	6	9

Pooling Layer … Over Volume



Perform pooling separately on each channel

Pooling Layer … Average Pooling

1	3	2	1
2	9	1	1
1	4	2	3
5	6	1	2

Filter: $f = 2$
Stride: $s = 2$

3.75	**1.25**
4	**2**

Pooling Layer … Hyperparameters

The hyperparameters for pooling layers are:

- f: filter size
- s: stride
- Max or average pooling
- Padding
  - (which is very rarely used)

Pooling Layer

Input:

$$n_H \times n_W \times n_C$$

Output:

$$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_C$$

Note: There is NO parameter to learn in the pooling layer

CNN … EXAMPLE

CNN … Example 3

Construct the CNN with the following specifications,

- The RGB image of size 32×32 is given as input
- **Layer 1-Conv1:** There are 6 filters of size 5×5, stride $= 1$ and no padding,
- **Layer 1-Pool1:** There are 6 filters of size 2×2, stride $= 2$ and no padding,
- **Layer 2-Conv2:** There are 16 filters of size 5×5, stride $= 1$ and no padding,
- **Layer 2-Pool2:** There are 16 filters of size 2×2, stride $= 2$ and no padding,
- **Layer 3-FC3:** There are 120 neurons in FC3.
- **Layer 4-FC4:** There are 84 neurons in FC4.
- **Output Layer:** with 10 classes.

CNN … Example 3 (Summary)

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,6)	4,704	$(5\times5\times3 + 1) \times 6 = 456$
POOL1	(14,14,6)	1,176	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	$(5\times5\times6 + 1) \times16 = 2416$
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	$400\times120 + 120 = 48120$
FC4	(84,1)	84	$120\times84 + 84 = 10164$
Softmax	(10,1)	10	$84\times10 + 10 = 850$

Acknowledgements

Stuart J. Russell and Peter Norvig, Tom M. Mitchell, Jiwon Jeong, Floydhub, Andrej Karpathy