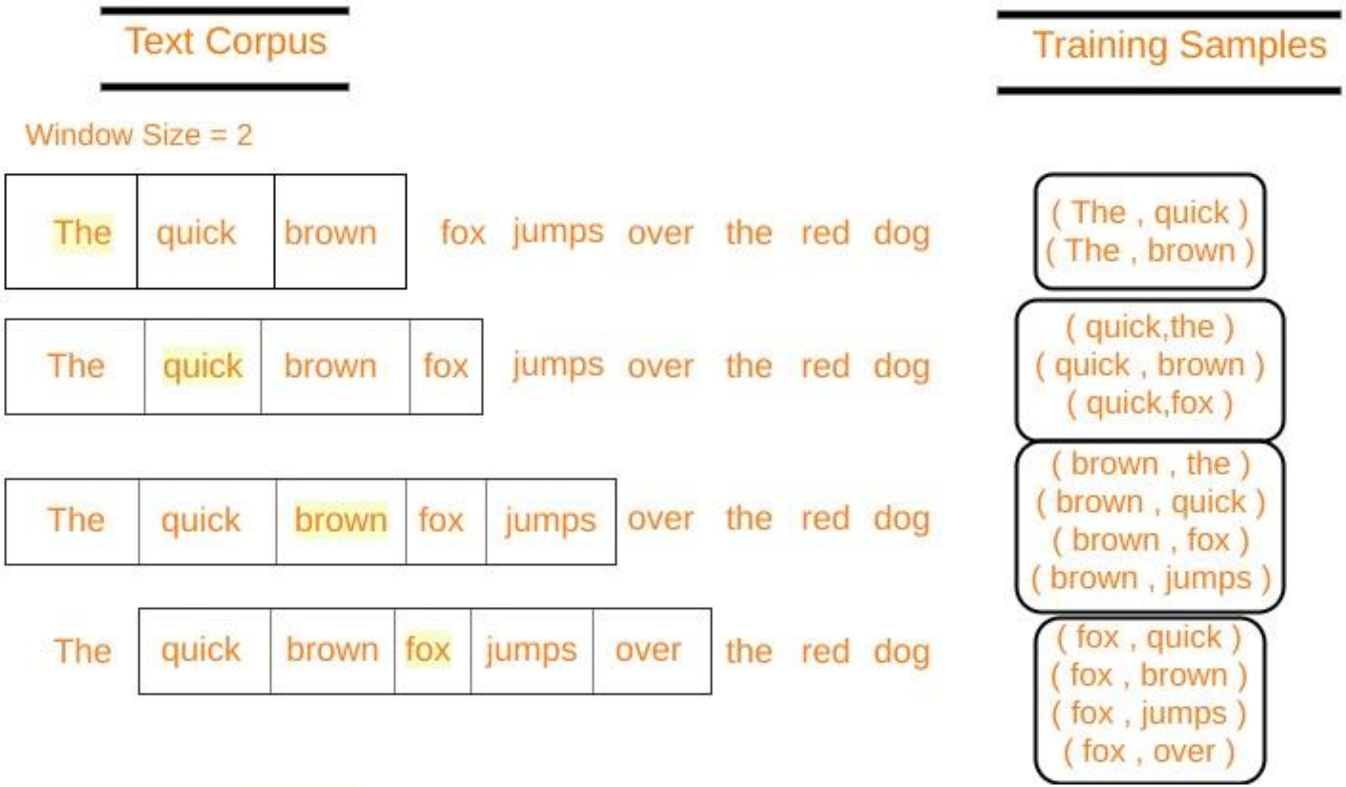
Keras

Keras for Deep Learning Research

Overview

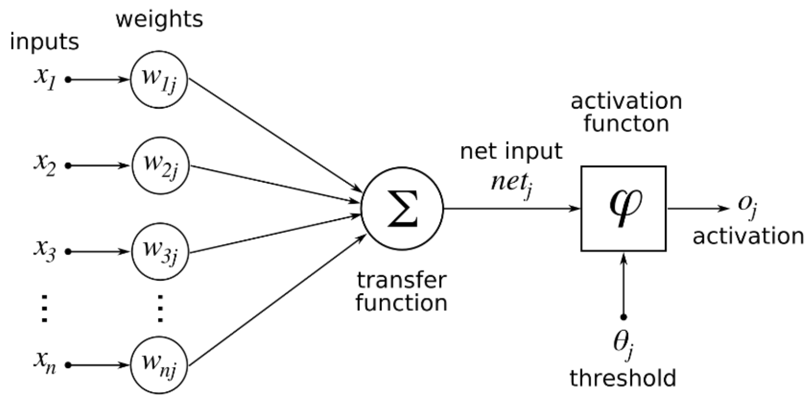
- Quick Review of Previous Topics and NNs
- Convolutional Neural Networks (CNN)
- Hyperparameters of CNN
- Image classification with CNN

Preparing data for w2vec



Deep Learning

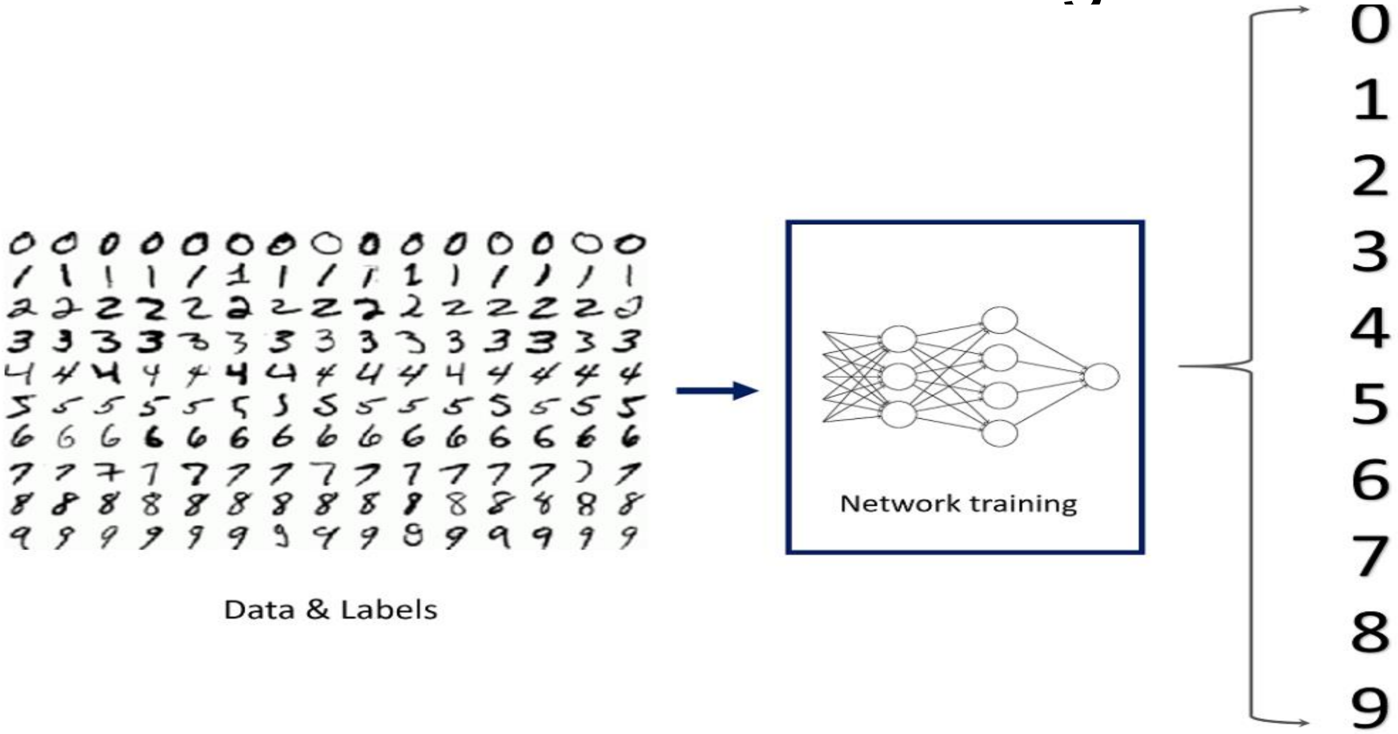
- Single Neuron
- We now understand how to perform a calculation in a neuron
  - $w \cdot x + b = z$
  - $a = \sigma(z)$



Source: https://www.udemy.com/complete-guide-to-tensorflow-for-deep-learning-with-python/learn/v4/overview

Previously we learned: Image classification on MNIST dataset using NN

.



Convolutional Neural Network (CNN)

UMKC

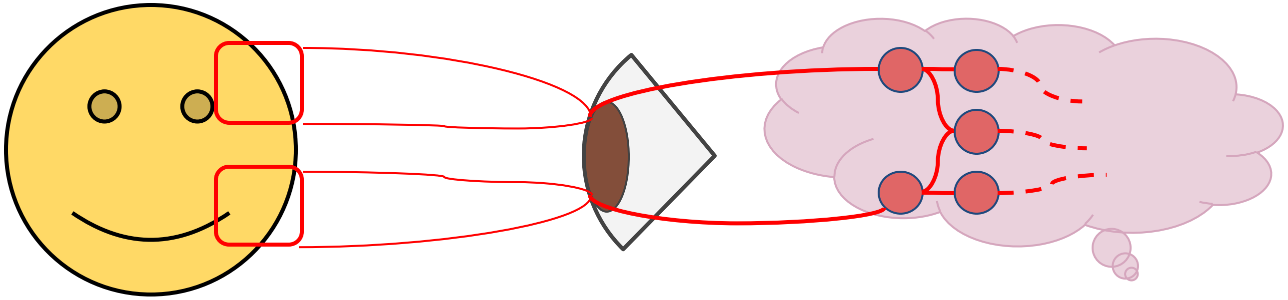
Convolutional Neural Network (CNN)

- Just like the simple perceptron, CNNs also have their origins in biological research.
- Hubel and Wiesel studied the structure of the visual cortex in mammals, winning a Nobel Prize in 1981.

Source: https://en.wikipedia.org/wiki/Convolutional_neural_network

Convolutional Neural Network (CNN)

- Their research revealed that neurons in the visual cortex had a small local receptive field.



Source: https://en.wikipedia.org/wiki/Convolutional_neural_network

Convolutional Neural Network (CNN)

- This idea then inspired an ANN architecture that would become CNN
- Famously implemented in the 1998 paper by Yann LeCun et al.
- The LeNet-5 architecture was first used to classify the MNIST data set.

Source: https://en.wikipedia.org/wiki/Convolutional_neural_network

Convolutional Neural Network (CNN)

- There are four main operations in the CNN

- Convolutions and Filters
- Pooling or Sub Sampling
- Non Linearity
- Classification (Fully Connected Layer)

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

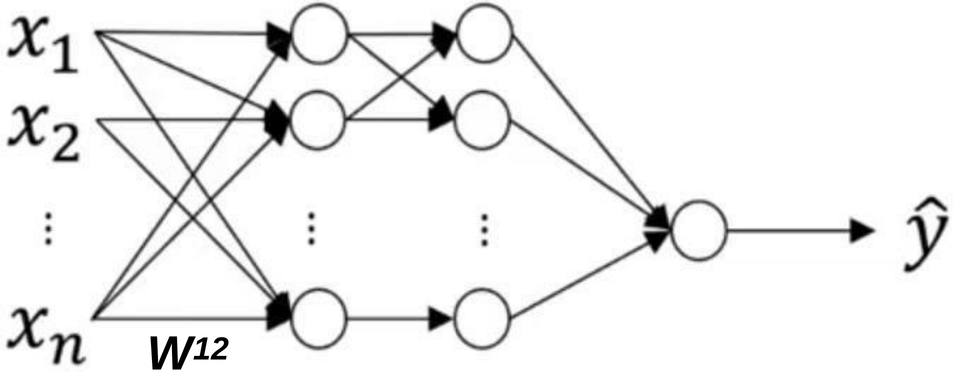
UMKC

Traditional Neural Network vs. Convolutional Neural Network

32

32

32 x 32 x 3 = ***3072*** ***1000***



W^{12}

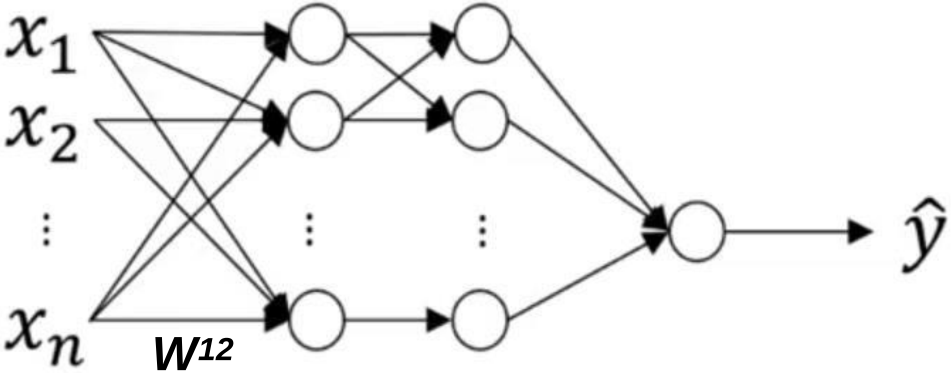
1000 x 3072
dimensional matrix

Traditional Neural Network vs. Convolutional Neural Network

1000

1000

1000 x 1000 x 3 = ***3,000,000*** ***1000***



W^{12}

1000 x 3,000,000

dimensional matrix!!!

= ***3,000,000,000 features***

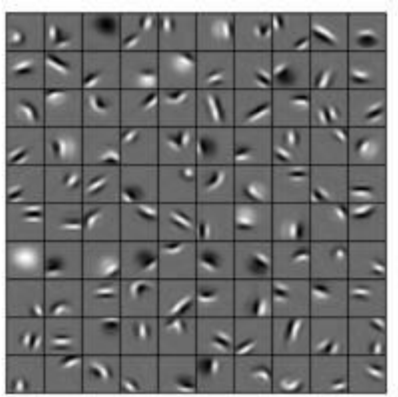
Traditional Neural Network vs. Convolutional Neural Network

Salient points:

- Spatial relation between features in image is not considered in NN.
- NN is not feasible for large images!
- In order to perform Computer Vision operations on large images, convolution operation plays an important role.
- Thus, CNNs are fundamentally important.

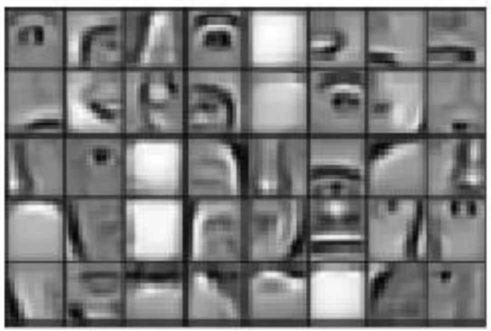
Stages of feature extraction by CNN

Low level features



Edges, curves and colour

Mid level features



Parts of objects

High level features

Complete objects

Foundation of Convolutional Neural Networks

- 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

*

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

=

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3 $^{-1}$	0 ^{0}	1 ^{1}	2	7	4
1 $^{-1}$	5 ^{0}	8 ^{1}	9	3	1
2 $^{-1}$	7 ^{0}	2 ^{1}	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

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

=

5			

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0^{-1}	1^{0}	2^{1}	7	4
1	5^{-1}	8^{0}	9^{1}	3	1
2	7^{-1}	2^{0}	5^{1}	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

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

=

5	4		

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1 [-1]	2 [0]	7 [1]	4
1	5	8 [-1]	9 [0]	3 [1]	1
2	7	2 [-1]	5 [0]	1 [1]	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

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

=

5	4	0	

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2 $^{-1}$	7 ^{0}	4 ^{1}
1	5	8	9 $^{-1}$	3 ^{0}	1 ^{1}
2	7	2	5 $^{-1}$	1 ^{0}	3 ^{1}
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

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

=

5	4	0	-8

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1 $^{-1}$	5 ^{0}	8 ^{1}	9	3	1
2 $^{-1}$	7 ^{0}	2 ^{1}	5	1	3
0 $^{-1}$	1 ^{0}	3 ^{1}	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

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

=

5	4	0	-8
10			

UMKC

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1	5 $^{-1}$	8 ^{0}	9 ^{1}	3	1
2	7 $^{-1}$	2 ^{0}	5 ^{1}	1	3
0	1 $^{-1}$	3 ^{0}	1 ^{1}	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

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

=

5	4	0	-8
10	2		

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1	5	8 [-1]	9 [0]	3 [1]	1
2	7	2 [-1]	5 [0]	1 [1]	3
0	1	3 [-1]	1 [0]	7 [1]	8
4	2	1	6	2	8
2	4	5	2	3	9

*

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

=

5	4	0	-8
10	2	2	

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1	5	8	9 [-1]	3 [0]	1 [1]
2	7	2	5 [-1]	1 [0]	3 [1]
0	1	3	1 [-1]	7 [0]	8 [1]
4	2	1	6	2	8
2	4	5	2	3	9

*

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

=

5	4	0	-8
10	2	2	-3

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1	5	8	9	3	1
2 $^{-1}$	7 ^{0}	2 ^{1}	5	1	3
0 $^{-1}$	1 ^{0}	3 ^{1}	1	7	8
4 $^{-1}$	2 ^{0}	1 ^{1}	6	2	8
2	4	5	2	3	9

*

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

=

5	4	0	-8
10	2	2	-3
0			

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1	5	8	9	3	1
2	7^{-1}	2^{0}	5^{1}	1	3
0	1^{-1}	3^{0}	1^{1}	7	8
4	2^{-1}	1^{0}	6^{1}	2	8
2	4	5	2	3	9

*

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

=

5	4	0	-8
10	2	2	-3
0	2		

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2^{-1}	5^{0}	1^{1}	3
0	1	3^{-1}	1^{0}	7^{1}	8
4	2	1^{-1}	6^{0}	2^{1}	8
2	4	5	2	3	9

*

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

=

5	4	0	-8
10	2	2	-3
0	2	4	

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5 $^{-1}$	1 ^{0}	3 ^{1}
0	1	3	1 $^{-1}$	7 ^{0}	8 ^{1}
4	2	1	6 $^{-1}$	2 ^{0}	8 ^{1}
2	4	5	2	3	9

*

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

=

5	4	0	-8
10	2	2	-3
0	2	4	7

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0^{-1}	1^{0}	3^{1}	1	7	8
4^{-1}	2^{0}	1^{1}	6	2	8
2^{-1}	4^{0}	5^{1}	2	3	9

*

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

=

5	4	0	-8
10	2	2	-3
0	2	4	7
3			

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1 $^{-1}$	3 ^{0}	1 ^{1}	7	8
4	2 $^{-1}$	1 ^{0}	6 ^{1}	2	8
2	4 $^{-1}$	5 ^{0}	2 ^{1}	3	9

*

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

=

5	4	0	-8
10	2	2	-3
0	2	4	7
3	2		

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3 $_{-1}$	1 $_{0}$	7 $_{1}$	8
4	2	1 $_{-1}$	6 $_{0}$	2 $_{1}$	8
2	4	5 $_{-1}$	2 $_{0}$	3 $_{1}$	9

*

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

=

5	4	0	-8
10	2	2	-3
0	2	4	7
3	2	3	

Foundation of Convolutional Neural Networks

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

*

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

=

5	4	0	-8
10	2	2	-3
0	2	4	7
3	2	3	16

UMKC

Foundation of Convolutional Neural Networks

- Padding

Image	*	Filter	=	Output Image
6 x 6		3 x 3		4 x 4
$n * n$		$f * f$		$nout * nout$

using $nout = n - f + 1$

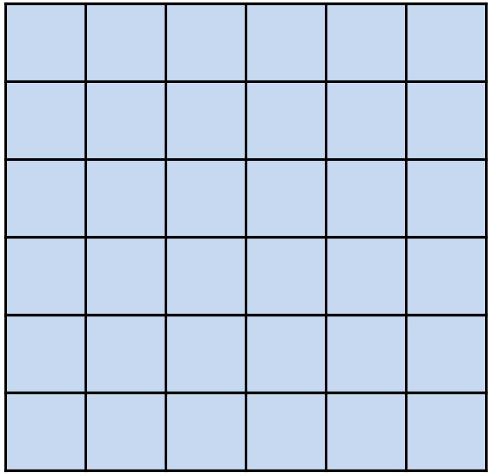
Shortcoming of this technique:

1) *Output goes on shrinking as the number of layers increase.*
2) *Information from boundary of the image remains unused.*

Solution is Zero padding around the edges of the image!

Foundation of Convolutional Neural Networks

- Padding



If Image is 6 x 6

$$
\begin{aligned}
nout &= n - f + 1 \\
&= 6 - 3 + 1 \\
&= 4
\end{aligned}
$$

Thus, Output Image = 4 x 4

Foundation of Convolutional Neural Networks

- Padding

0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

If $p=1$

$$
\begin{aligned}
nout &= n + 2p - f + 1 \\
&= 6 + (2 * 1) - 3 + 1 \\
&= 6
\end{aligned}
$$

Thus, Output Image = 6 x 6

Foundation of Convolutional Neural Networks

- Padding

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0							0	0
0	0							0	0
0	0							0	0
0	0							0	0
0	0							0	0
0	0							0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

If $p = 2$

$$
\begin{aligned}
nout &= n + 2p - f + 1 \\
&= 6 + (2 * 2) - 3 + 1 \\
&= 6 + 4 - 3 + 1 \\
&= 8
\end{aligned}
$$

Thus, Output Image = 8 x 8

Foundation of Convolutional Neural Networks

- Padding

How much to pad?

1) *Valid* = no padding.

Follows the formula $nout = n - f + 1$.

2) *Same* = Output dimension is Same as input.

Follows the formula $nout = n + 2p - f + 1$.

To keep Output size same as input size: $n = n + 2p - f + 1$

$p = \frac{(f-1)}{2}$ Thus, filters are generally odd.

Foundation of Convolutional Neural Networks

- Strided Convolution: Shifting of filter by s pixels during convolution. Here, s =2.

3 $^{-1}$	0 ^{0}	1 ^{1}	2	7	4	2
1 $^{-1}$	5 ^{0}	8 ^{1}	9	3	1	1
2 $^{-1}$	7 ^{0}	2 ^{1}	5	1	3	5
0	1	3	1	7	8	4
4	2	1	6	2	8	3
2	4	5	2	3	9	8
2	3	6	4	2	0	1

*

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

=

5		

UMKC

Foundation of Convolutional Neural Networks

- Strided Convolution

3	0	1^{-1}	2^{0}	7^{1}	4	2
1	5	8^{-1}	9^{0}	3^{1}	1	1
2	7	2^{-1}	5^{0}	1^{1}	3	5
0	1	3	1	7	8	4
4	2	1	6	2	8	3
2	4	5	2	3	9	8
2	3	6	4	2	0	1

*

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

=

5	0	

Foundation of Convolutional Neural Networks

- Strided Convolution

3	0	1	2	7 $^{-1}$	4 ^{0}	2 ^{1}
1	5	8	9	3 $^{-1}$	1 ^{0}	1 ^{1}
2	7	2	5	1 $^{-1}$	3 ^{0}	5 ^{1}
0	1	3	1	7	8	4
4	2	1	6	2	8	3
2	4	5	2	3	9	8
2	3	6	4	2	0	1

*

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

=

5	0	-3

Foundation of Convolutional Neural Networks

- Strided Convolution

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

*

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

=

5	0	-3
0		

Foundation of Convolutional Neural Networks

- Strided Convolution

3	0	1	2	7	4	2
1	5	8	9	3	1	1
2	7	2 $^{-1}$	5 ^{0}	1 ^{1}	3	5
0	1	3 $^{-1}$	1 ^{0}	7 ^{1}	8	4
4	2	1 $^{-1}$	6 ^{0}	2 ^{1}	8	3
2	4	5	2	3	9	8
2	3	6	4	2	0	1

*

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

=

5	0	-3
0	4	

Foundation of Convolutional Neural Networks

- Strided Convolution

3	0	1	2	7	4	2
1	5	8	9	3	1	1
2	7	2	5	1^{-1}	3^{0}	5^{1}
0	1	3	1	7^{-1}	8^{0}	4^{1}
4	2	1	6	2^{-1}	8^{0}	3^{1}
2	4	5	2	3	9	8
2	3	6	4	2	0	1

*

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

=

5	0	-3
0	4	2

Foundation of Convolutional Neural Networks

- Strided Convolution

3	0	1	2	7	4	2
1	5	8	9	3	1	1
2	7	2	5	1	3	5
0	1	3	1	7	8	4
4^{-1}	2^{0}	1^{1}	6	2	8	3
2^{-1}	4^{0}	5^{1}	2	3	9	8
2^{-1}	3^{0}	6^{1}	4	2	0	1

*

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

=

5	0	-3
0	4	2
4		

Foundation of Convolutional Neural Networks

- Strided Convolution

3	0	1	2	7	4	2
1	5	8	9	3	1	1
2	7	2	5	1	3	5
0	1	3	1	7	8	4
4	2	1^{-1}	6^{0}	2^{1}	8	3
2	4	5^{-1}	2^{0}	3^{1}	9	8
2	3	6^{-1}	4^{0}	2^{1}	0	1

*

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

=

5	0	-3
0	4	2
4	-5	

UMKC

Foundation of Convolutional Neural Networks

- Strided Convolution

3	0	1	2	7	4	2
1	5	8	9	3	1	1
2	7	2	5	1	3	5
0	1	3	1	7	8	4
4	2	1	6	2^{-1}	8^{0}	3^{1}
2	4	5	2	3^{-1}	9^{0}	8^{1}
2	3	6	4	2^{-1}	0^{0}	1^{1}

*

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

=

5	0	-3
0	4	2
4	-5	5

Foundation of Convolutional Neural Networks

- Strided Convolution

3	0	1	2	7	4	2
1	5	8	9	3	1	1
2	7	2	5	1	3	5
0	1	3	1	7	8	4
4	2	1	6	2^{-1}	8^{0}	3^{1}
2	4	5	2	3^{-1}	9^{0}	8^{1}
2	3	6	4	2^{-1}	0^{0}	1^{1}

*

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

=

5	0	-3
0	4	2
4	-5	5

Here, Stride(S)=2,

Thus, $nout = \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor = \left\lfloor \frac{7+2*0-3}{2} + 1 \right\rfloor = \left\lfloor \frac{4}{2} + 1 \right\rfloor$

$= \left\lfloor \frac{4}{2} + 1 \right\rfloor = 3$

If input image is 6x6 then Output Image will be ?

Foundation of Convolutional Neural Networks

- Strided Convolution

3	0	1	2	7	4	2
1	5	8	9	3	1	1
2	7	2	5	1	3	5
0	1	3	1	7	8	4
4	2	1	6	2^{-1}	8^{0}	3^{1}
2	4	5	2	3^{-1}	9^{0}	8^{1}
2	3	6	4	2^{-1}	0^{0}	1^{1}

*

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

=

5	0	-3
0	4	2
4	-5	5

Here, Stride(S)=2,

Thus, $nout = \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor = \left\lfloor \frac{7+2*0-3}{2} + 1 \right\rfloor = \left\lfloor \frac{4}{2} + 1 \right\rfloor$

$$= \left\lfloor \frac{4}{2} + 1 \right\rfloor = 3$$

If input image is 6x6 then Output Image will be still 2x2 if all other factors are constant.

Foundation of Convolutional Neural Networks

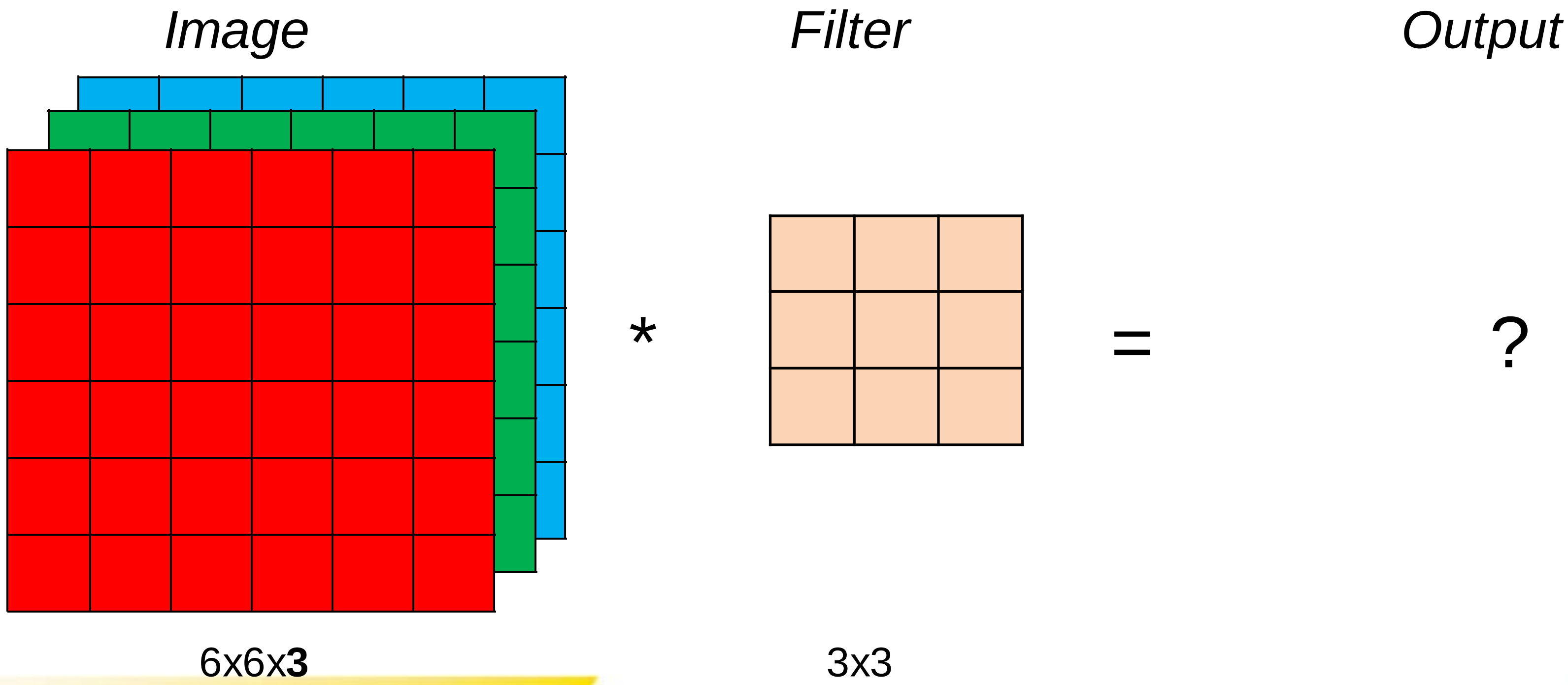
- Summary of Convolution

For an $n * n$ image and filter $f * f$ with padding p and a stride of s pixels,

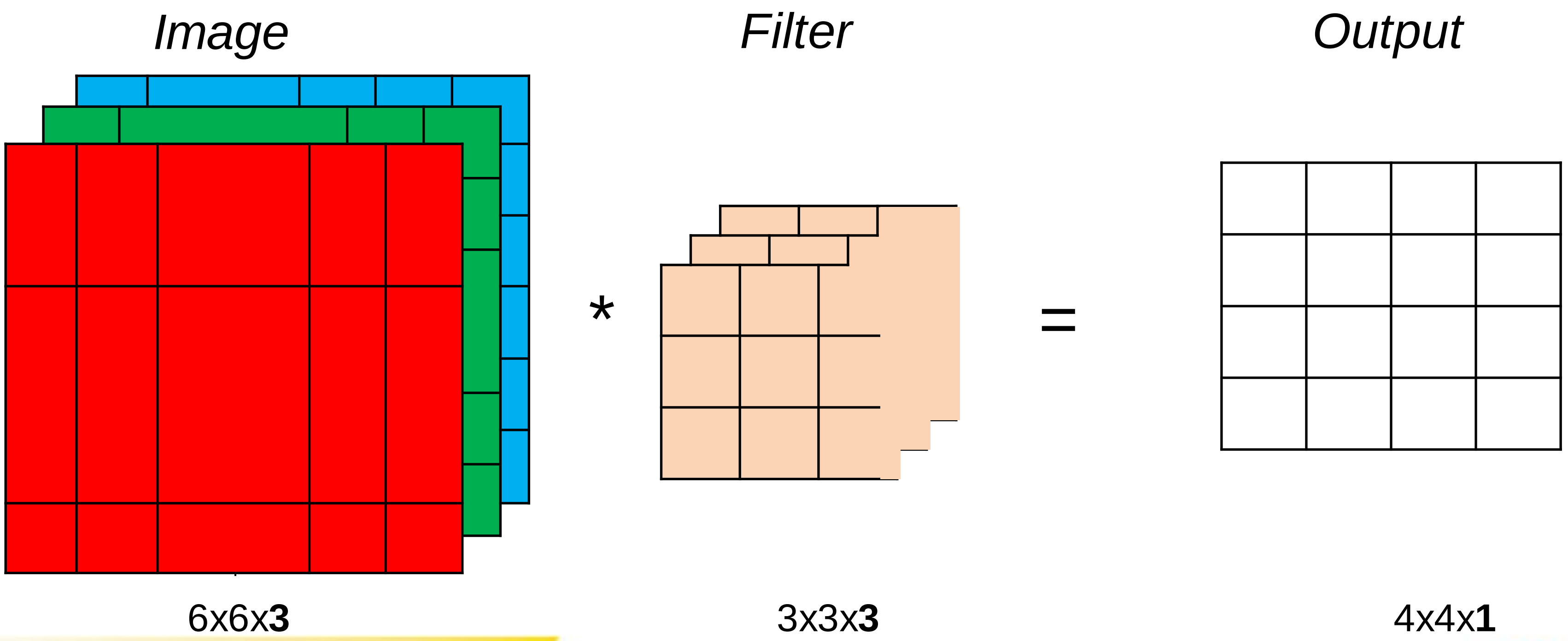
Output image dimension is given by:

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

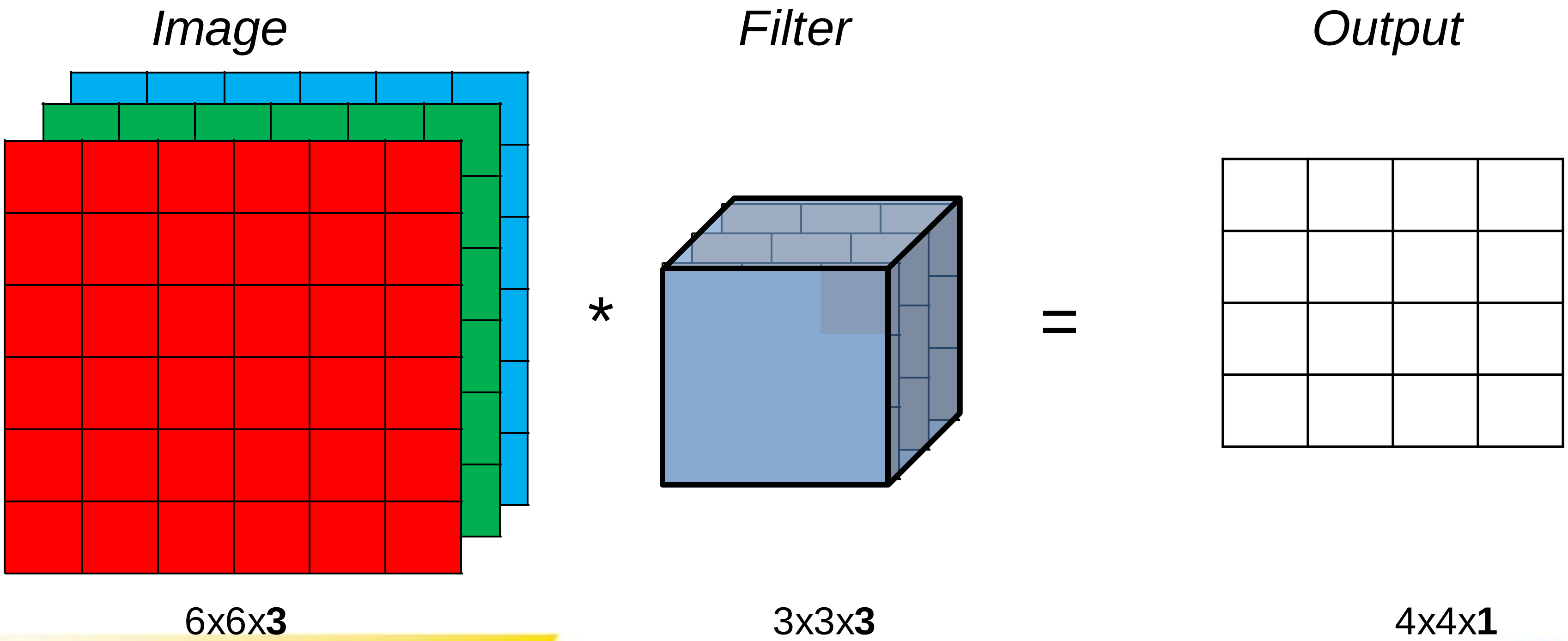
Convolution on Colour images



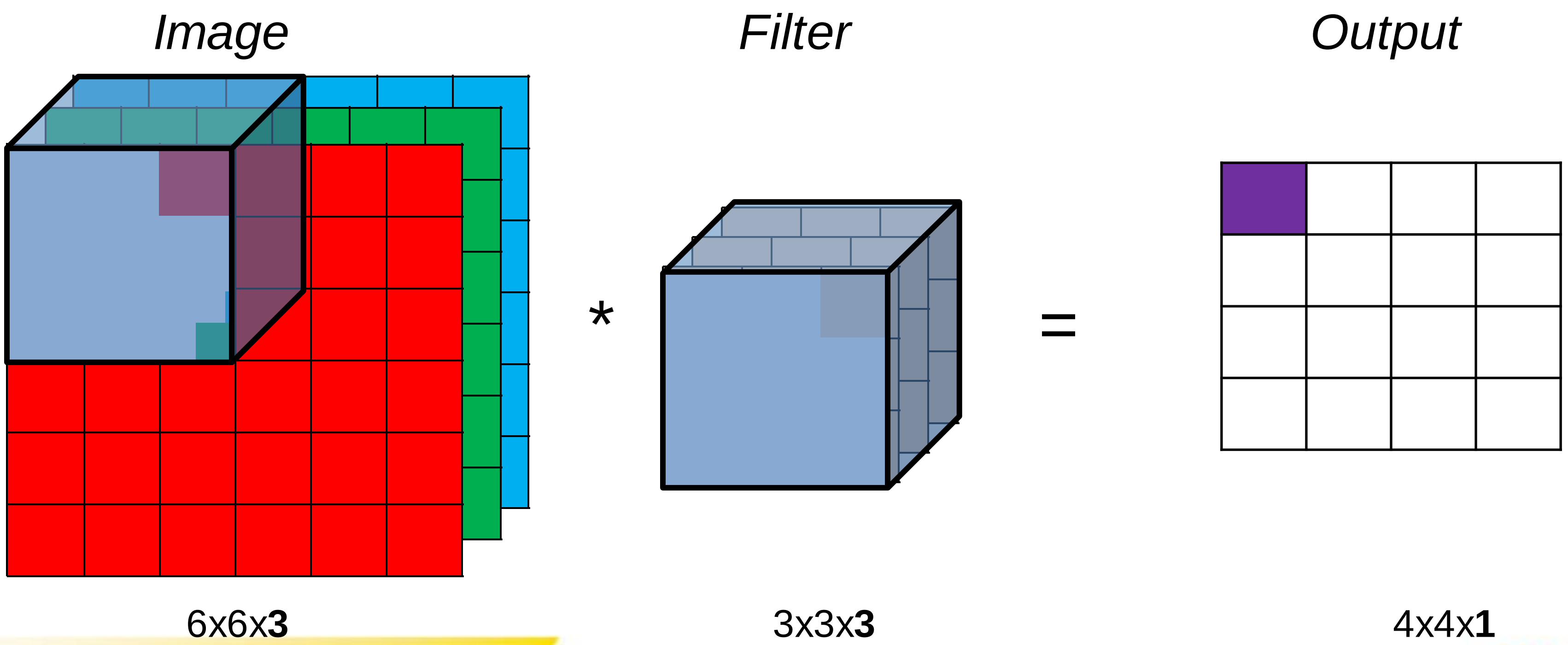
Convolution on Colour images



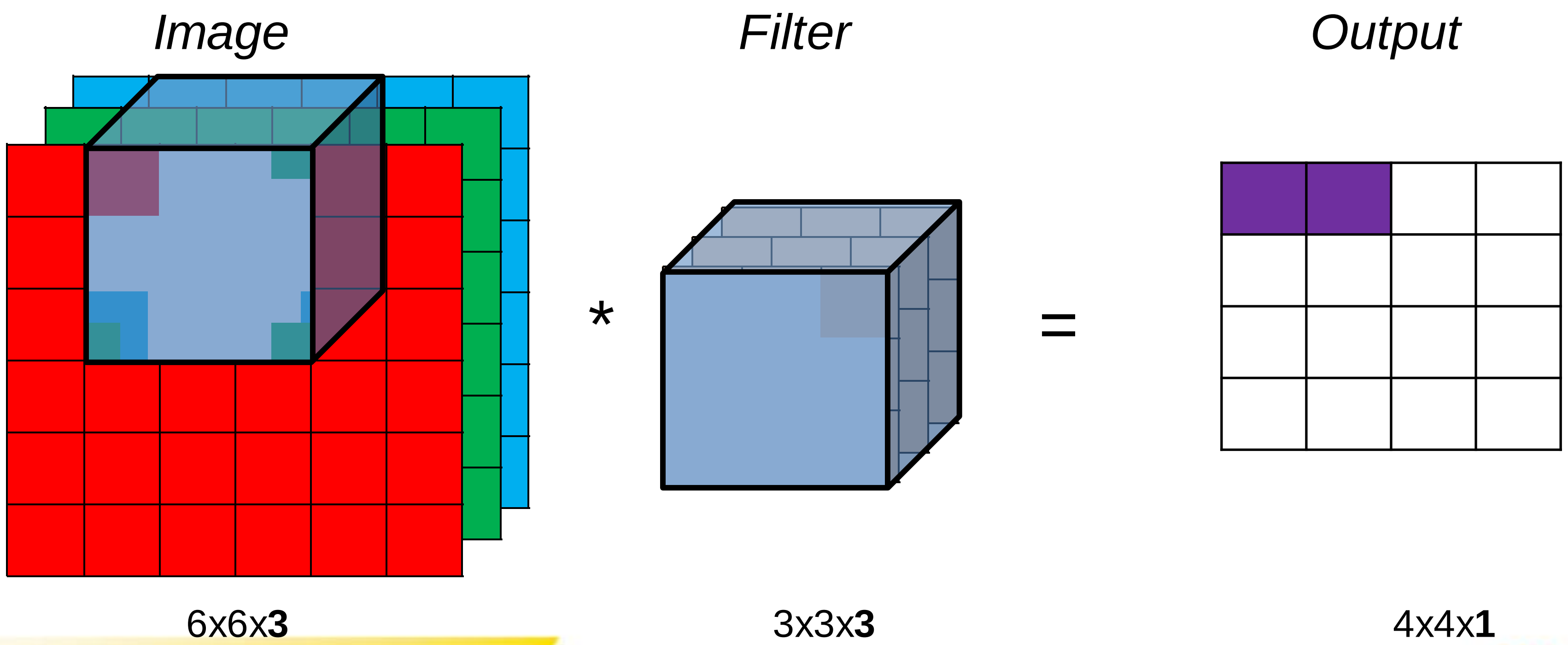
Convolution on Colour images



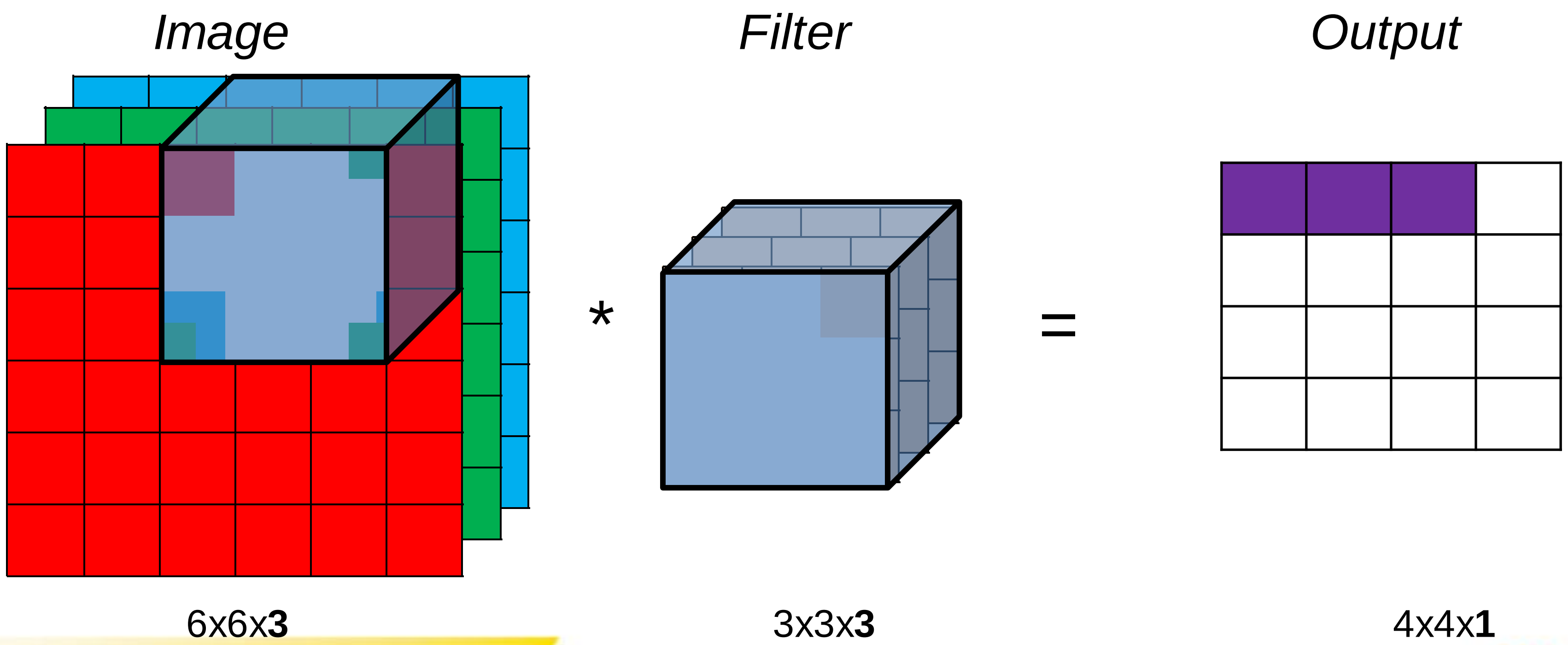
Convolution on Colour images



Convolution on Colour images

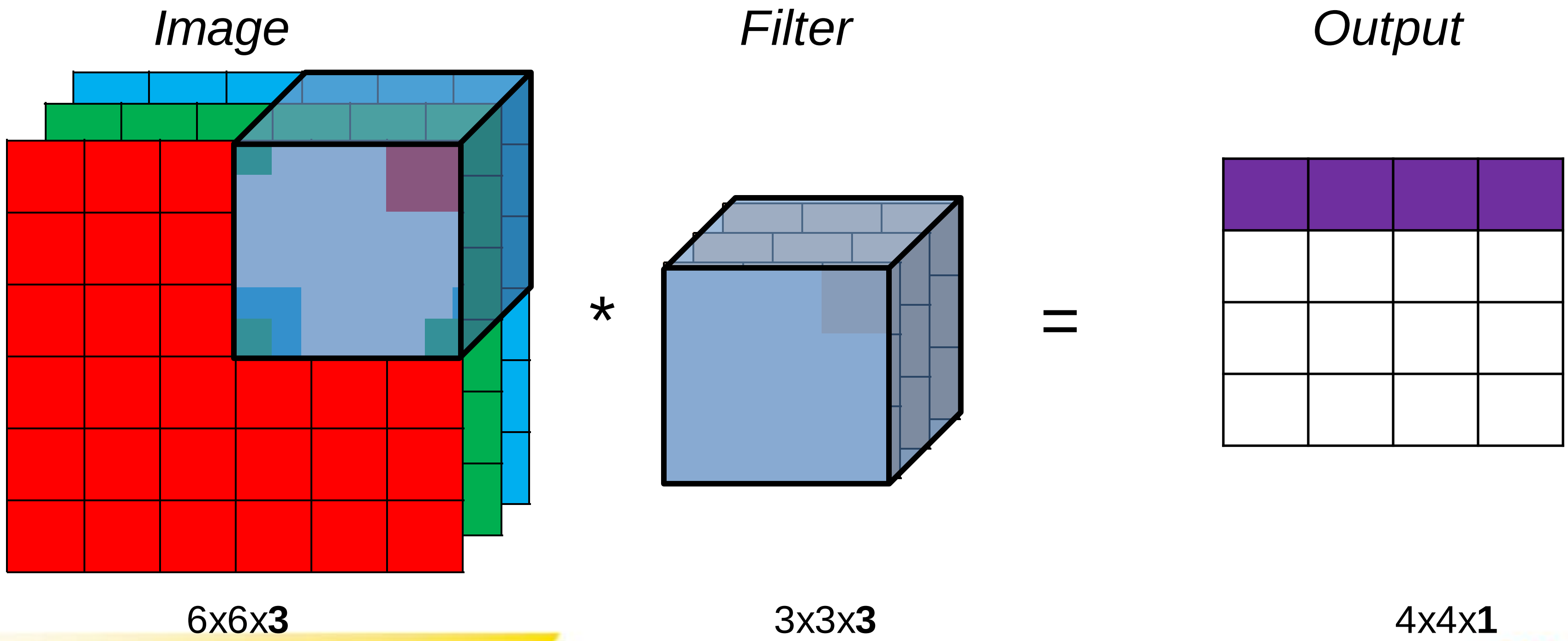


Convolution on Colour images

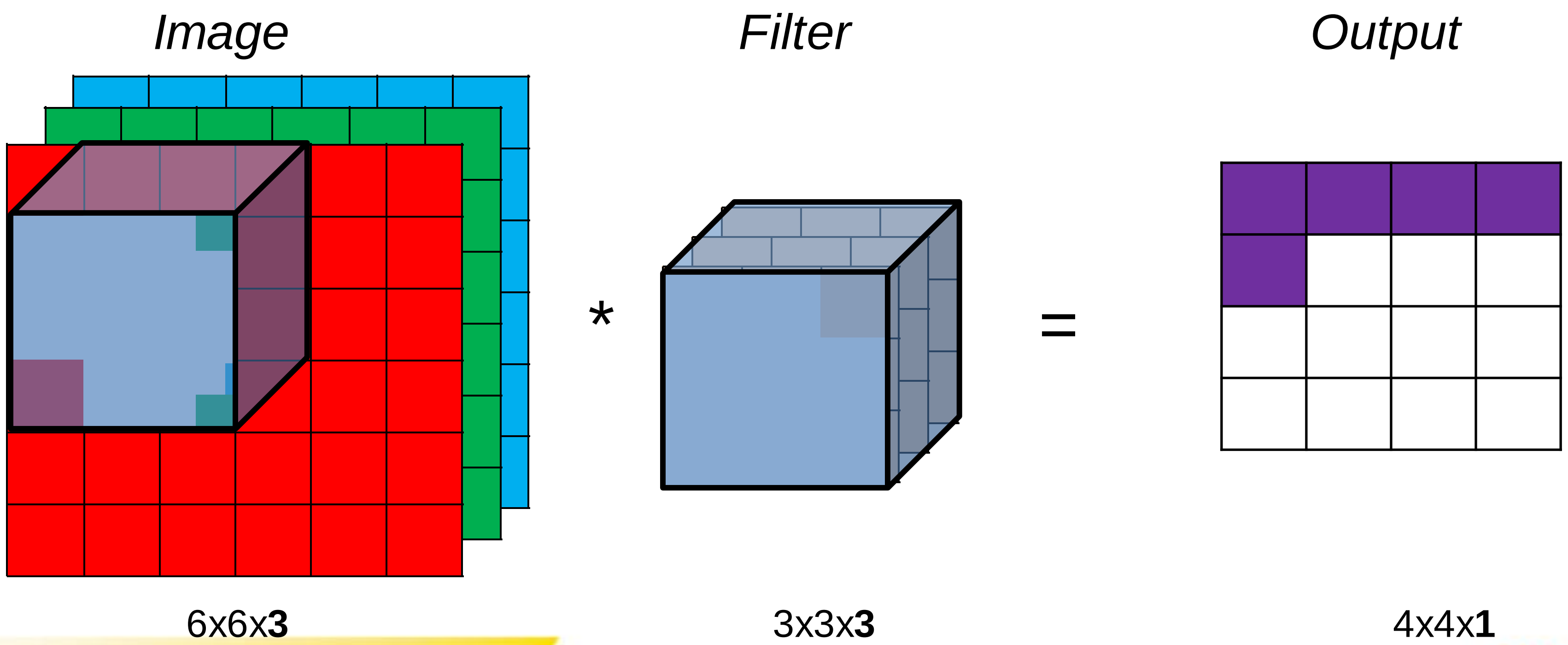


UMKC

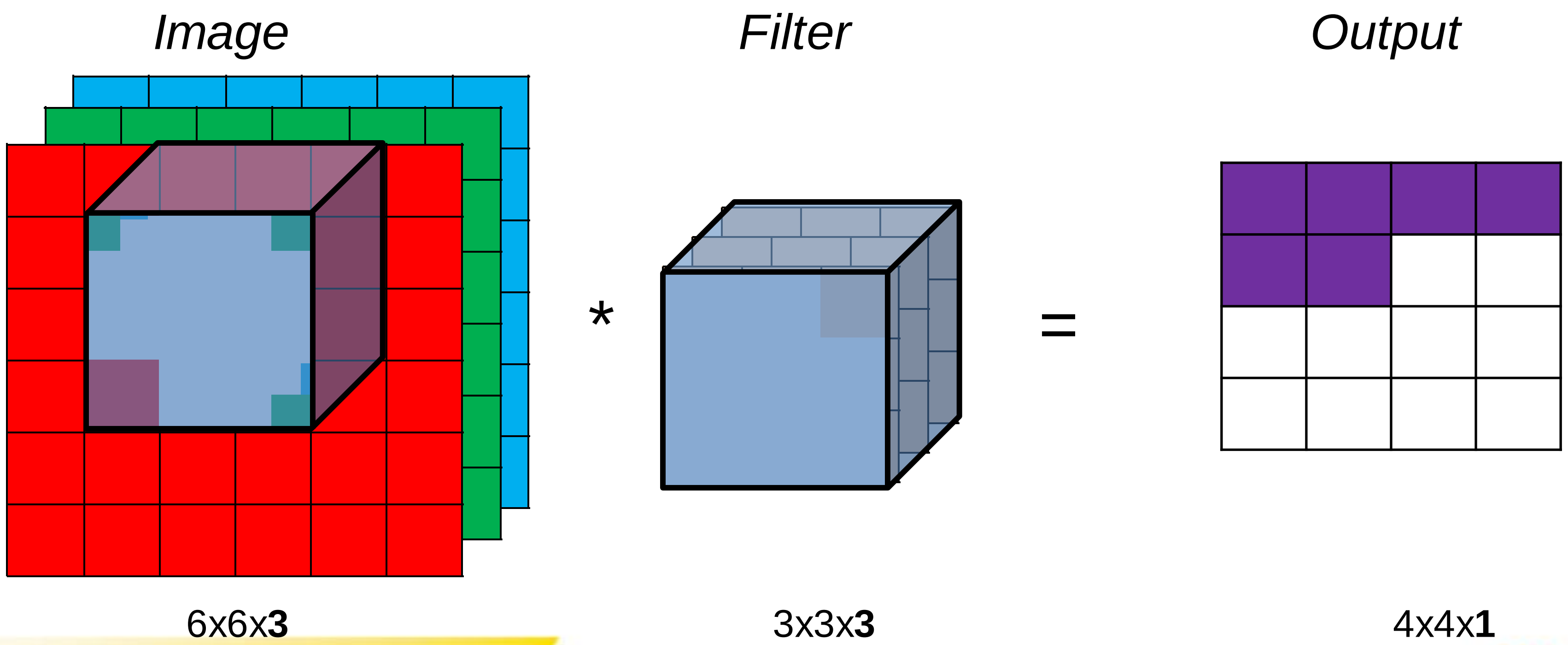
Convolution on Colour images



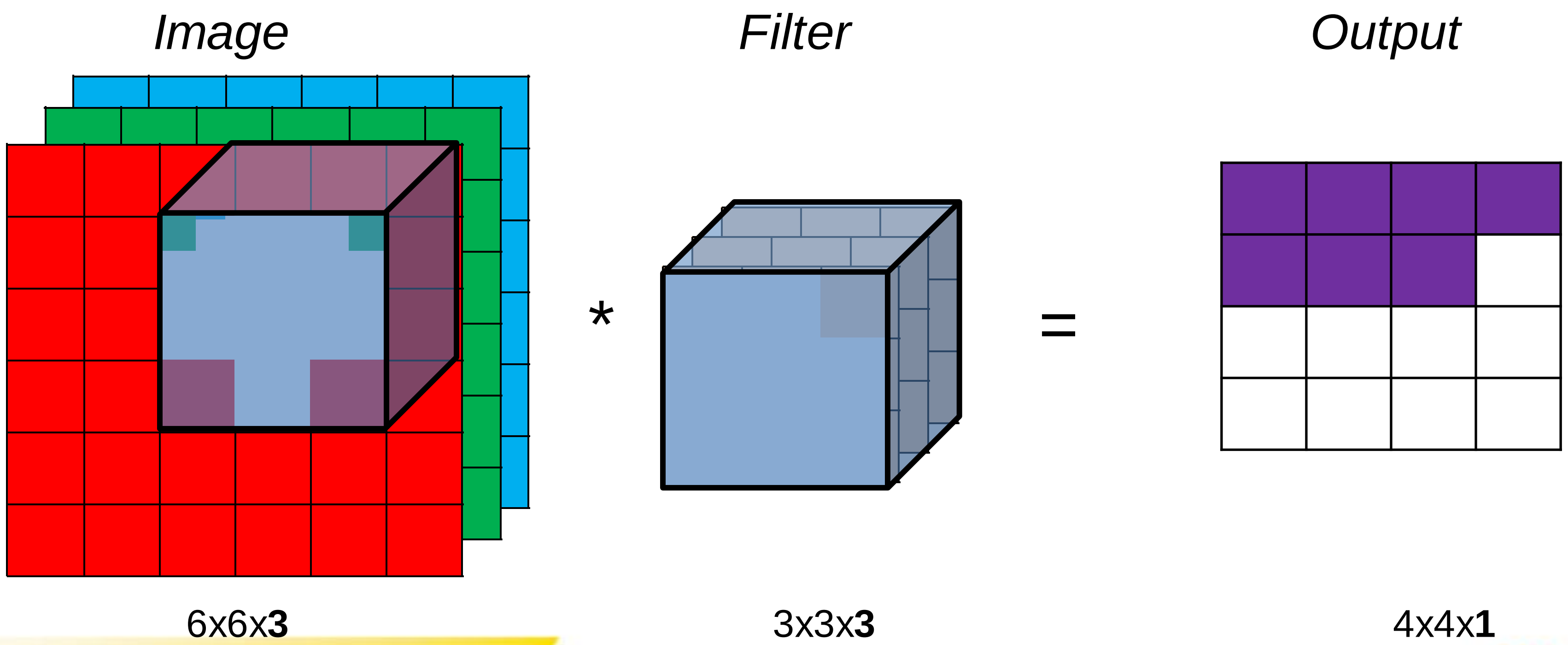
Convolution on Colour images



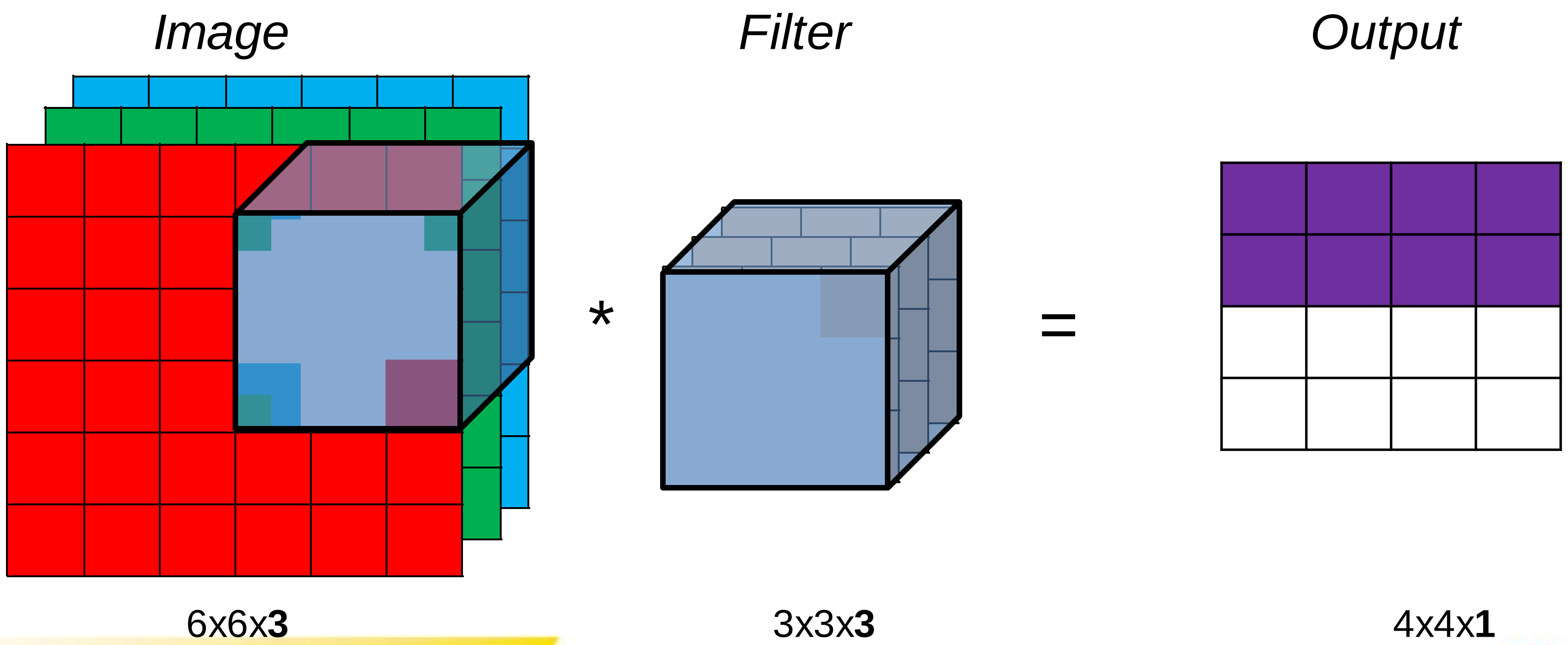
Convolution on Colour images



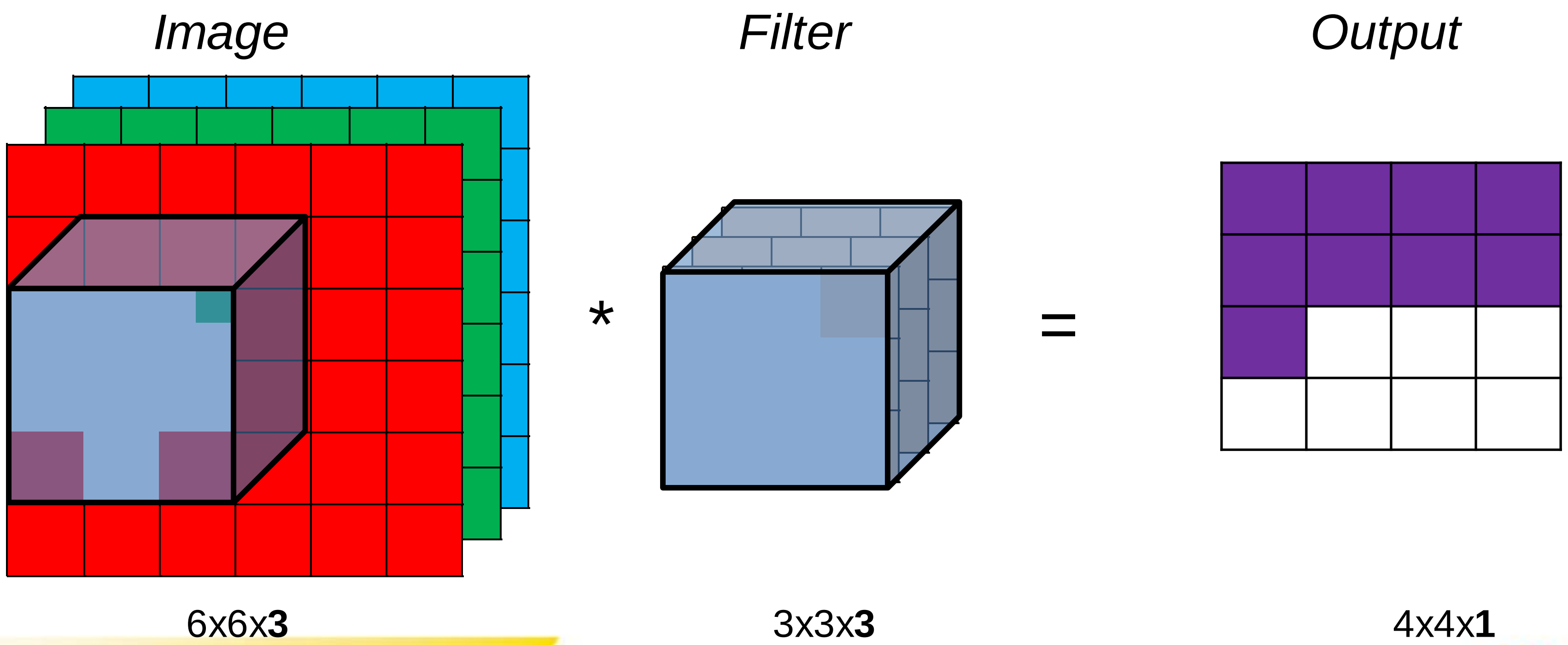
Convolution on Colour images



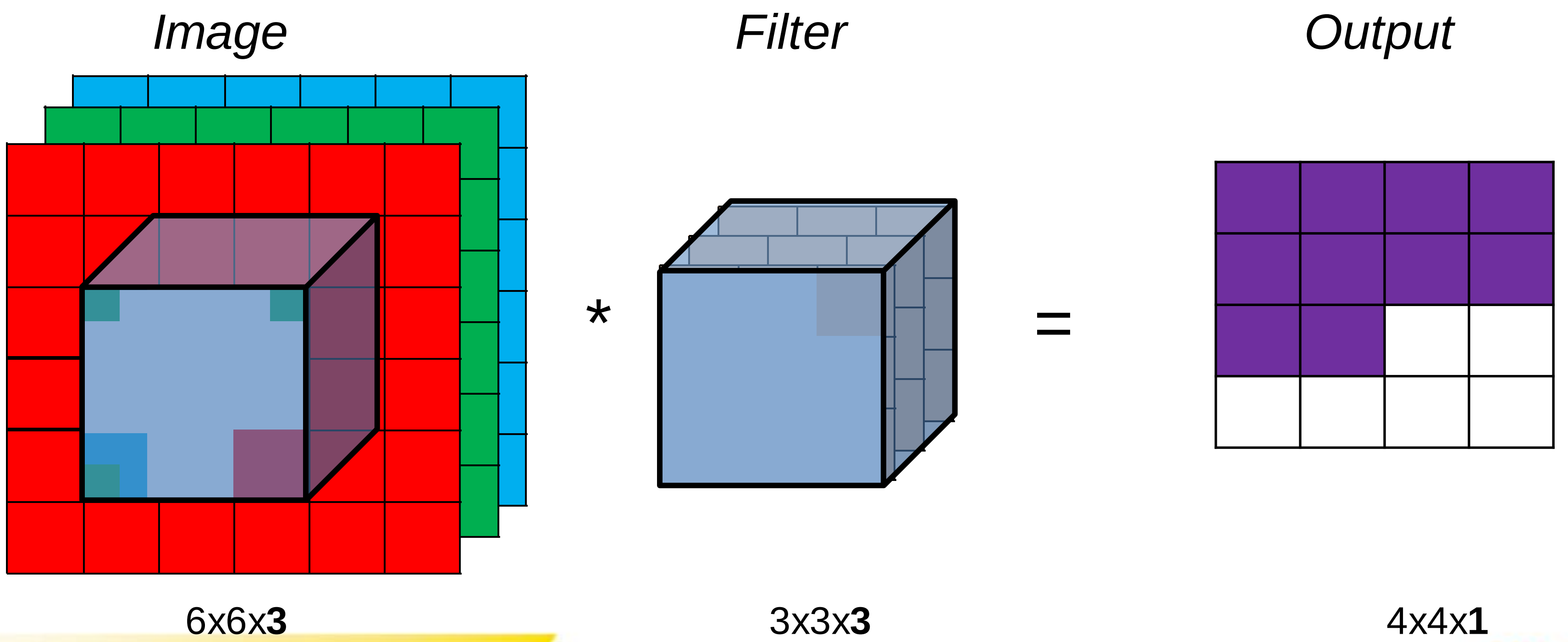
Convolution on Colour images



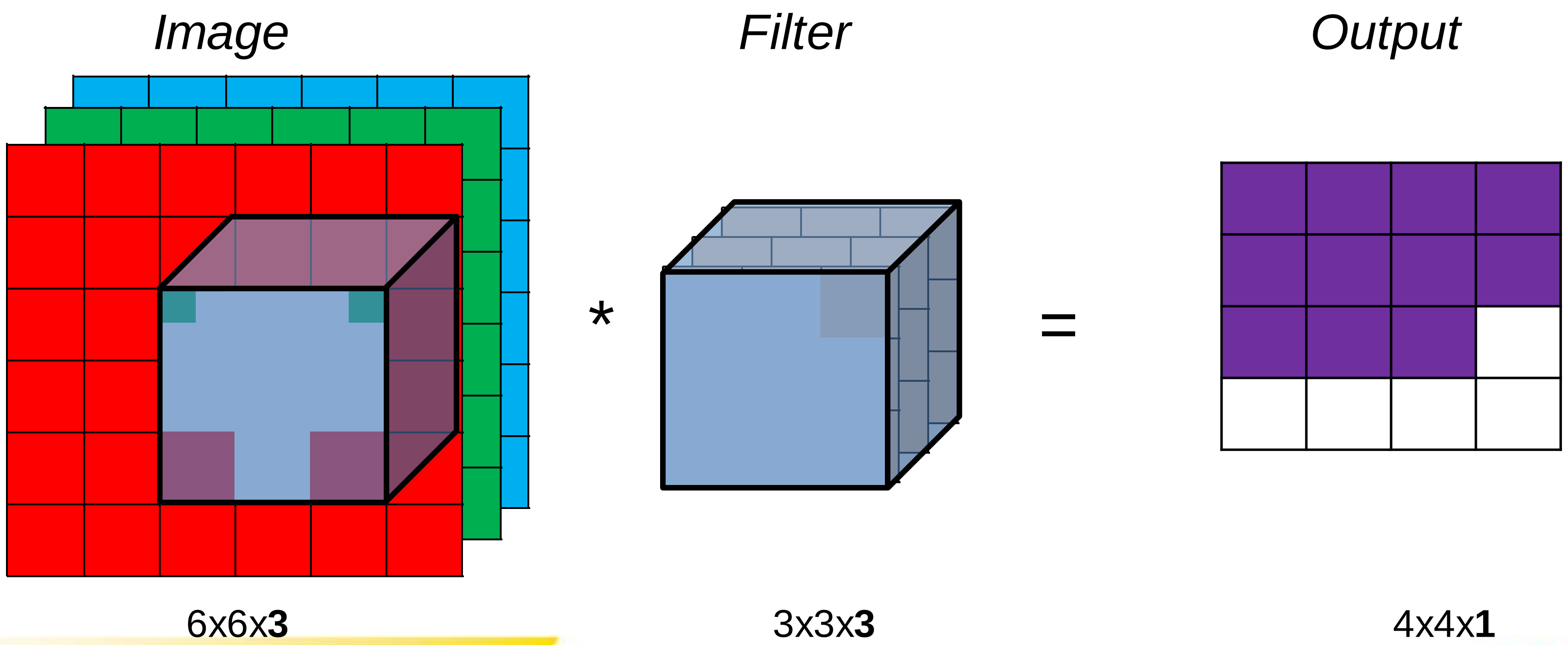
Convolution on Colour images



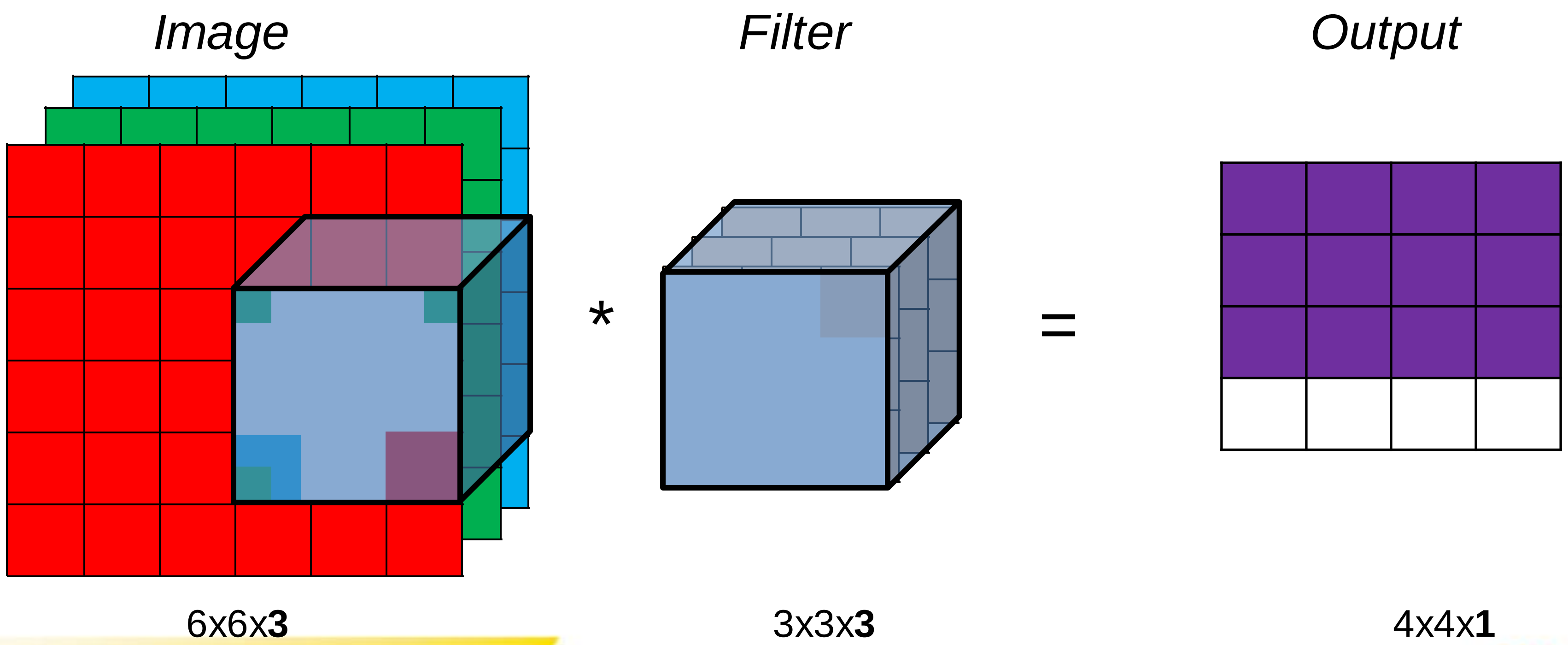
Convolution on Colour images



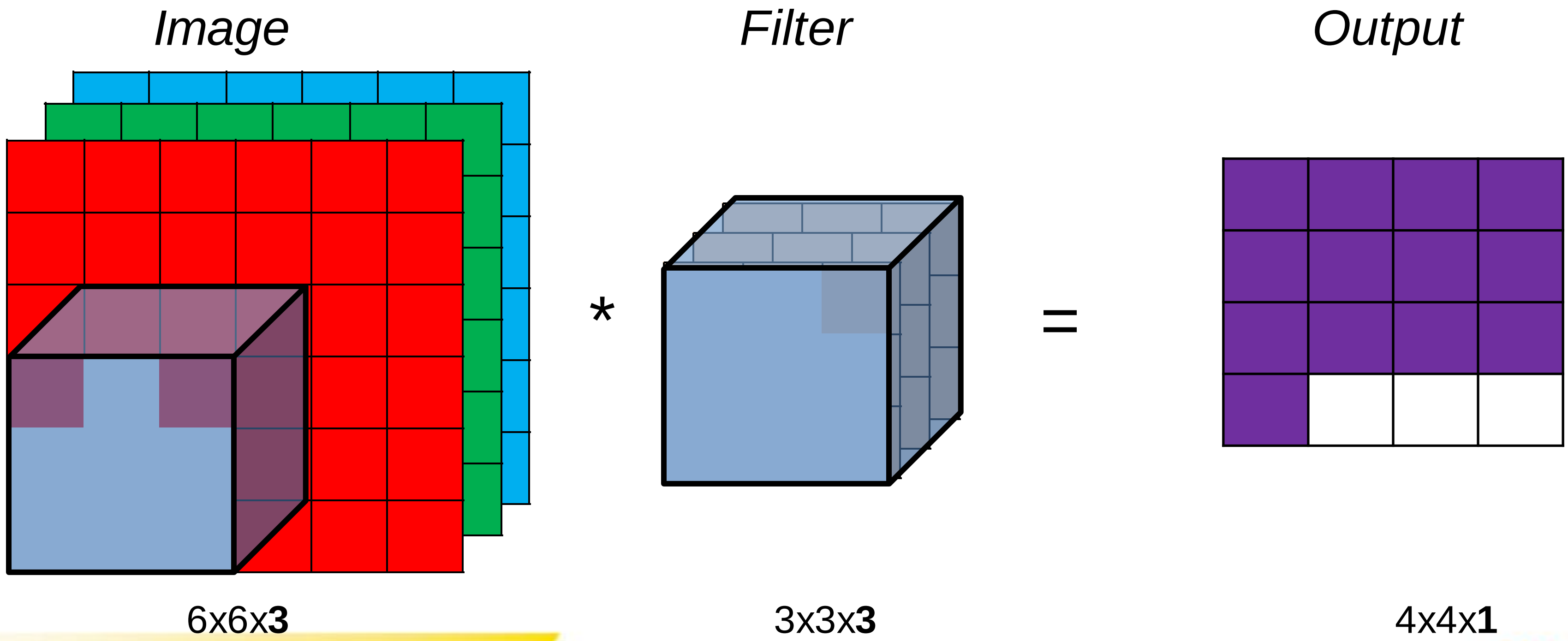
Convolution on Colour images



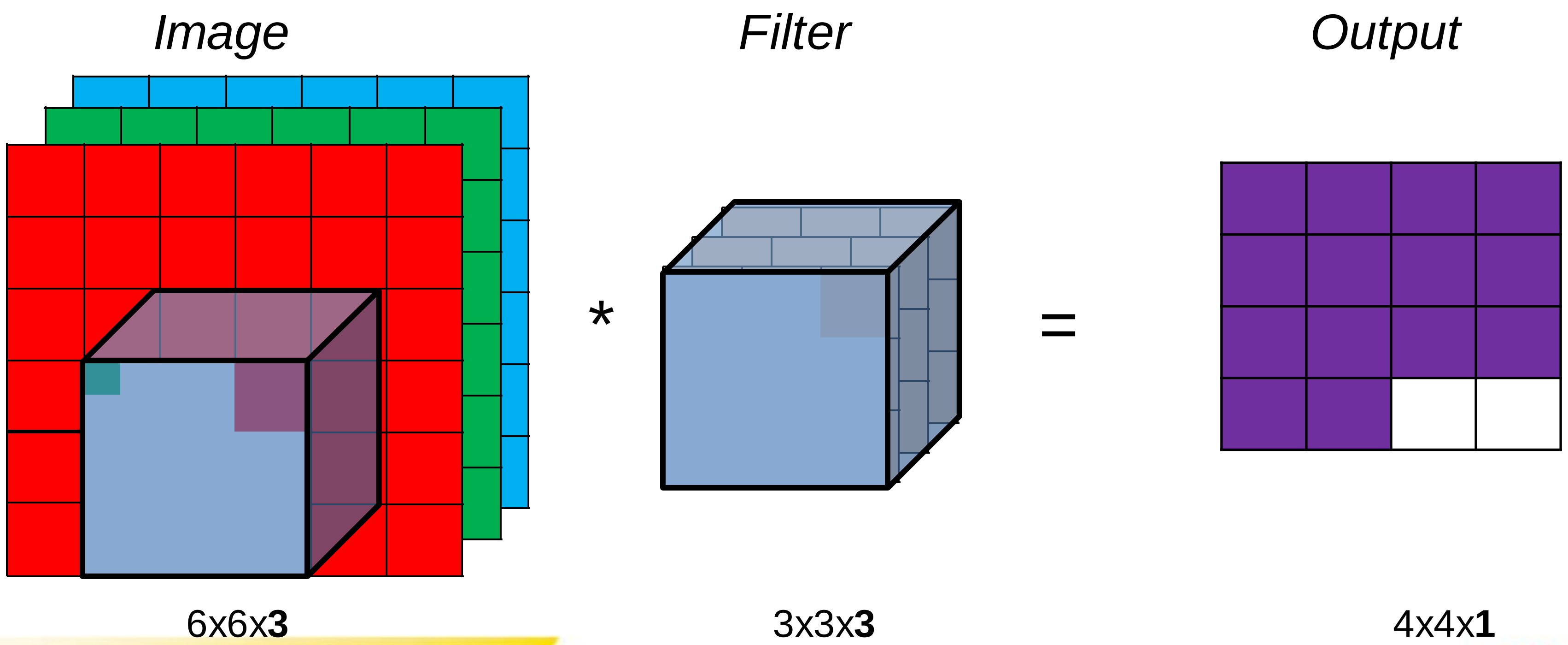
Convolution on Colour images



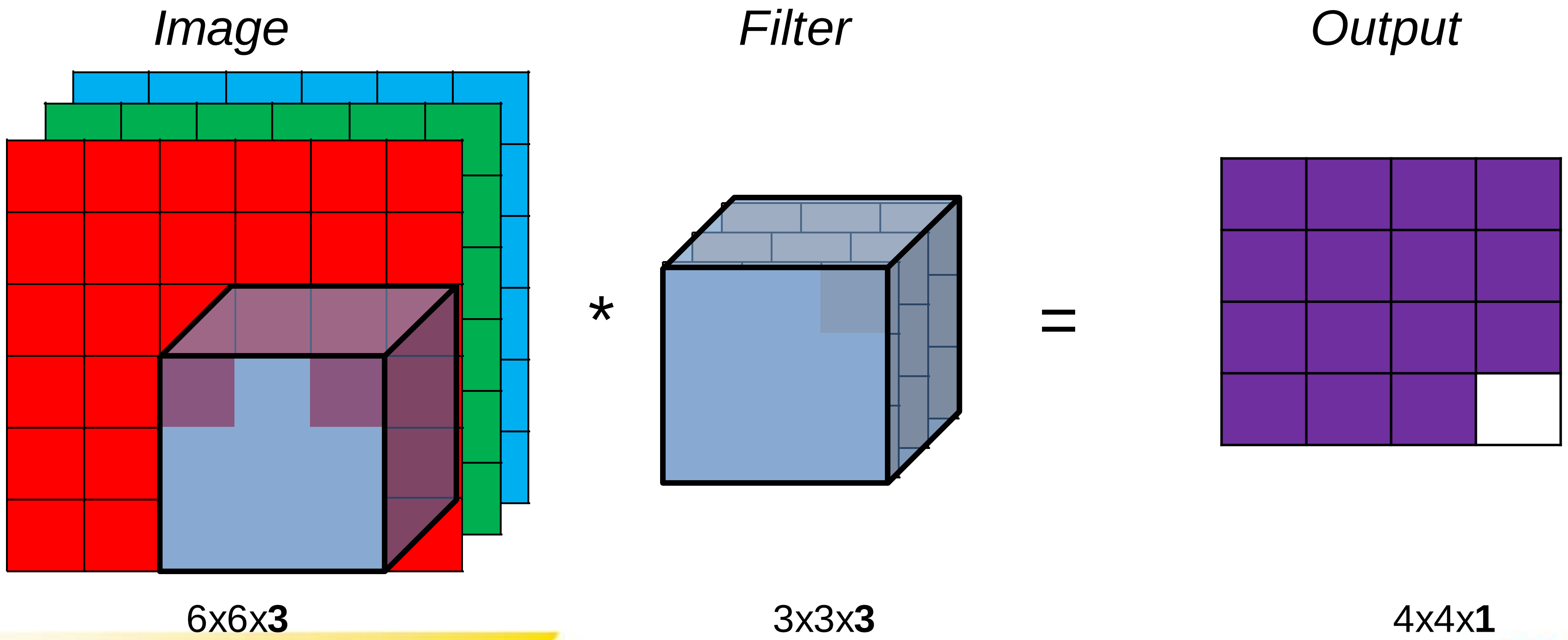
Convolution on Colour images



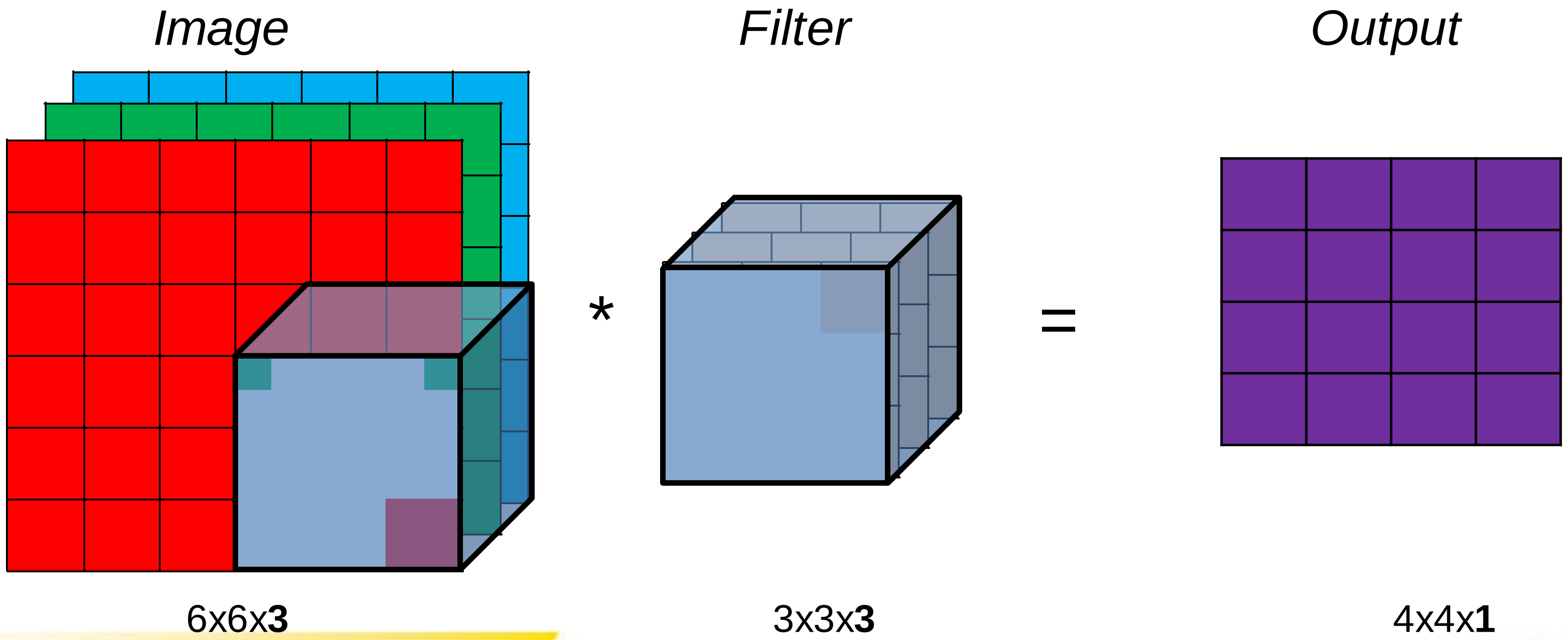
Convolution on Colour images



Convolution on Colour images



Convolution on Colour images



Pooling Layer

Purpose:

1) To reduce the size of the layer to speed up computation.

2) To retain robust features.

Types:

1) Max Pooling

2) Average Pooling

Pooling Layer

1	3	2	1
3	6	8	9
8	3	3	9
4	6	8	2

$\xrightarrow{\text{Max}}$

6	9
8	9

Pooling Layer

1	3	2	1
3	6	8	9
8	3	3	9
4	6	8	2

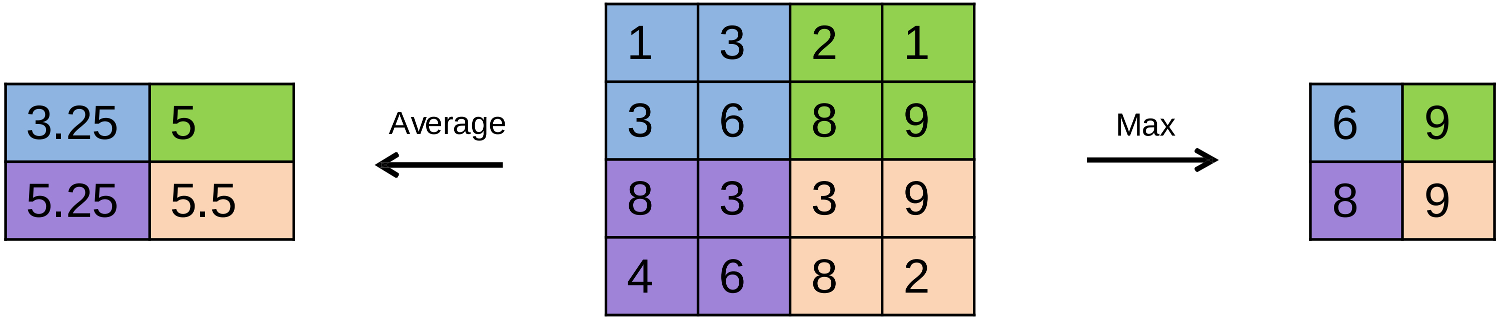
Average

3.25	5
5.25	5.5

Max

6	9
8	9

Pooling Layer



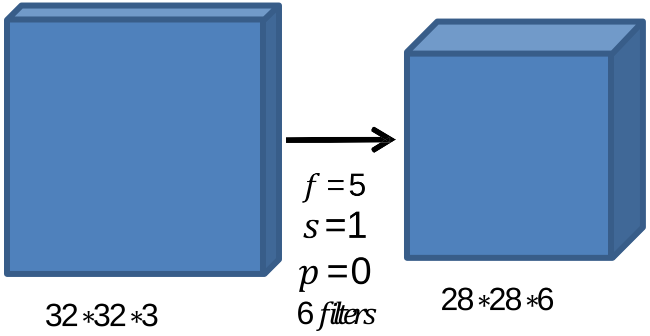
Salient features:

1)Follows the process of convolution with filters of size f, stride s and padding p (not used much) as the hyperparameters.

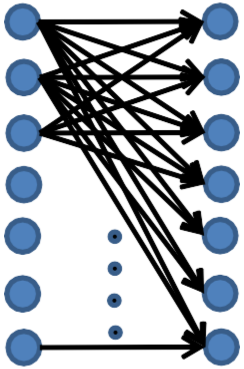
2) Filters have no weights. So, no trainable parameters.

3)Pooling operation is carried out channel-wise. Thus, number of channels remain unchanged.

Why Convolutional ?



FC Neural Network



$32*32*3$
$=3072$

$28*28*6$
$=4704$

$3072*4704$
$=\mathbf{1,44,50,688}$ parameters

Conv. Neural Network

$f=5$
$s=1$
$p=0$
6 filters

6 filters * 5*5*3
= 450 weights + 6 biases
= **456** parameters

Why Convolutional ?

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

*

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

=

0			

Why Convolutional ?

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

*

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

=

0	30		

Parameter Sharing: A feature detector that is useful in one part of the image may also be useful on another part of the same image.

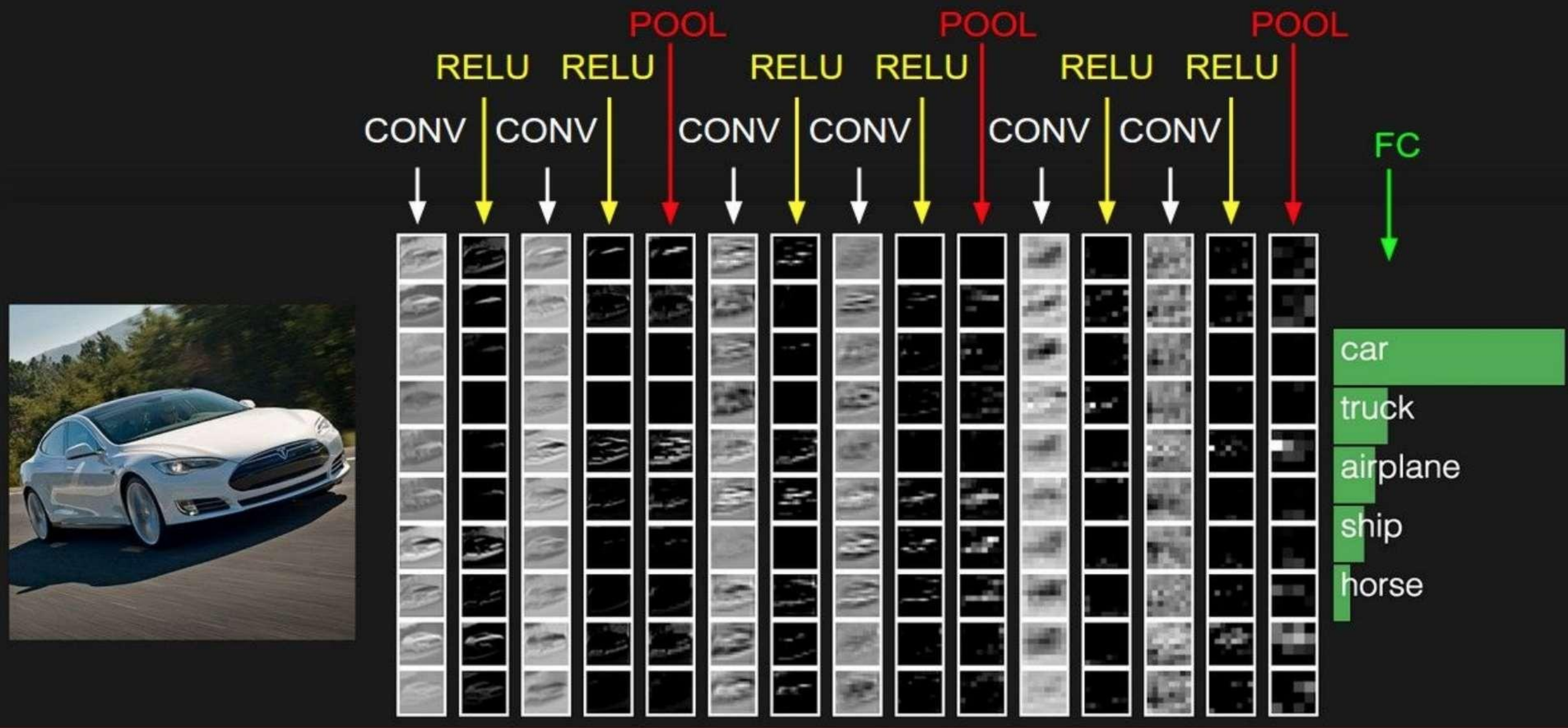
Why Convolutional ?

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

Parameter Sharing: A feature detector that is useful in one part of the image may also be useful on another part of the same image.

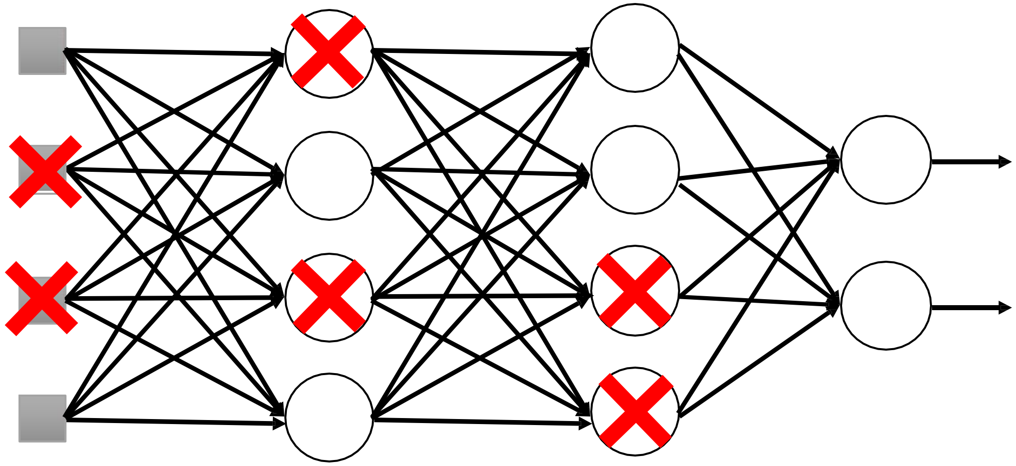
Sparcity of connections: In each layer, output depends only on small number of inputs.

LAYERS IN CNN



Dropout

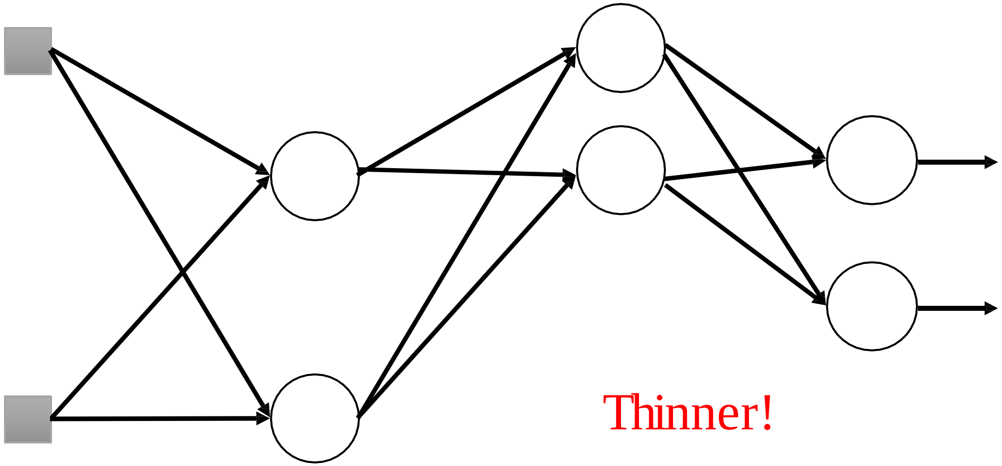
Training:



- **Each time before updating the parameters**
  - Each neuron has p% to dropout

Dropout

Training:



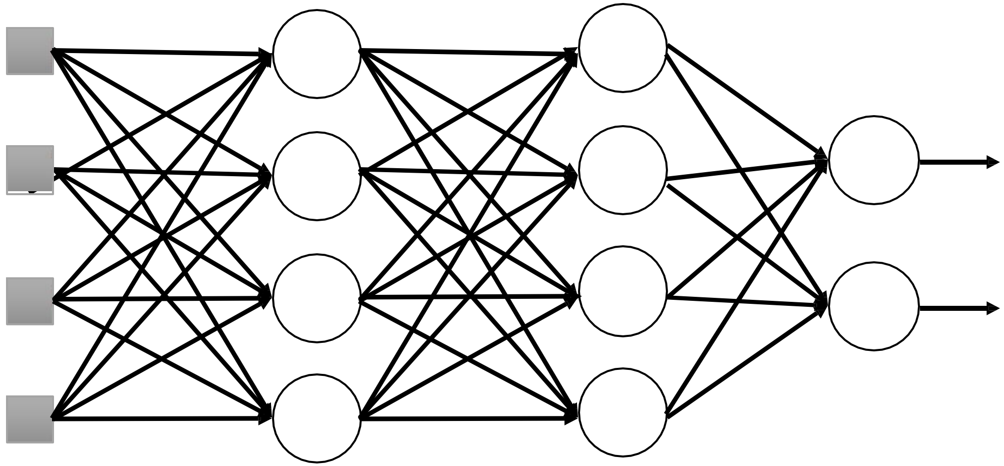
Thinner!

- **Each time before updating the parameters**
  - Each neuron has p% to dropout
    - **The structure of the network is changed.**
  - Using the new network for training

For each mini-batch, we resample the dropout neurons

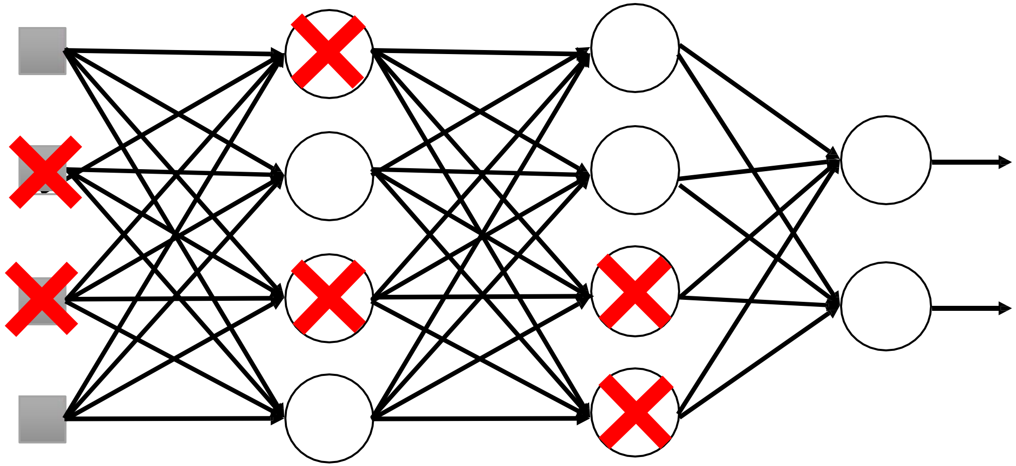
Dropout

Testing:



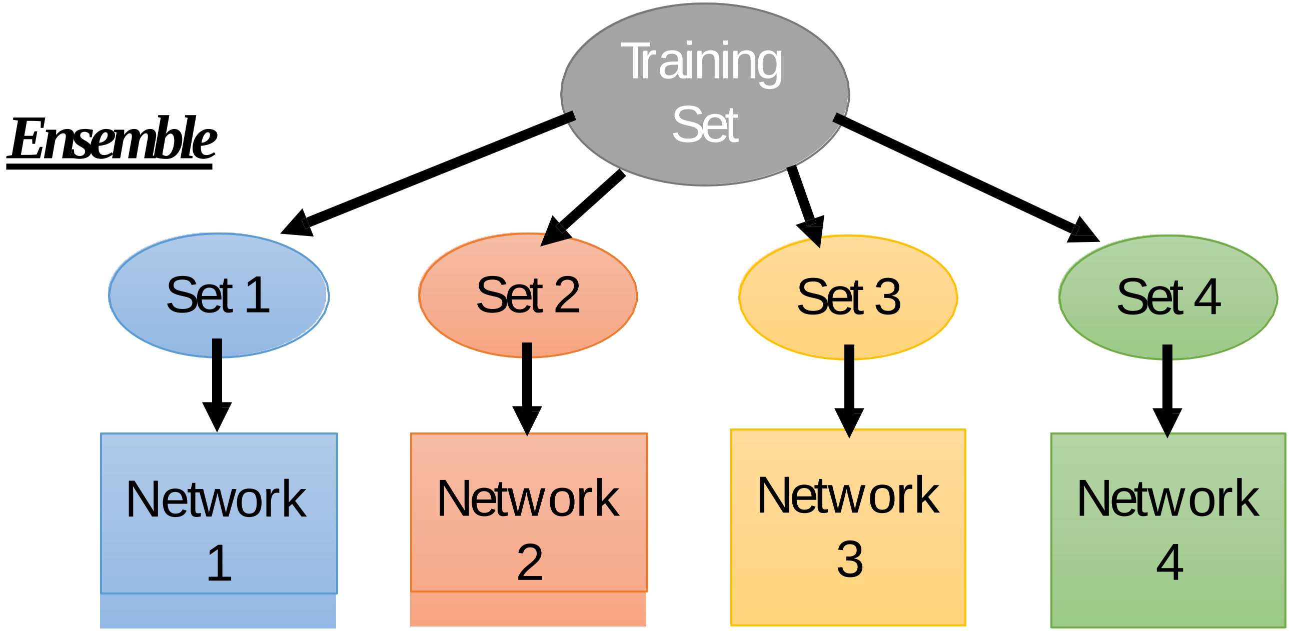
- **No dropout**
  - If the dropout rate at training is p%, all the weights times (1-p)%
  - Assume that the dropout rate is 50%. If a weight $w = 1$ by training, set $w = 0.5$ for testing.

Dropout - Intuitive Reason



- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

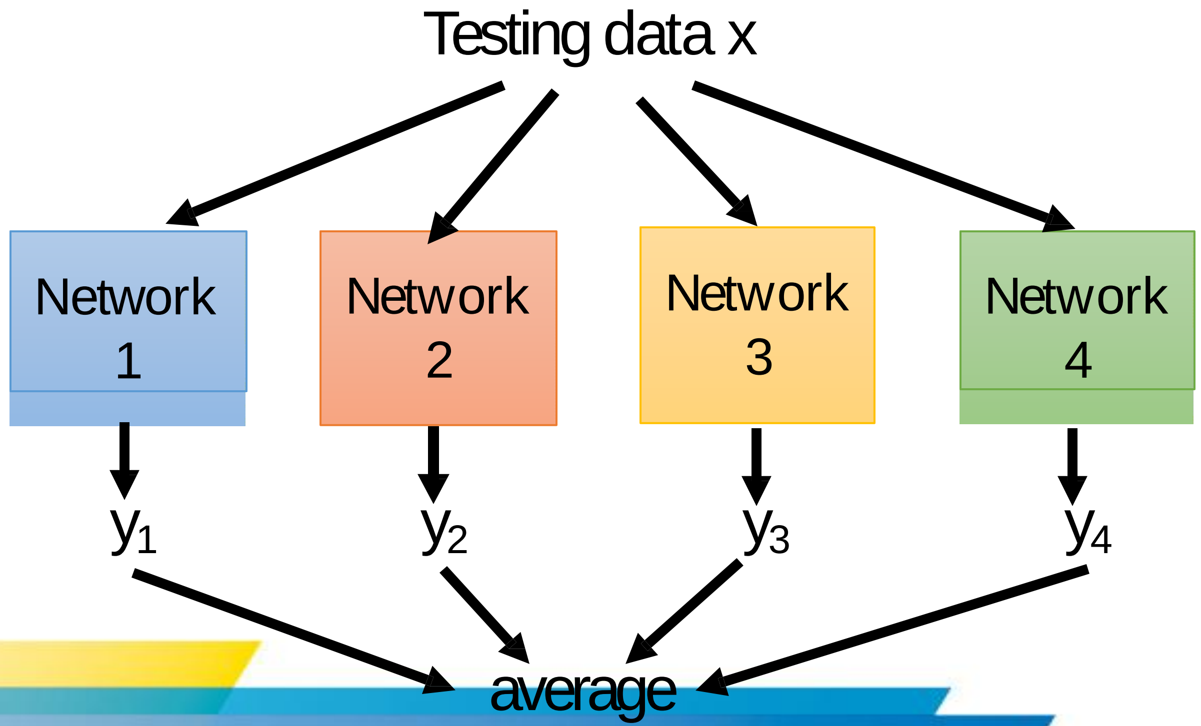
Dropout is a kind of ensemble.



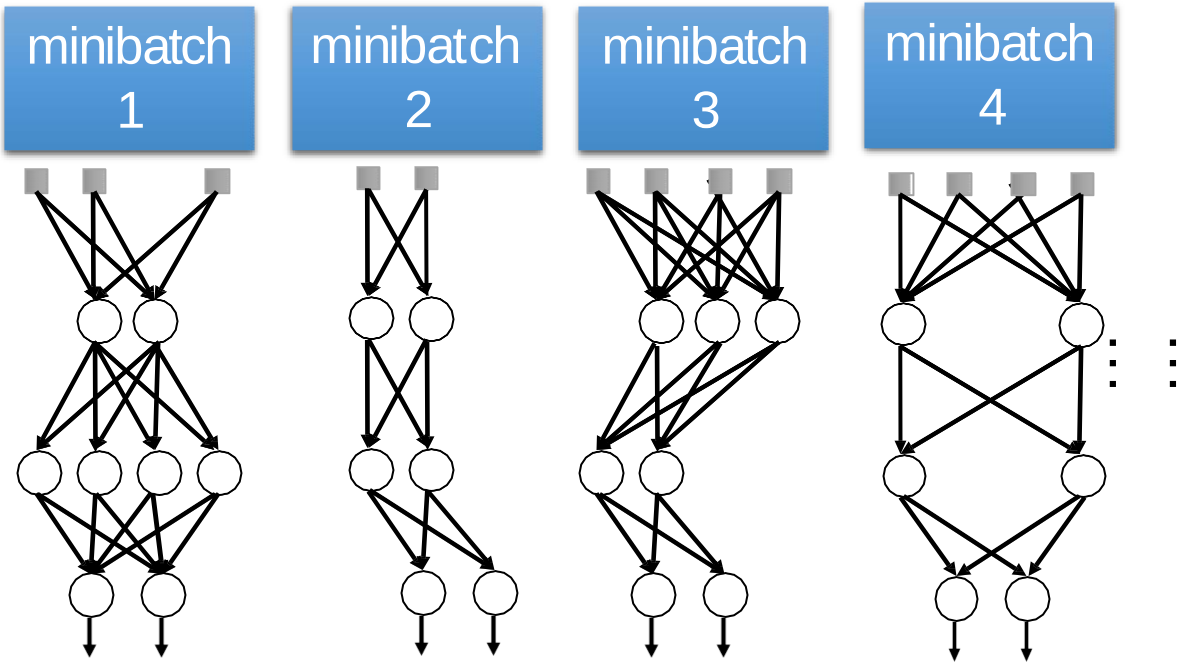
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble



Dropout is a kind of ensemble.



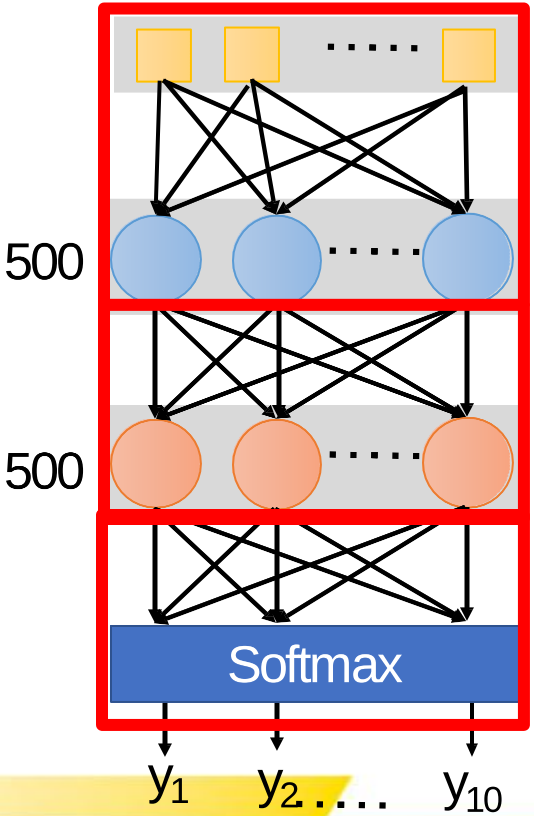
<u>Training of Dropout</u>

M neurons

2^M possible networks

- ➢Using one mini-batch to train one network
- ➢Some parameters in the network are shared

Let's try it




```
model = Sequential()
```

```
model.add( Dense( input_dim=28*28,
                  output_dim=500 ))
model.add( Activation('sigmoid') )
```

model.add(dropout(0.8))

```
model.add( Dense( output_dim=500 ) )
model.add( Activation('sigmoid') )
```

model.add(dropout(0.8))

```
model.add( Dense(output_dim=10 ) )
model.add( Activation('softmax') )
```

A few important terms

- ***Forward pass***: Process of passing the data from input layer to output layer.
- ***Cost Function***: Difference between the predicted and true output of a NN.
- ***Backpropagation***: The process of updating parameters of a network depending on the cost function (using optimization algorithms viz., Gradient Descent, SGDM, ADAGrad, etc.) to minimize the cost.
- ***Mini-batch***: Number of images passing at once through the network.
- ***Learning Rate***: Speed by which the parameters are updated.
- ***Iteration***: A mini-batch performing a forward and a backward pass through the network is an iteration.
- ***Epoch***: When the complete dataset undergoes a forward and a backward pass, an epoch is completed.

Visualize the graph on Tensorboard

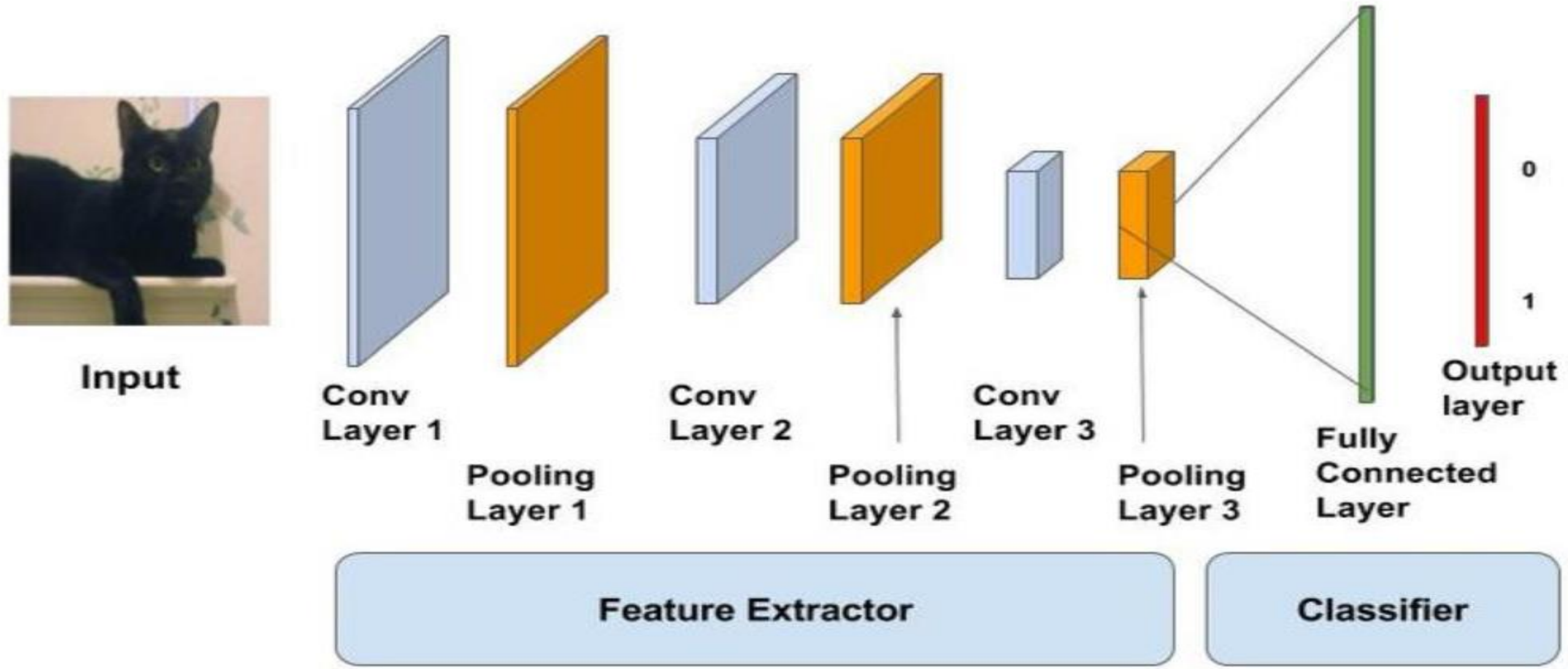
```
tbCallBack = keras.callbacks.TensorBoard(log_dir='./Graph', histogram_freq=0,
 write_graph=True, write_images=True)
model.fit(...inputs and parameters..., callbacks=[tbCallBack])
```

If you want to visualize the files created during training, run in your terminal

```
tensorboard --logdir path_to_current_dir/Graph
```

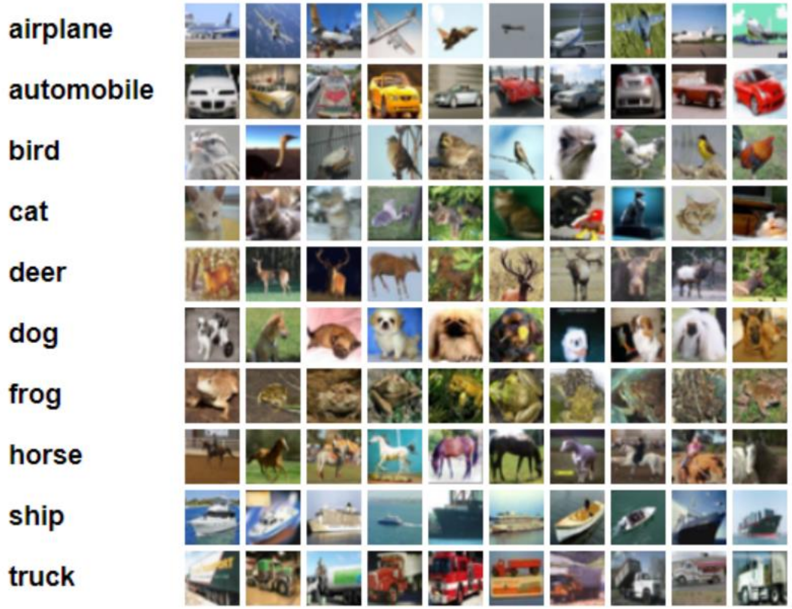
Use case: Image classification

Image Classification on Cifar10



Data set: Cifar10

- containing 60.000 different images
- size of all images in this dataset is 32x32x3 (RGB)



Load data set

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)

X_train /= 255.0
X_test /= 255.0
```

Using one hot encoding to transform the output variable into a binary matrix

Normalizing the data

structure of the model

- We will use a structure with two convolutional layers
- followed by max pooling
- and a flattening out of the network to fully connected layers to make predictions

Our baseline network structure can be summarized as follows:

- Convolutional input layer, 32 feature maps with a size of 3×3, a rectifier activation function and a weight constraint of max norm set to 3.
- Dropout set to 20%.
- Convolutional layer, 32 feature maps with a size of 3×3, a rectifier activation function and a weight constraint of max norm set to 3.
- Max Pool layer with size 2×2.
- Flatten layer.
- Fully connected layer with 512 units and a rectifier activation function.
- Dropout set to 50%.
- Fully connected output layer with 10 units and a softmax activation function

UMKC

Convolutional layer

```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))
```

Flatten layer

```
model.add(Flatten())
model.add(Dense(80))
model.add(Activation('relu'))
model.add(Dropout(0.3))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

Fitting model

```
model.compile(loss='categorical_crossentropy',
              optimizer=SGD,
              metrics=['accuracy'])
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_split=0.2,
          shuffle=True)
scores = model.evaluate(x_test, y_test, verbose=1)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
model.save('./model' + '.h5')
```

References

- http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/
- https://towardsdatascience.com/a-walkthrough-of-convolutional-neural-network-7f474f91d7bd
- http://cs231n.github.io/convolutional-networks/
- https://www.youtube.com/watch?v=AgkfIQ4IGaM
- https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/
- https://www.youtube.com/watch?v=YRhxdVk_sIs
- https://www.youtube.com/watch?v=FmpDIaiMIeA&t=748s
- https://medium.com/@2017csm1006/forward-and-backpropagation-in-convolutional-neural-network-4dfa96d7b37e

UMKC