Convolutional Neural Networks
- A breakthrough in Computer Vision

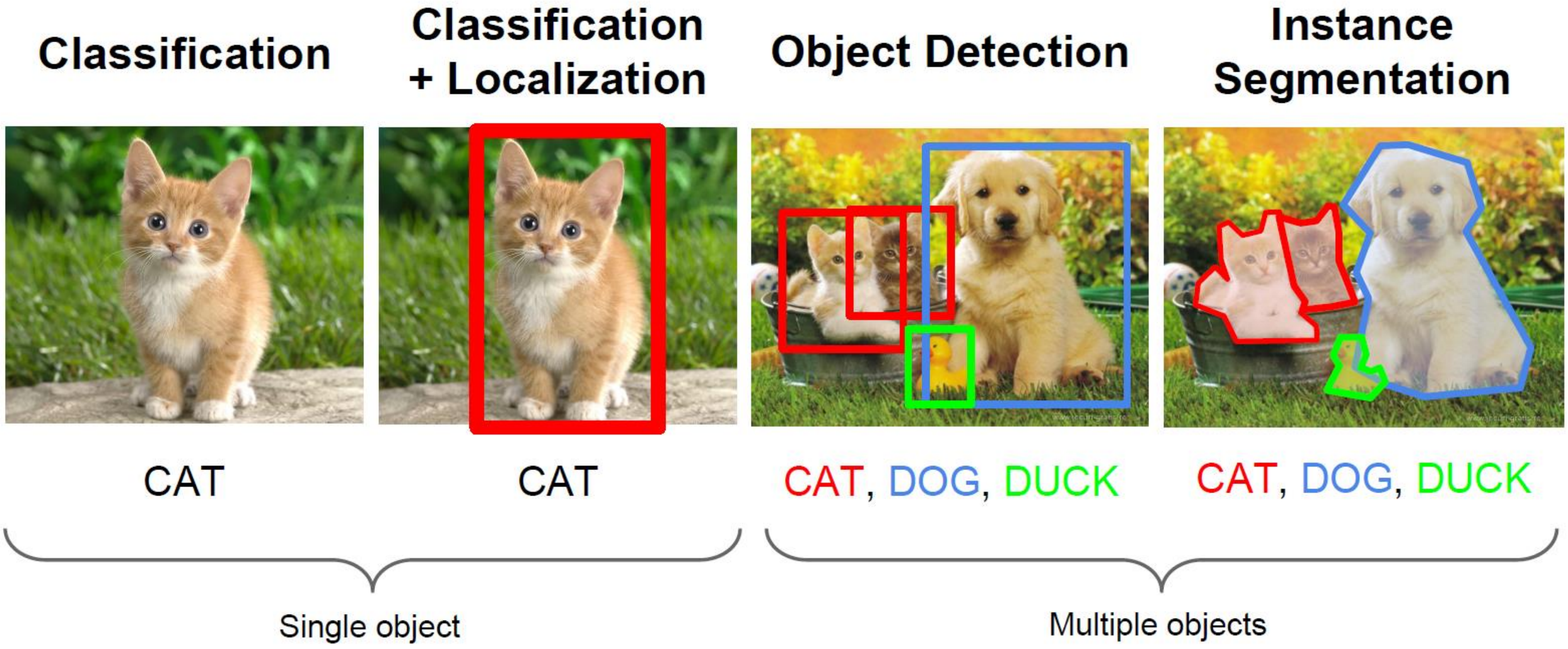
by

Paresh Kamble & Rucha Gole
PhD Student MTech Student
Department of Electronics and Communication Engineering,
VNIT, Nagpur, Maharashtra, India.

Introduction

- Pre-requisite: i) Linear Algebra, ii) Neural Network
- Ted talk - Fei Fei Li – How we teach computers to understand pictures
- Buzz about Deep Learning and CNN
- Applications of CNN – Image recognition, video analysis, medical imaging, Games, etc.
- Companies that use CNN – Google, Facebook, Twitter, Instagram, IBM, Intel, etc.
- Disclaimer – The presentation is loosely based on Coursera – Deep Learning Specialization and CS231n course of Stanford University.

Common problems in Computer Vision

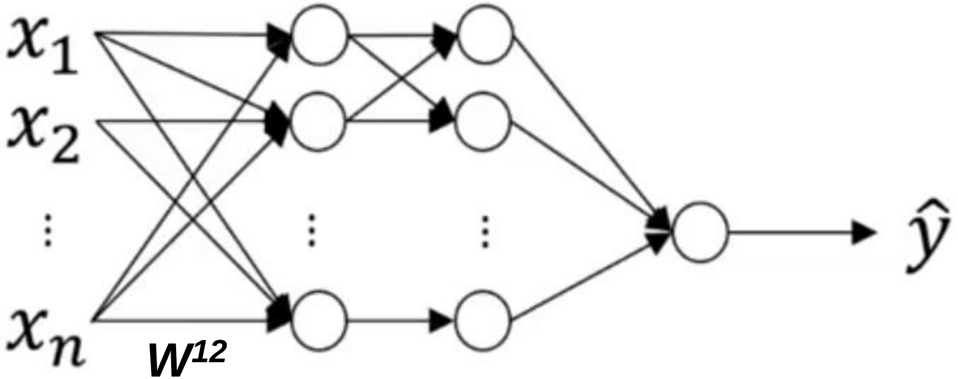


Traditional Neural Network vs. Convolutional Neural Network

32

32

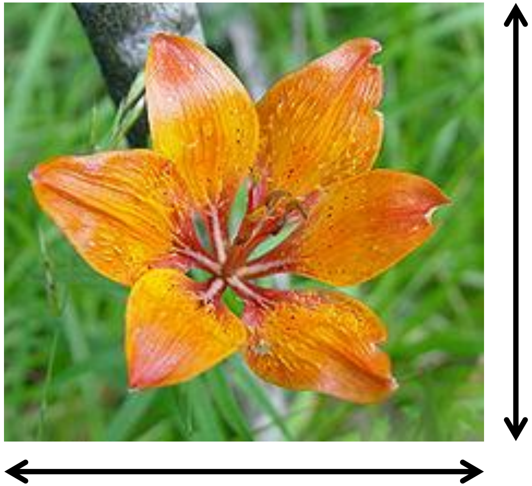
32 x 32 x 3 = ***3072*** ***1000***



W^{12}

1000 x 3072
dimensional matrix

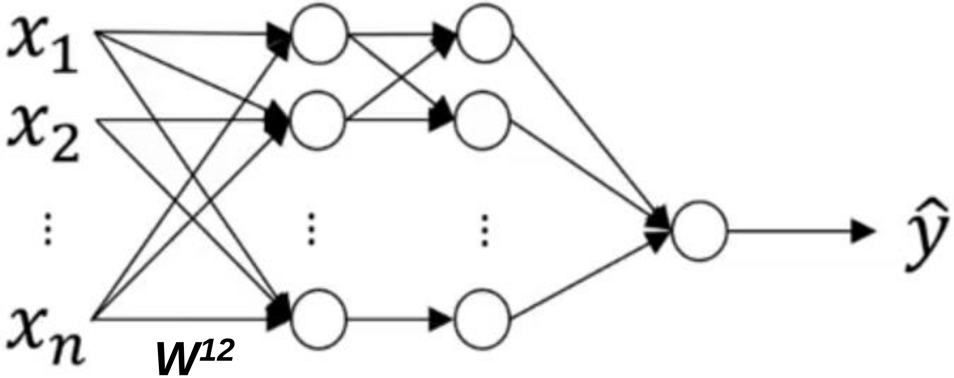
Traditional Neural Network vs. Convolutional Neural Network



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***

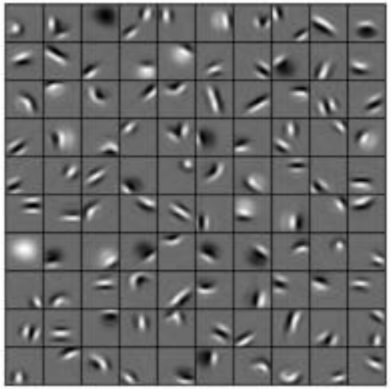
Traditional Neural Network vs. Convolutional Neural Network

Salient points:

- Spatial relation between features in image is not considered in NN.
- NN needs huge data to prevent overfitting!
- 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.

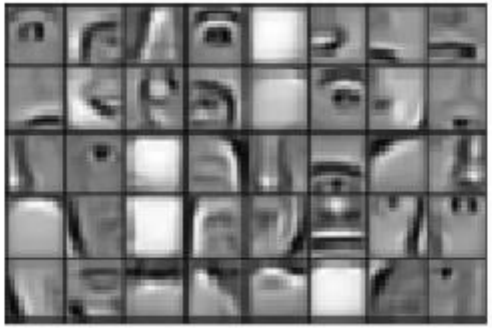
Stages of feature extraction by CNN

Low level features



Edges, curves and colour

Mid level features



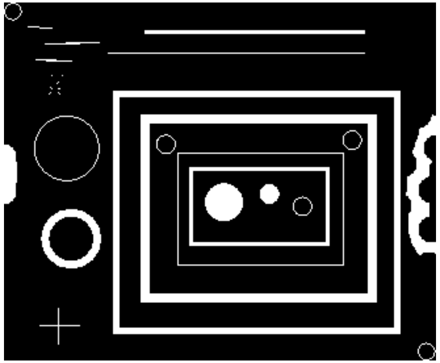
Parts of objects

High level features

Complete objects

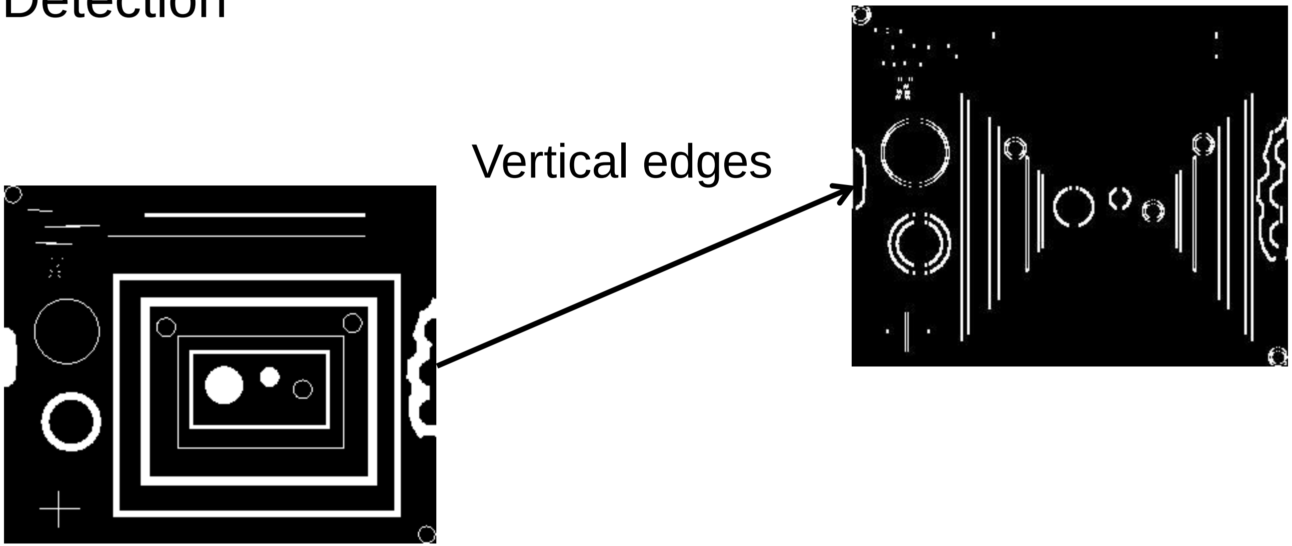
Foundation of Convolutional Neural Networks

- Edge Detection



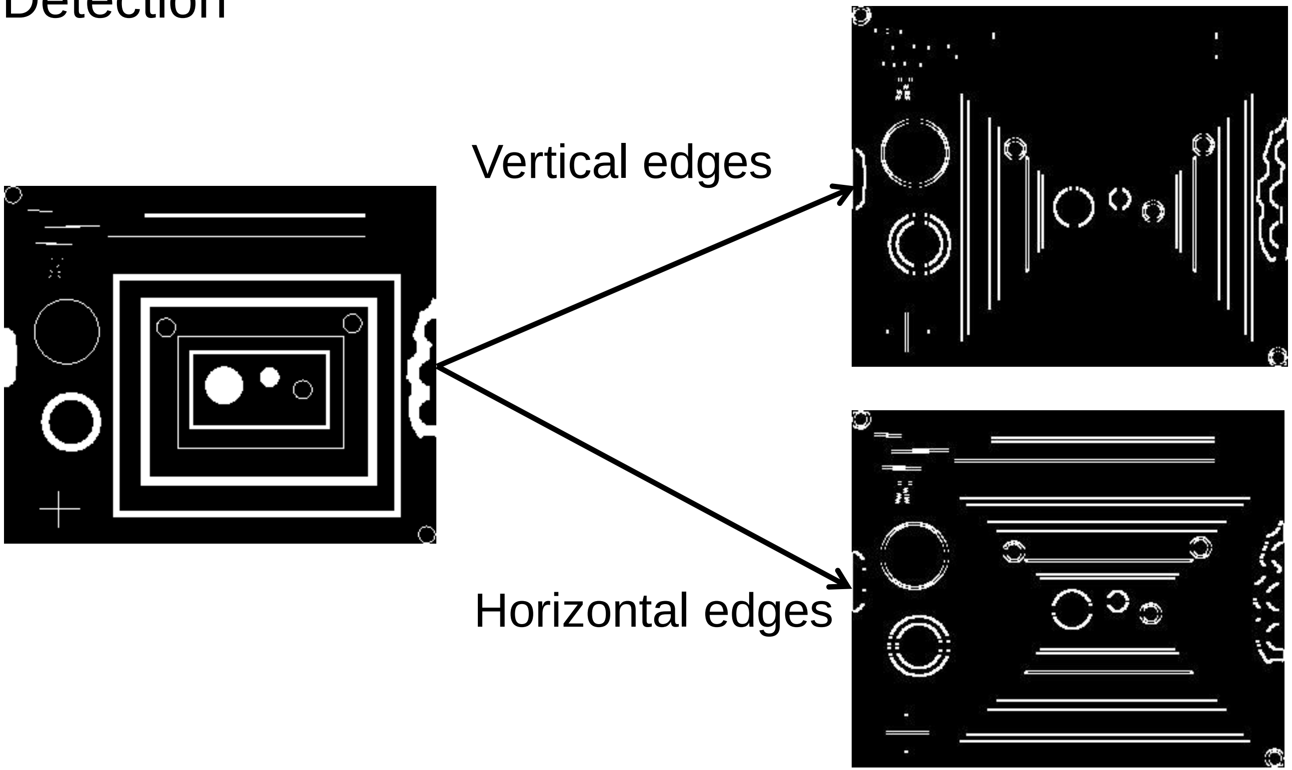
Foundation of Convolutional Neural Networks

- Edge Detection



Foundation of Convolutional Neural Networks

- Edge Detection



Foundation of Convolutional Neural Networks

- Vertical Edge Detection

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
3 & 0 & 1 & 2 & 7 & 4 \\ \hline
1 & 5 & 8 & 9 & 3 & 1 \\ \hline
2 & 7 & 2 & 5 & 1 & 3 \\ \hline
0 & 1 & 3 & 1 & 7 & 8 \\ \hline
4 & 2 & 1 & 6 & 2 & 8 \\ \hline
2 & 4 & 5 & 2 & 3 & 9 \\ \hline
\end{array}
\;*\;
\begin{array}{|c|c|c|}
\hline
-1 & 0 & 1 \\ \hline
-1 & 0 & 1 \\ \hline
-1 & 0 & 1 \\ \hline
\end{array}
\;=\;
\begin{array}{|c|c|c|c|}
\hline
 & & & \\ \hline
 & & & \\ \hline
 & & & \\ \hline
 & & & \\ \hline
\end{array}
$$

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

$$\begin{bmatrix} 3 & 0^{-1} & 1^{0} & 2^{1} & 7 & 4 \\ 1 & 5^{-1} & 8^{0} & 9^{1} & 3 & 1 \\ 2 & 7^{-1} & 2^{0} & 5^{1} & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 4 & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

$$
\begin{bmatrix}
3 & 0 & 1^{-1} & 2^{0} & 7^{1} & 4 \\
1 & 5 & 8^{-1} & 9^{0} & 3^{1} & 1 \\
2 & 7 & 2^{-1} & 5^{0} & 1^{1} & 3 \\
0 & 1 & 3 & 1 & 7 & 8 \\
4 & 2 & 1 & 6 & 2 & 8 \\
2 & 4 & 5 & 2 & 3 & 9
\end{bmatrix}
*
\begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1
\end{bmatrix}
=
\begin{bmatrix}
5 & 4 & 0 & \\
 & & & \\
 & & & \\
 & & &
\end{bmatrix}
$$

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

$$\begin{bmatrix} 3 & 0 & 1 & 2^{-1} & 7^{0} & 4^{1} \\ 1 & 5 & 8 & 9^{-1} & 3^{0} & 1^{1} \\ 2 & 7 & 2 & 5^{-1} & 1^{0} & 3^{1} \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 4 & 0 & -8 \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

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			

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

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
3 & 0 & 1 & 2 & 7 & 4 \\ \hline
1 & 5 & 8 & 9^{\scriptscriptstyle -1} & 3^{\scriptscriptstyle 0} & 1^{\scriptscriptstyle 1} \\ \hline
2 & 7 & 2 & 5^{\scriptscriptstyle -1} & 1^{\scriptscriptstyle 0} & 3^{\scriptscriptstyle 1} \\ \hline
0 & 1 & 3 & 1^{\scriptscriptstyle -1} & 7^{\scriptscriptstyle 0} & 8^{\scriptscriptstyle 1} \\ \hline
4 & 2 & 1 & 6 & 2 & 8 \\ \hline
2 & 4 & 5 & 2 & 3 & 9 \\ \hline
\end{array}
\;*\;
\begin{array}{|c|c|c|}
\hline
-1 & 0 & 1 \\ \hline
-1 & 0 & 1 \\ \hline
-1 & 0 & 1 \\ \hline
\end{array}
\;=\;
\begin{array}{|c|c|c|c|}
\hline
5 & 4 & 0 & -8 \\ \hline
10 & 2 & 2 & -3 \\ \hline
 & & & \\ \hline
 & & & \\ \hline
\end{array}
$$

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

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1^{\scriptscriptstyle -1} & 7^{\scriptscriptstyle 0} & 8^{\scriptscriptstyle 1} \\ 4 & 2 & 1 & 6^{\scriptscriptstyle -1} & 2^{\scriptscriptstyle 0} & 8^{\scriptscriptstyle 1} \\ 2 & 4 & 5 & 2^{\scriptscriptstyle -1} & 3^{\scriptscriptstyle 0} & 9^{\scriptscriptstyle 1} \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 4 & 0 & -8 \\ 10 & 2 & 2 & -3 \\ 0 & 2 & 4 & 7 \\ 3 & 2 & 3 & 16 \end{bmatrix}$$

Foundation of Convolutional Neural Networks

- Vertical Edge Detection

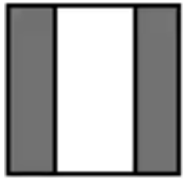
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



Foundation of Convolutional Neural Networks

- Vertical Edge Detection

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

*

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

=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0

Foundation of Convolutional Neural Networks

- Vertical & Horizontal Edge Detection

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

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

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

Foundation of Convolutional Neural Networks

- Vertical & Horizontal Edge Detection

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

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

1	0	-1
2	0	-2
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

*

w1	*w2*	*w3*
w4	*w5*	*w6*
w7	*w8*	*w9*

=

Learning the filter weights!

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 $nout = n - f + 1$

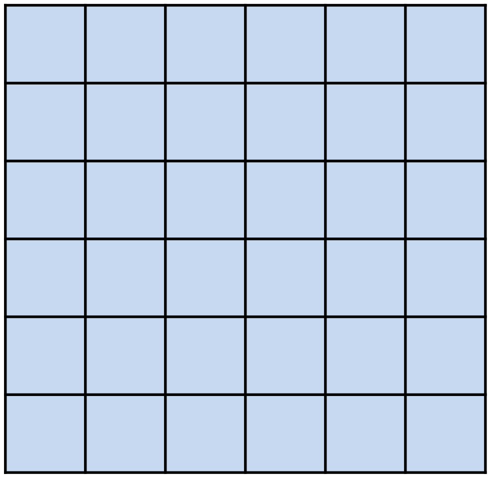
Shortcoming of this technique:

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		

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

$$
\begin{array}{|c|c|c|c|c|c|c|}
\hline
3 & 0 & 1 & 2 & 7^{-1} & 4^{0} & 2^{1} \\
\hline
1 & 5 & 8 & 9 & 3^{-1} & 1^{0} & 1^{1} \\
\hline
2 & 7 & 2 & 5 & 1^{-1} & 3^{0} & 5^{1} \\
\hline
0 & 1 & 3 & 1 & 7 & 8 & 4 \\
\hline
4 & 2 & 1 & 6 & 2 & 8 & 3 \\
\hline
2 & 4 & 5 & 2 & 3 & 9 & 8 \\
\hline
2 & 3 & 6 & 4 & 2 & 0 & 1 \\
\hline
\end{array}
*
\begin{array}{|c|c|c|}
\hline
-1 & 0 & 1 \\
\hline
-1 & 0 & 1 \\
\hline
-1 & 0 & 1 \\
\hline
\end{array}
=
\begin{array}{|c|c|c|}
\hline
5 & 0 & -3 \\
\hline
 & & \\
\hline
 & & \\
\hline
\end{array}
$$

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

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 & 2 \\ 1 & 5 & 8 & 9 & 3 & 1 & 1 \\ 2 & 7 & 2^{-1} & 5^{0} & 1^{1} & 3 & 5 \\ 0 & 1 & 3^{-1} & 1^{0} & 7^{1} & 8 & 4 \\ 4 & 2 & 1^{-1} & 6^{0} & 2^{1} & 8 & 3 \\ 2 & 4 & 5 & 2 & 3 & 9 & 8 \\ 2 & 3 & 6 & 4 & 2 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0 & -3 \\ 0 & 4 & \\ & & \end{bmatrix}$$

Foundation of Convolutional Neural Networks

- Strided Convolution

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 & 2 \\ 1 & 5 & 8 & 9 & 3 & 1 & 1 \\ 2 & 7 & 2 & 5 & 1^{-1} & 3^{0} & 5^{1} \\ 0 & 1 & 3 & 1 & 7^{-1} & 8^{0} & 4^{1} \\ 4 & 2 & 1 & 6 & 2^{-1} & 8^{0} & 3^{1} \\ 2 & 4 & 5 & 2 & 3 & 9 & 8 \\ 2 & 3 & 6 & 4 & 2 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0 & -3 \\ 0 & 4 & 2 \\ & & \end{bmatrix}$$

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	

Foundation of Convolutional Neural Networks

- Strided Convolution

$$
\begin{array}{|c|c|c|c|c|c|c|}
\hline
3 & 0 & 1 & 2 & 7 & 4 & 2 \\ \hline
1 & 5 & 8 & 9 & 3 & 1 & 1 \\ \hline
2 & 7 & 2 & 5 & 1 & 3 & 5 \\ \hline
0 & 1 & 3 & 1 & 7 & 8 & 4 \\ \hline
4 & 2 & 1 & 6 & 2^{-1} & 8^{0} & 3^{1} \\ \hline
2 & 4 & 5 & 2 & 3^{-1} & 9^{0} & 8^{1} \\ \hline
2 & 3 & 6 & 4 & 2^{-1} & 0^{0} & 1^{1} \\ \hline
\end{array}
*
\begin{array}{|c|c|c|}
\hline
-1 & 0 & 1 \\ \hline
-1 & 0 & 1 \\ \hline
-1 & 0 & 1 \\ \hline
\end{array}
=
\begin{array}{|c|c|c|}
\hline
5 & 0 & -3 \\ \hline
0 & 4 & 2 \\ \hline
4 & -5 & 5 \\ \hline
\end{array}
$$

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:

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

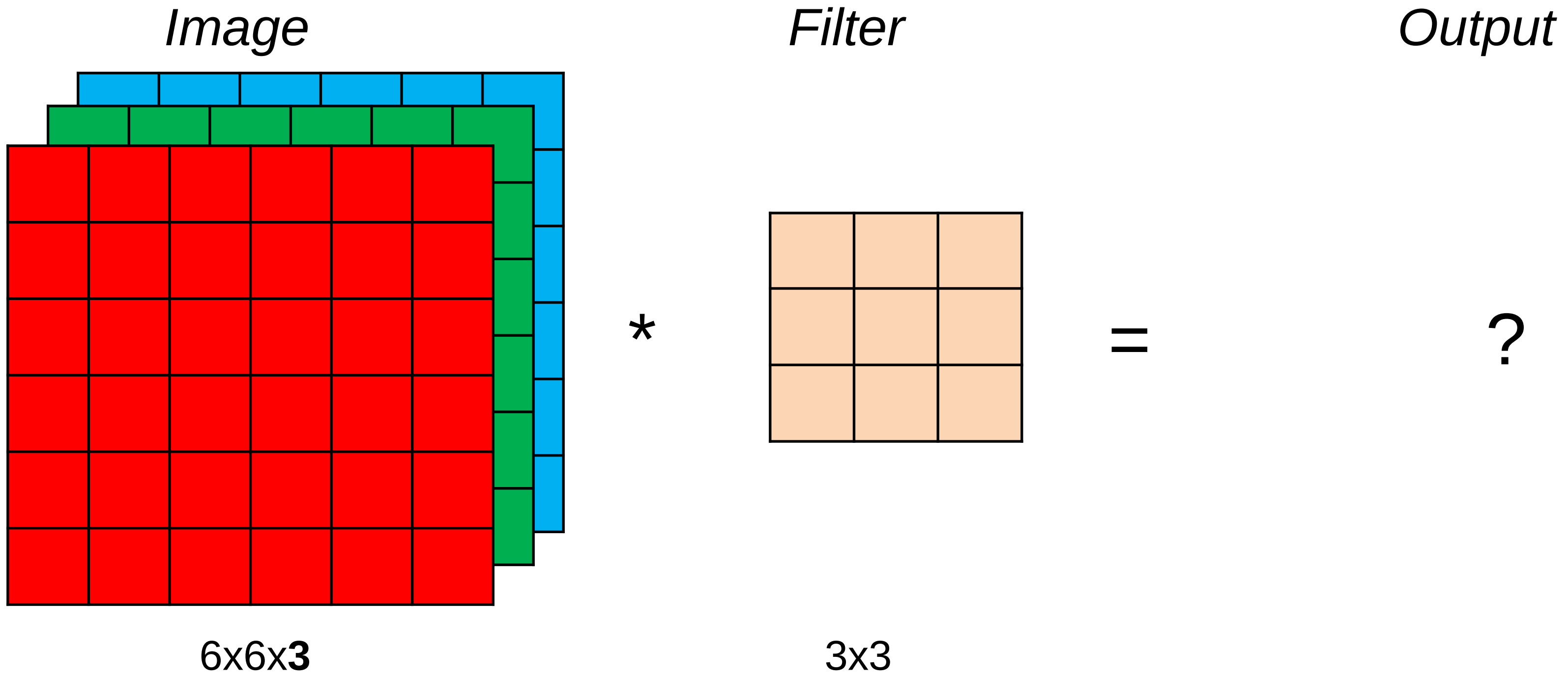
Convolution vs. Cross-correlation

Convolution involves: flip on axes -> multiply -> sum

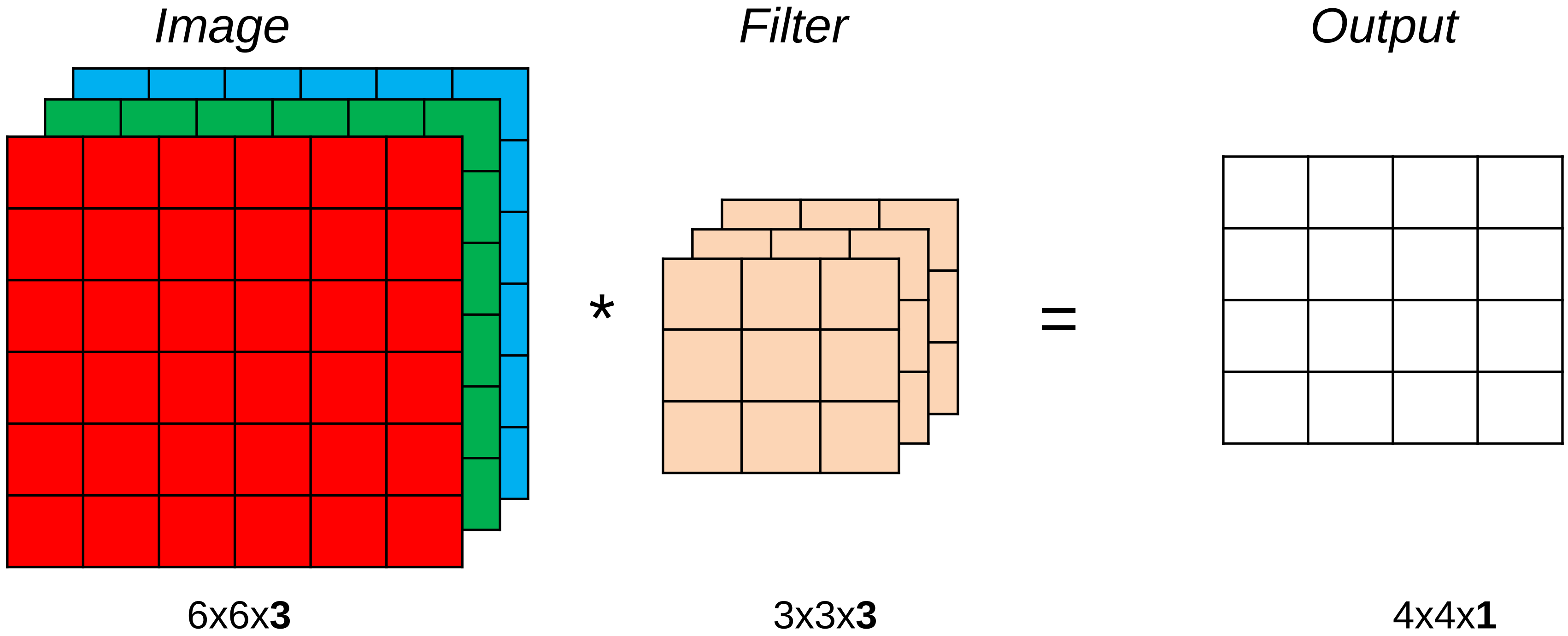
Cross-correlation involves: multiply -> sum

In CNNs, the process is ***cross-correlation*** but is referred to as ***convolution*** by convention.

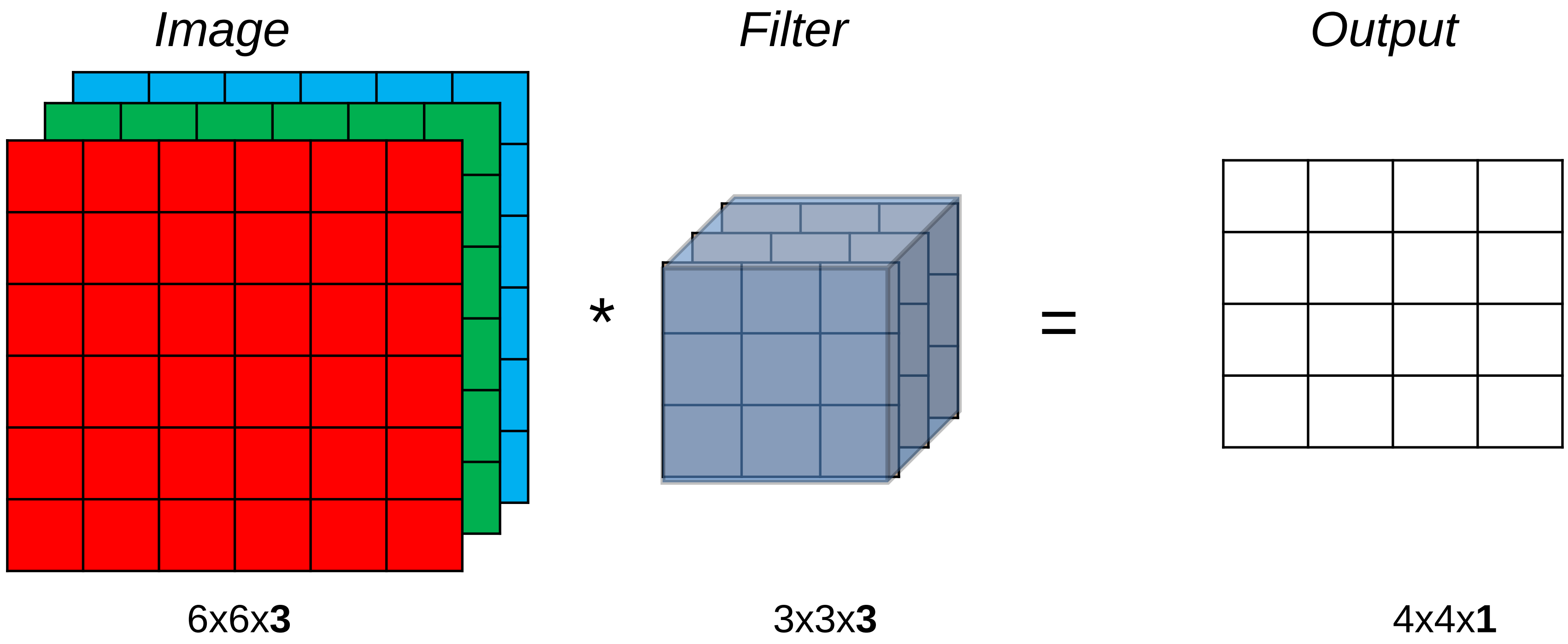
Convolution on Colour images



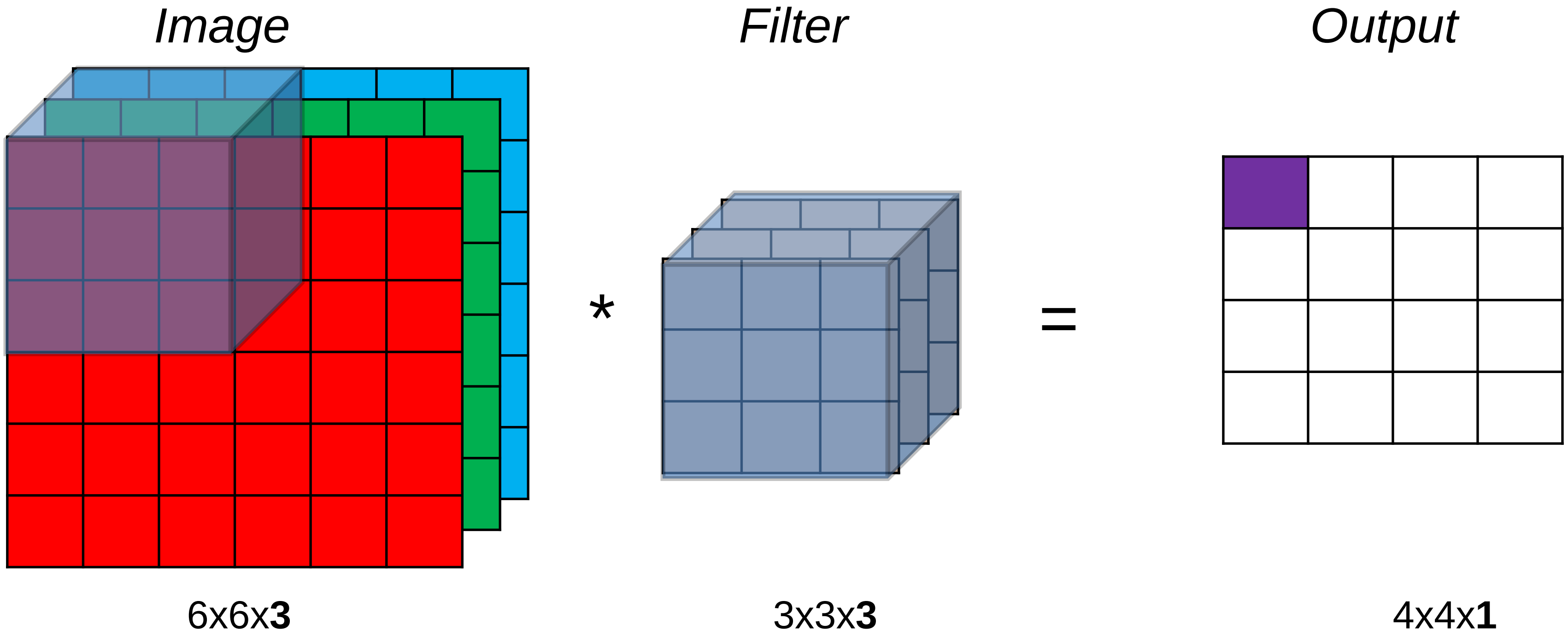
Convolution on Colour images



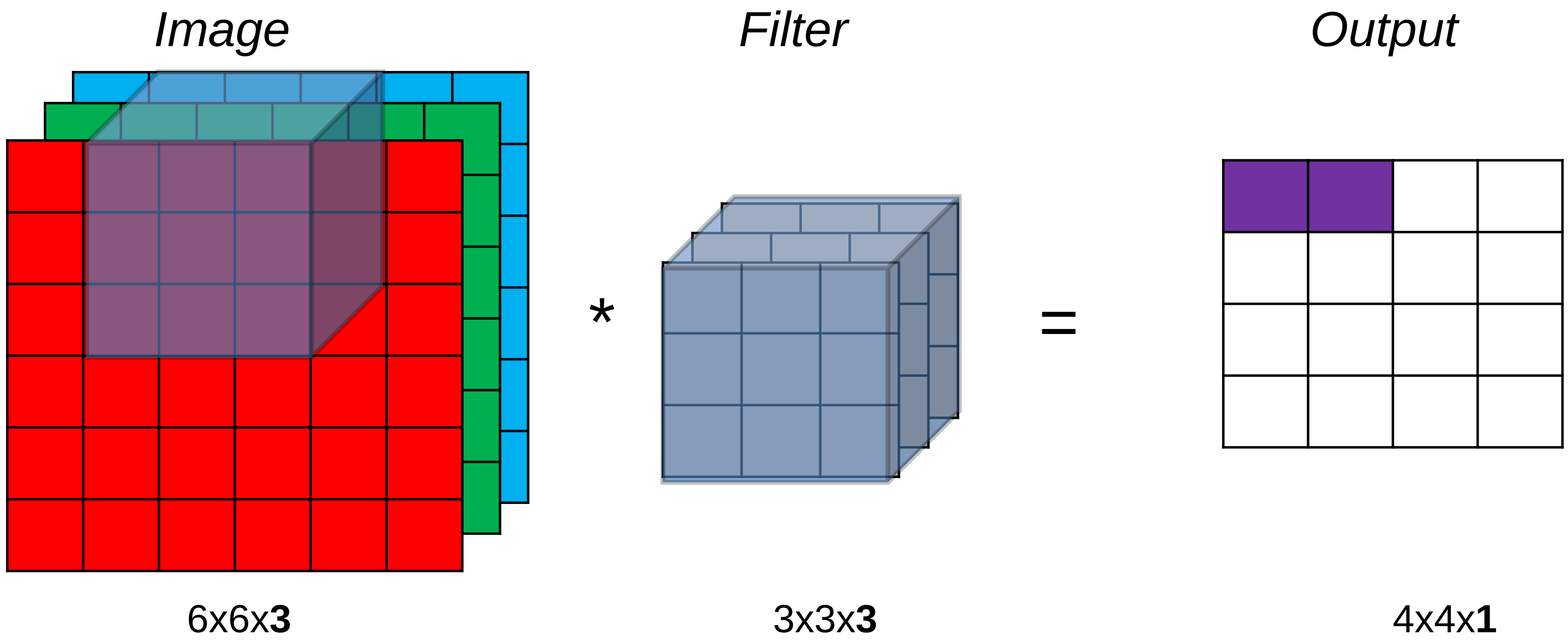
Convolution on Colour images



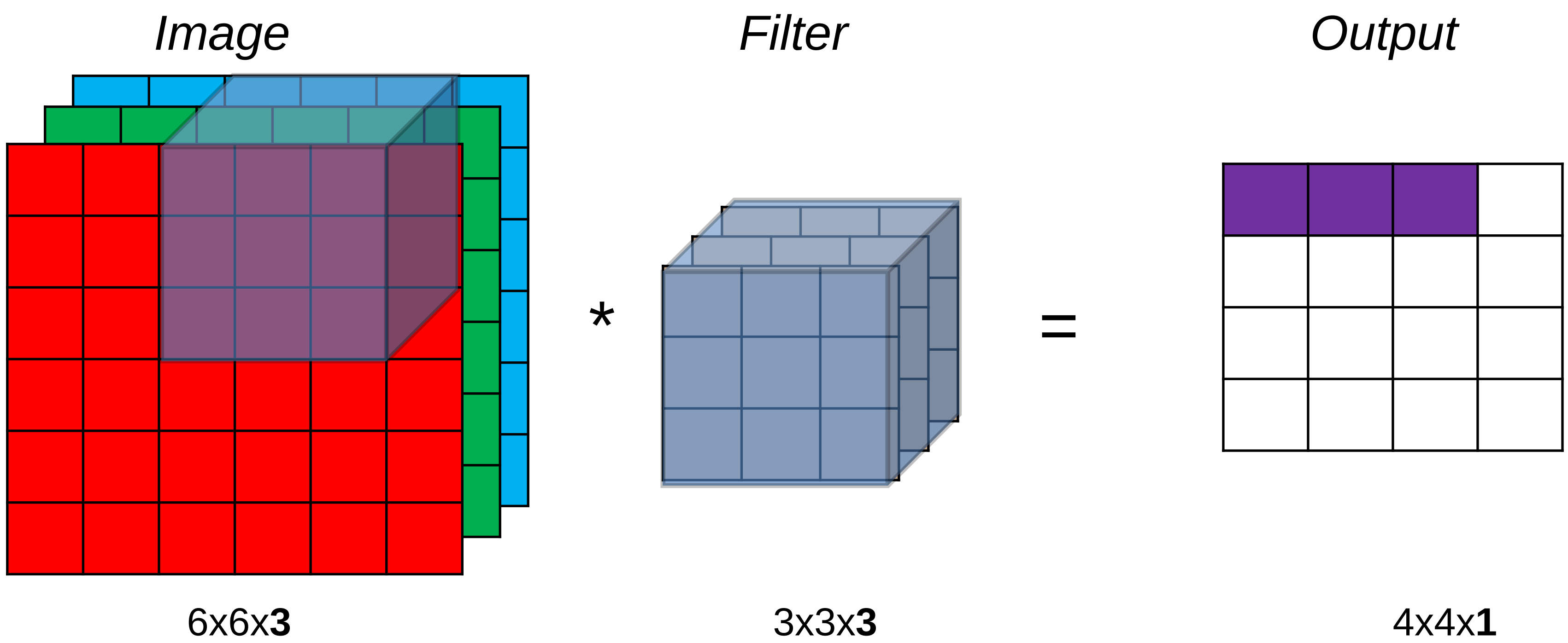
Convolution on Colour images



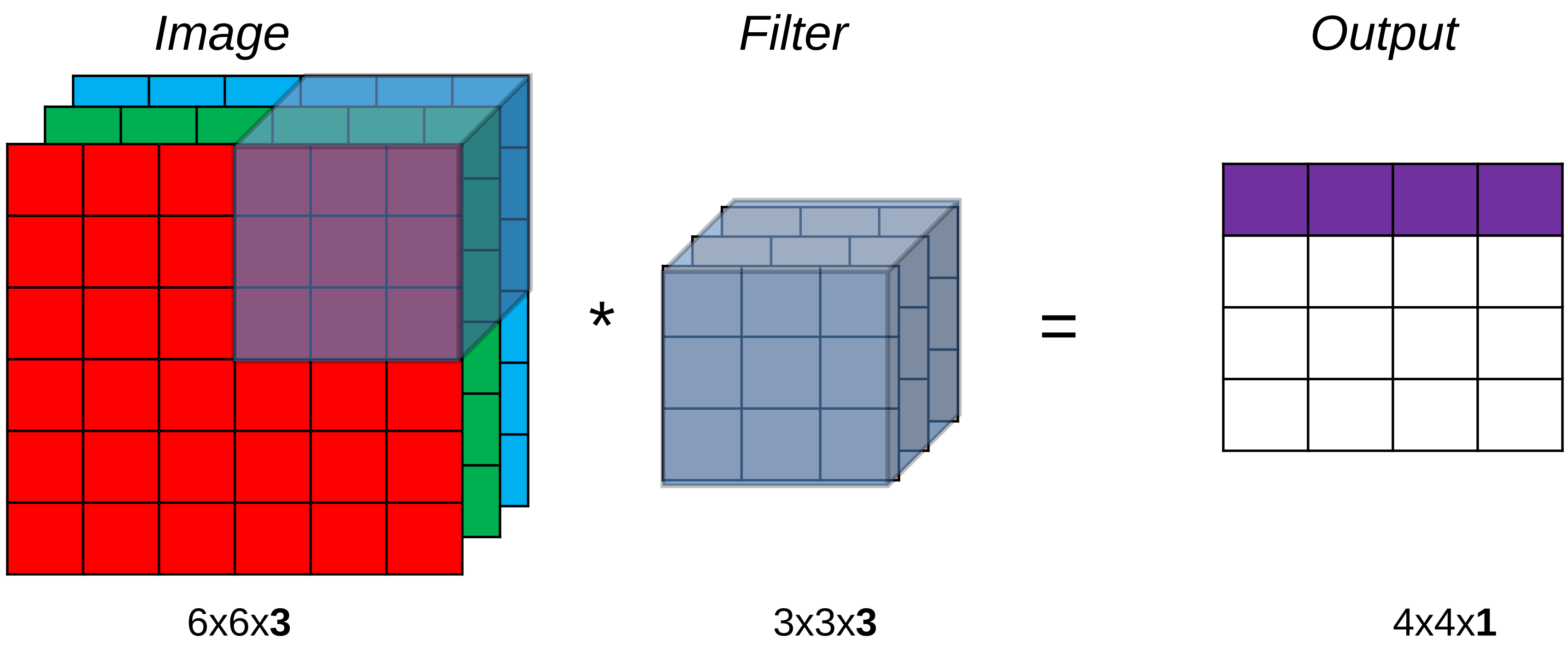
Convolution on Colour images



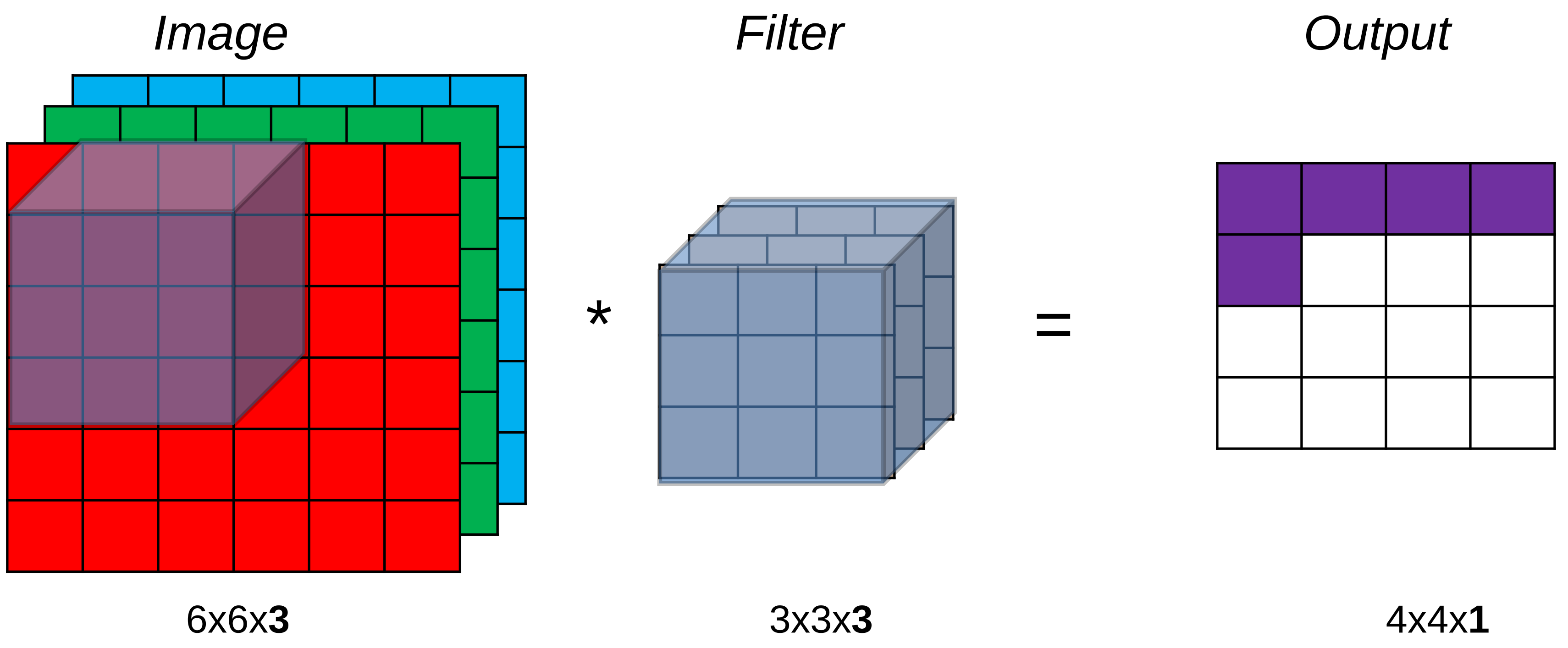
Convolution on Colour images



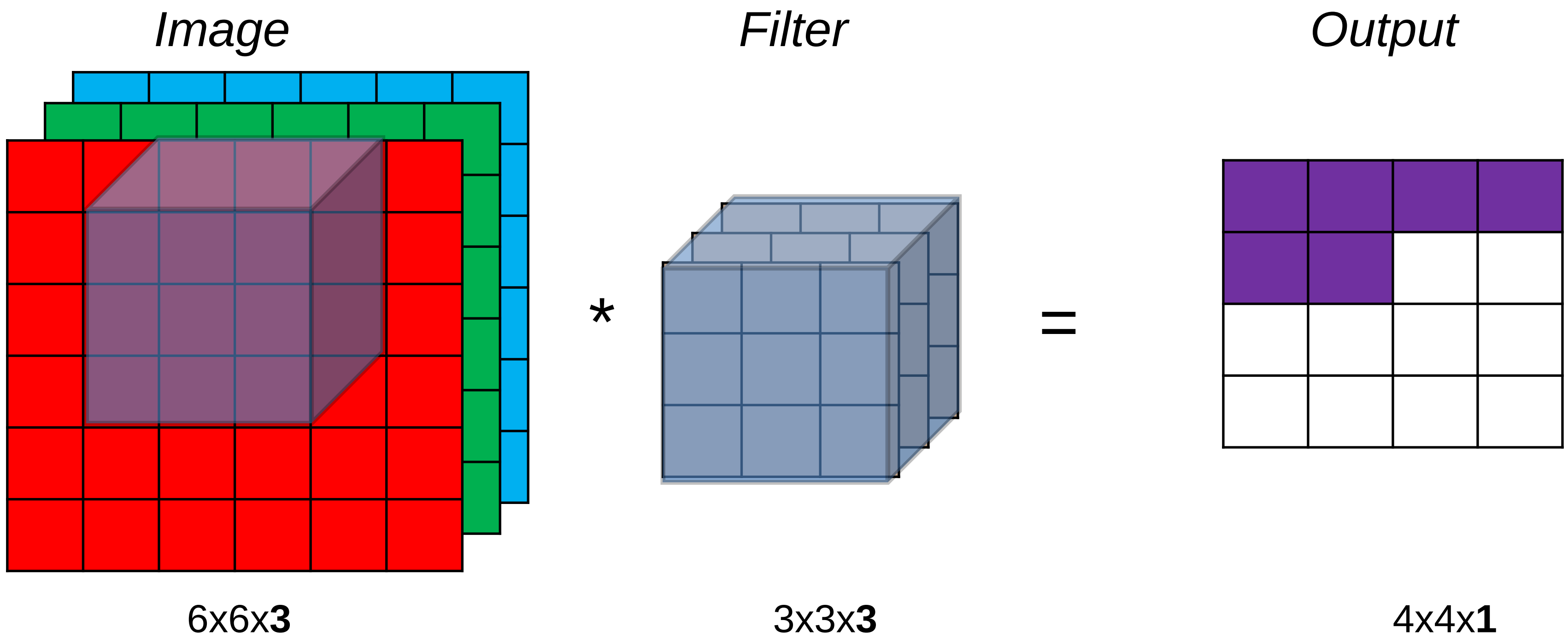
Convolution on Colour images



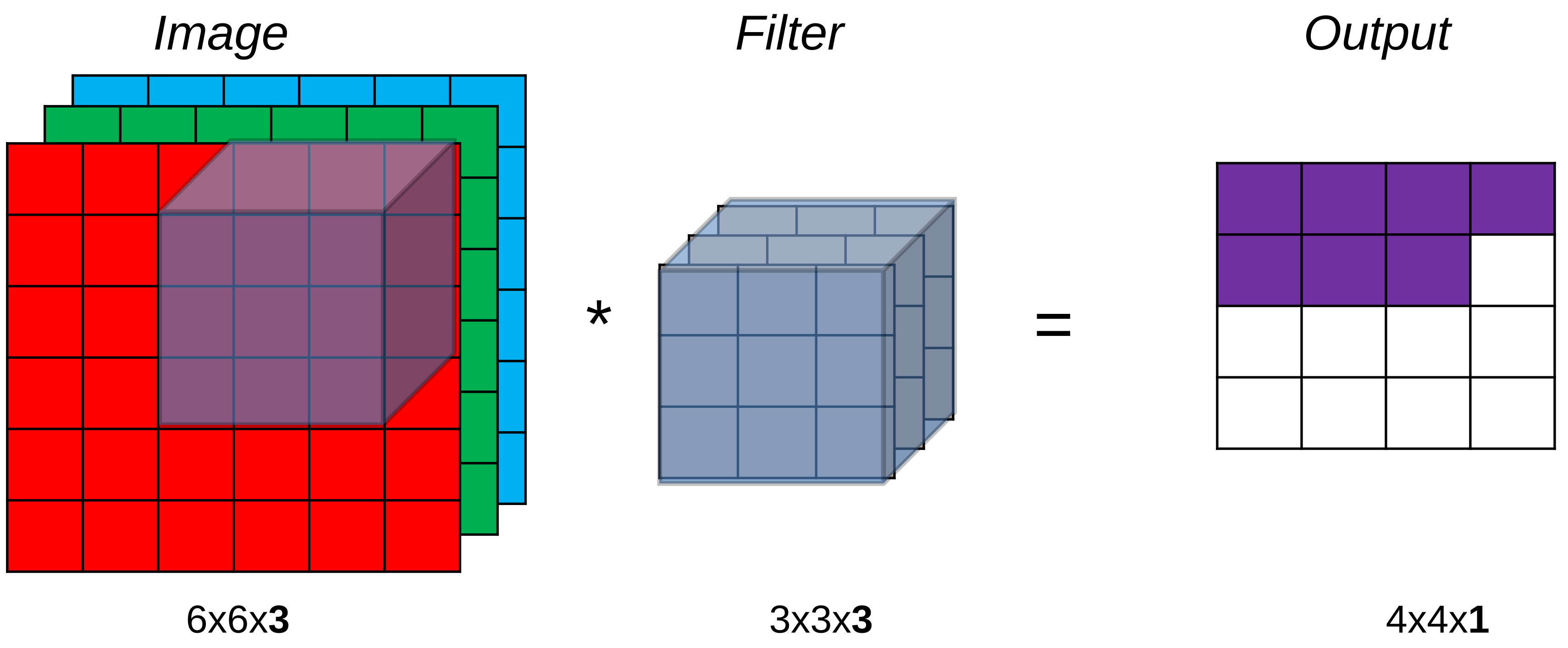
Convolution on Colour images



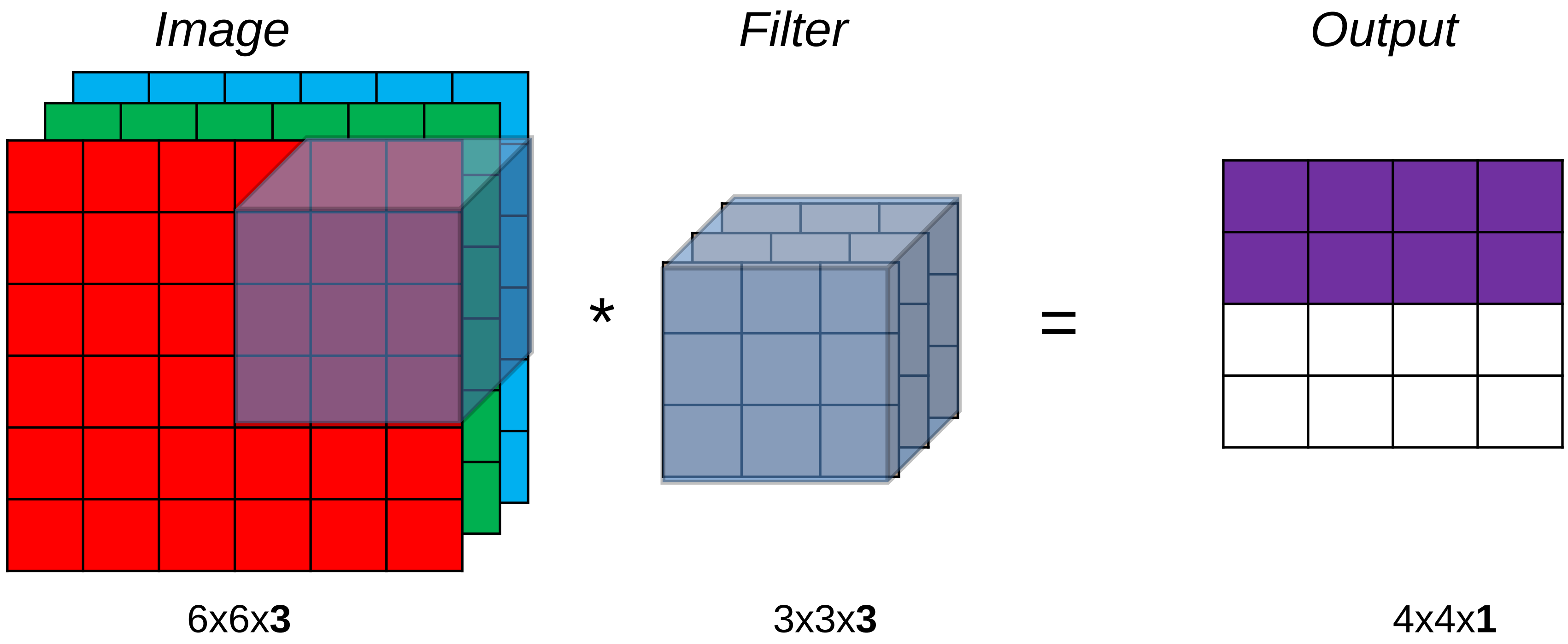
Convolution on Colour images



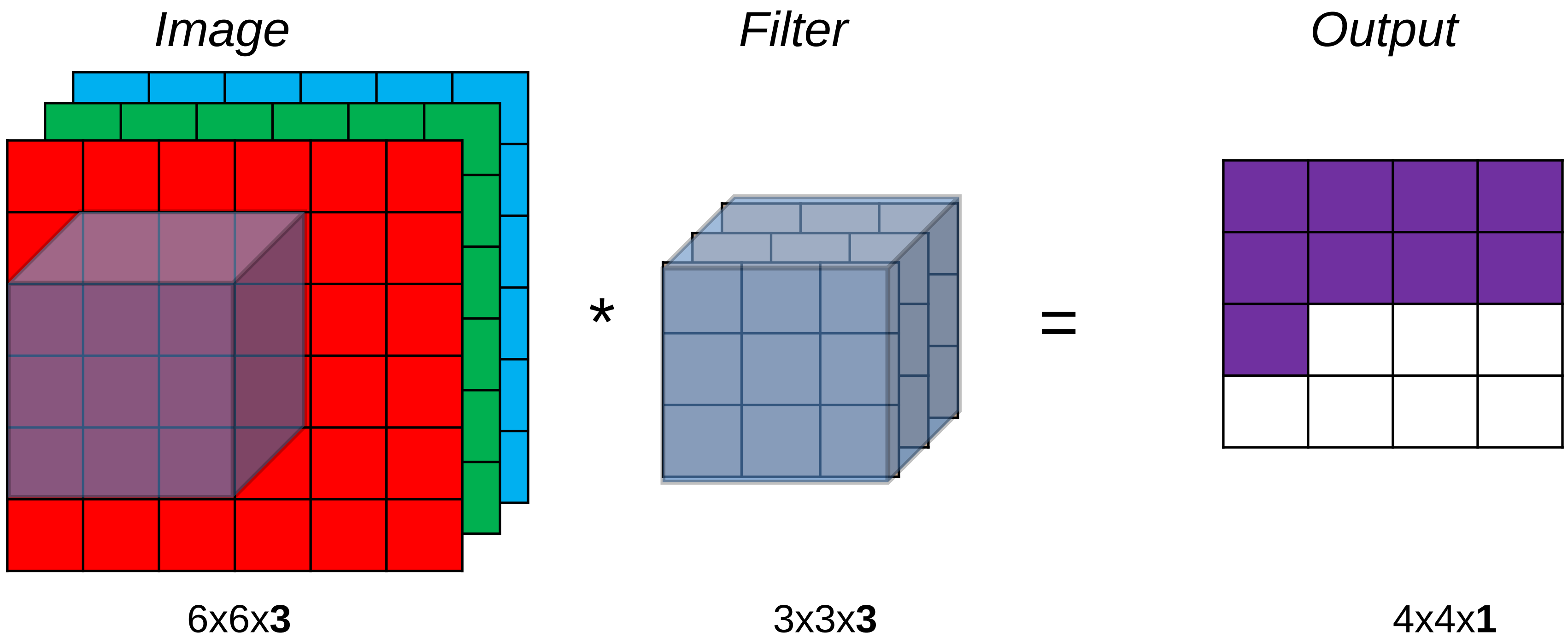
Convolution on Colour images



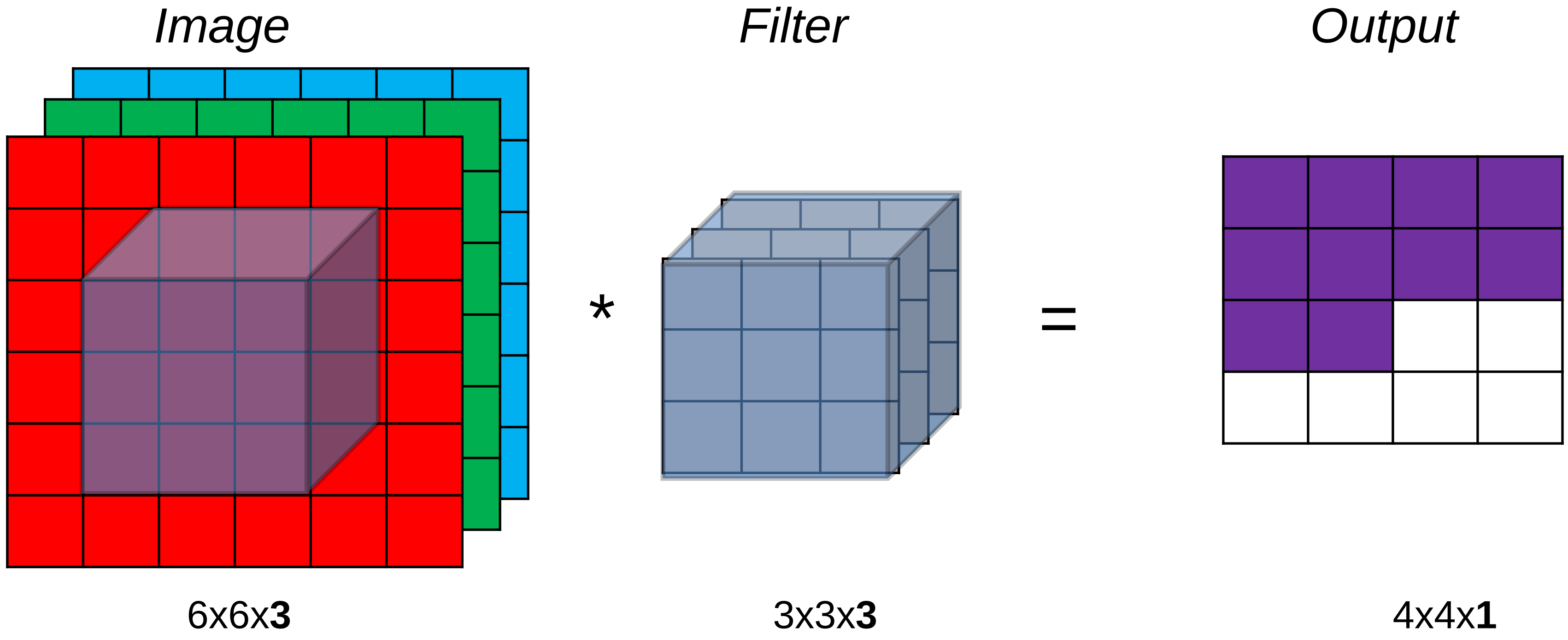
Convolution on Colour images



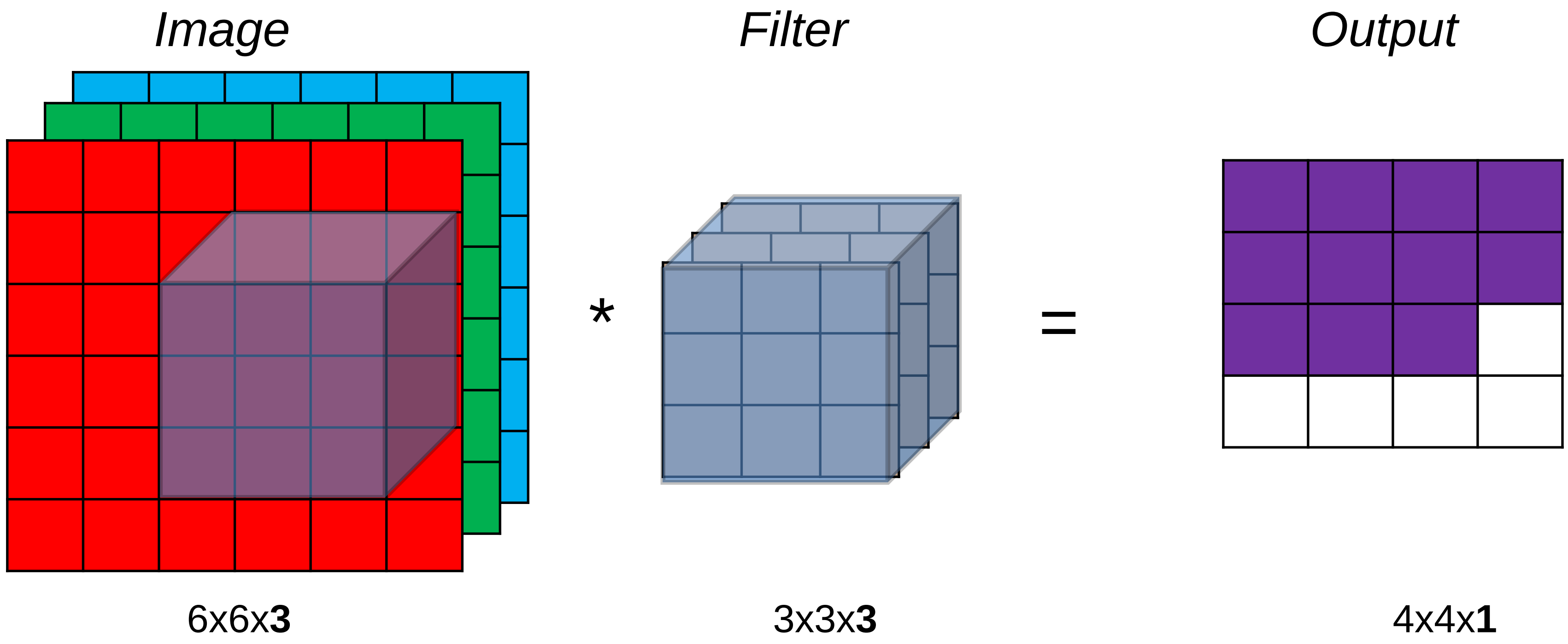
Convolution on Colour images



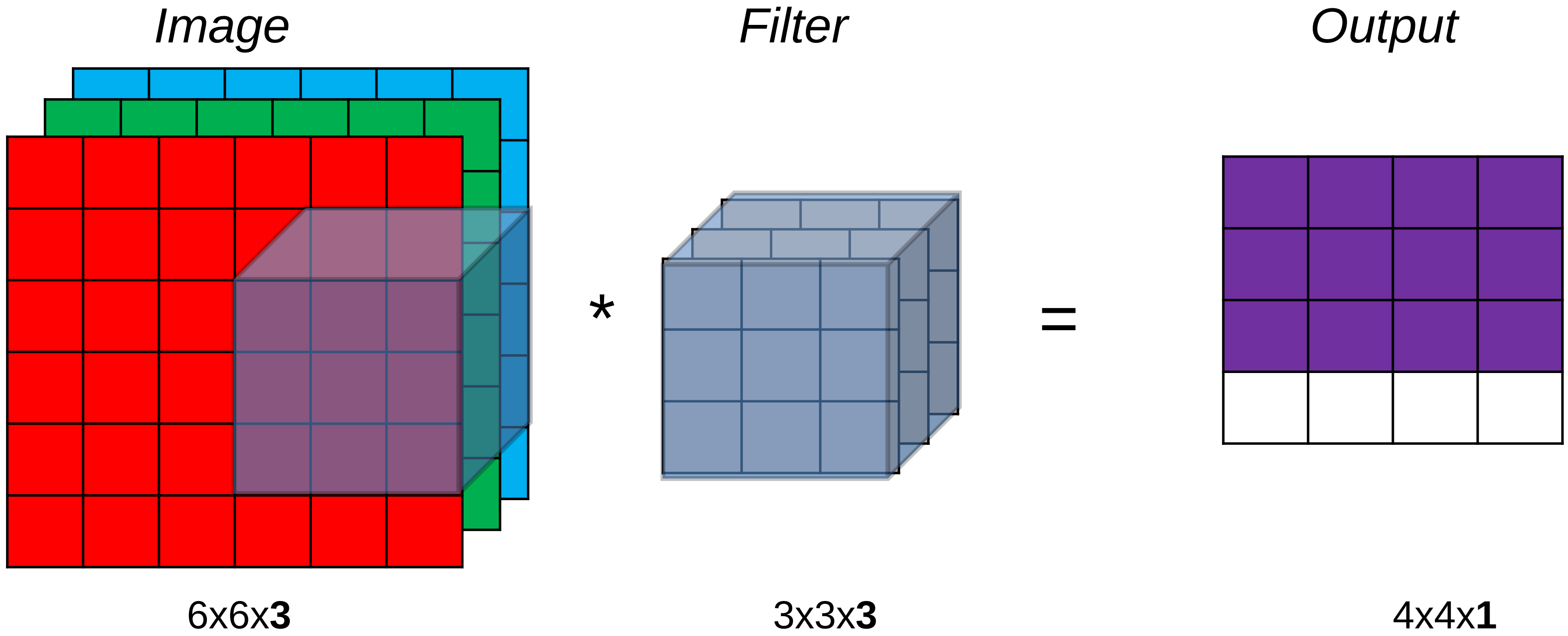
Convolution on Colour images



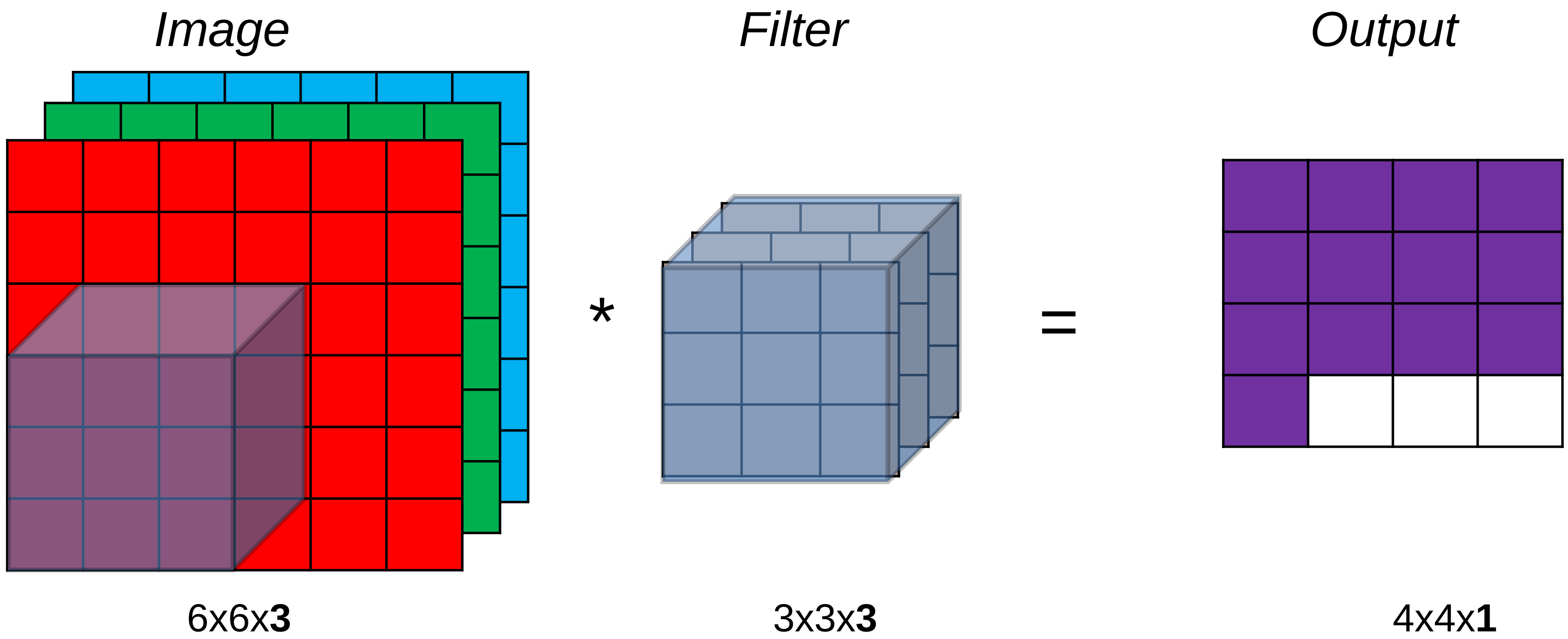
Convolution on Colour images



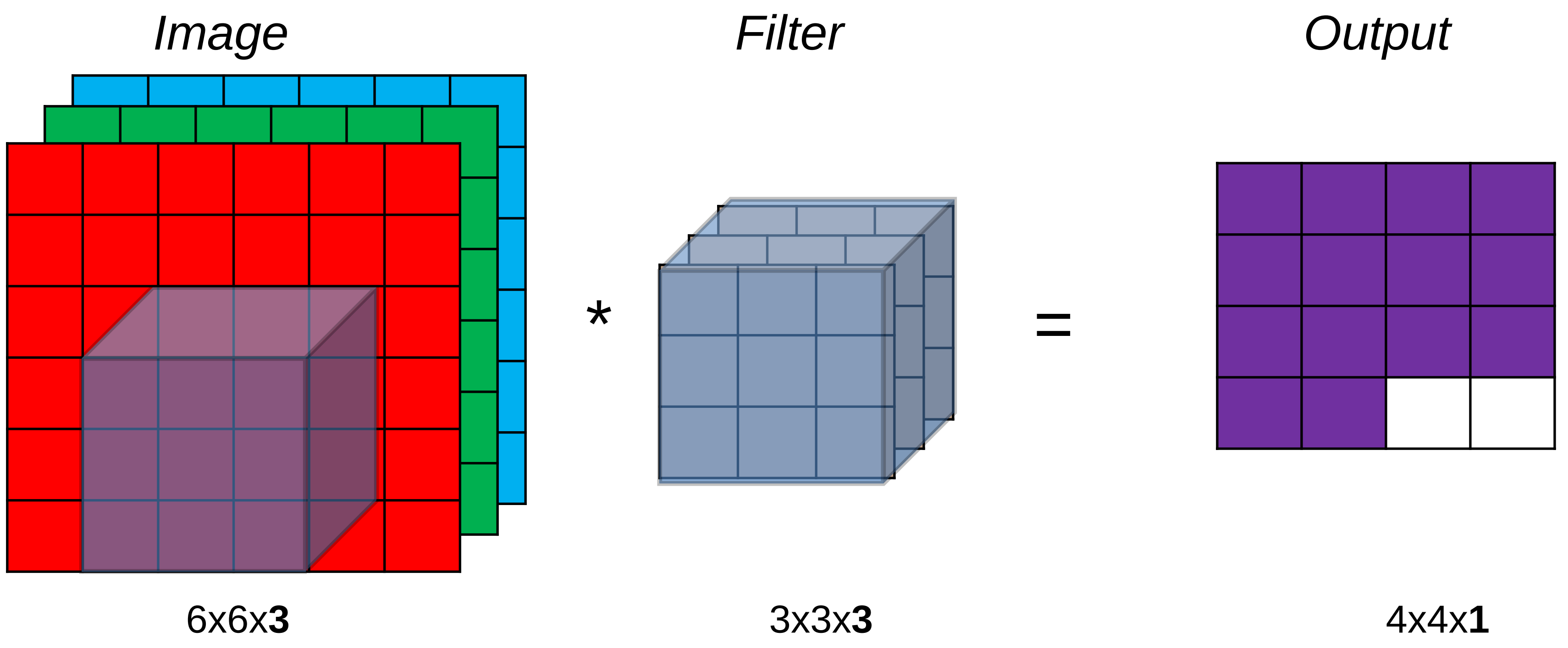
Convolution on Colour images



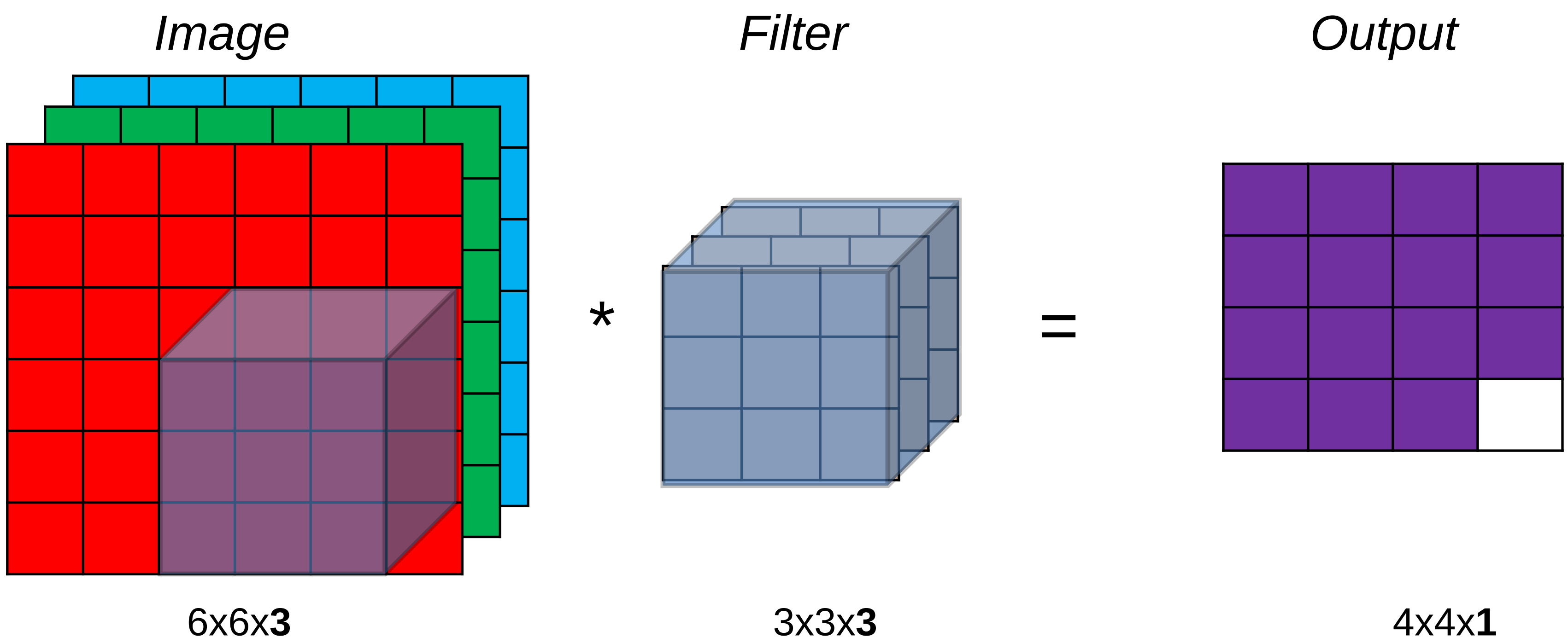
Convolution on Colour images



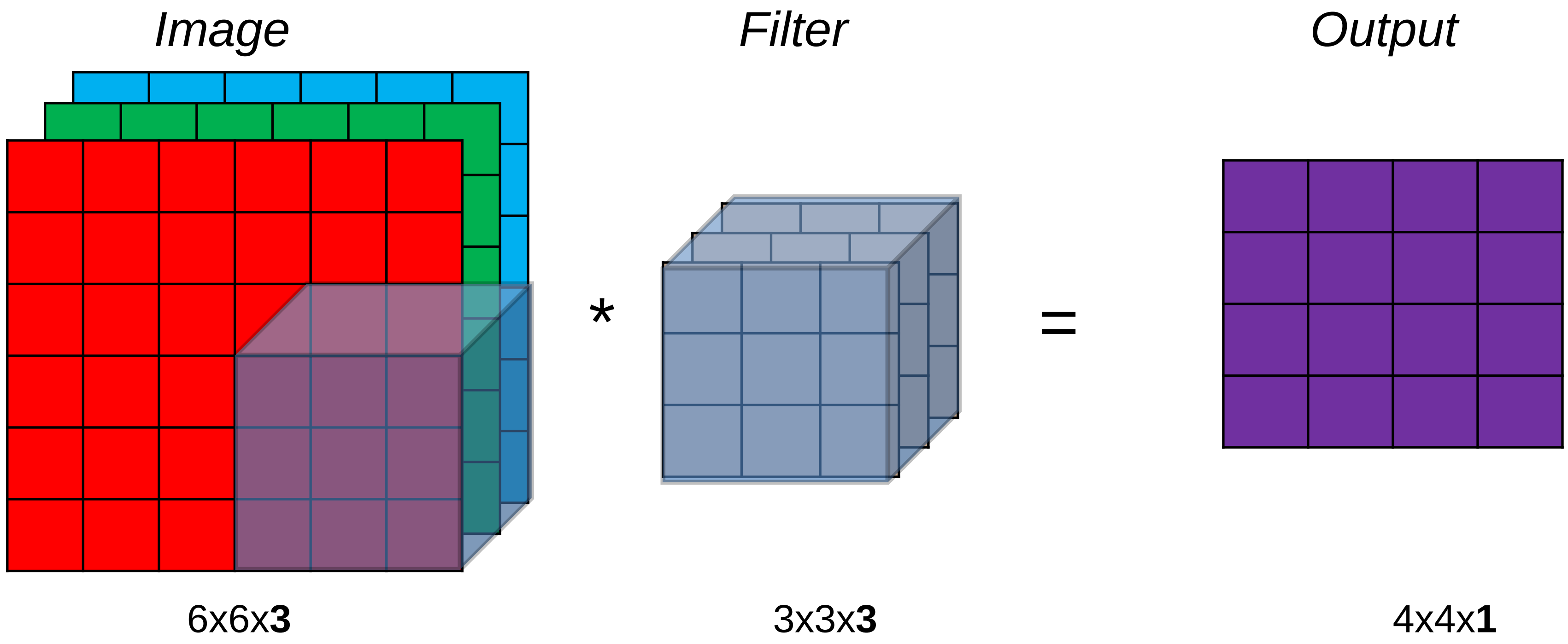
Convolution on Colour images



Convolution on Colour images



Convolution on Colour images



Quiz 1

Filter 1

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

0	0	0
0	0	0
0	0	0

0	0	0
0	0	0
0	0	0

Detect *in* *-plane*

Quiz 1

Filter 1

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

Detect vertical edges in R-plane

Quiz 1

Filter 1

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

Detect vertical edges in R-plane

Filter 2

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

Detect   in   plane

Quiz 1

Filter 1

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

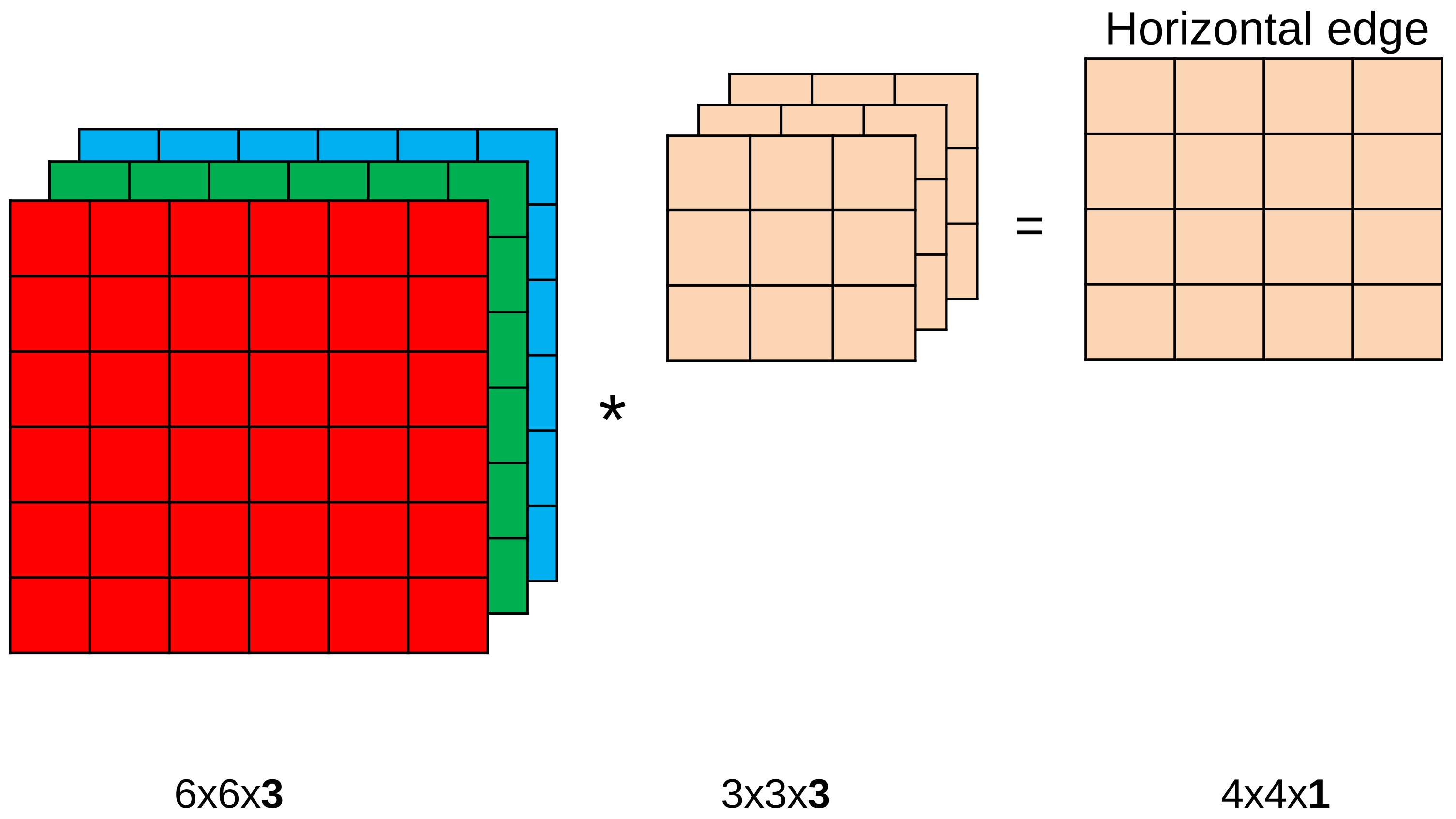
Detect vertical edges in R-plane

Filter 2

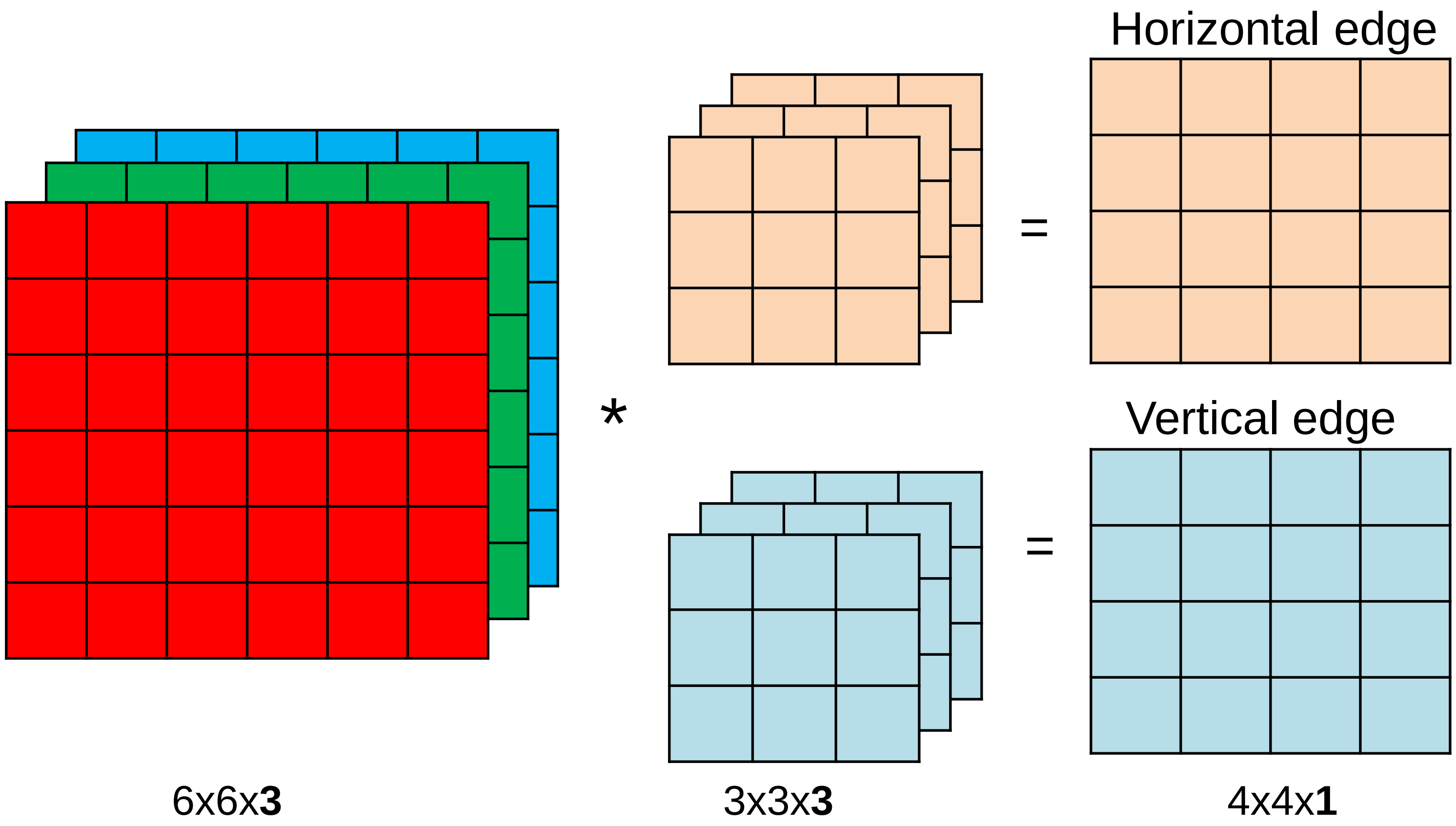
1	0	-1	1	0	-1	1	0	-1
1	0	-1	1	0	-1	1	0	-1
1	0	-1	1	0	-1	1	0	-1

Detect vertical edges in all-plane

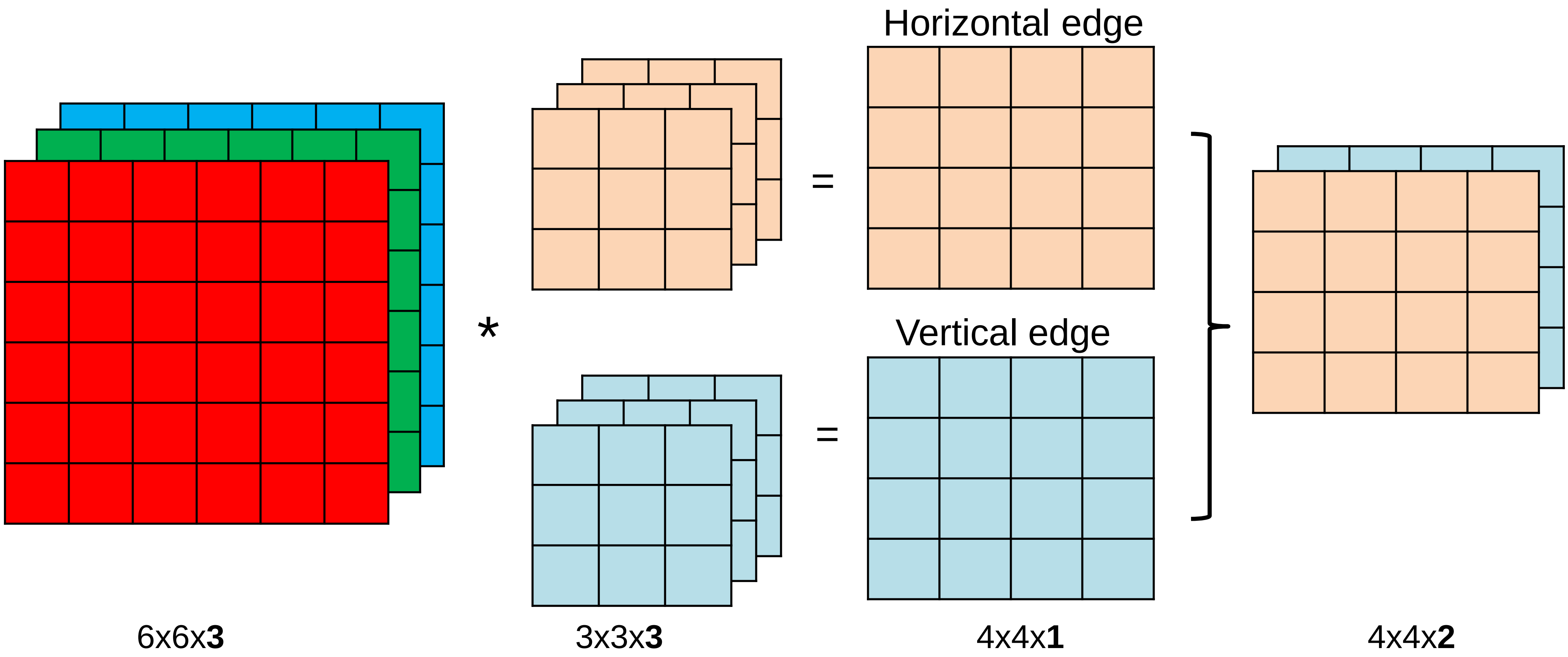
Multiple Filters



Multiple Filters



Multiple Filters



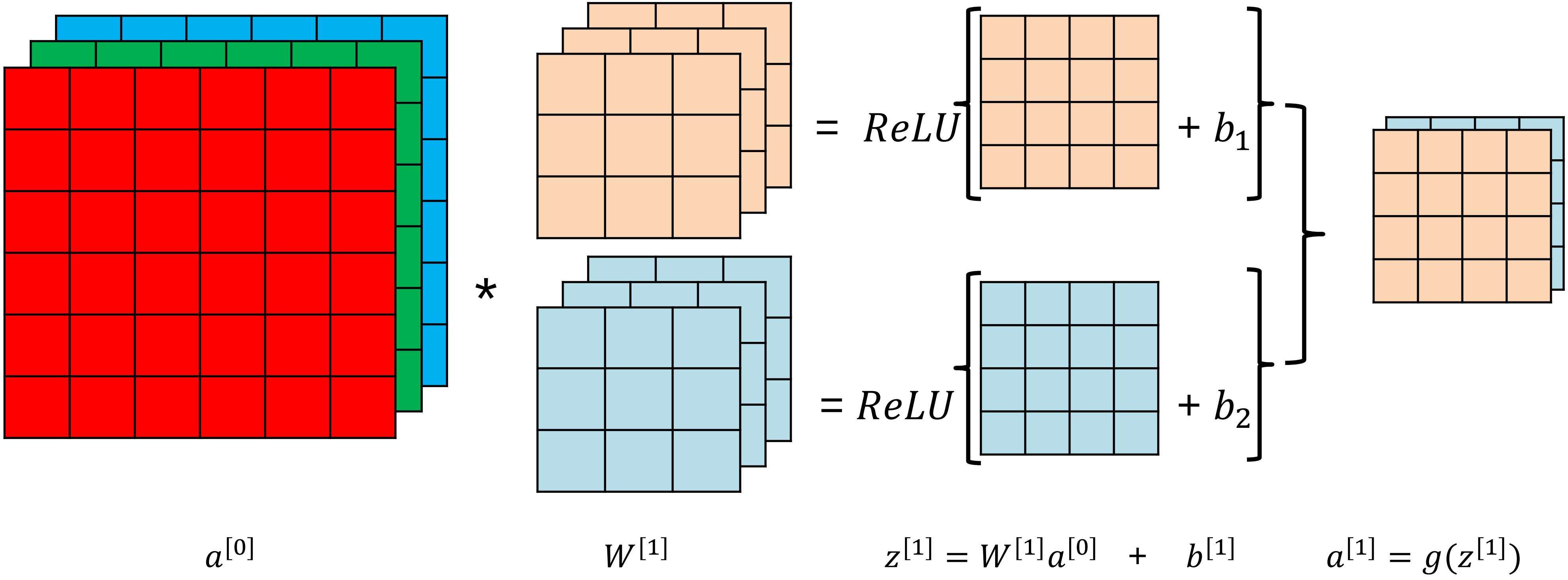
Summary of Convolution with Multiple filters

For an $n * n * c$ image and N filters each $f * f * c$ with padding p and a stride of s pixels,

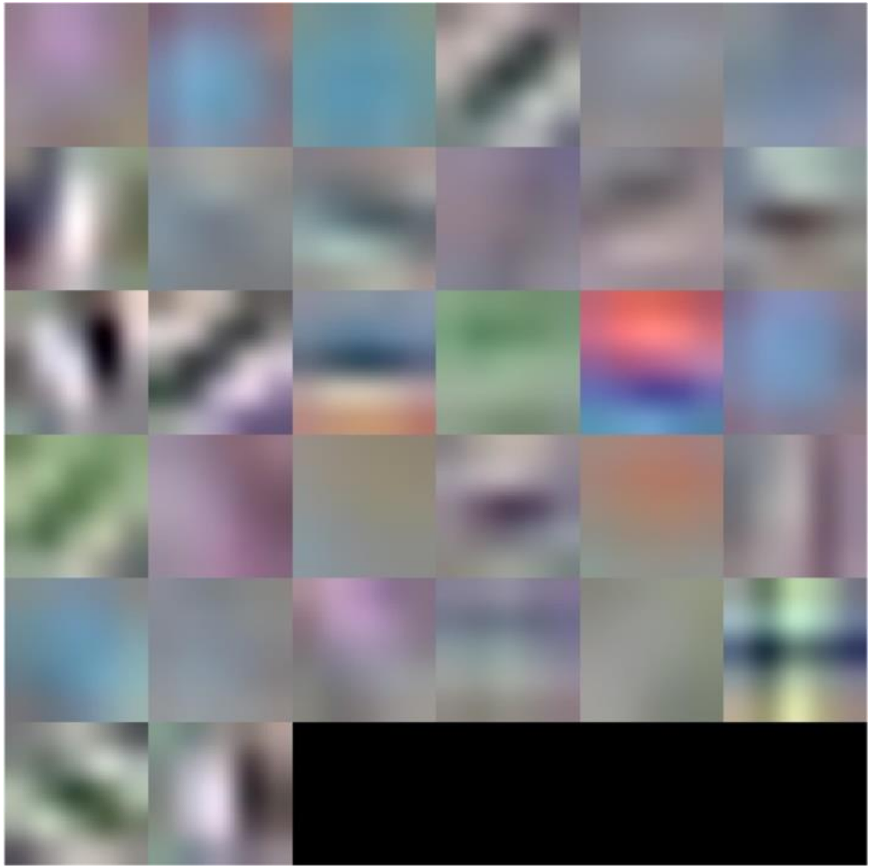
Output image dimension is given by:

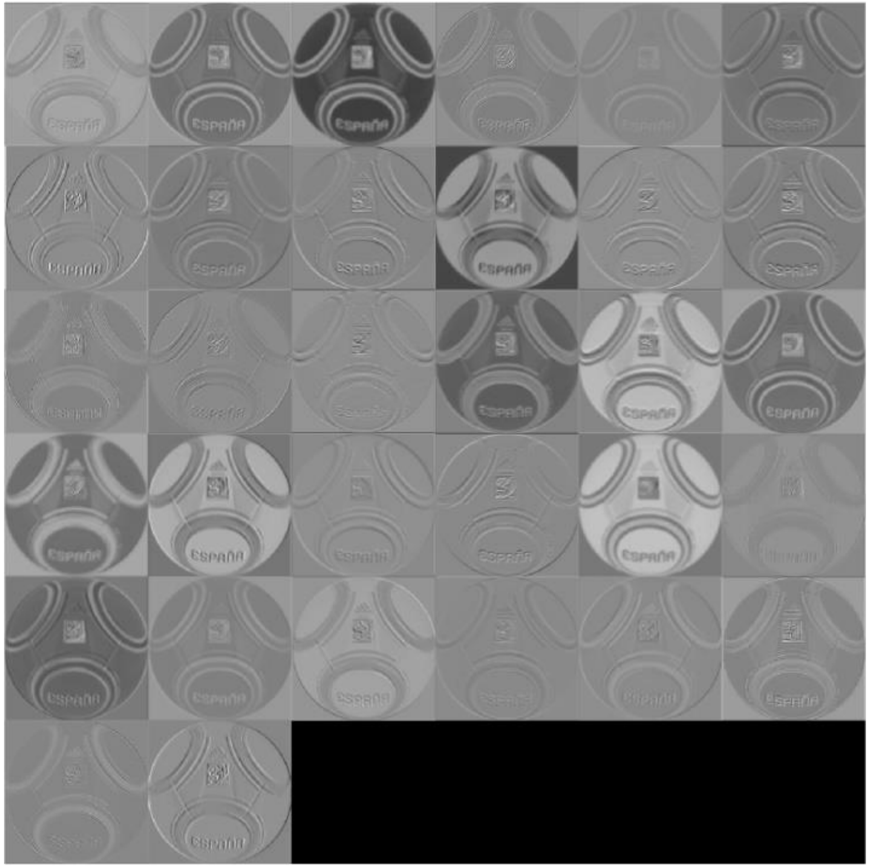
$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor * \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor * N$$

Single Convolution Layer



Learnt filters and corresponding activations





Quiz 2

- No. of **Trainable parameters** (Weights + biases) in a one convolution layer?

No. of filters: 10

Volume of each filter: 3x3x3

Input volume: 32x32x3

Quiz 2

- No. of **Trainable parameters** (Weights + biases) in a one convolution layer?

No. of filters: 10

Volume of each filter: 3x3x3

Input volume: 32x32x3

Solution: No need of input volume. It is a catch.

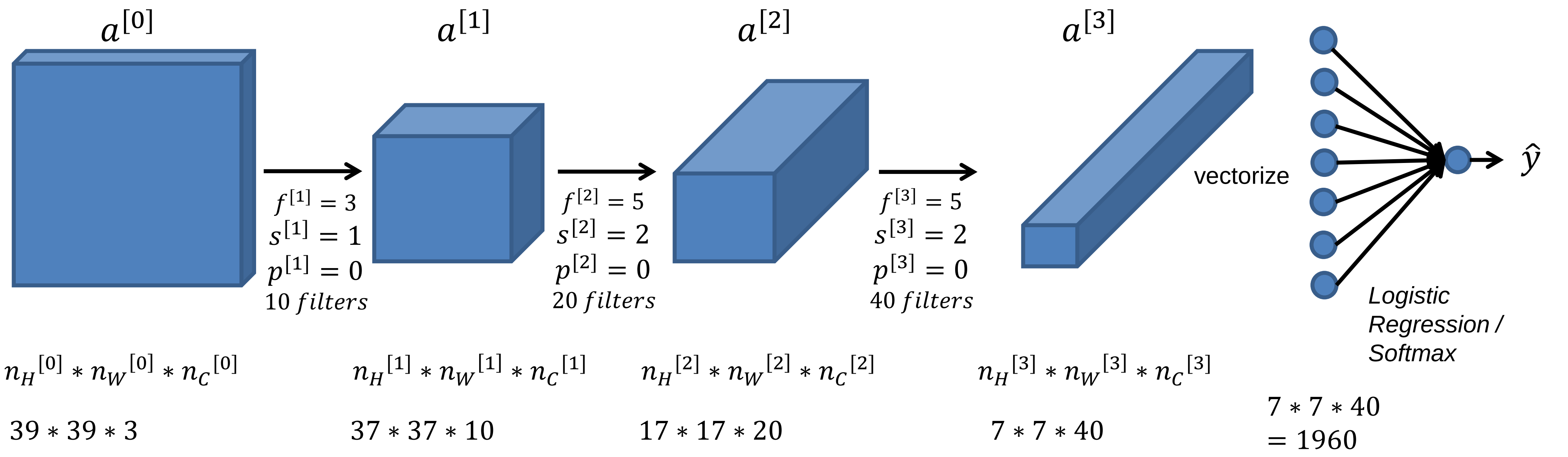
Total trainable parameters = (3x3x3 weight + 1 bias) of each filter x 10 filters

= (27+1) x 10

= (28) x10

= 280.

CNN Example # 1 (Basic)



$$\frac{n + 2p - f}{s} + 1 = \frac{39+2*0-3}{1} + 1 = 37$$

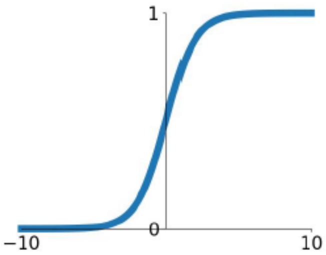
$$\frac{n + 2p - f}{s} + 1 = \frac{37+2*0-5}{2} + 1 = 17$$

$$\frac{n + 2p - f}{s} + 1 = \frac{17+2*0-5}{2} + 1 = 7$$

Non-linear activation

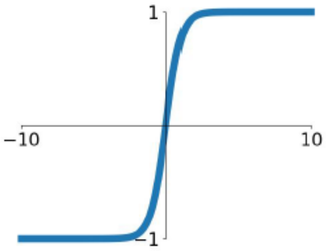
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



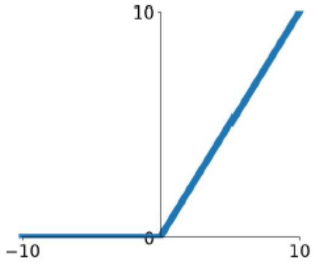
tanh

$\tanh(x)$



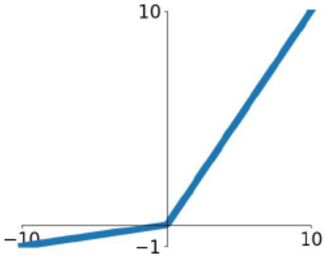
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

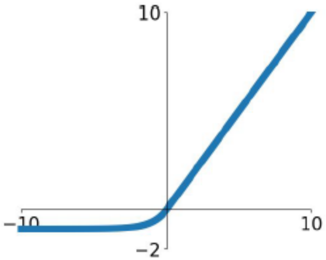


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Pooling Layer

Purpose:

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

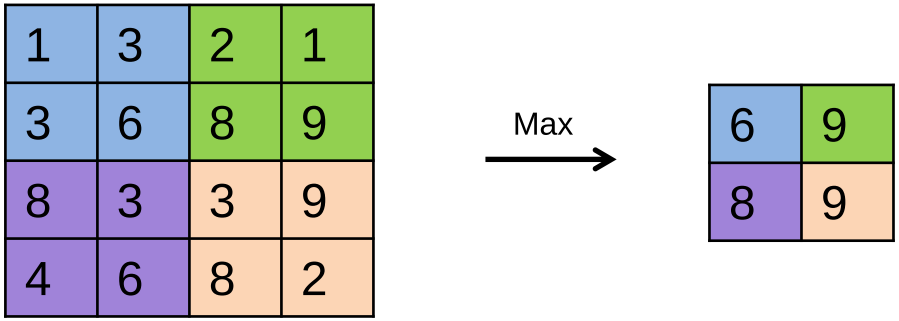
2) To retain robust features.

Types:

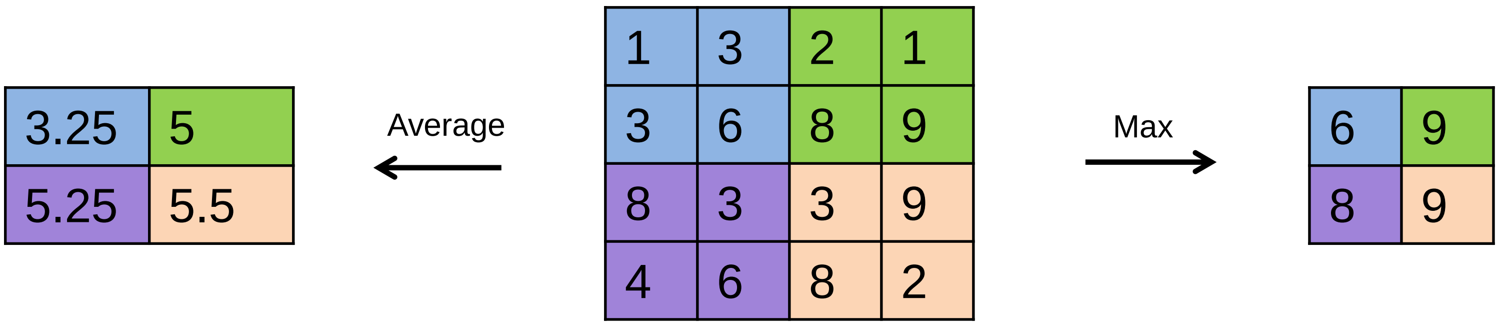
1) Max Pooling

2) Average Pooling

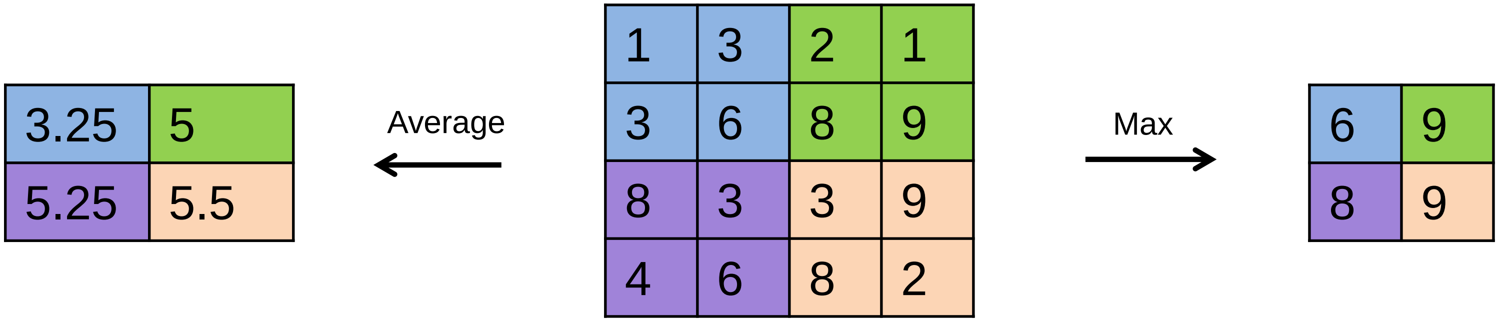
Pooling Layer



Pooling Layer



Pooling Layer



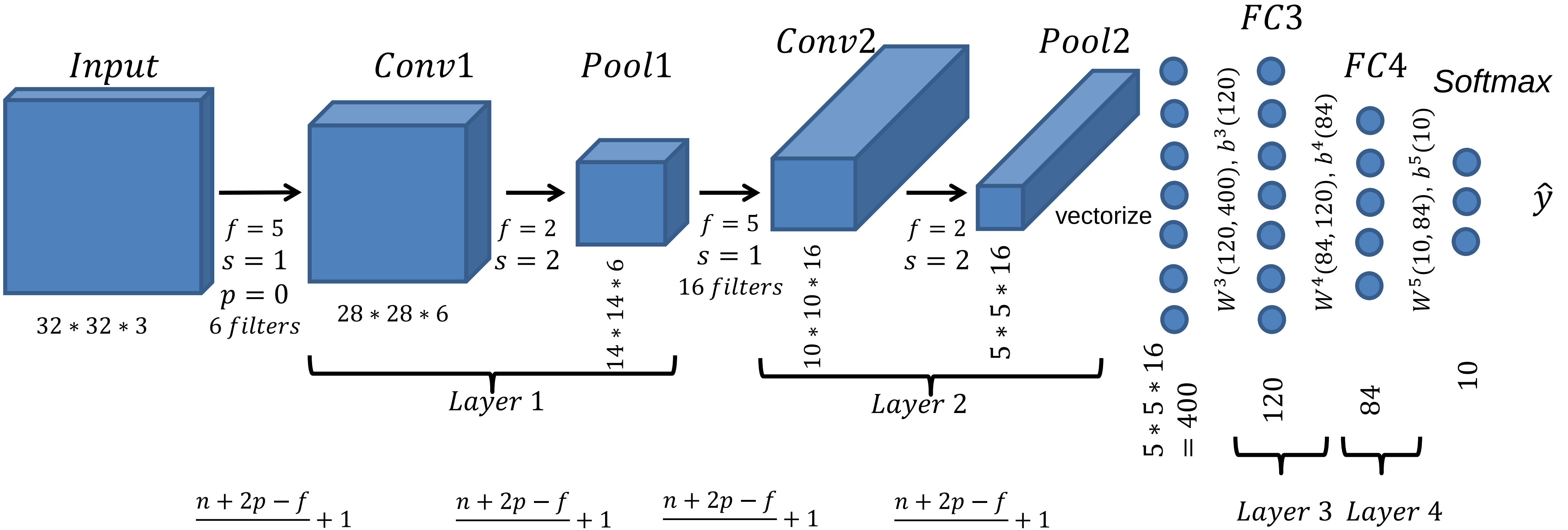
Salient features:

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.

CNN Example # 2 (Fully Loaded)



$$\frac{n+2p-f}{s}+1 = \frac{32+2*0-5}{1}+1 = 28$$

$$\frac{n+2p-f}{s}+1 = \frac{28+2*0-2}{2}+1 = 14$$

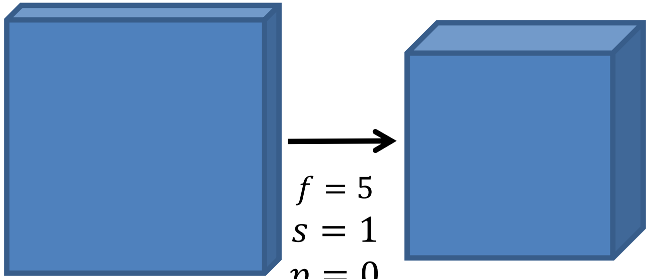
$$\frac{n+2p-f}{s}+1 = \frac{14+2*0-5}{1}+1 = 10$$

$$\frac{n+2p-f}{s}+1 = \frac{10+2*0-2}{2}+1 = 5$$

Computation of parameters

No	Layer Type	Input data dim.	Filter dim. (if any)	Output data dim.	Activation size	Weights	Biases	No. of learnable parameters
1	**Input**	-	-	32*32*3	3072			
2	**Conv Layer 1**	32*32*3	6 filters * 5*5*3	28*28*6	4704	450	6	456
3	**ReLU 1**	28*28*6	-	28*28*6	4704			
4	**Maxpool 1**	28*28*6	2*2*6	14*14*6	1176			
5	**Conv Layer 2**	14*14*6	16 filters * 5*5*6	10*10*16	1600	2400	16	2416
6	**ReLU 2**	10*10*16	-	10*10*16	1600			
7	**Maxpool 2**	10*10*16	2*2*16	5*5*16	400			
8	**FC Layer 3**	5*5*16 = 400	-	120	120	48000	1	48001
9	**ReLU 3**	120	-	120	120			
10	**FC Layer 4**	120	-	84	84	10080	1	10081
11	**ReLU 4**	84	-	84	84			
12	**Softmax Output**	84	-	10	10	840	1	841
						Total Learnable Parameters:		**61,795**

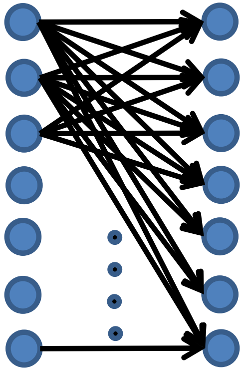
Why Convolutional ?



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

$32 * 32 * 3$ $\quad$ $28 * 28 * 6$

FC Neural Network



$32 * 32 * 3 = 3072$ $\quad$ $28 * 28 * 6 = 4704$

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

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

Why Convolutional ?

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
\end{array}
*
\begin{array}{|c|c|c|}
\hline
1 & 0 & -1 \\ \hline
1 & 0 & -1 \\ \hline
1 & 0 & -1 \\ \hline
\end{array}
=
\begin{array}{|c|c|c|c|}
\hline
0 & 30 & & \\ \hline
 & & & \\ \hline
 & & & \\ \hline
 & & & \\ \hline
\end{array}
$$

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 ?

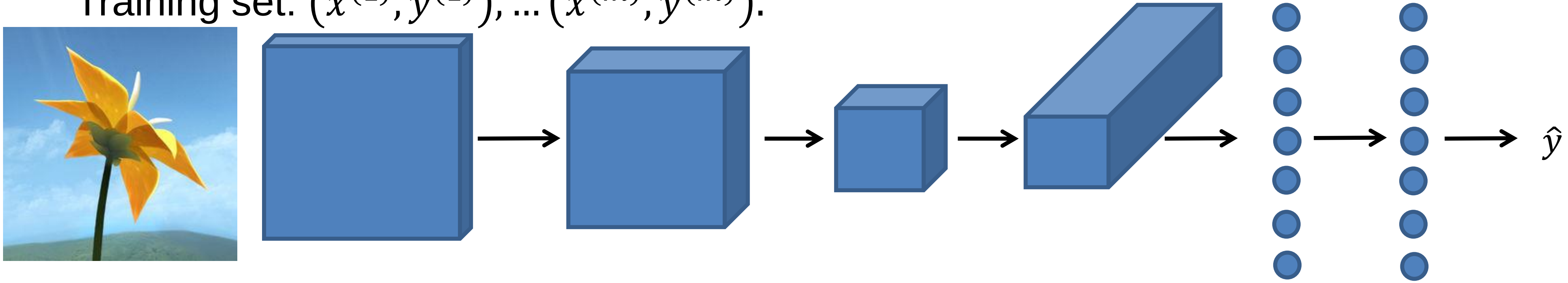
$$
\begin{bmatrix}
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0 \\
10 & 10 & 10 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
1 & 0 & -1 \\
1 & 0 & -1 \\
1 & 0 & -1
\end{bmatrix}
=
\begin{bmatrix}
0 & 30 & 30 &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

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.

Putting it altogether

Training set: $\left(x^{(1)}, y^{(1)}\right), \ldots \left(x^{(m)}, y^{(m)}\right)$.



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

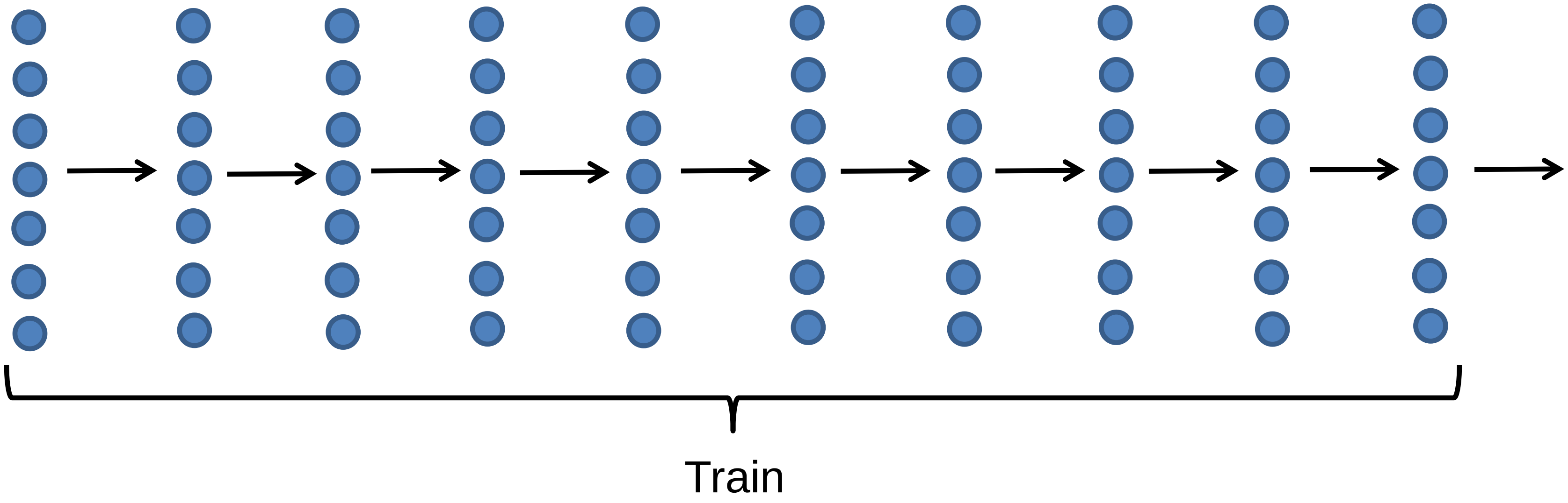
Use gradient descent (or other optimization algorithms) to optimize parameters to reduce J.

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.

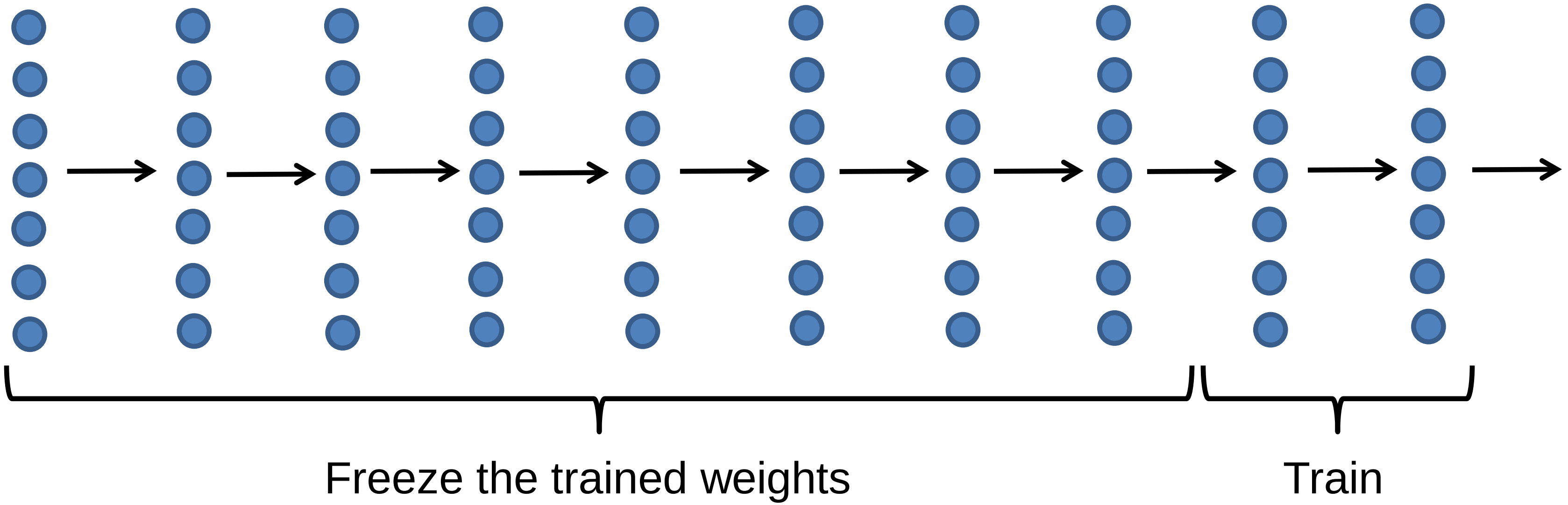
Transfer Learning

1) Train from scratch



Transfer Learning

2) Transfer Learning



Pretrained models of AlexNet, VGG-16, VGG-19, GoogleNet, ZFNet, etc. trained on (say) ImageNet dataset.

Transfer Learning

Parameters	Training from scratch	Transfer Learning
Method	Build CNN from scratch	Only last few layers need to be trained
Tuning	Need to tune large number of hyperparameters	Only a few hyperparameters need to be tuned
Computation	Large computation power is required (multiple GPUs)	Less computation power needed (can even work with CPUs)
Dataset	Huge dataset needed to avoid overfitting	A small dataset is enough
Training time	May take weeks or even months	May take hours to train

Popular Languages for Deep Learning

- Python
- Java
- R
- C++
- C
- Javascript
- Scala
- Julia

Popular Deep Learning Libraries

- TensorFlow
- Pytorch
- Caffe
- Keras
- Theano
- Deep Learning Toolbox – Matlab
- MatConvNet

Popular Software / IDE for Deep Learning

- PyCharm
- Spyder
- Jupyter Notebook
- Matlab (Deep Learning toolbox) R2018b
- Anaconda (contains Sypder, Jupyter Notebook, numpy, scipy, etc.)
- Google CoLab (similar to Jupyter Notebook; gives K80 GPU for 12hrs at a time.)

Resources

- http://cs231n.github.io/convolutional-networks/
- https://www.coursera.org/learn/convolutional-neural-networks
- https://itunes.apple.com/us/app/flower/id1279174518?mt=8
- https://fossbytes.com/popular-top-programming-languages-machine-learning-data-science/

Thank you!

Any Queries?