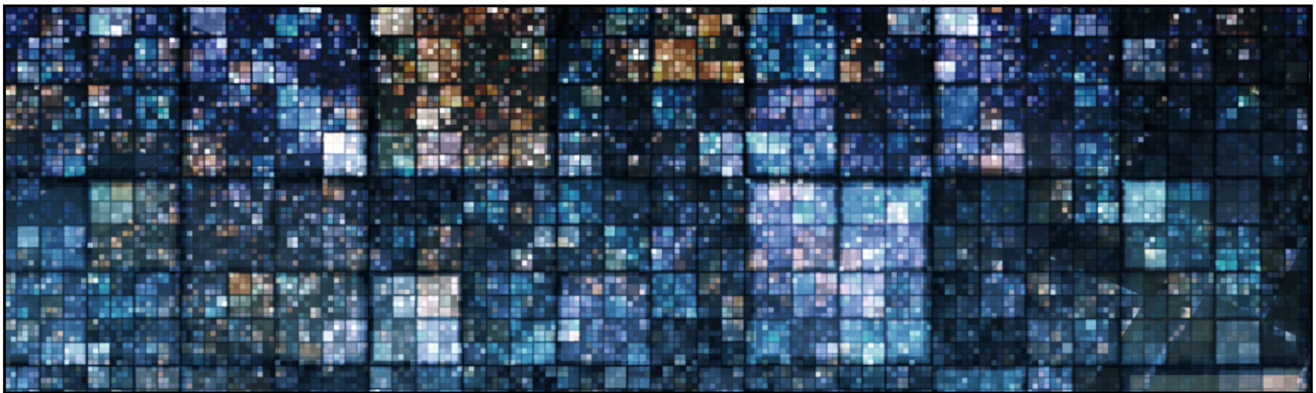


COMP47590

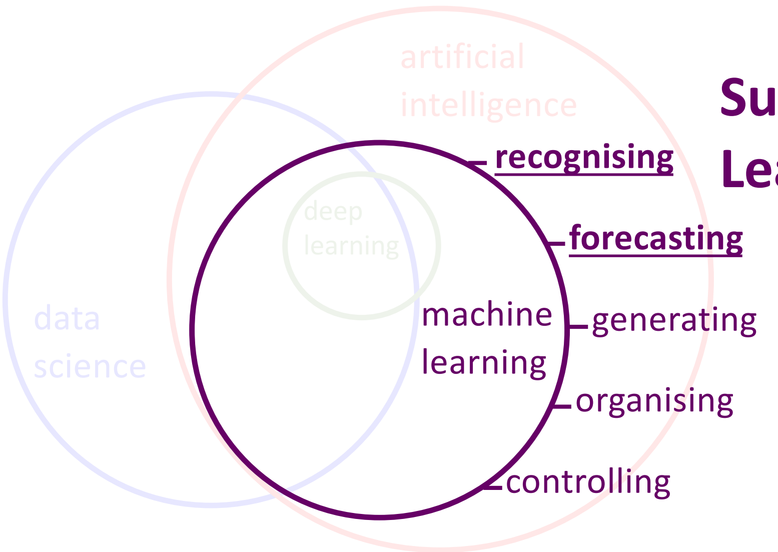
Advanced Machine Learning

Deep Learning - CNNs

Dr. Brian Mac Namee

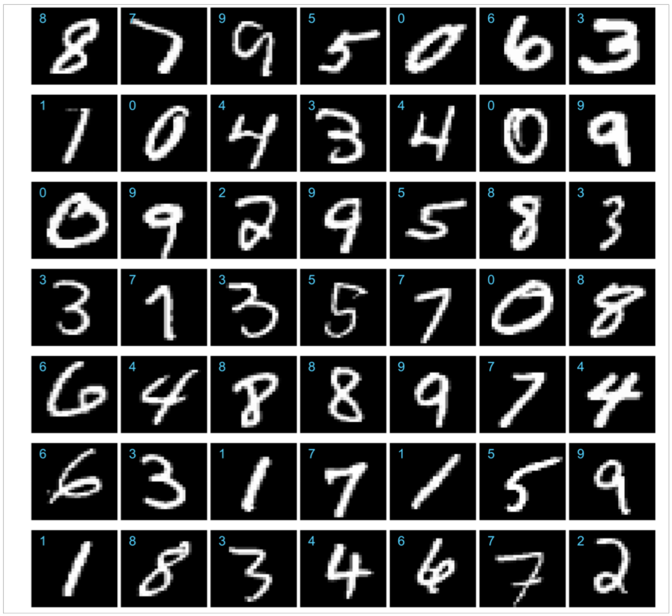
Information

Email:	Brian.MacNamee@ucd.ie
Course Materials:	All material posted on UCD CS moodle https://csmoodle.ucd.ie/moodle/course/view.php?id=663 Enrolment key **UCDAdvML2017**



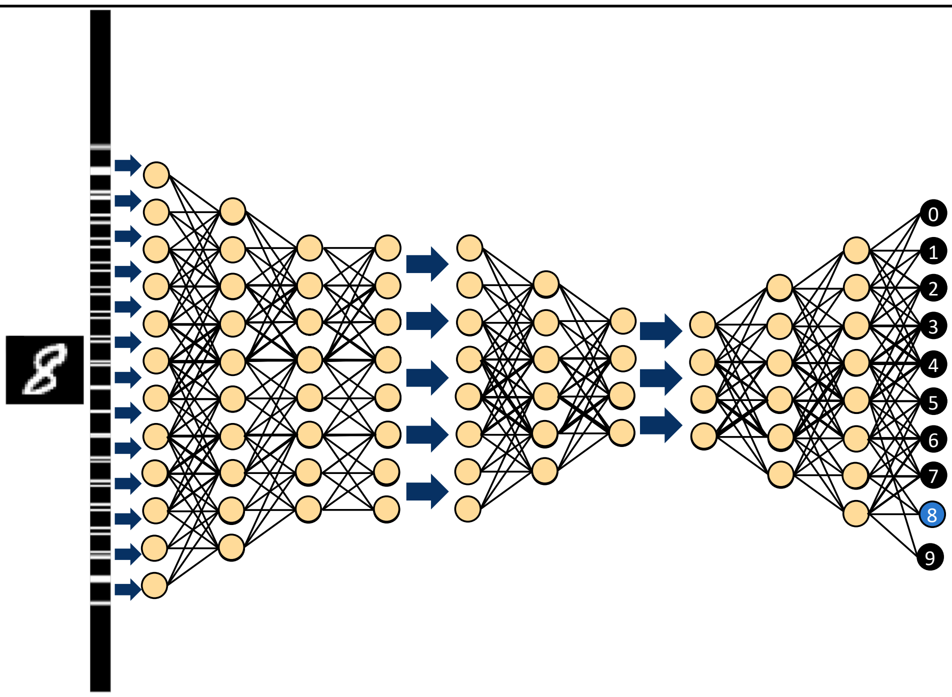
Supervised Learning

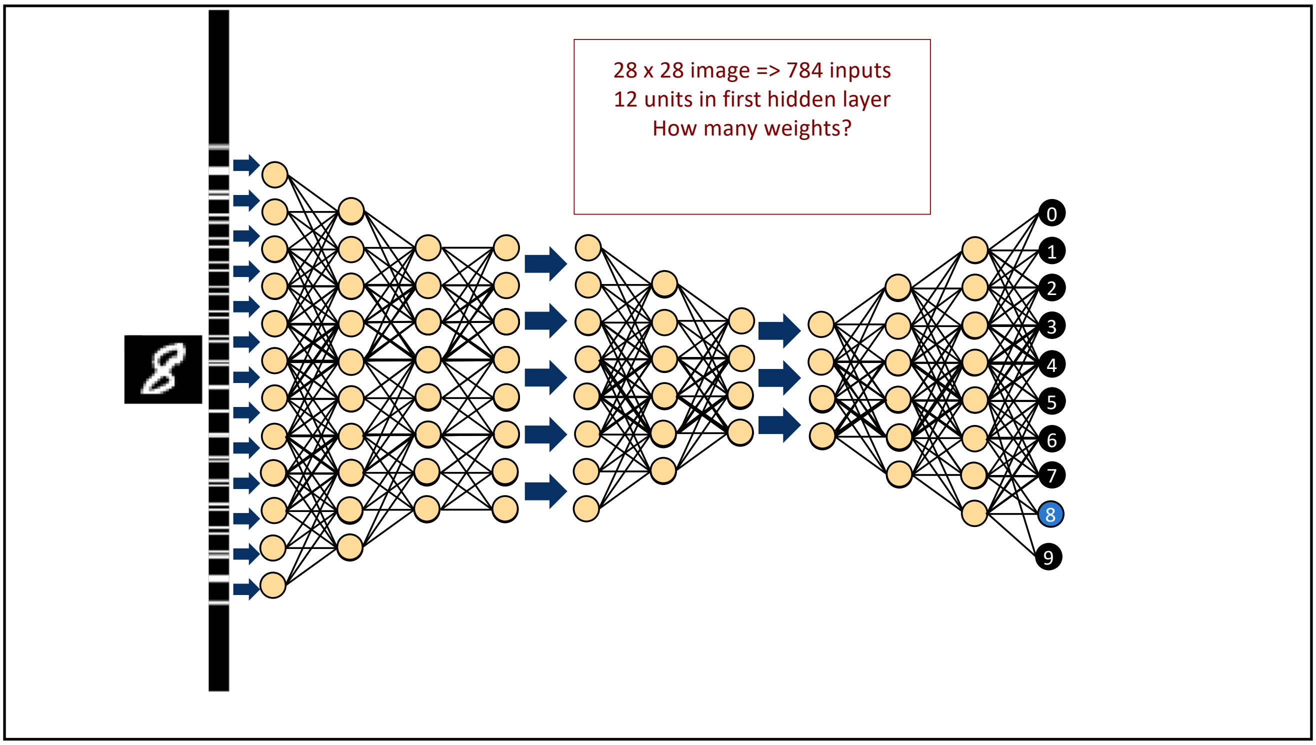
CONVOLUTIONAL NEURAL NETWORKS

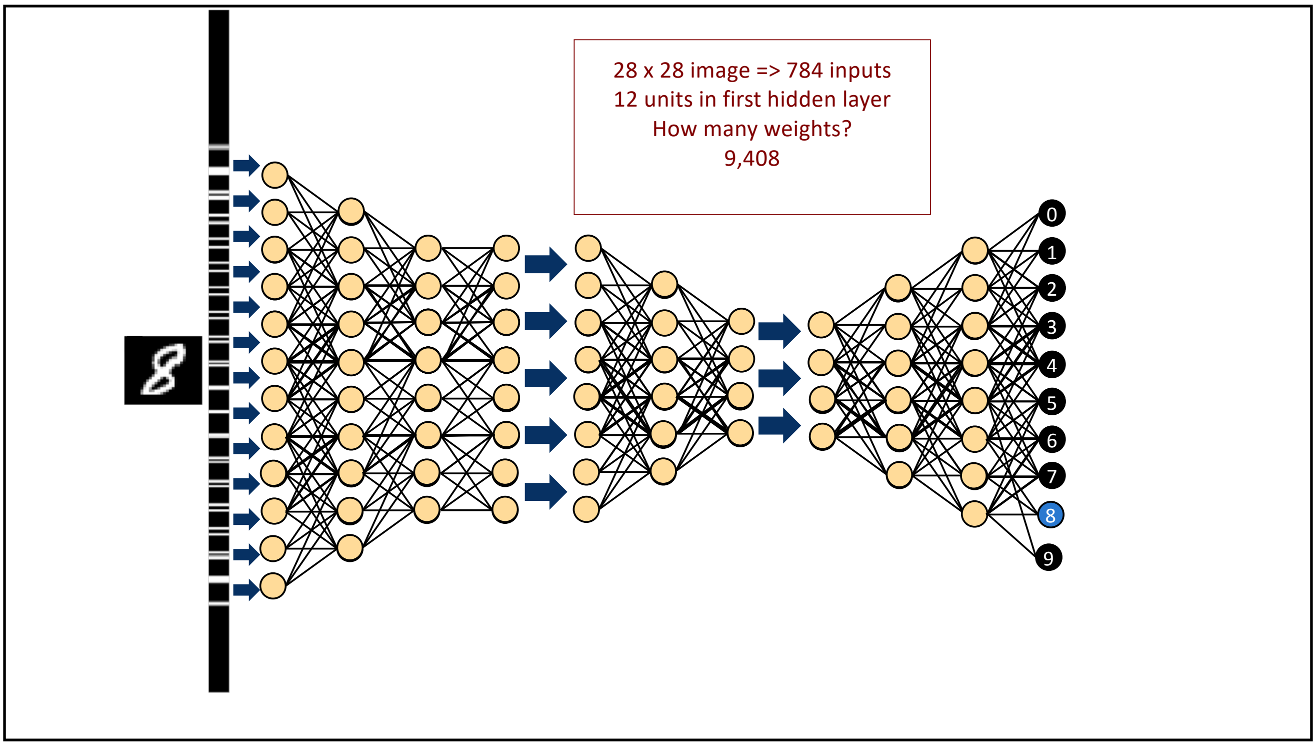


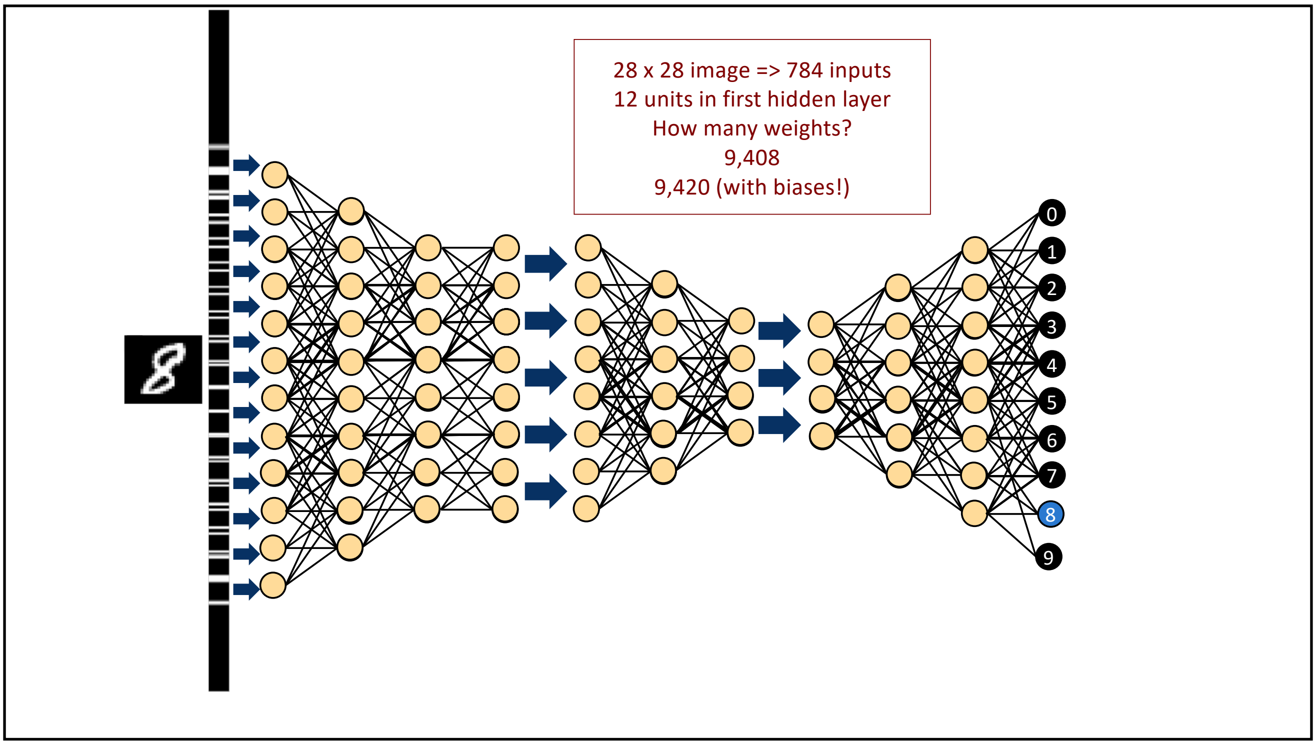
MNIST Dataset from Yan LeCun
http://yann.lecun.com/exdb/mnist/index.html

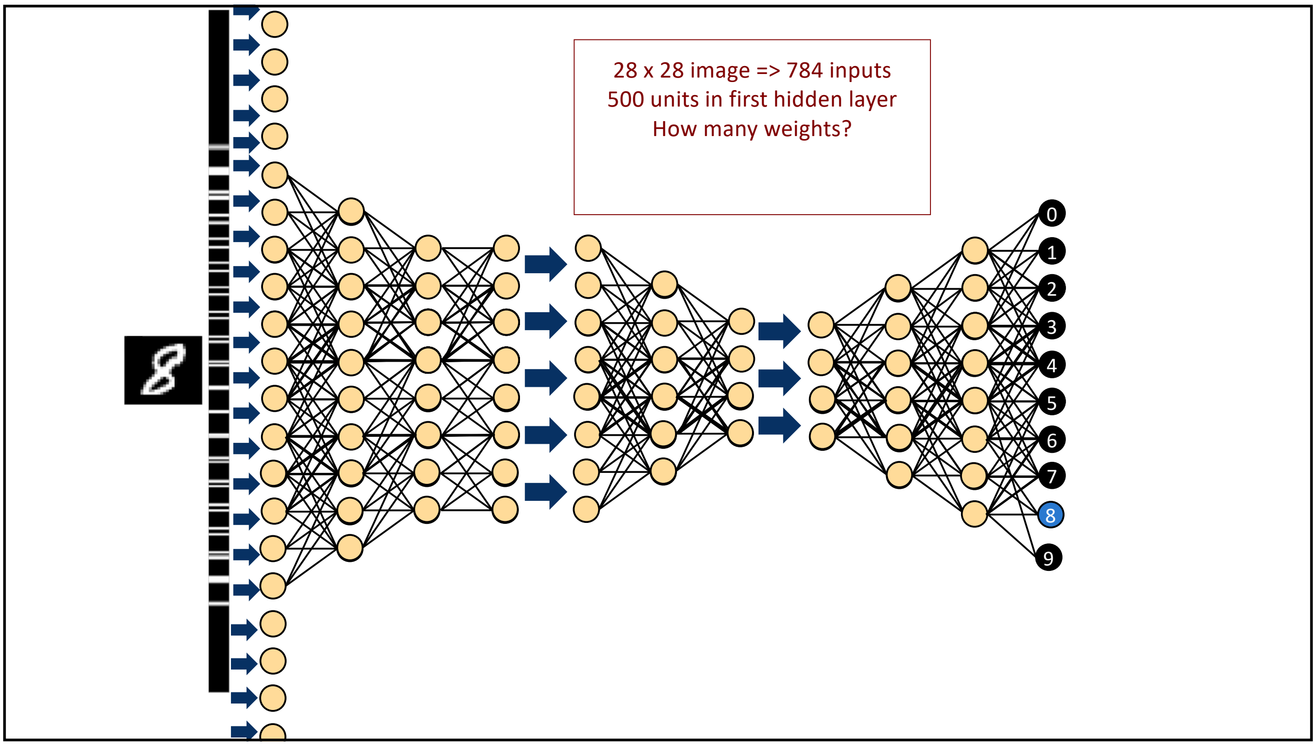
Kaggle Digit Recogniser Contest
https://www.kaggle.com/c/digit-recognizer

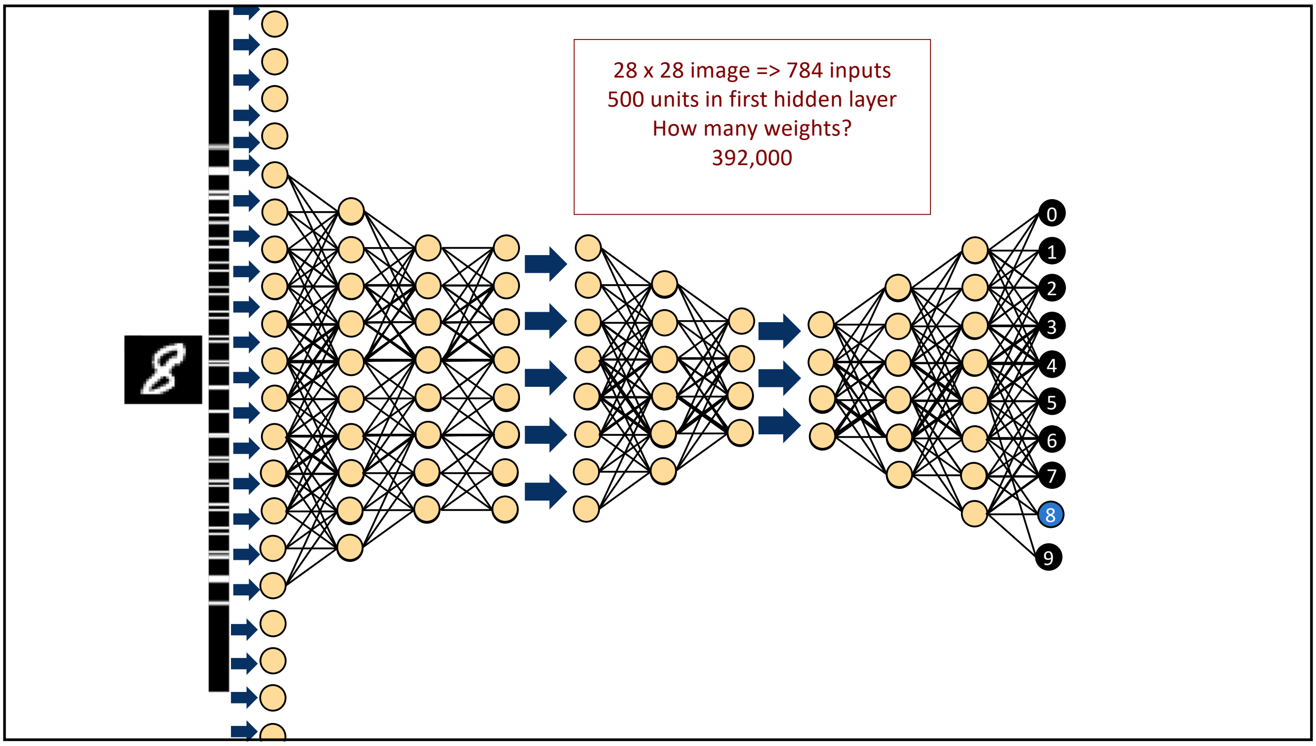


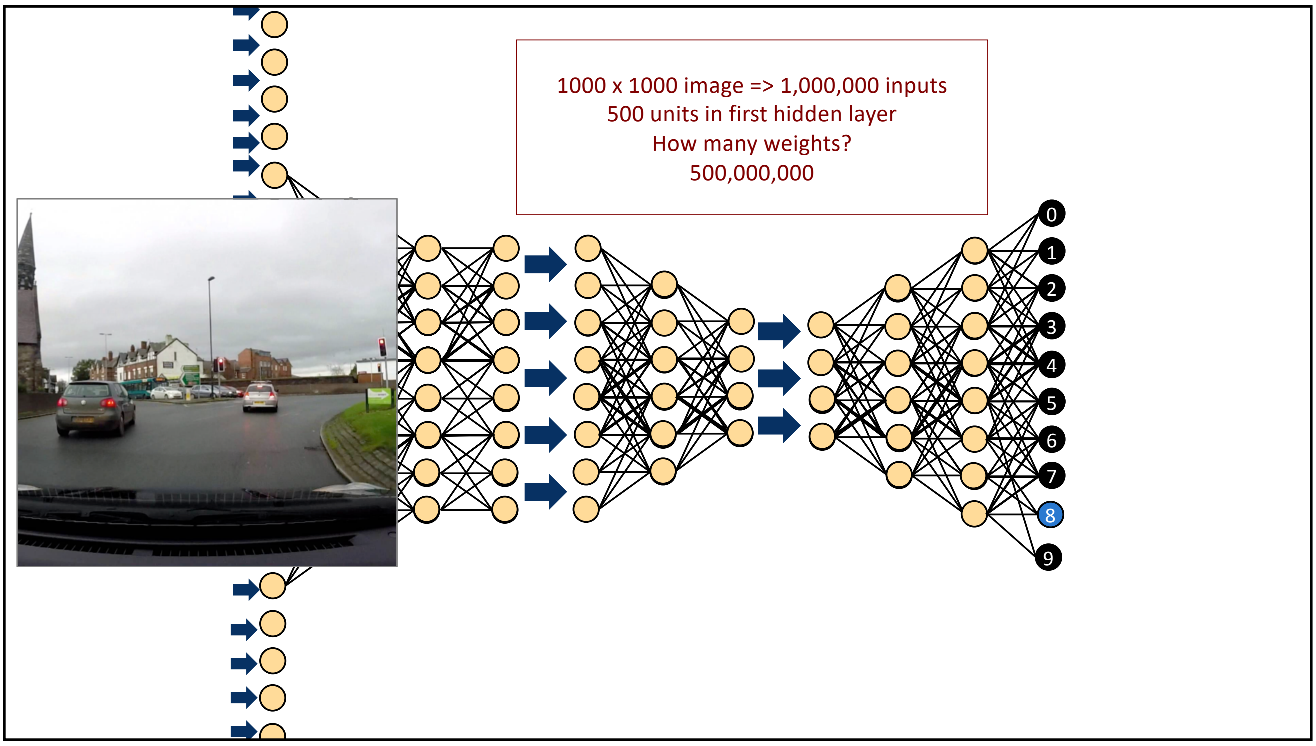

28 x 28 image => 784 inputs
12 units in first hidden layer
How many weights?
0
1
2
3
4
5
6
7
8
9

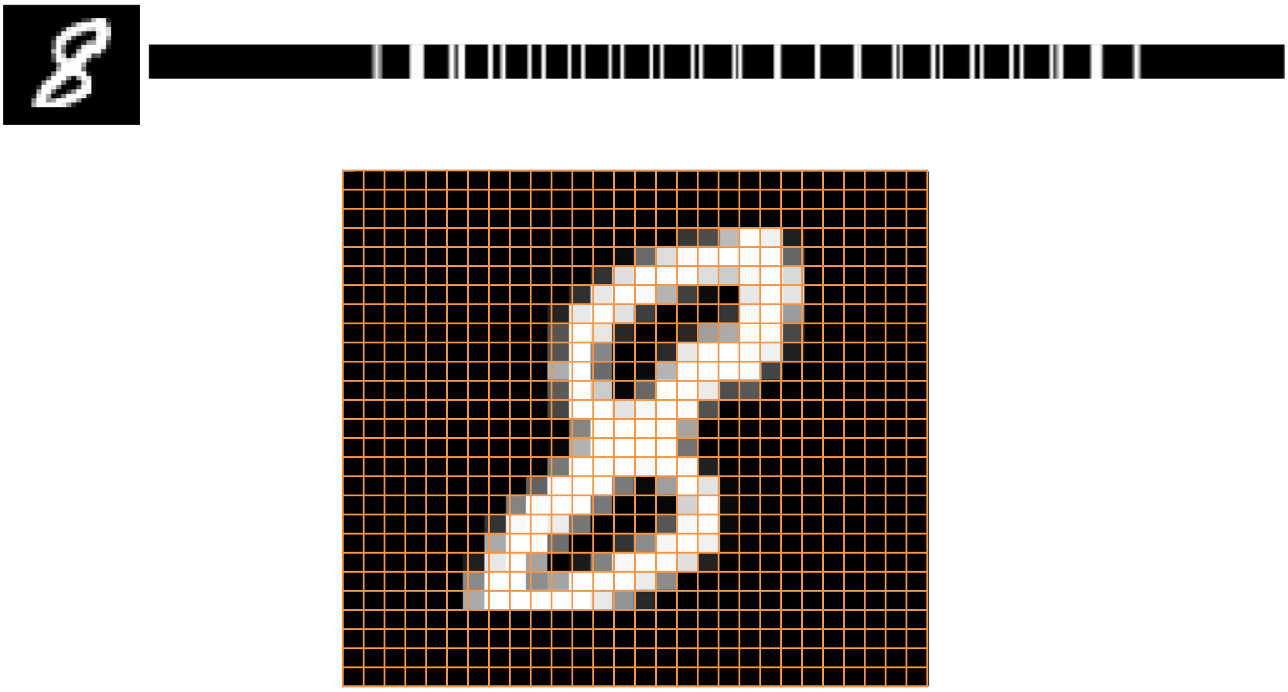

28 x 28 image => 784 inputs
12 units in first hidden layer
How many weights?
9,408
0
1
2
3
4
5
6
7
8
9


28 x 28 image => 784 inputs
12 units in first hidden layer
How many weights?
9,408
9,420 (with biases!)
0
1
2
3
4
5
6
7
8
9


28 x 28 image => 784 inputs
500 units in first hidden layer
How many weights?
0
1
2
3
4
5
6
7
8
9


28 x 28 image => 784 inputs
500 units in first hidden layer
How many weights?
392,000
0
1
2
3
4
5
6
7
8
9


1000 x 1000 image => 1,000,000 inputs
500 units in first hidden layer
How many weights?
500,000,000
0
1
2
3
4
5
6
7
8
9

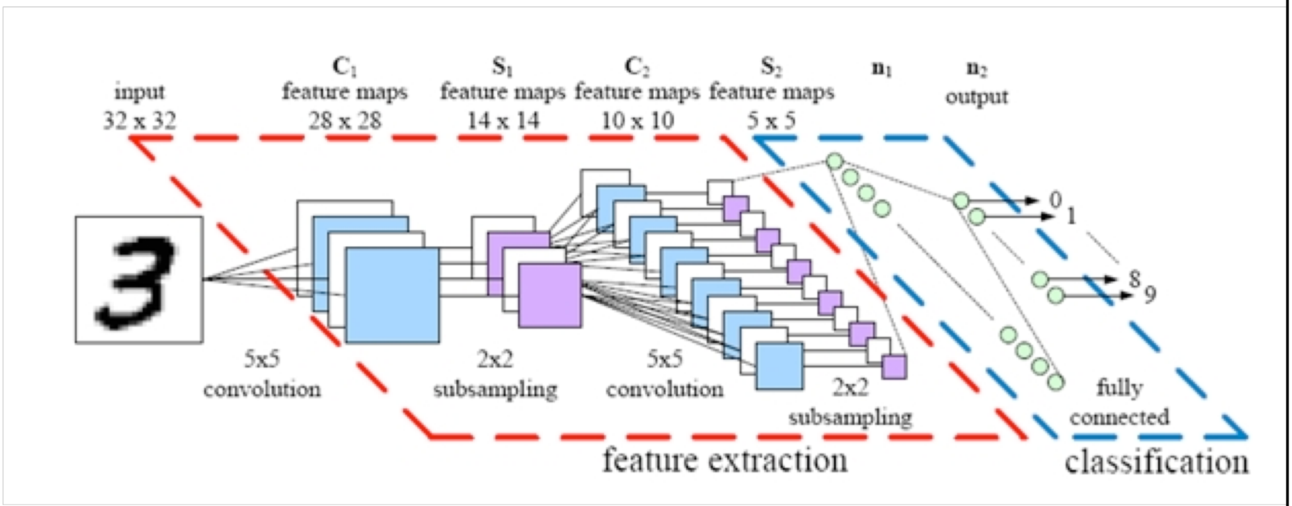


Convolutional Neural Networks

The advent of **convolutional neural networks** (CNN) has been a significant driver of advances in the performance of machine learning systems in recent years

We will look at their key components:

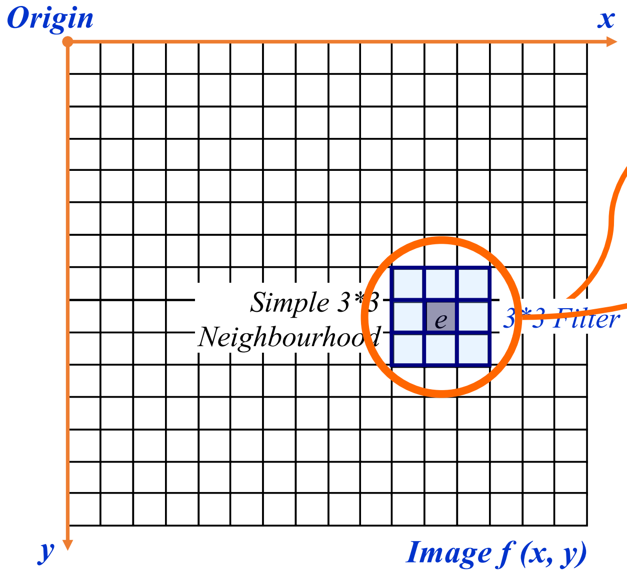
- Convolution
- Convolutional layers
- Pooling layers
- CNN architectures
- Why convolutions?



Efficient mapping of the training of Convolutional Neural Networks to a CUDA-based cluster
Jonathan Ward, Sergey Andreev, Francisco Heredia, Bogdan Lazar, Zlatka Manevska
http://parse.ele.tue.nl/education/cluster2

CONVOLUTION

Convolution



a	b	c
d	e	f
g	h	i

Original Image Pixels

$*$

r	s	t
u	v	w
x	y	z

Filter

$$
\begin{aligned}
e_{processed} = {} & v*e + \\
& r*a + s*b + t*c + \\
& u*d + w*f + \\
& x*g + y*h + z*i
\end{aligned}
$$

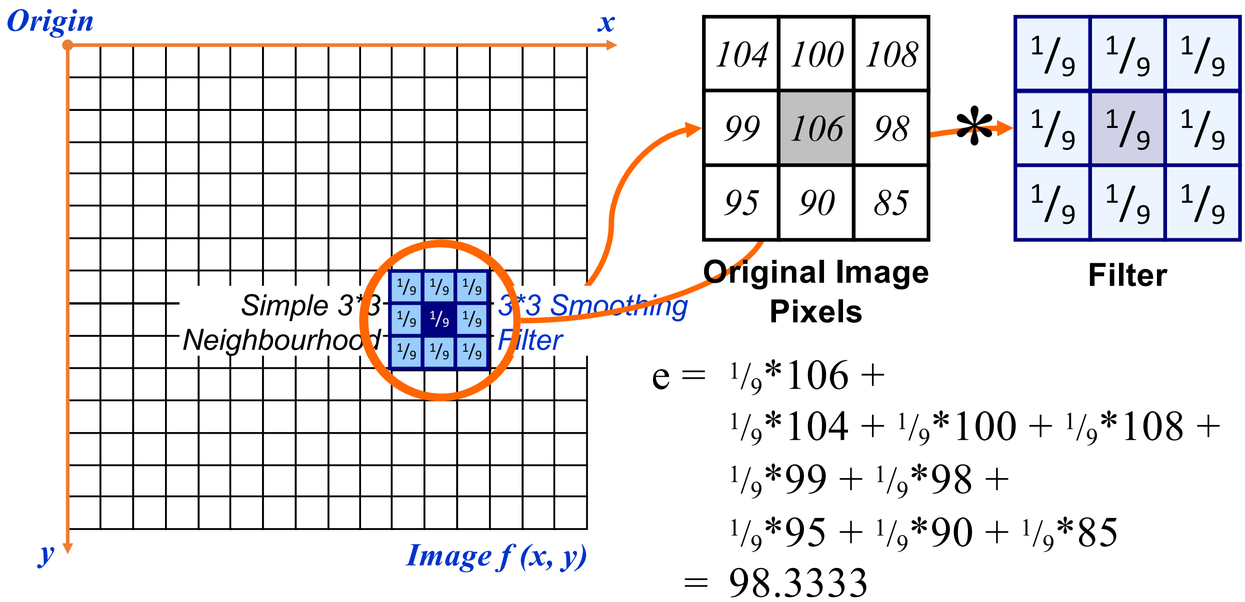
Convolution

Image processing uses convolutions for all kinds of basic image processing operations

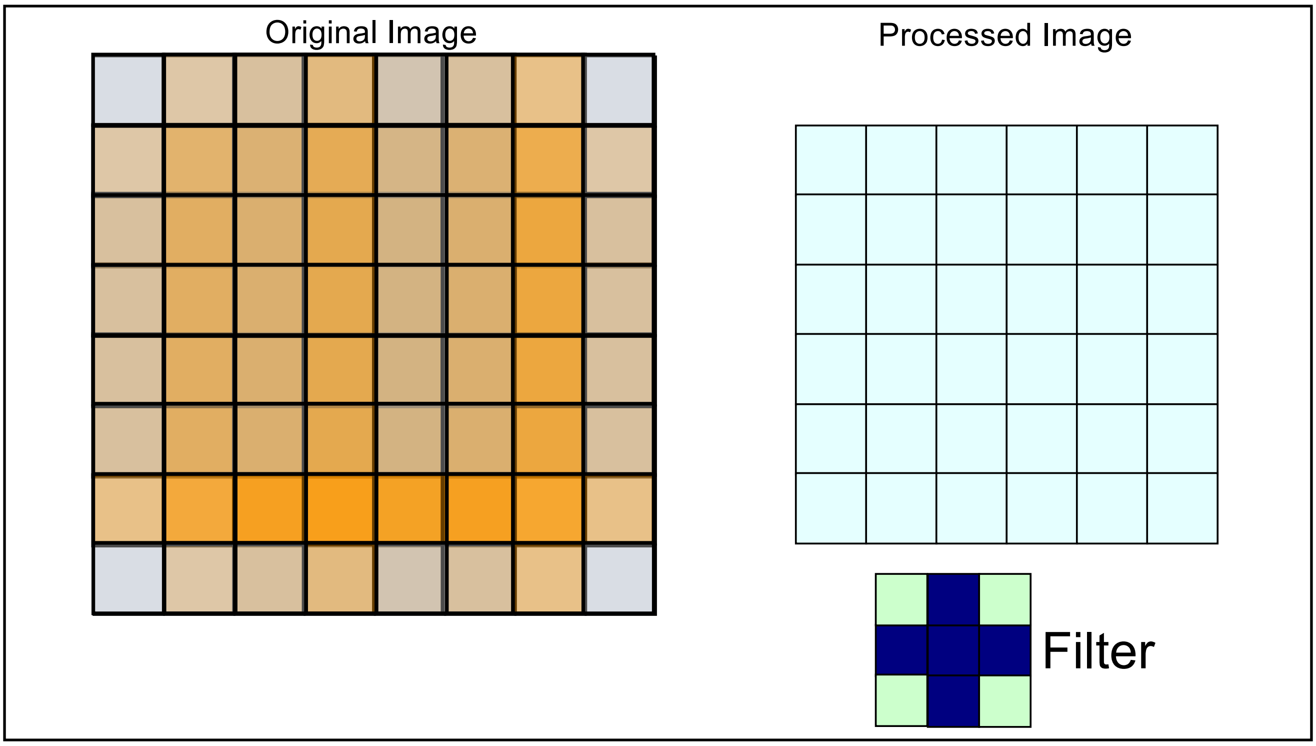
- Smoothing filter
- Useful for removing noise
- Also useful for highlighting gross detail

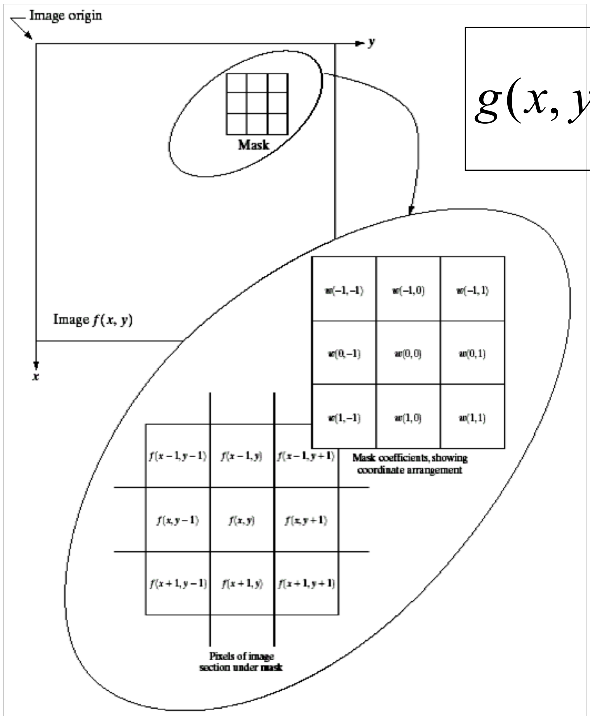
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

Simple smoothing filter



$$
\begin{aligned}
e = {} & \tfrac{1}{9}*106 + \\
& \tfrac{1}{9}*104 + \tfrac{1}{9}*100 + \tfrac{1}{9}*108 + \\
& \tfrac{1}{9}*99 + \tfrac{1}{9}*98 + \\
& \tfrac{1}{9}*95 + \tfrac{1}{9}*90 + \tfrac{1}{9}*85 \\
= {} & 98.3333
\end{aligned}
$$





$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x+s,y+t)$$

Convolution can be given in equation form as shown above

Notations are based on the image shown to the left

Gonzalez & Woods, Digital Image Processing, 2002)

Original Image

123	127	128	119	115	130
140	145	148	153	167	172
133	154	183	192	194	191
194	199	207	210	198	195
164	170	175	162	173	151

*

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

=

Filtered Image

Original Image

123	127	128	119	115	130
140	145	148	153	167	172
133	154	183	192	194	191
194	199	207	210	198	195
164	170	175	162	173	151

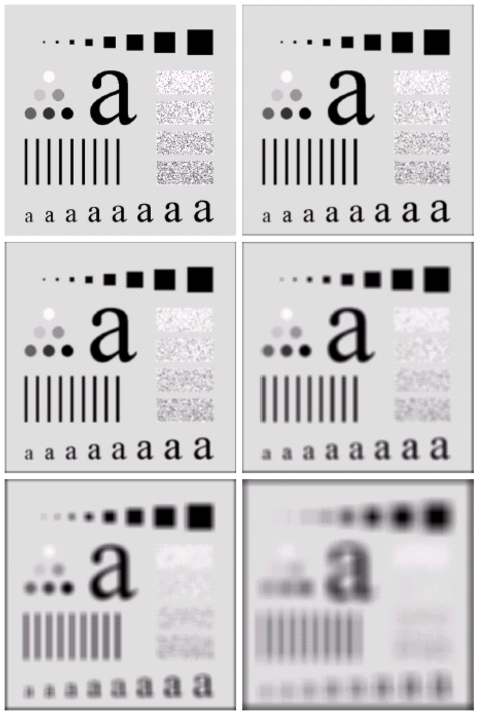
$$*\ \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} =$$

Filtered Image

142.33	149.89	155.44	159.22
167.00	176.78	183.56	185.78
175.44	183.56	188.22	185.11

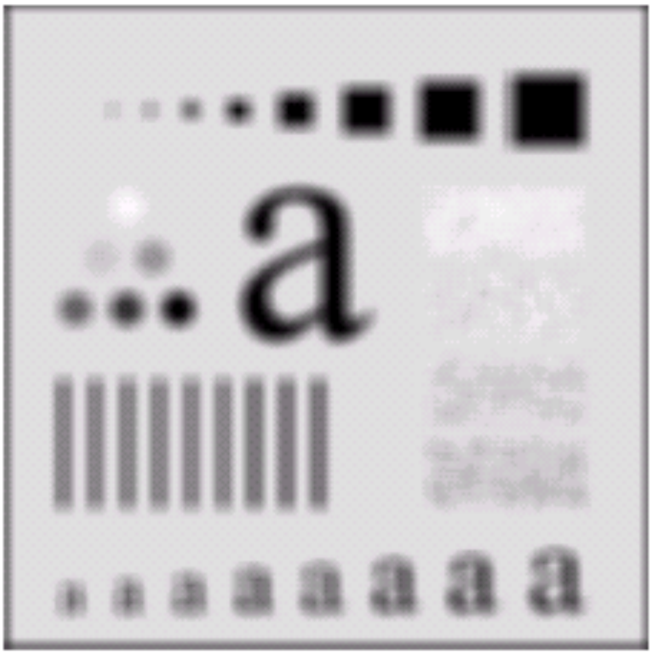
Smoothing Filter

- The image at the top left is an original image of size 500 x 500 pixels
- The subsequent images show the image after filtering with an averaging filter of increasing sizes
  - 3, 5, 9, 15 and 35



Gonzalez & Woods, Digital Image Processing, 2002)

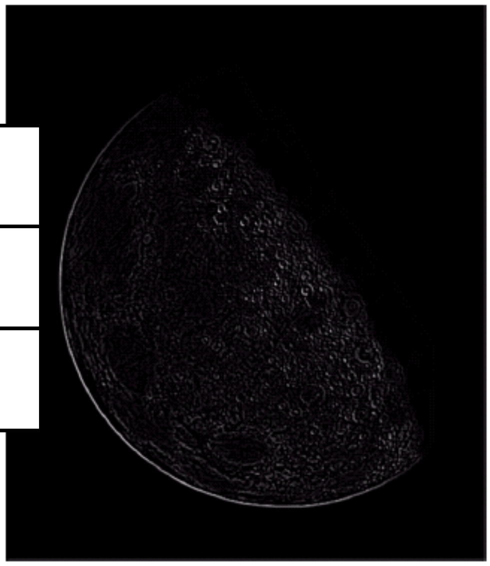
Gonzalez & Woods, Digital Image Processing, 2002)



Gonzalez & Woods, Digital Image Processing, 2002)

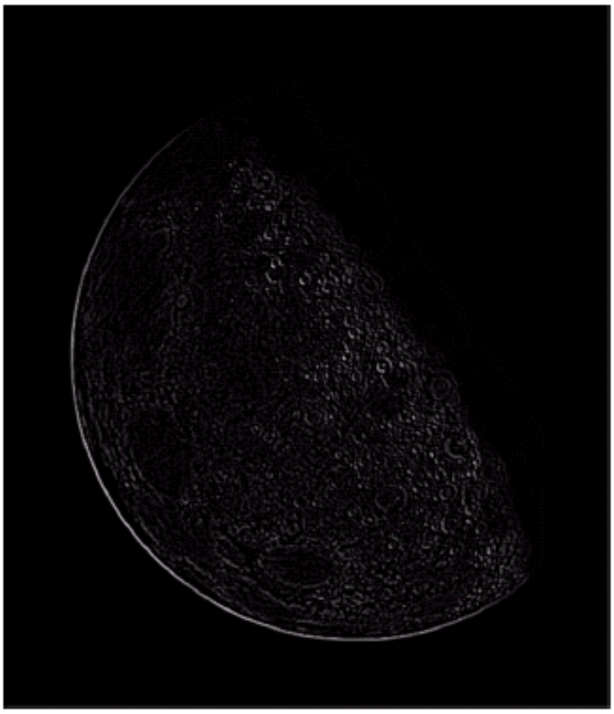
Edge Detecting Filter

0	1	0
1	-4	1
0	1	0



Gonzalez & Woods, Digital Image Processing, 2002)

Gonzalez & Woods, Digital Image Processing, 2002)



Gonzalez & Woods, Digital Image Processing, 2002)

Sharpening Filter

0	-1	0
-1	5	-1
0	-1	0

Gonzalez & Woods, Digital Image Processing, 2002)


Gonzalez & Woods, Digital Image Processing, 2002)


Gonzalez & Woods, Digital Image Processing, 2002)

CONVOLUTION VS. CORRELATION

Convolution vs. Correlation

Strictly speaking what we have been talking about is actually not convolution - it is cross correlation

Convolution involves a flipping the filter

a	b	c
d	e	f
g	h	i

Original Image Pixels

$*$

r	s	t
u	v	w
x	y	z

Filter

$$\begin{aligned} e_{processed} = {} & v*e + \\ & z*a + y*b + x*c + \\ & w*d + u*f + \\ & t*g + s*h + r*i \end{aligned}$$

Convolution vs. Correlation

Strictly speaking what we have been talking about is actually not convolution - it is cross correlation

Convolution involves a flipping the filter

a	b	c
d	e	f
g	h	i

Original Image Pixels

$*$

z	y	x
w	v	u
t	s	r

Filter

$$
\begin{aligned}
e_{processed} = {} & v*e + \\
& z*a + y*b + x*c + \\
& w*d + u*f + \\
& t*g + s*h + r*i
\end{aligned}
$$

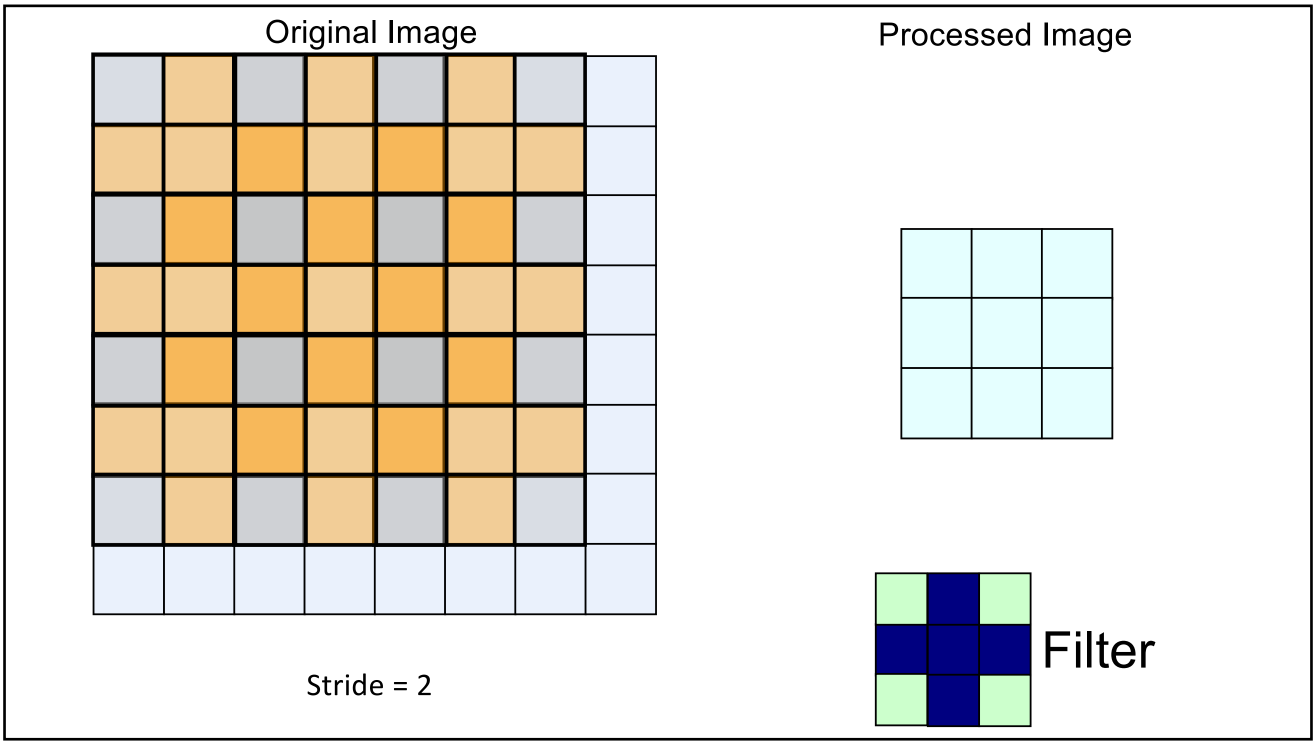
Convolution vs. Correlation

When training a neural network there is no need to worry about this (we are learning the weights anyway) and most implementations use cross correlation

STRIDE

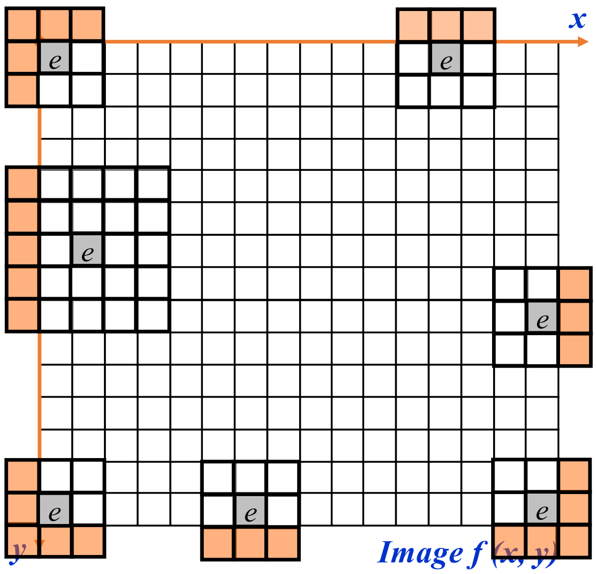
Stride

- We don’t have to place the filter over every pixel in the original image - we can *stride* across some pixels
- This essentially achieves dimensionality reduction

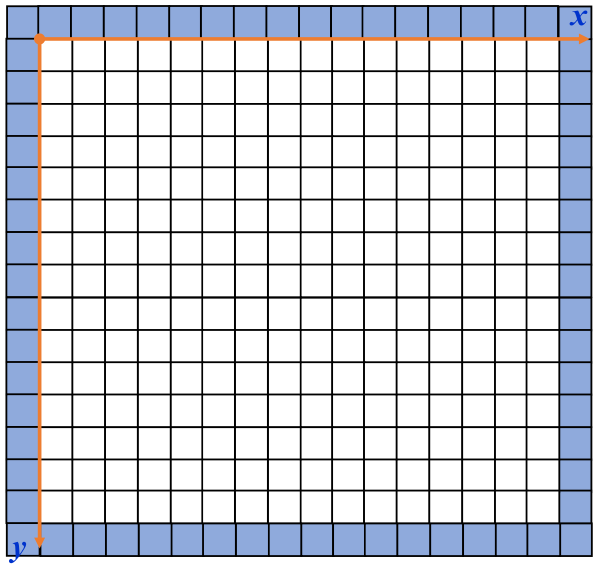


PADDING

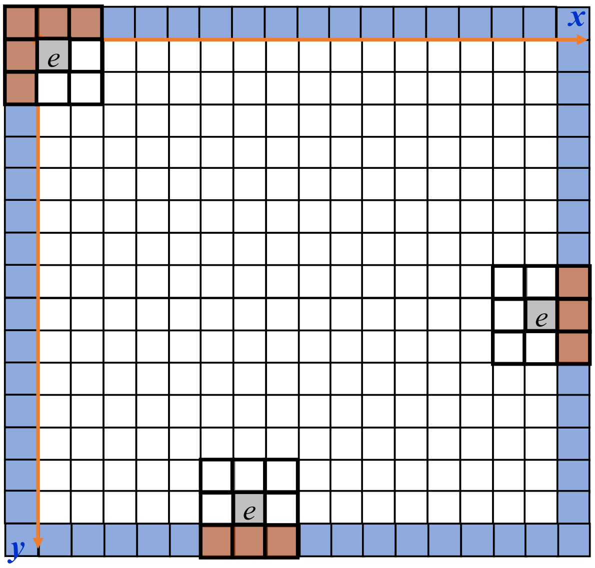
At the edges of an image we have a problem



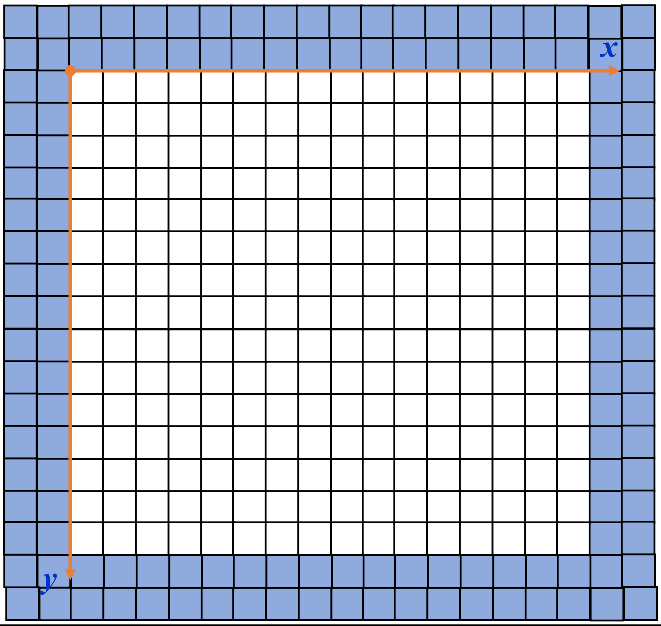
At the edges of an image we have a problem



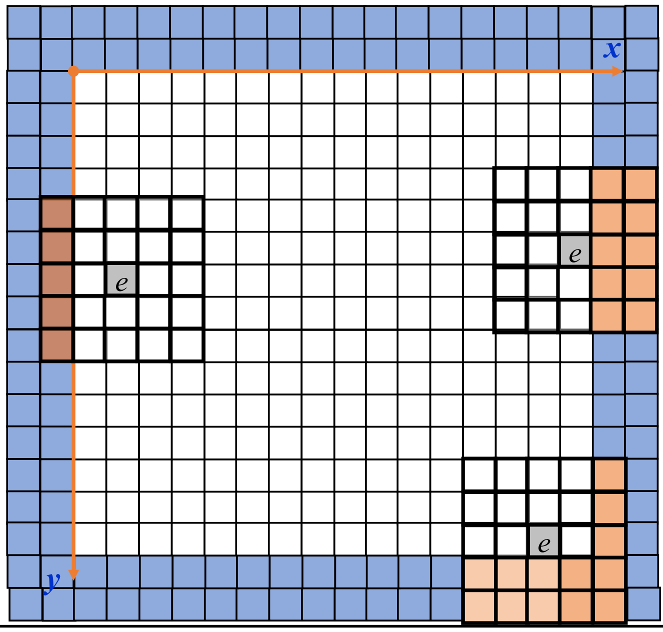
At the edges of an image we have a problem



At the edges of an image we have a problem



At the edges of an image we have a problem



Original Image

123	127	128	119	115	130
140	145	148	153	167	172
133	154	183	192	194	191
194	199	207	210	198	195
164	170	175	162	173	151

*

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

=

Filtered Image

Original Image

123	127	128	119	115	130
140	145	148	153	167	172
133	154	183	192	194	191
194	199	207	210	198	195
164	170	175	162	173	151

*

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

=

Filtered Image

59.44	90.11	91.11	92.22	95.11	64.89
91.33	142.33	149.89	155.44	159.22	107.67
107.22	167	176.78	183.56	185.78	124.11
112.67	175.44	183.56	188.22	185.11	122.44
80.78	123.22	124.78	125	121	79.67

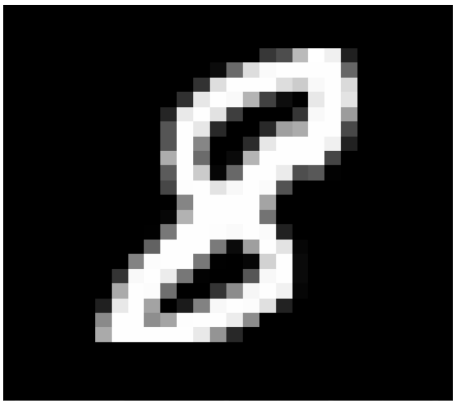
Padding

People often refer to two different types of convolution depending on whether or not padding is used:

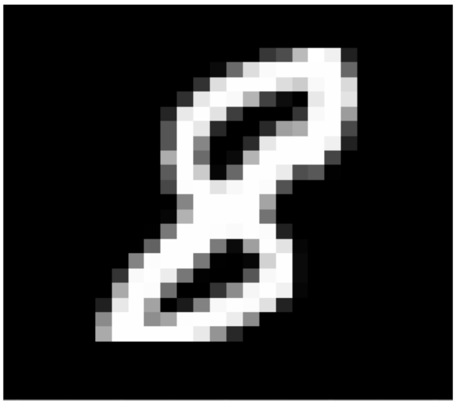
- **Valid convolution:** no padding
- **Same convolution:** padding

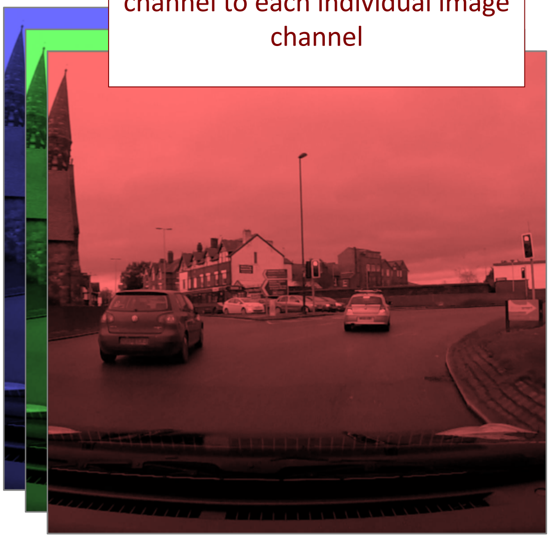
CONVOLUTIONS ON VOLUMES

Convolutions On Volumes



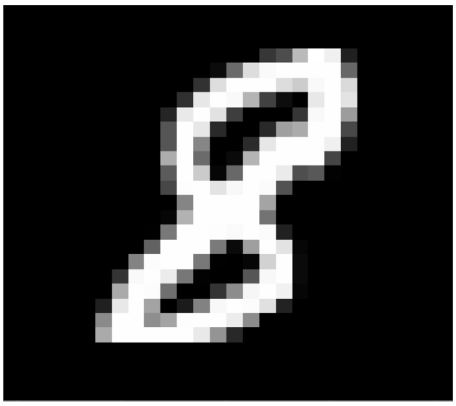
Convolutions On Volumes

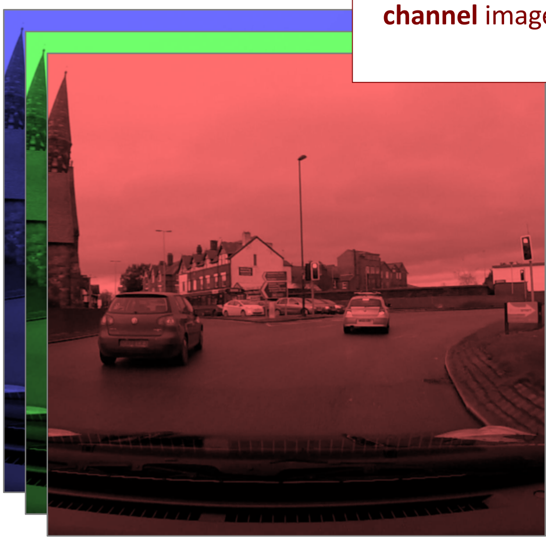




We sum the results of applying each individual convolution channel to each individual image channel

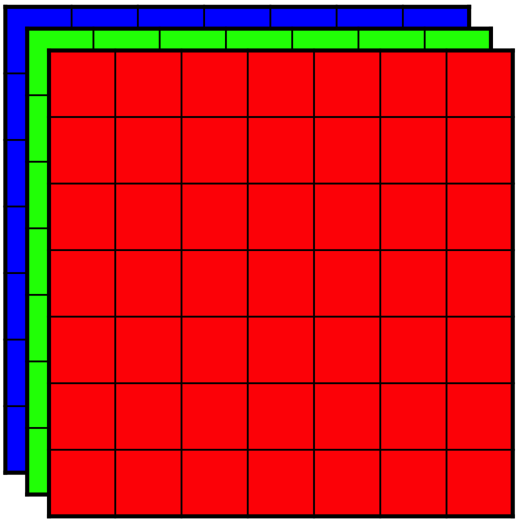
Convolutions On Volumes



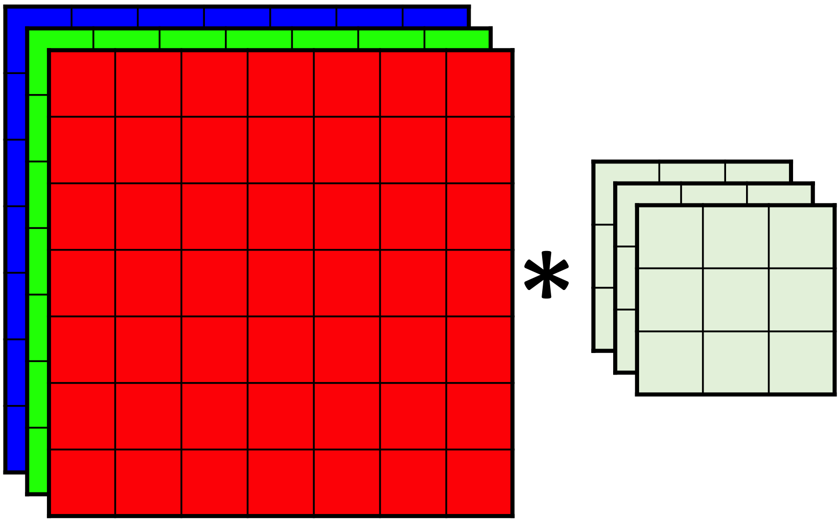


We refer to this as a **multi-channel** image

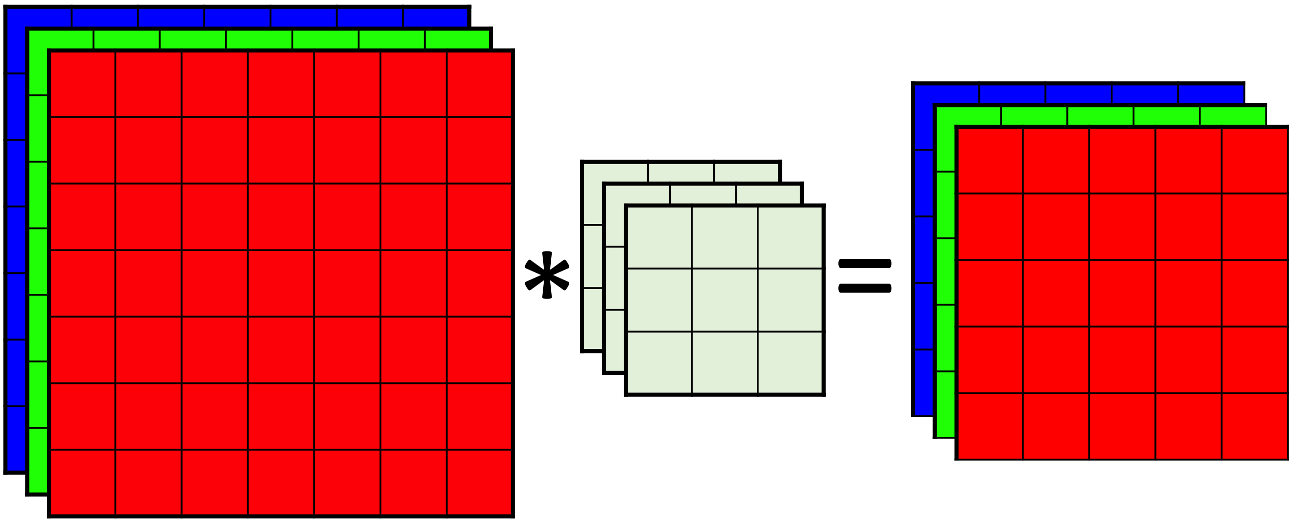
Convolutions On Volumes



Convolutions On Volumes

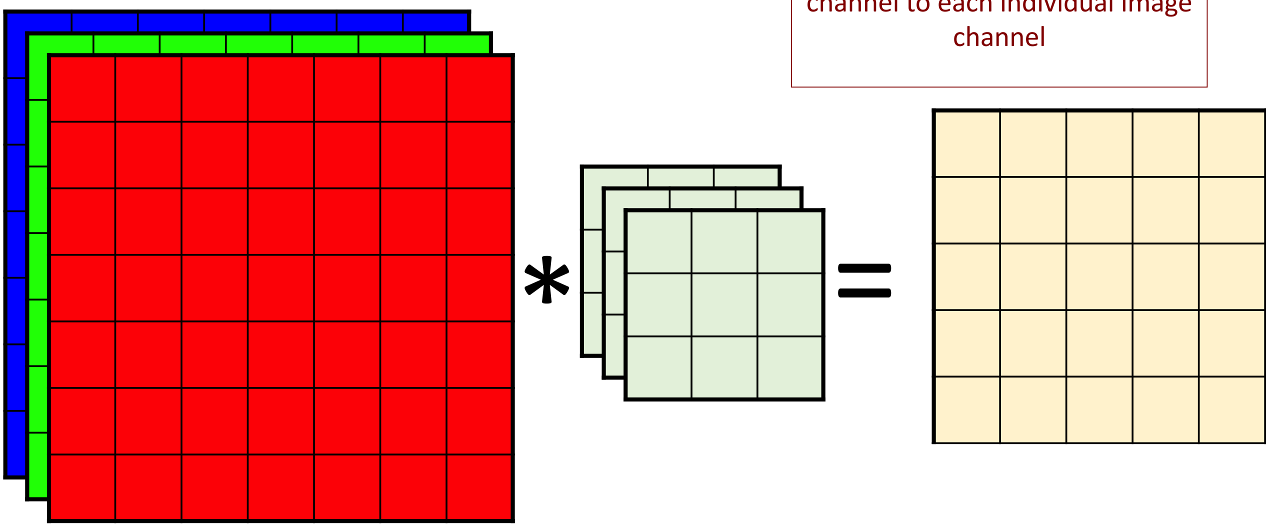


Convolutions On Volumes

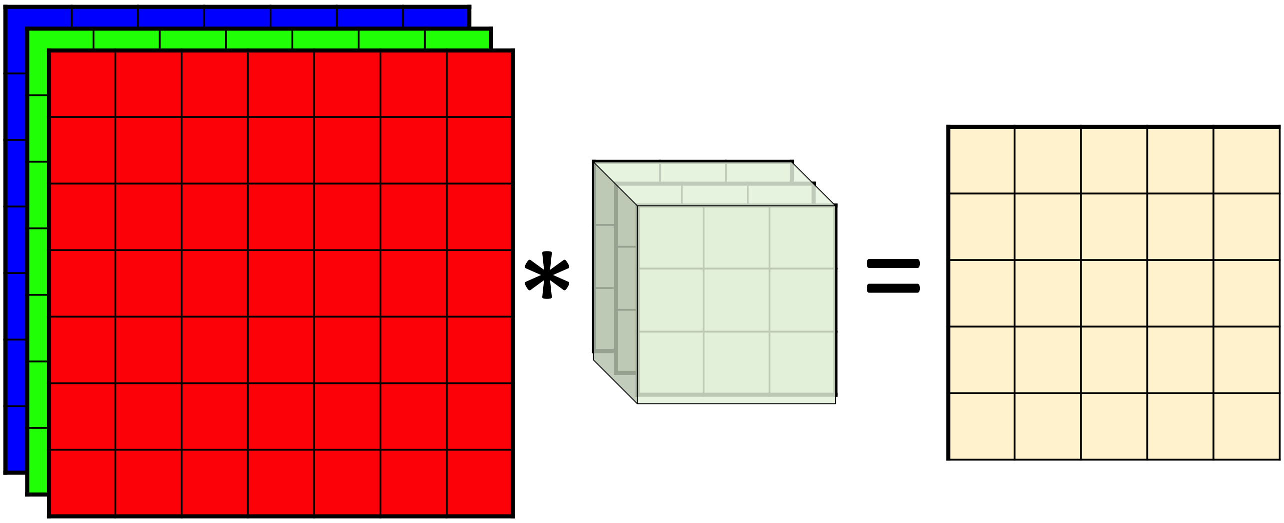


Convolutions On Volumes

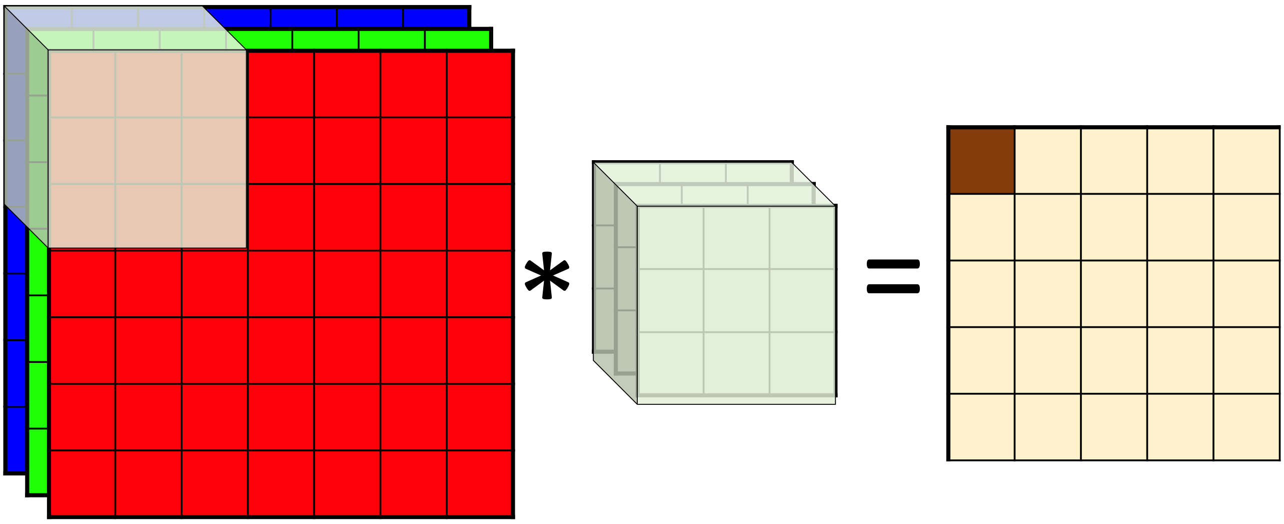
We sum the results of applying each individual convolution channel to each individual image channel



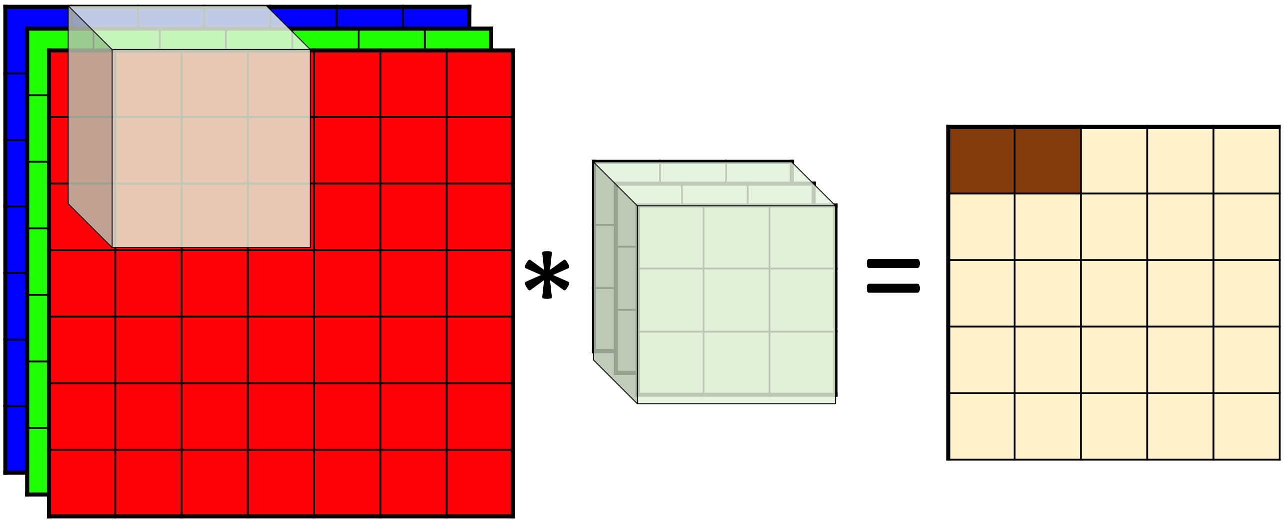
Convolutions On Volumes



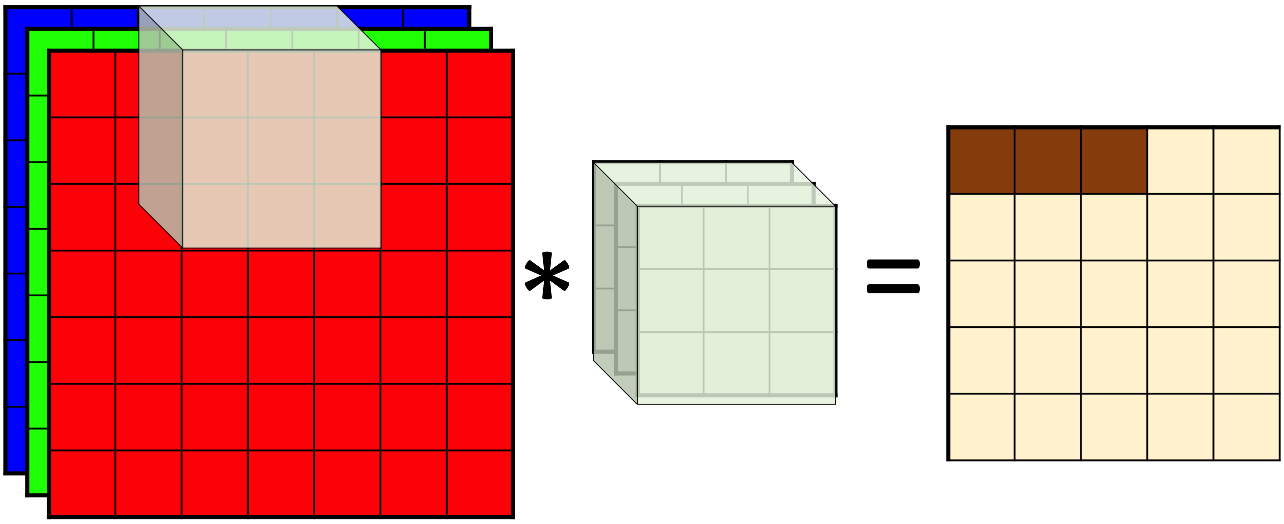
Convolutions On Volumes



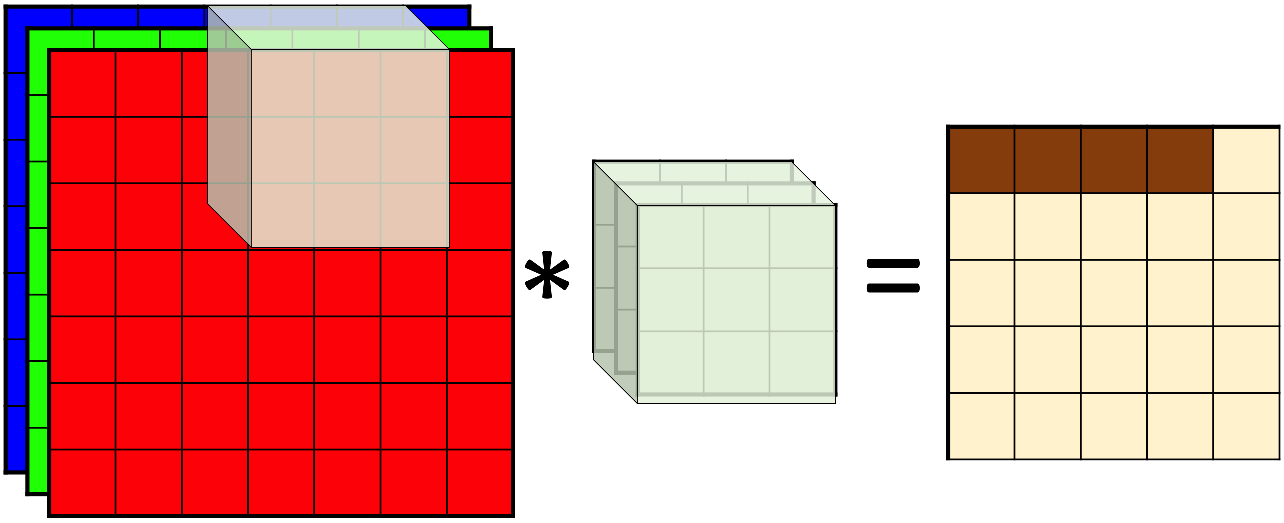
Convolutions On Volumes



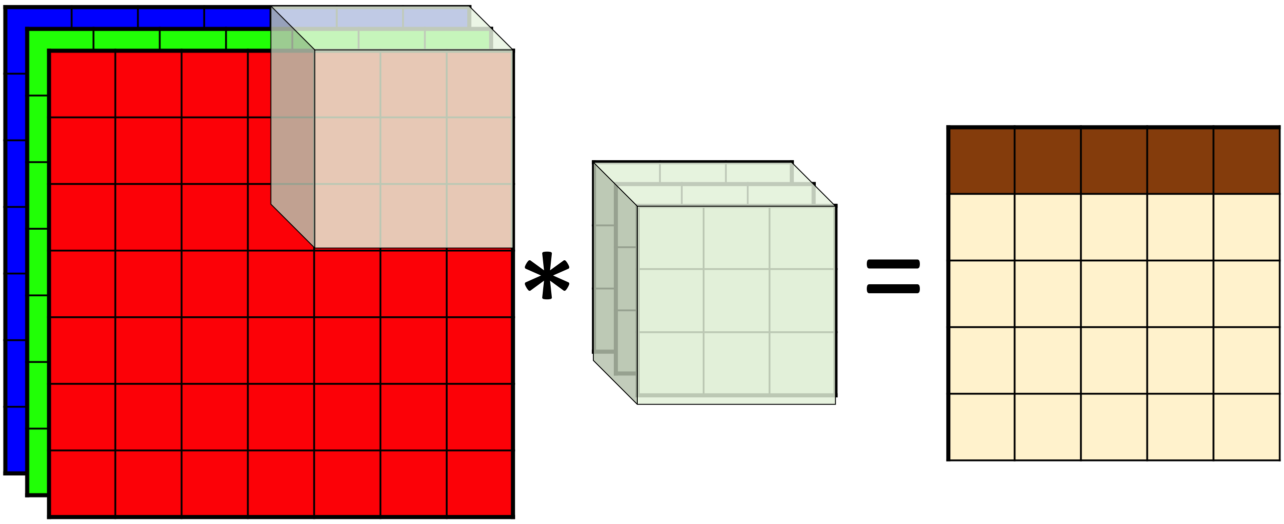
Convolutions On Volumes



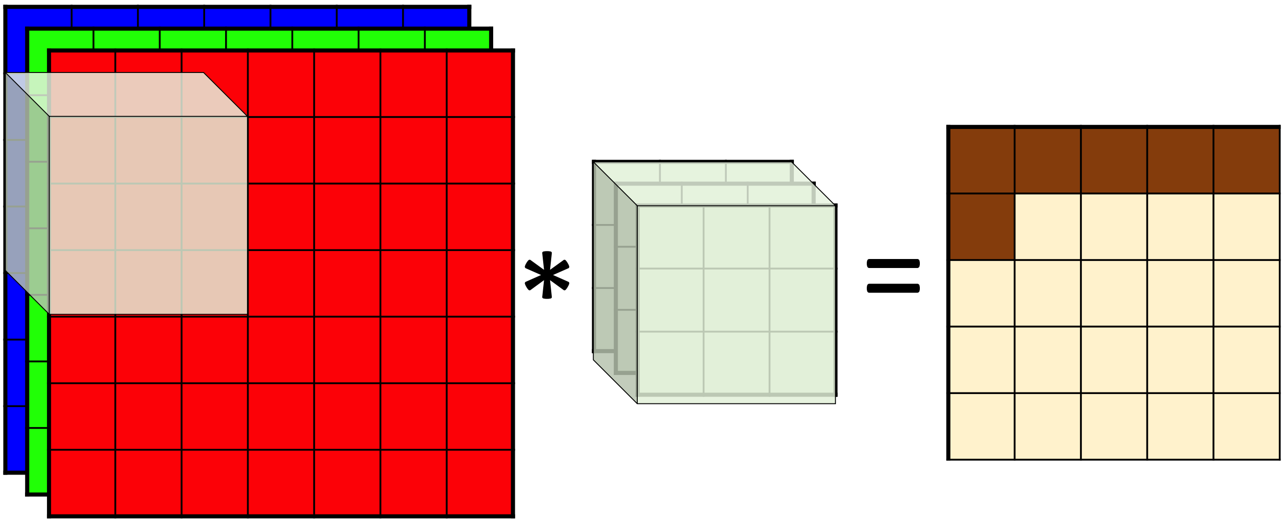
Convolutions On Volumes



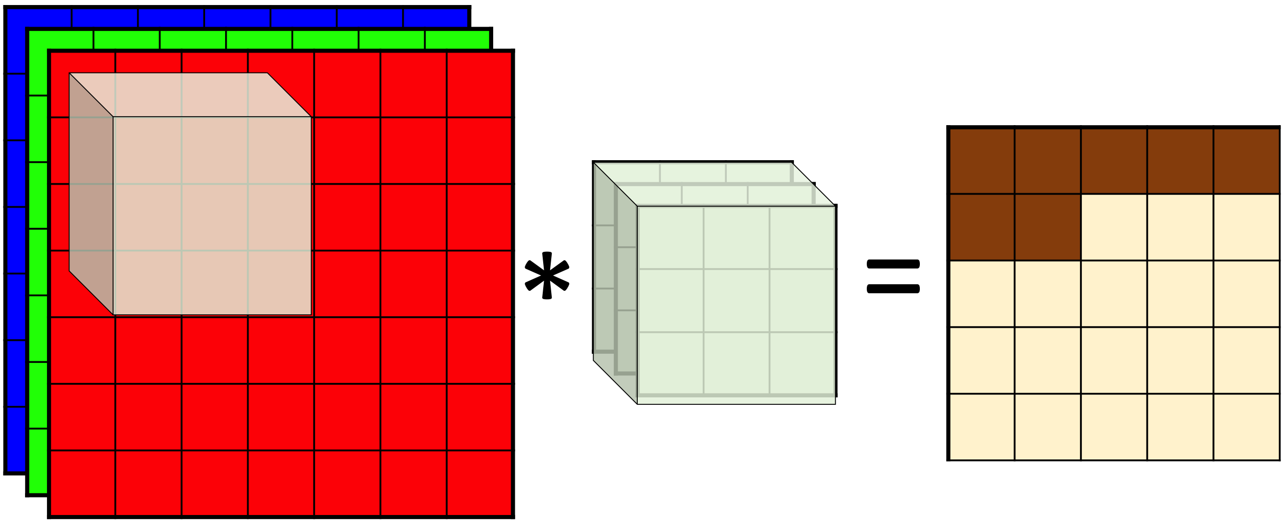
Convolutions On Volumes



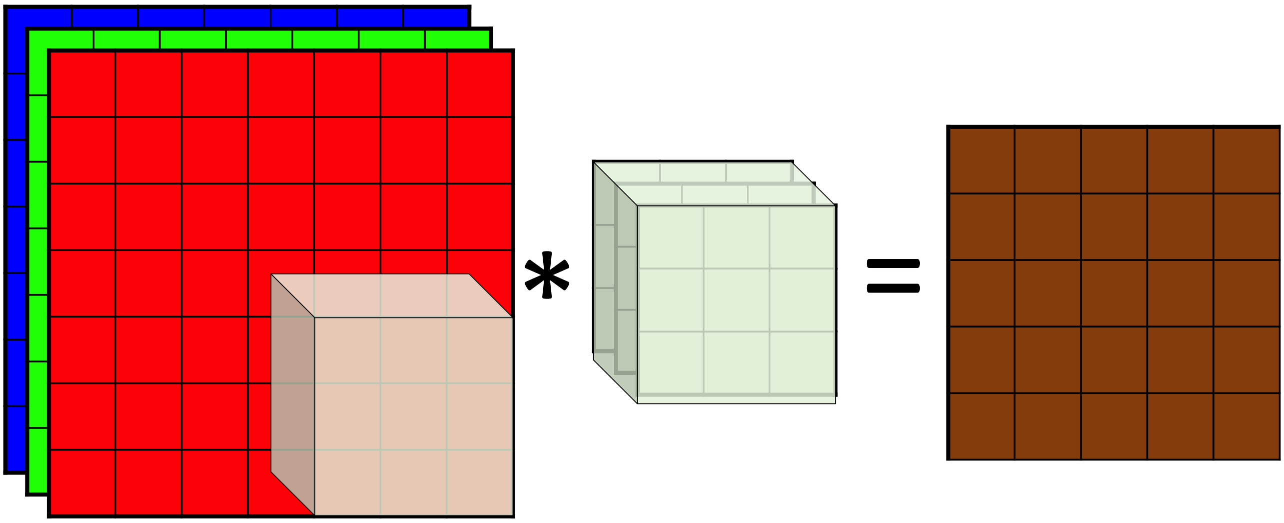
Convolutions On Volumes

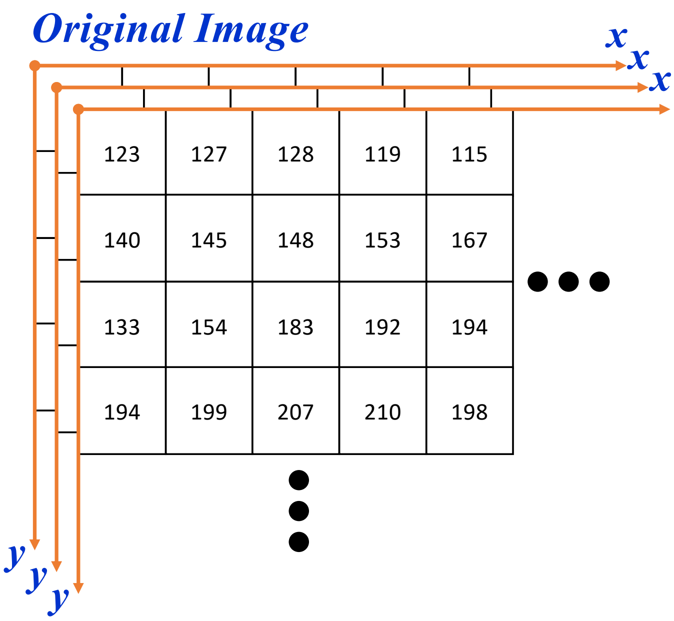


Convolutions On Volumes

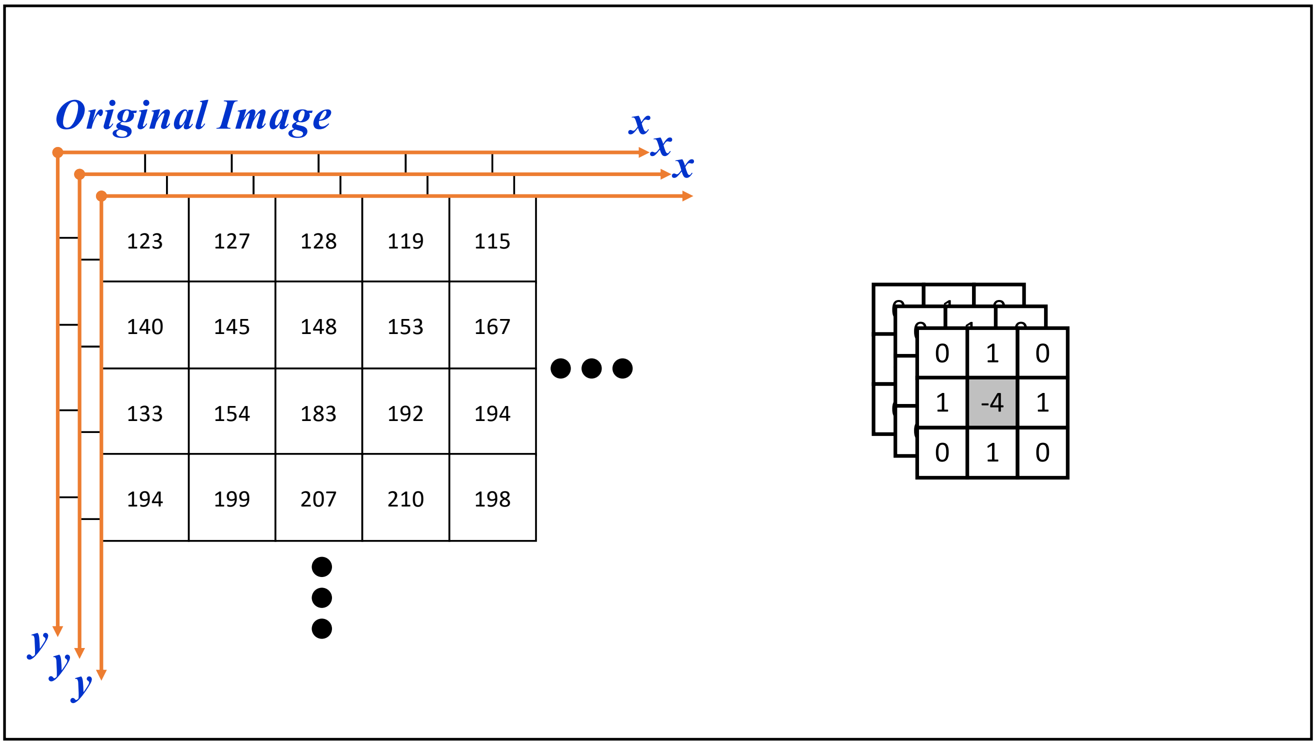


Convolutions On Volumes



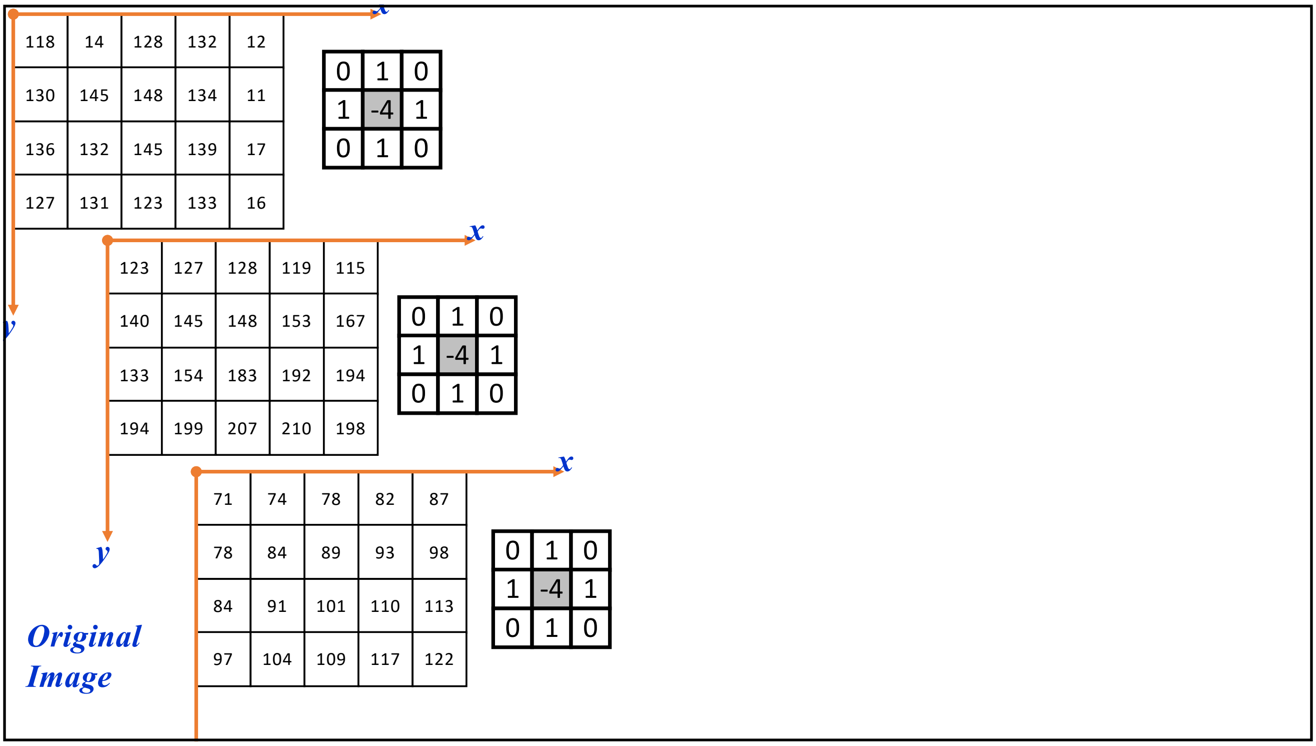


123	127	128	119	115
140	145	148	153	167
133	154	183	192	194
194	199	207	210	198



123	127	128	119	115
140	145	148	153	167
133	154	183	192	194
194	199	207	210	198

0	1	0
1	-4	1
0	1	0

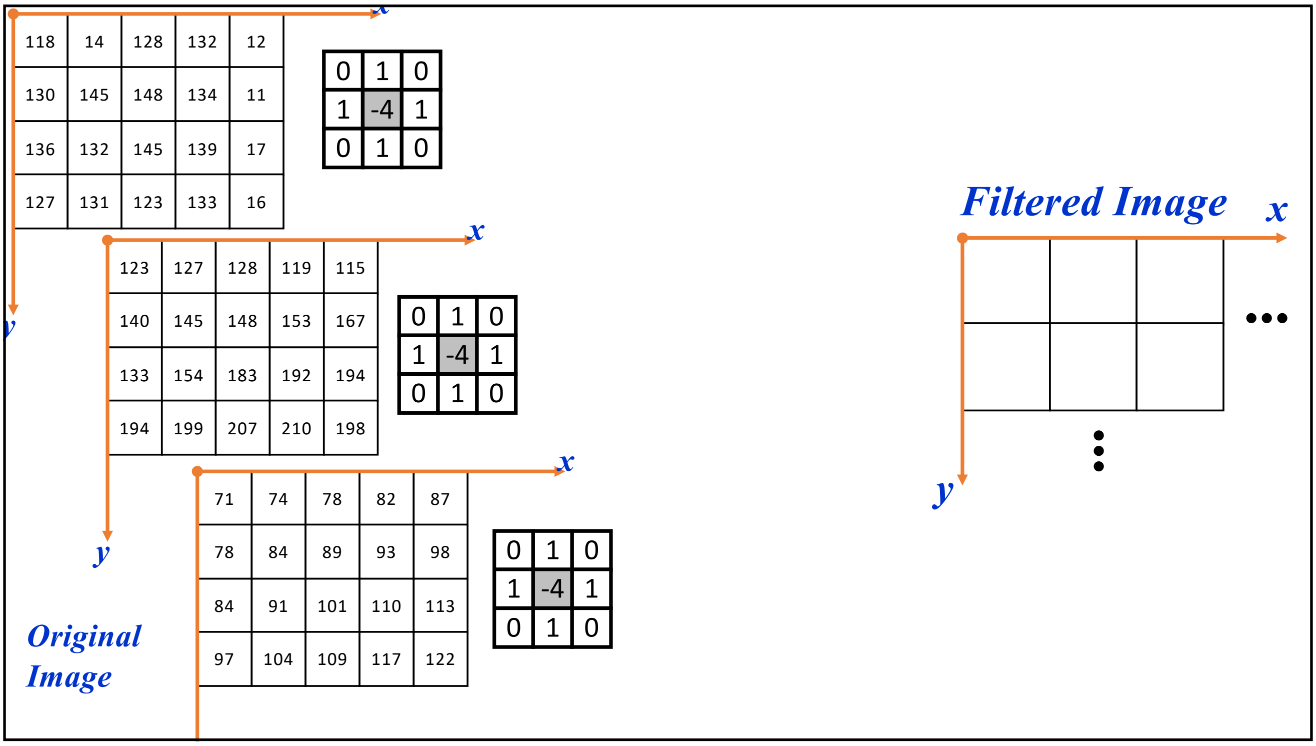


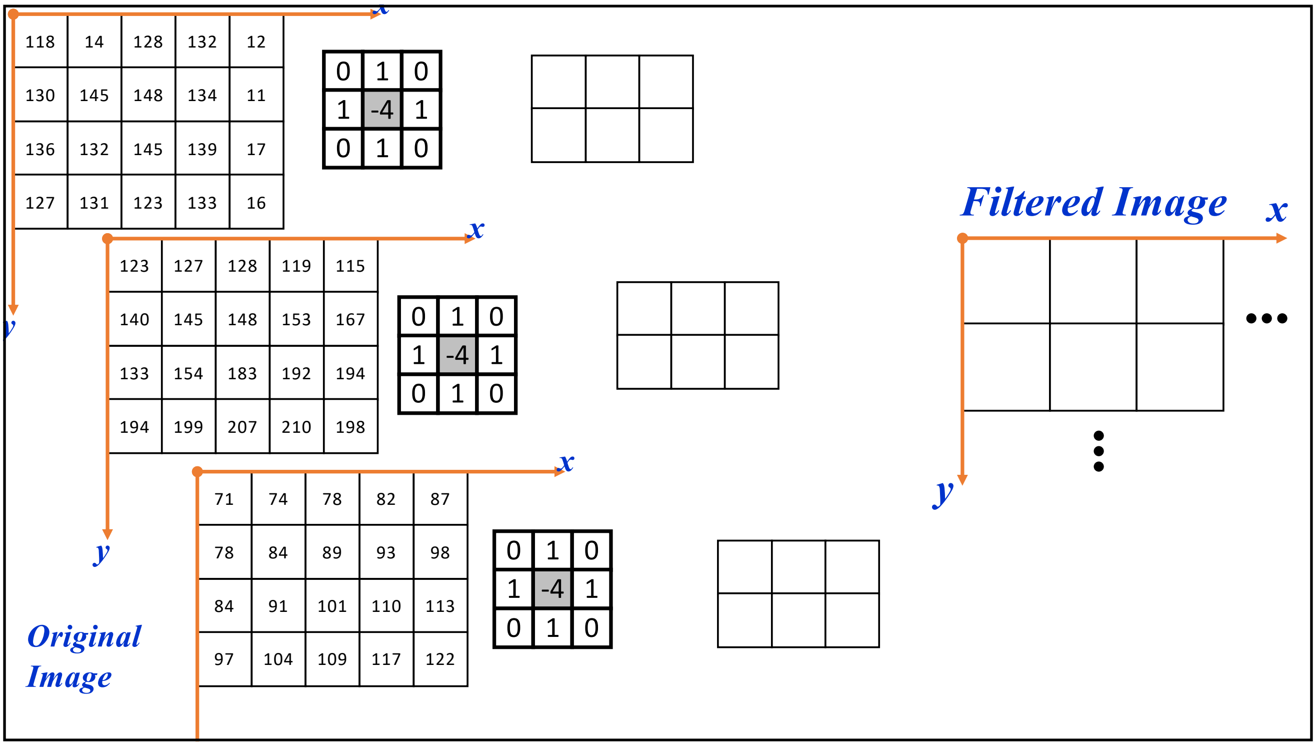
118	14	128	132	12
130	145	148	134	11
136	132	145	139	17
127	131	123	133	16

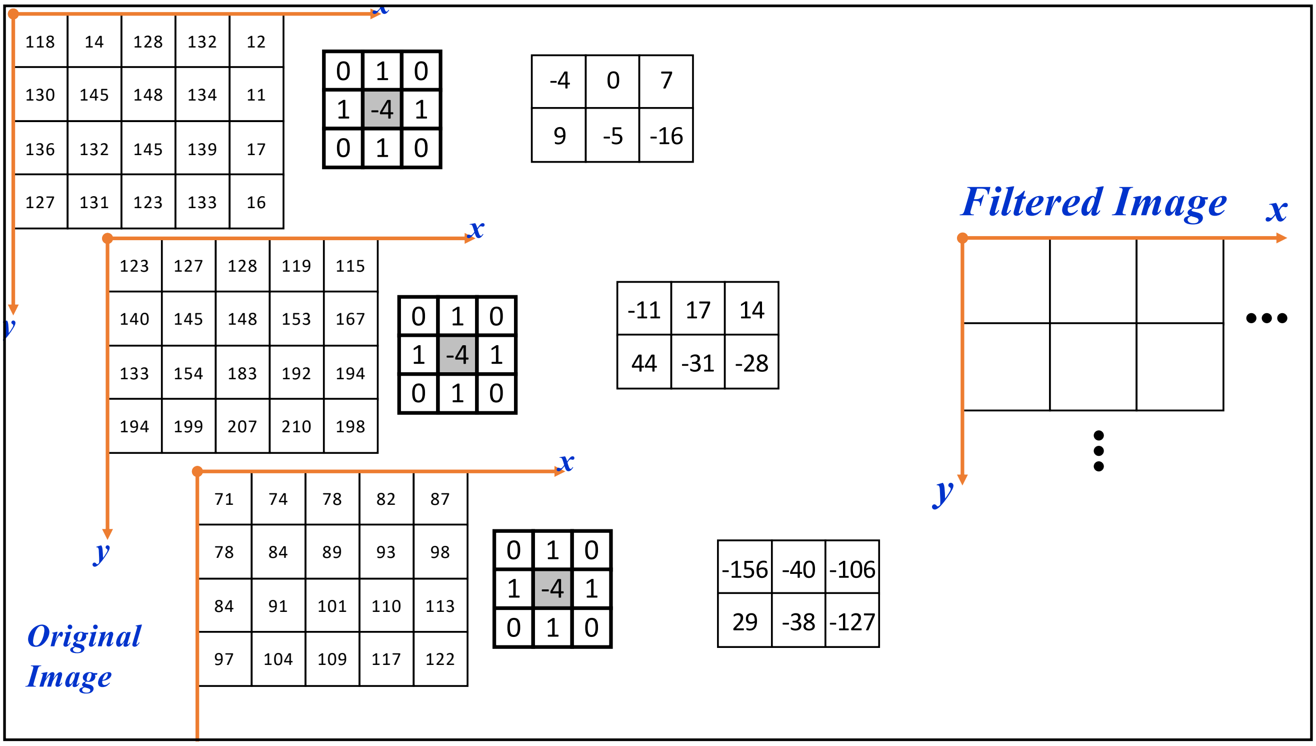
123	127	128	119	115
140	145	148	153	167
133	154	183	192	194
194	199	207	210	198

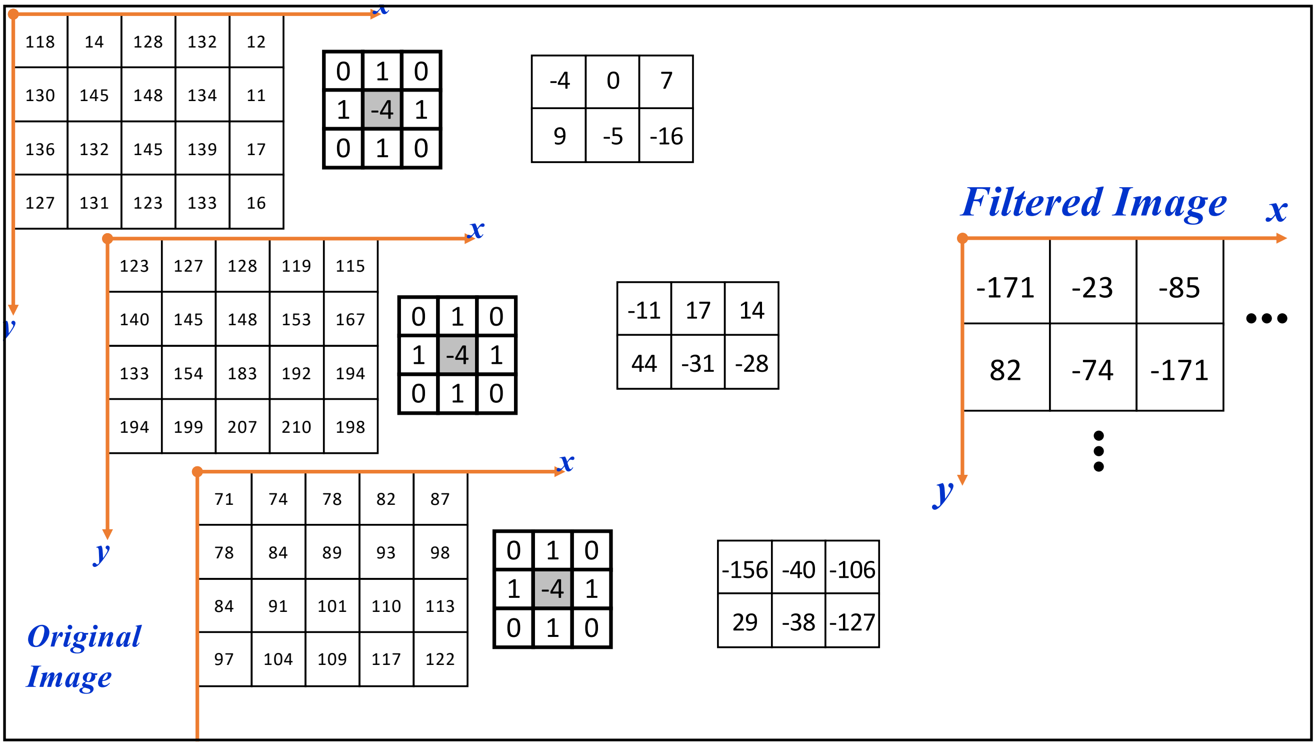
71	74	78	82	87
78	84	89	93	98
84	91	101	110	113
97	104	109	117	122

0	1	0
1	-4	1
0	1	0


118 14 128 132 12
130 145 148 134 11
136 132 145 139 17
127 131 123 133 16
123 127 128 119 115
140 145 148 153 167
133 154 183 192 194
194 199 207 210 198
71 74 78 82 87
78 84 89 93 98
84 91 101 110 113
97 104 109 117 122
0 1 0
1 -4 1
0 1 0
x
y
Original Image
Filtered Image

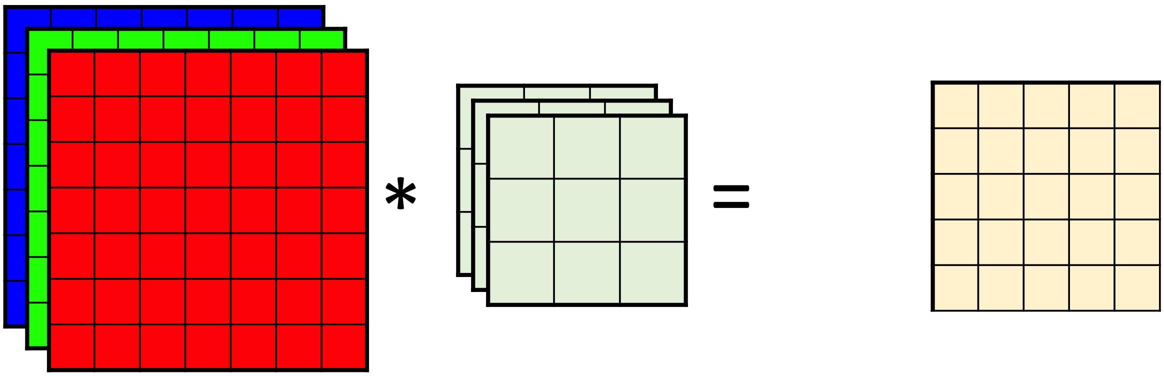

118 14 128 132 12
130 145 148 134 11
136 132 145 139 17
127 131 123 133 16
123 127 128 119 115
140 145 148 153 167
133 154 183 192 194
194 199 207 210 198
71 74 78 82 87
78 84 89 93 98
84 91 101 110 113
97 104 109 117 122
0 1 0
1 -4 1
0 1 0
x
y
Original Image
Filtered Image


118 14 128 132 12
130 145 148 134 11
136 132 145 139 17
127 131 123 133 16
123 127 128 119 115
140 145 148 153 167
133 154 183 192 194
194 199 207 210 198
71 74 78 82 87
78 84 89 93 98
84 91 101 110 113
97 104 109 117 122
0 1 0
1 -4 1
0 1 0
-4 0 7
9 -5 -16
-11 17 14
44 -31 -28
-156 -40 -106
29 -38 -127
x
y
Original Image
Filtered Image

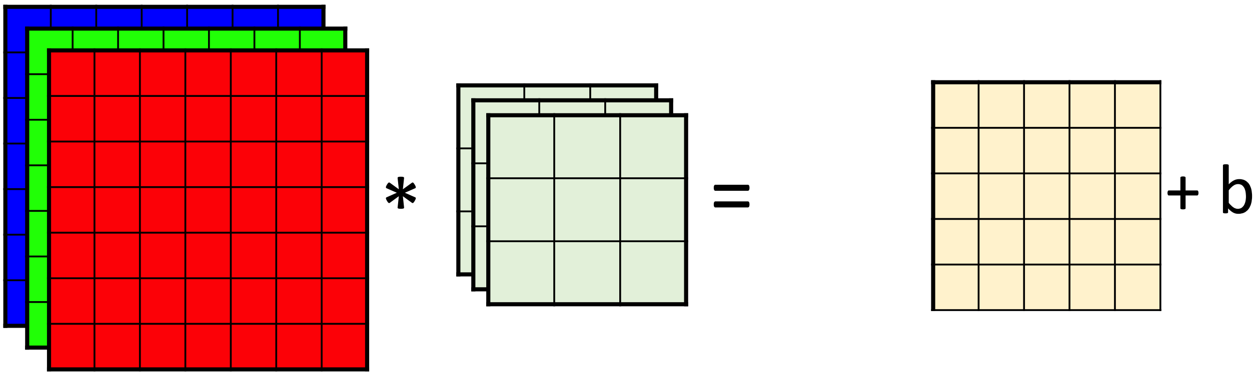

118 14 128 132 12
130 145 148 134 11
136 132 145 139 17
127 131 123 133 16
123 127 128 119 115
140 145 148 153 167
133 154 183 192 194
194 199 207 210 198
71 74 78 82 87
78 84 89 93 98
84 91 101 110 113
97 104 109 117 122
0 1 0
1 -4 1
0 1 0
-4 0 7
9 -5 -16
-11 17 14
44 -31 -28
-156 -40 -106
29 -38 -127
-171 -23 -85
82 -74 -171
x
y
Original Image
Filtered Image

CNN LAYER

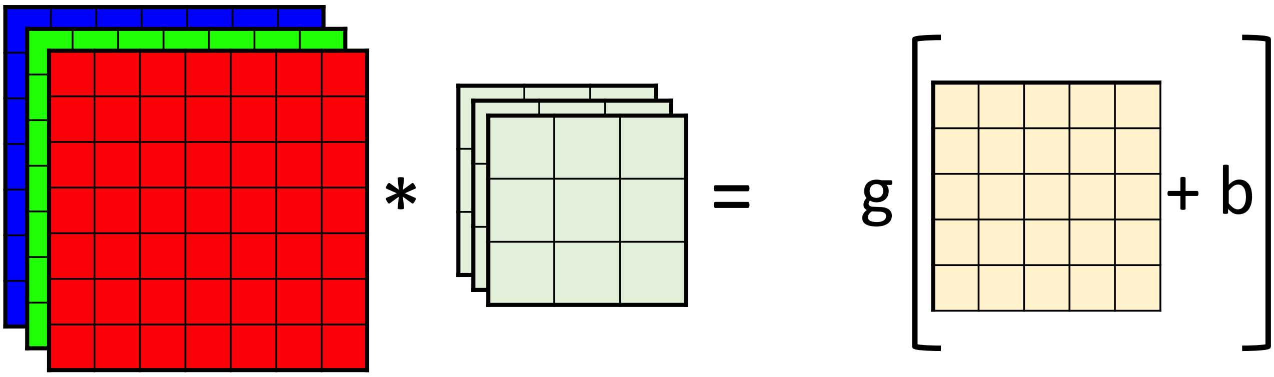
CNN Layer



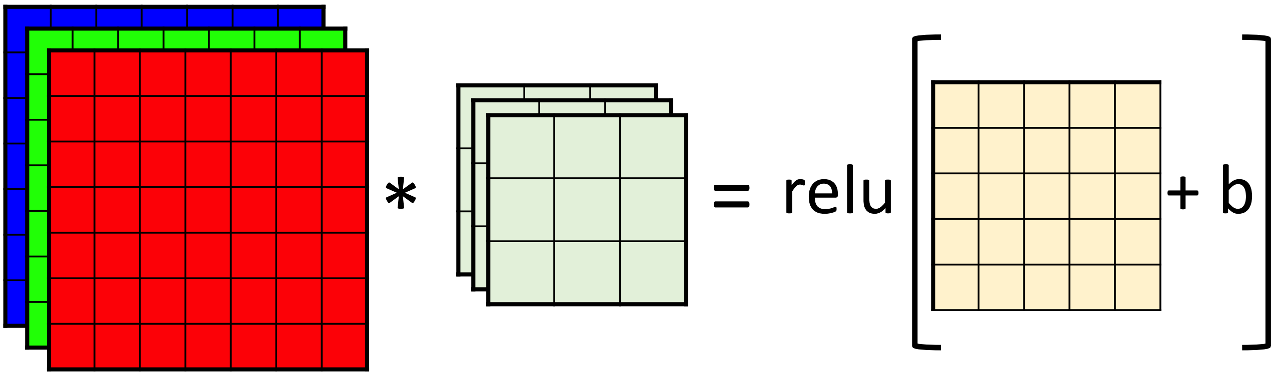
CNN Layer



CNN Layer

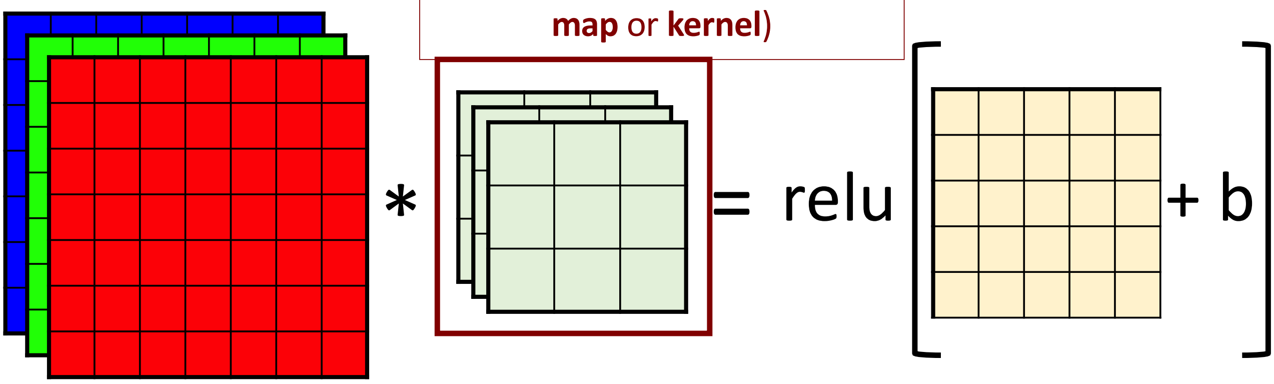


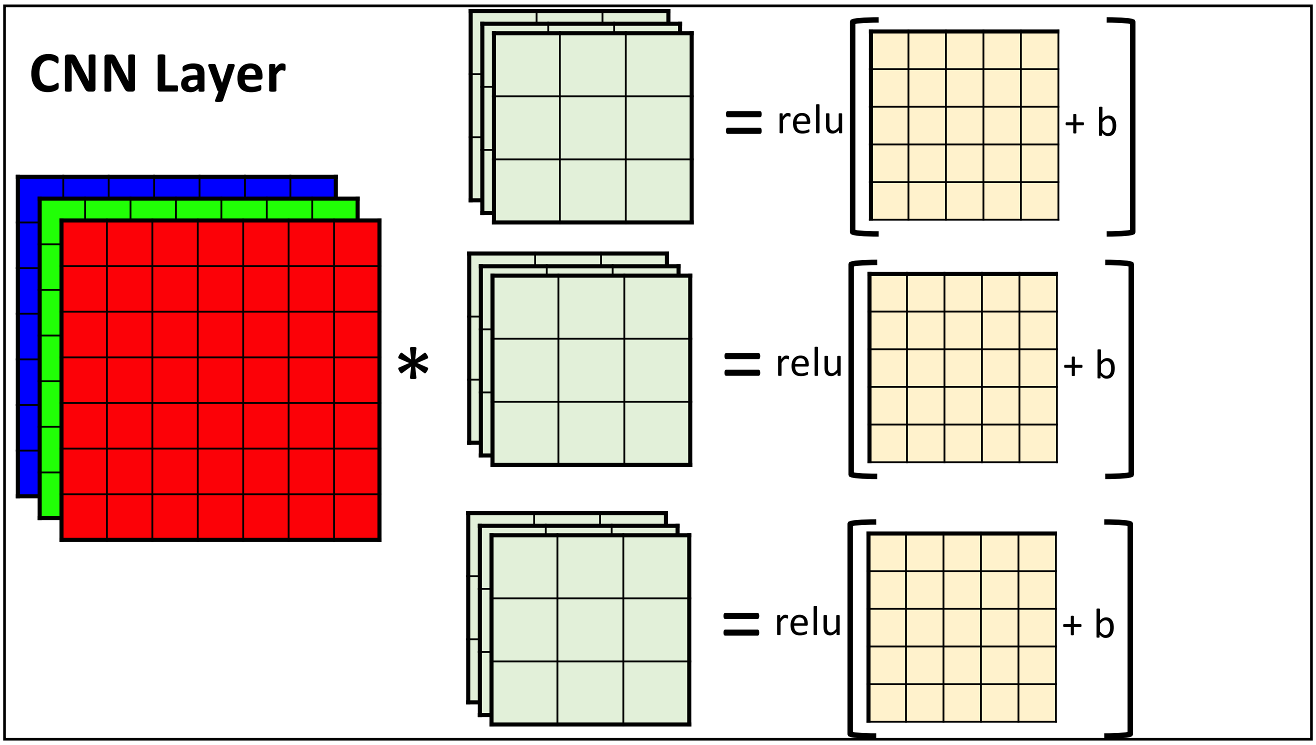
CNN Layer

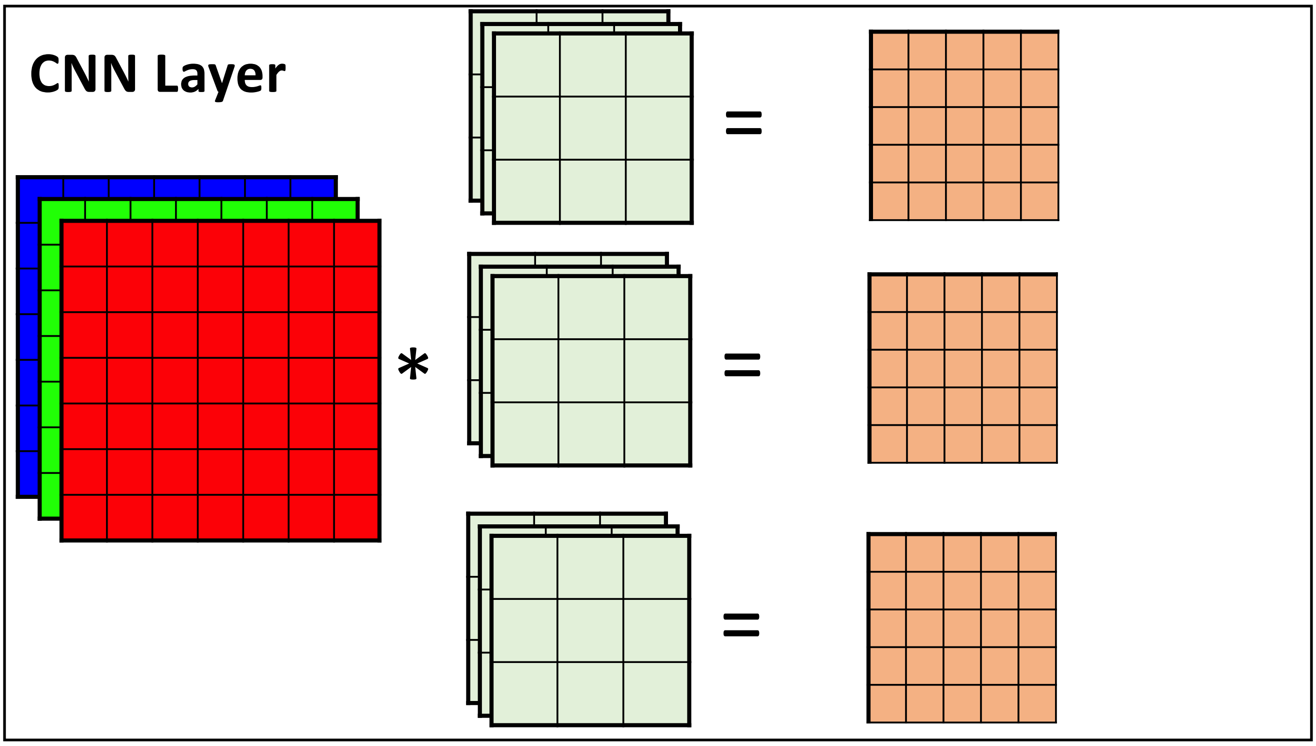


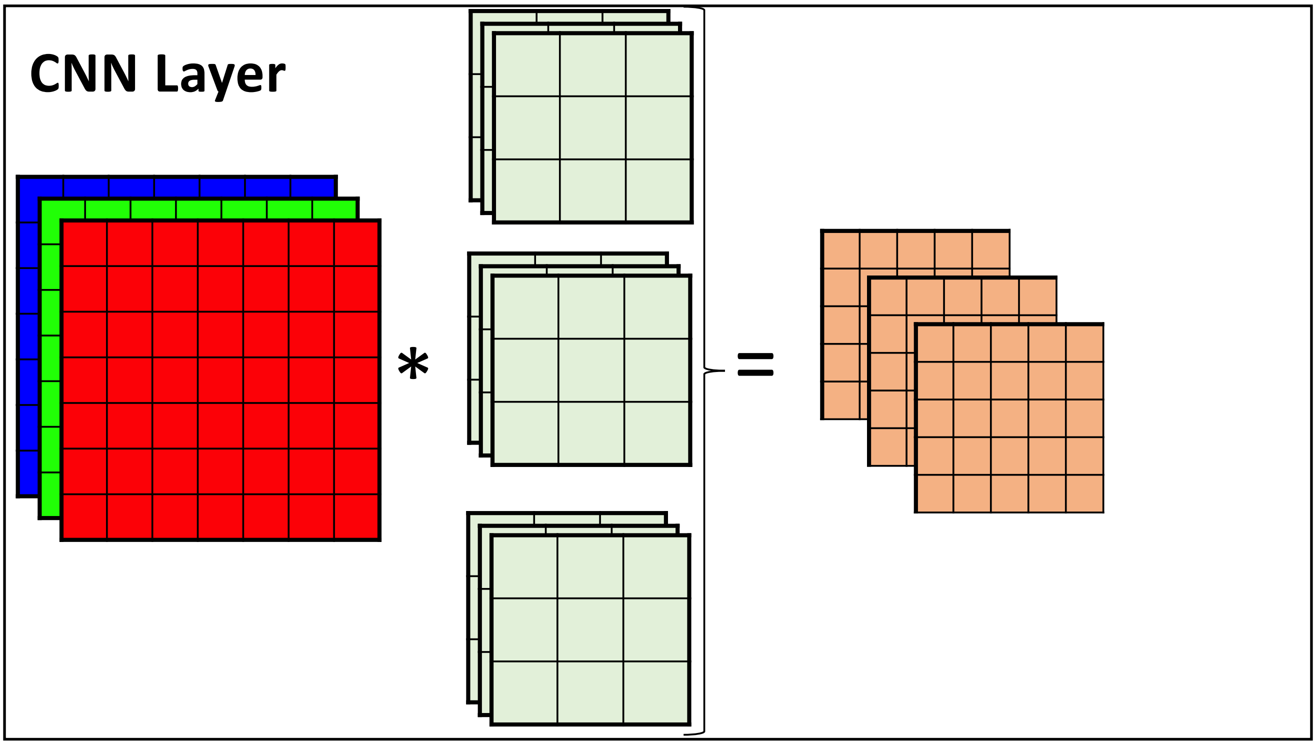
CNN Layer

In our CNN we are going to learn the weights of this **filter** (also sometimes called a **filter map** or **kernel**)

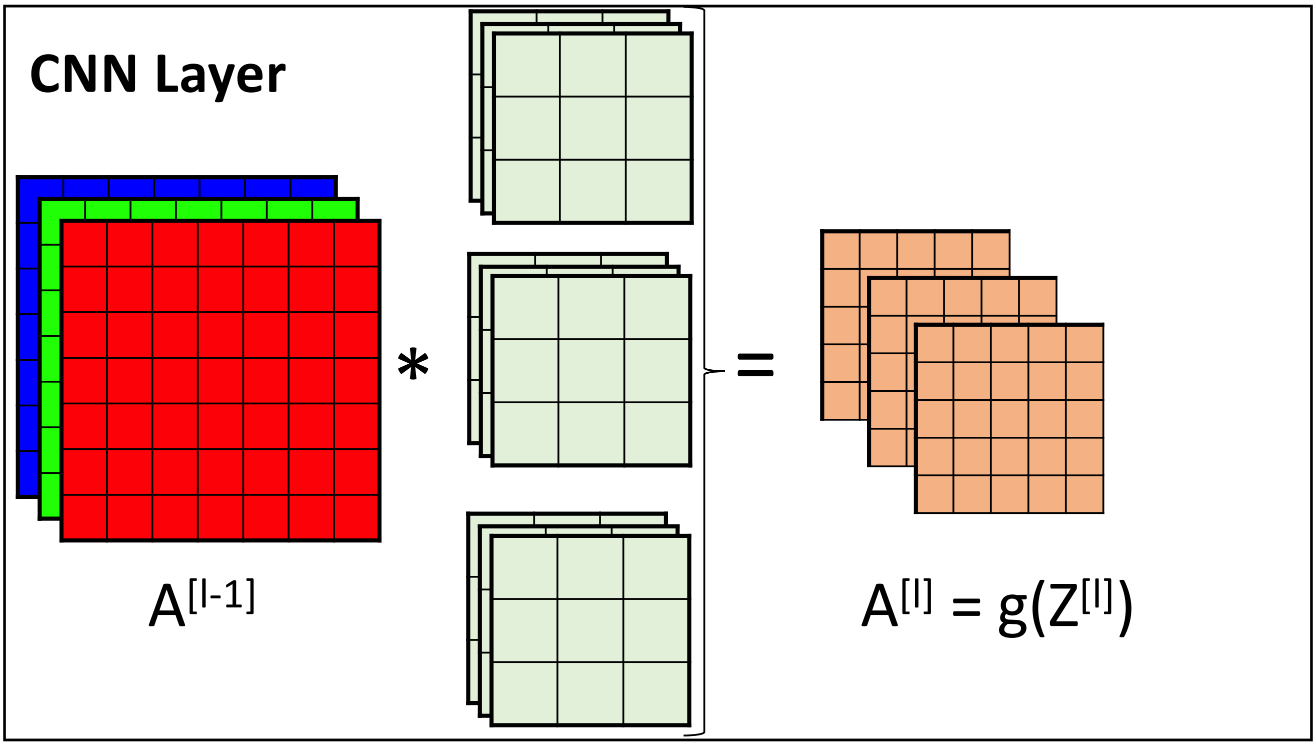



CNN Layer
*
= relu
+ b
= relu
+ b
= relu
+ b


CNN Layer
*
=
=
=



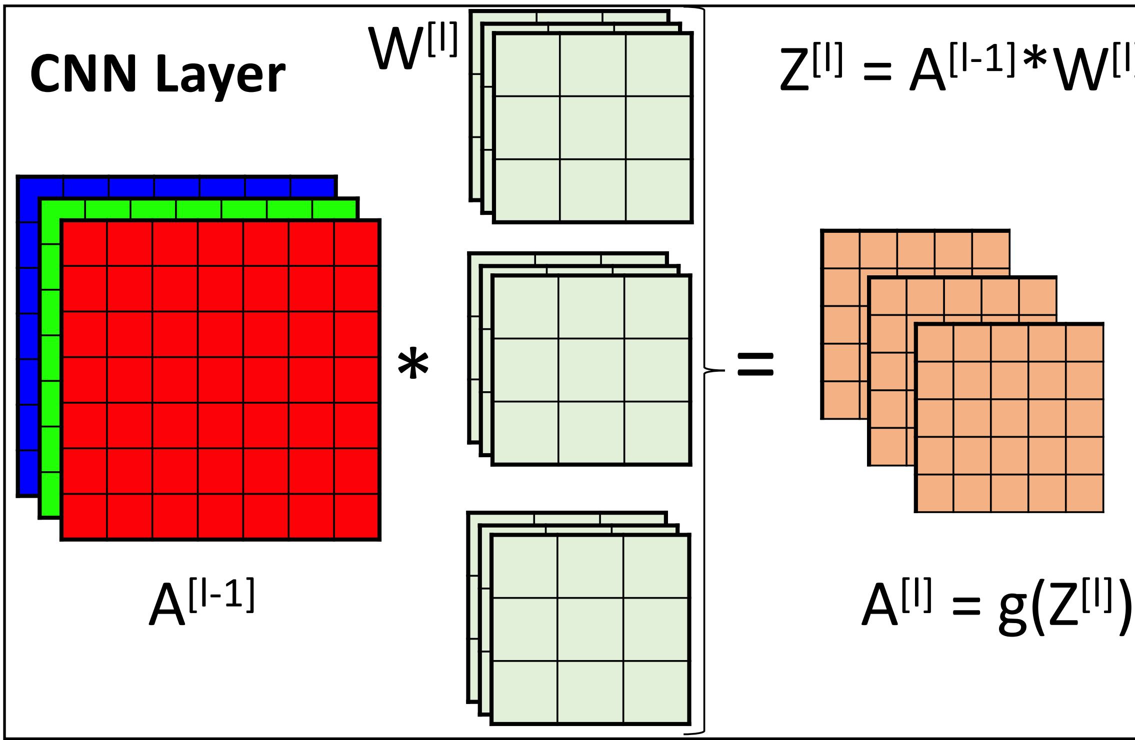
CNN Layer
*
=



CNN Layer
*
=
$A^{[l-1]}$
$A^{[l]} = g(Z^{[l]})$

CNN Layer

$$Z^{[l]} = A^{[l-1]} * W^{[l]} + B^{[l]}$$



POOLING

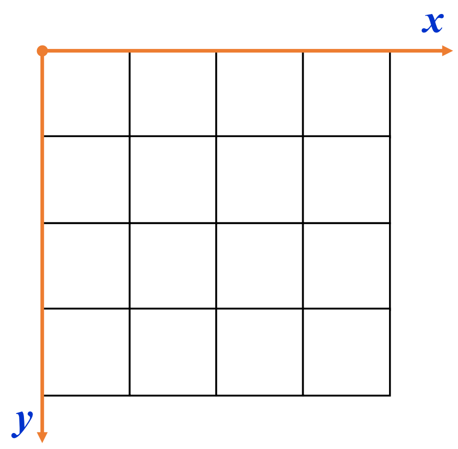
Pooling Layer

Pooling is a simple dimensionality reduction technique that is almost always used in combination with convolutional layers in CNNS

Max pooling is by far the most common type of pooling, but others like **average pooling** are sometimes used

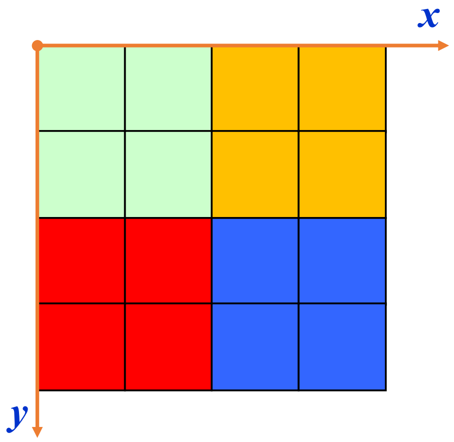
A pooling layer is defined by a **size** and a **stride**

Pooling

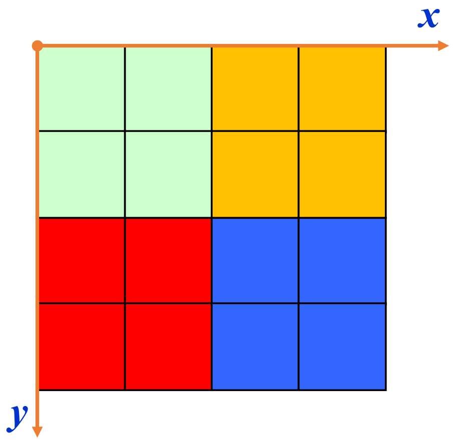


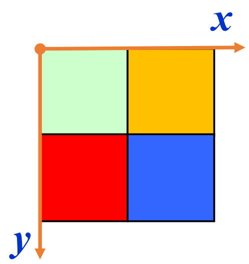
Let's apply max pooling with size 2 and stride 2

Pooling



Pooling





Pooling

Let's apply max pooling with size 2 and stride 2

123	127	128	119
140	145	148	153
133	154	183	192
194	199	207	210

Pooling

Let's apply max pooling with size 2 and stride 2

123	127	128	119
140	145	148	153
133	154	183	192
194	199	207	210

145	153
199	210

Pooling

Let's apply max pooling with size 2 and stride 1

123	127	128	119
140	145	148	153
133	154	183	192
194	199	207	210

Image from Andrei Karpathy

Pooling

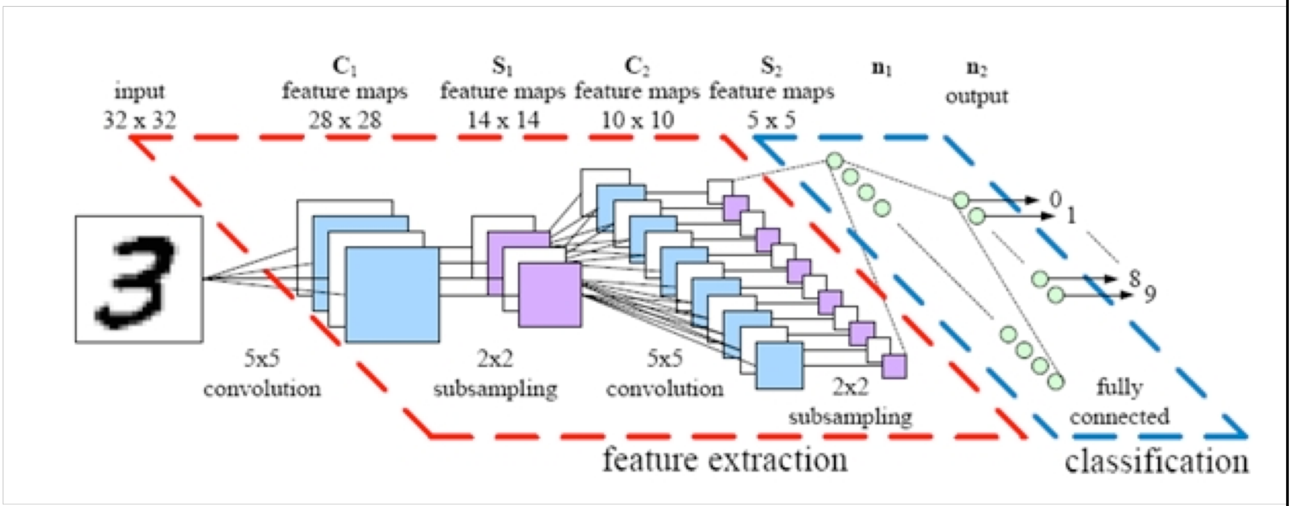
Let's apply max pooling with size 2 and stride 1

123	127	128	119
140	145	148	153
133	154	183	192
194	199	207	210

145	148	153
154	183	192
199	207	210

Image from Andrei Karpathy

CONVOLUTIONAL NEURAL NETWORK



Efficient mapping of the training of Convolutional Neural Networks to a CUDA-based cluster
Jonathan Ward, Sergey Andreev, Francisco Heredia, Bogdan Lazar, Zlatka Manevska
http://parse.ele.tue.nl/education/cluster2

Convolutional Neural Network

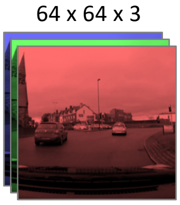
Although there are variants a convolutional neural network is typically composed of a number of pairs of convolutional layers followed by max pooling layers ultimately followed by a number of fully connected layers

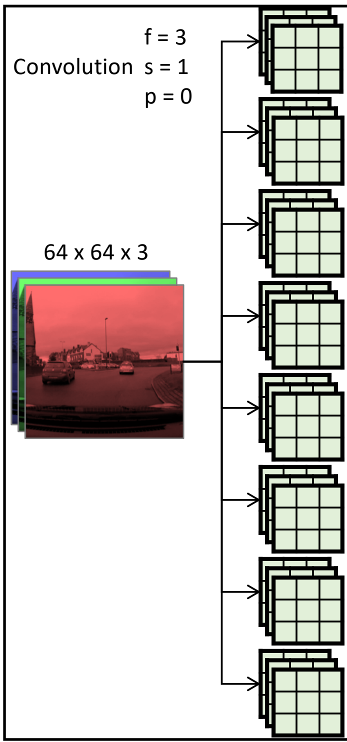
Convolutional Neural Network

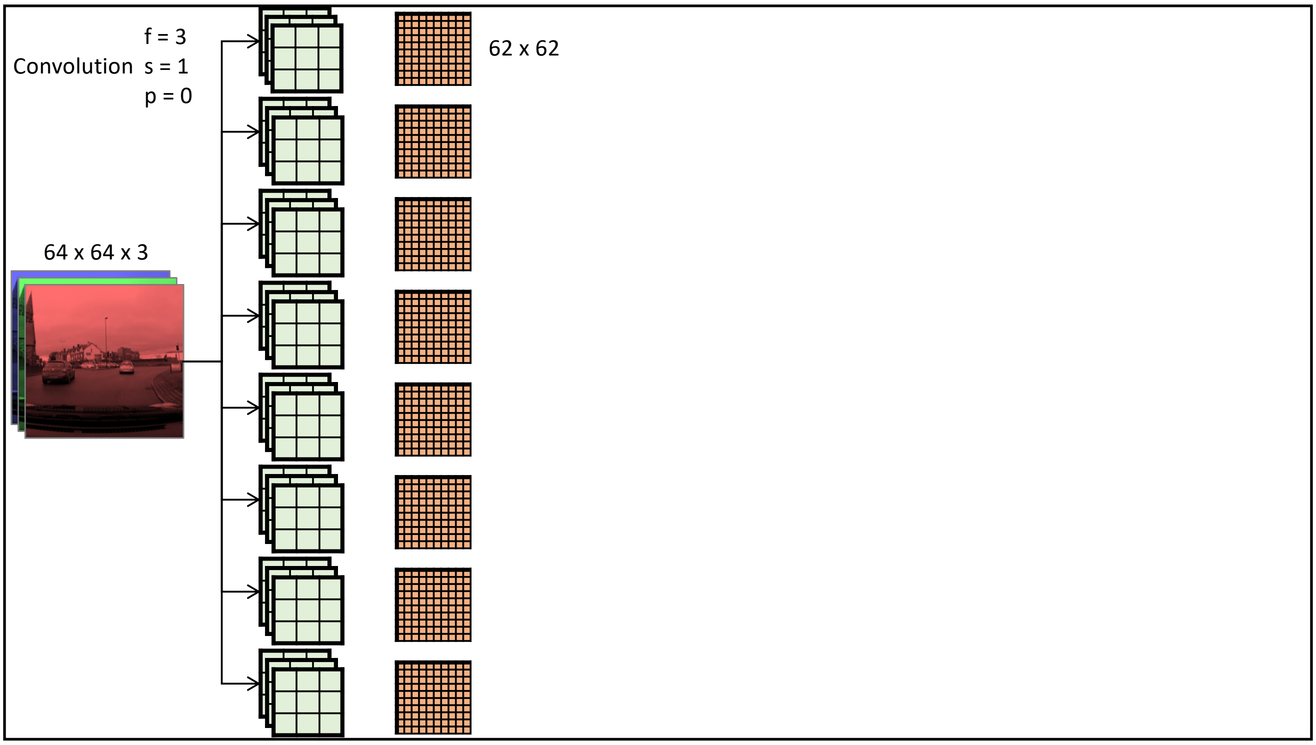
Let’s imagine a scenario where we will train a model to issue the control input for a self-driving car based on the view through the dashboard

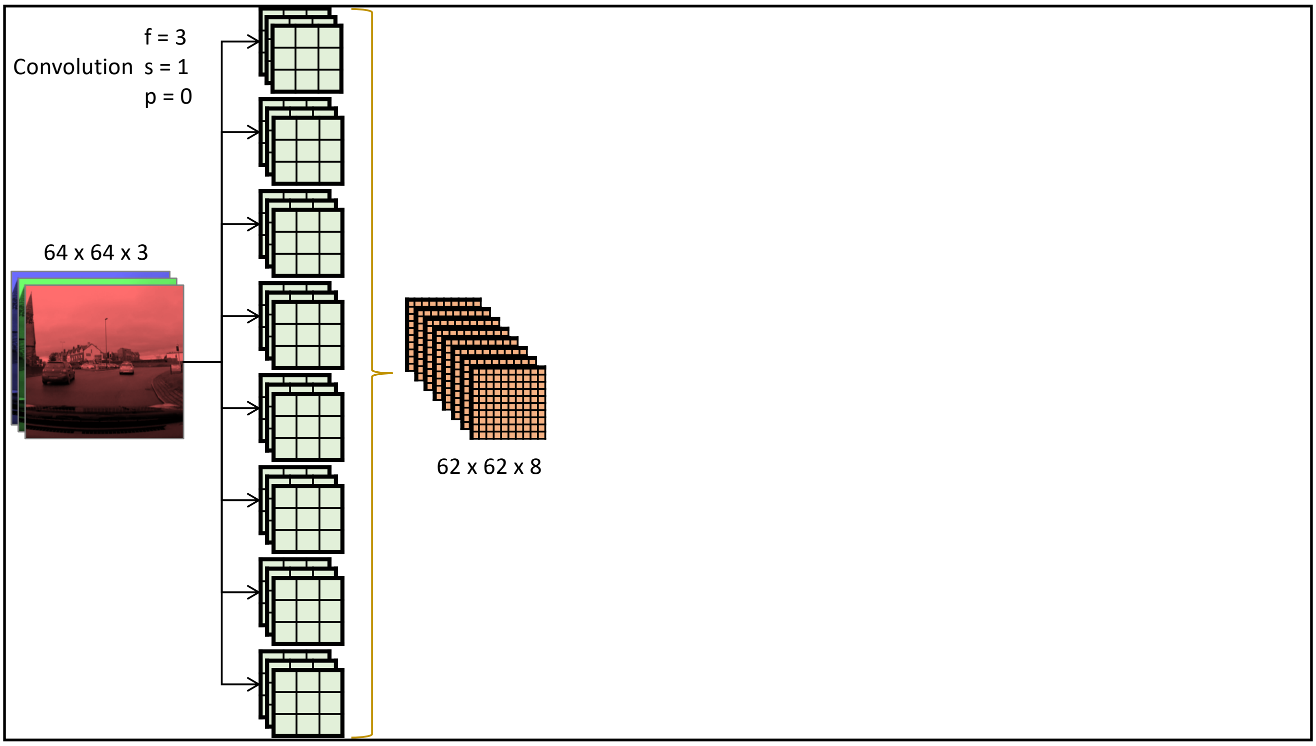
Let’s imagine four control outputs

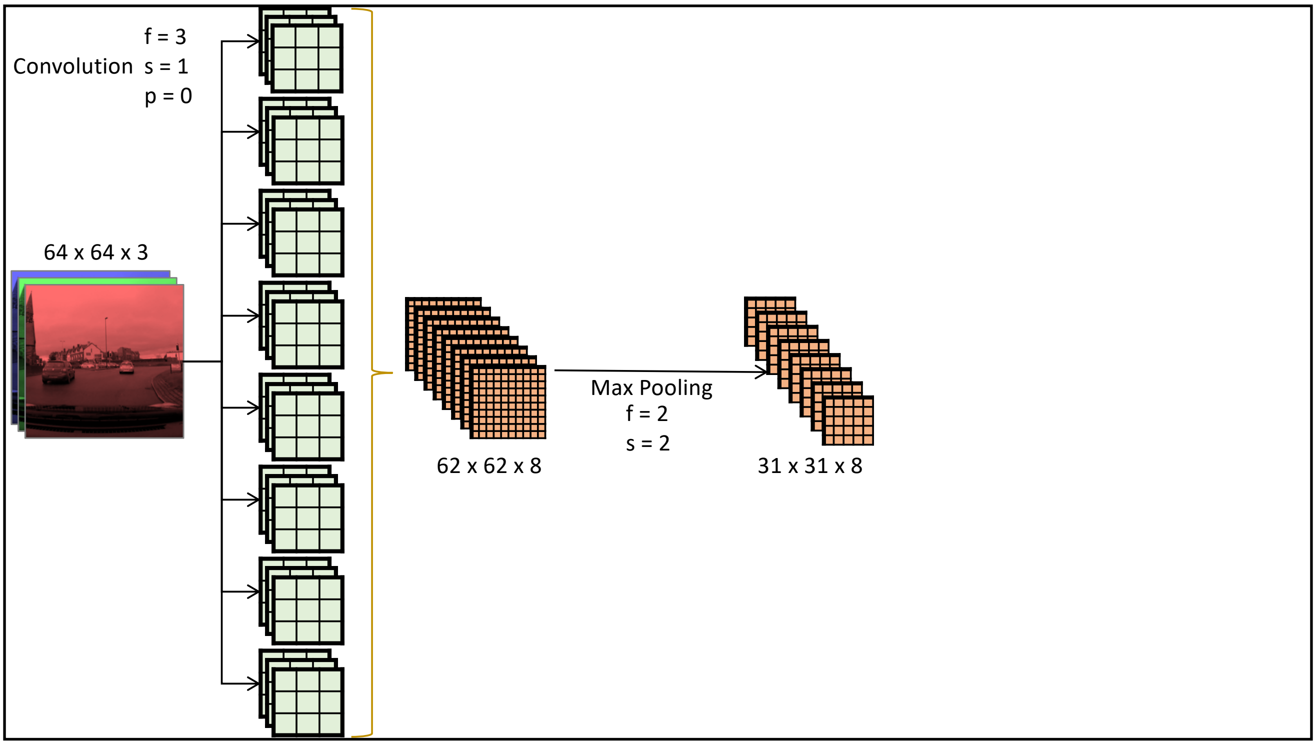
- Accelerate
- Brake
- Left
- Right

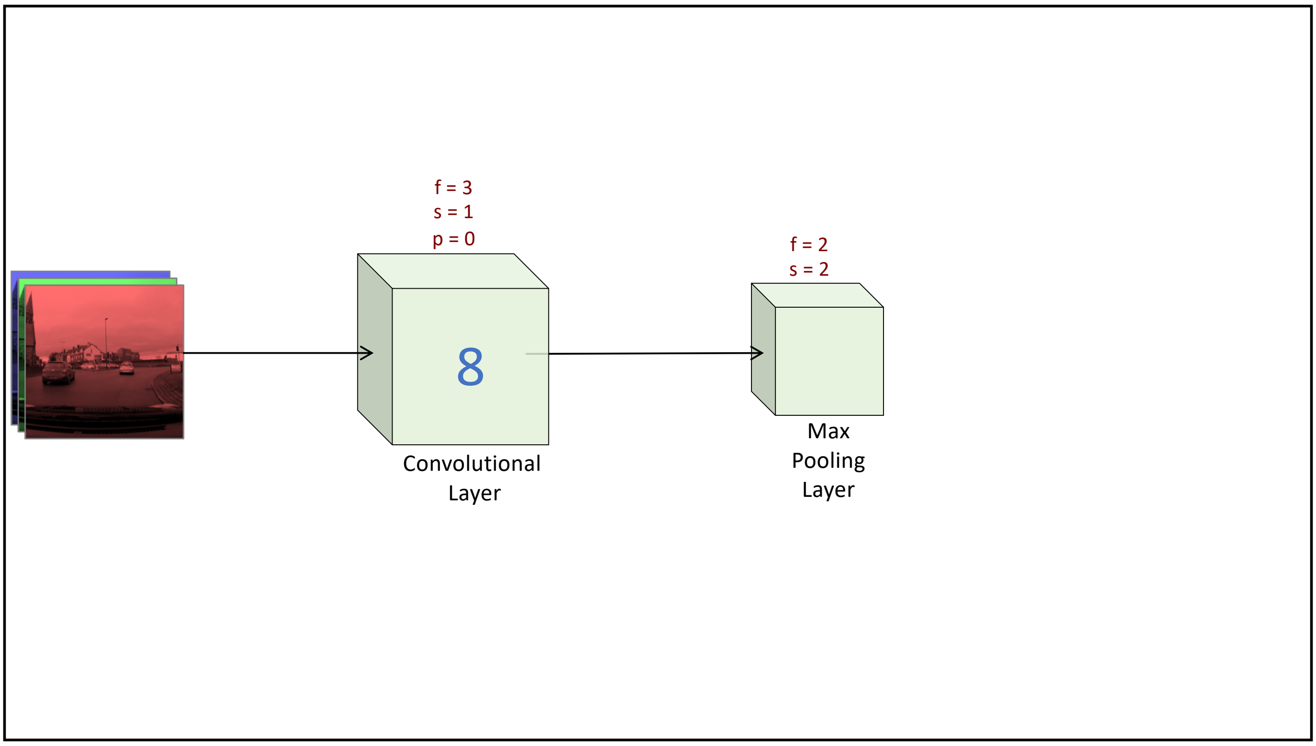

64 x 64 x 3

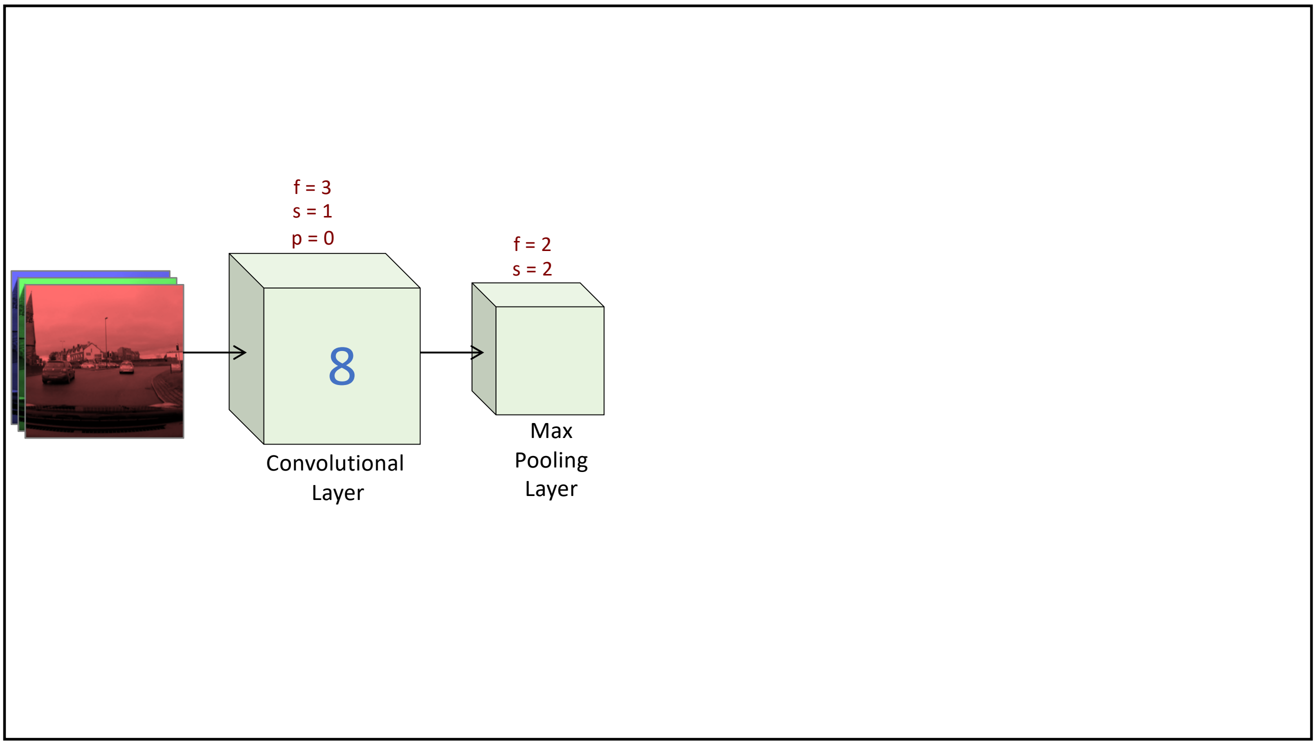

Convolution
f = 3
s = 1
p = 0
64 x 64 x 3

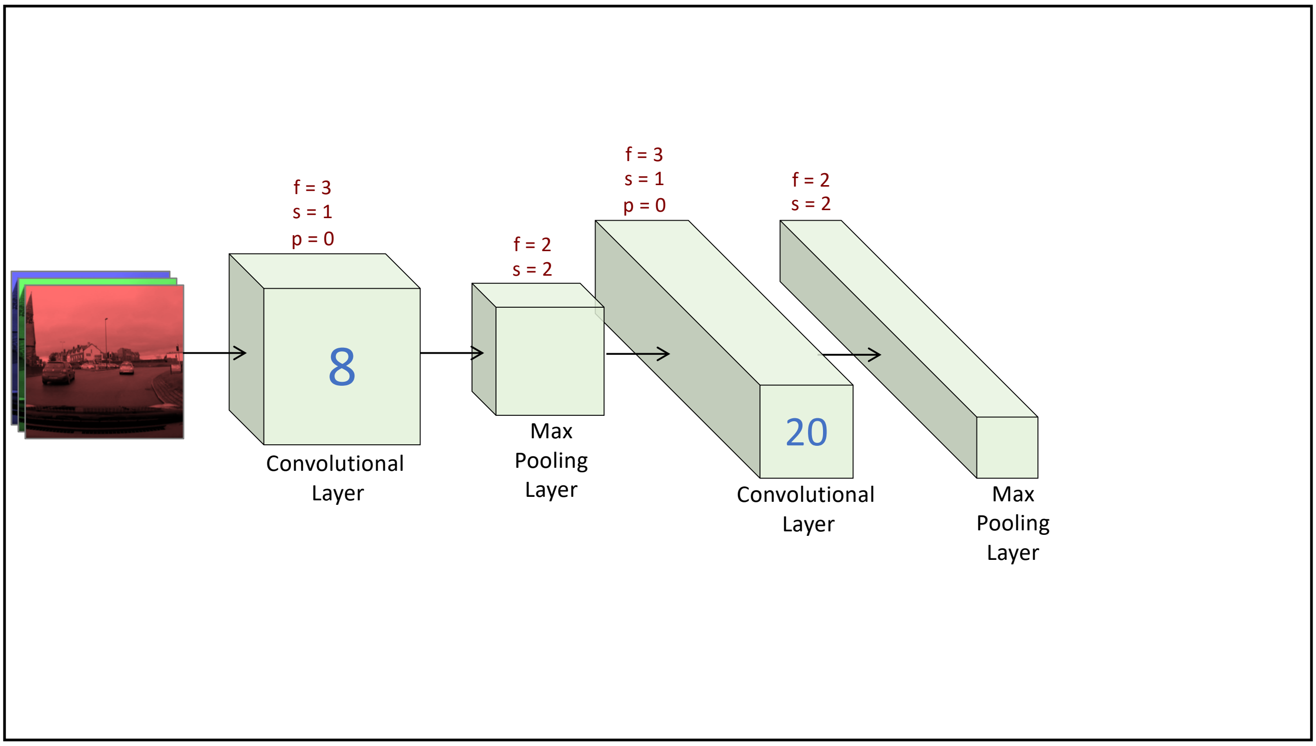

f = 3
Convolution s = 1
p = 0
64 x 64 x 3
62 x 62

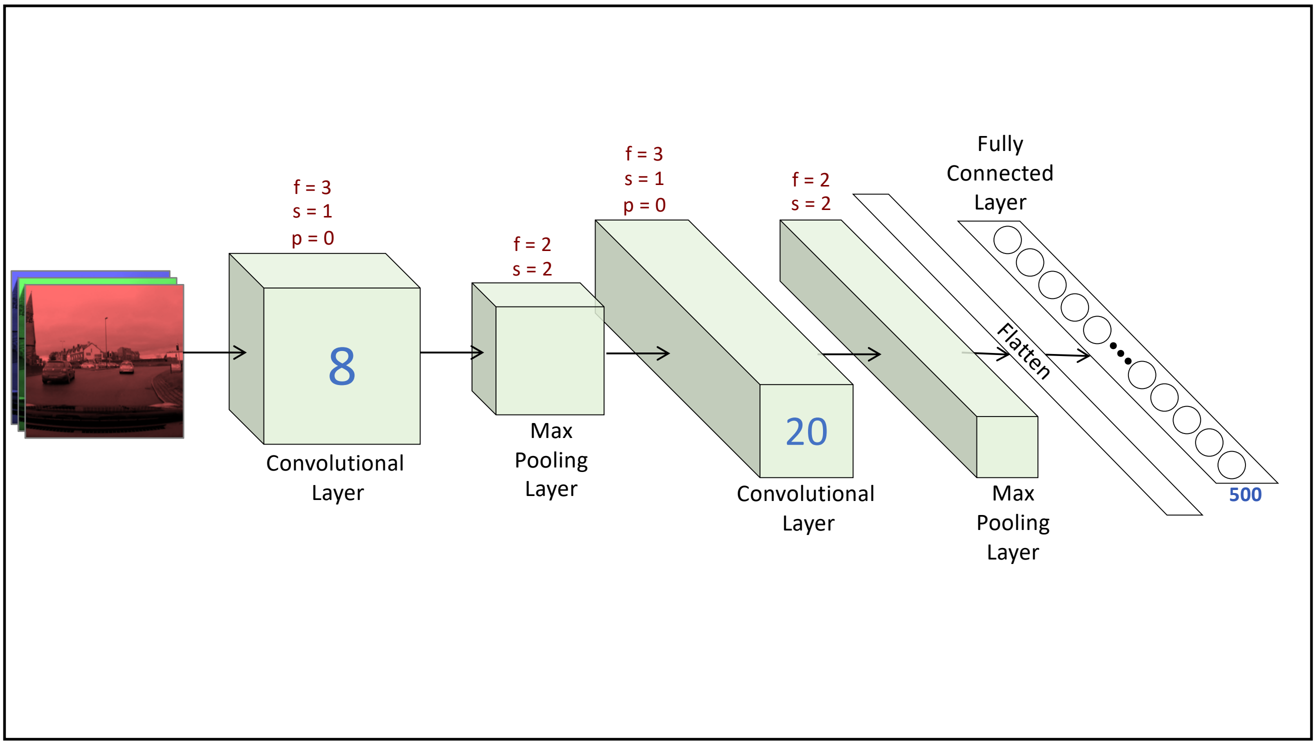

f = 3
Convolution s = 1
p = 0
64 x 64 x 3
62 x 62 x 8

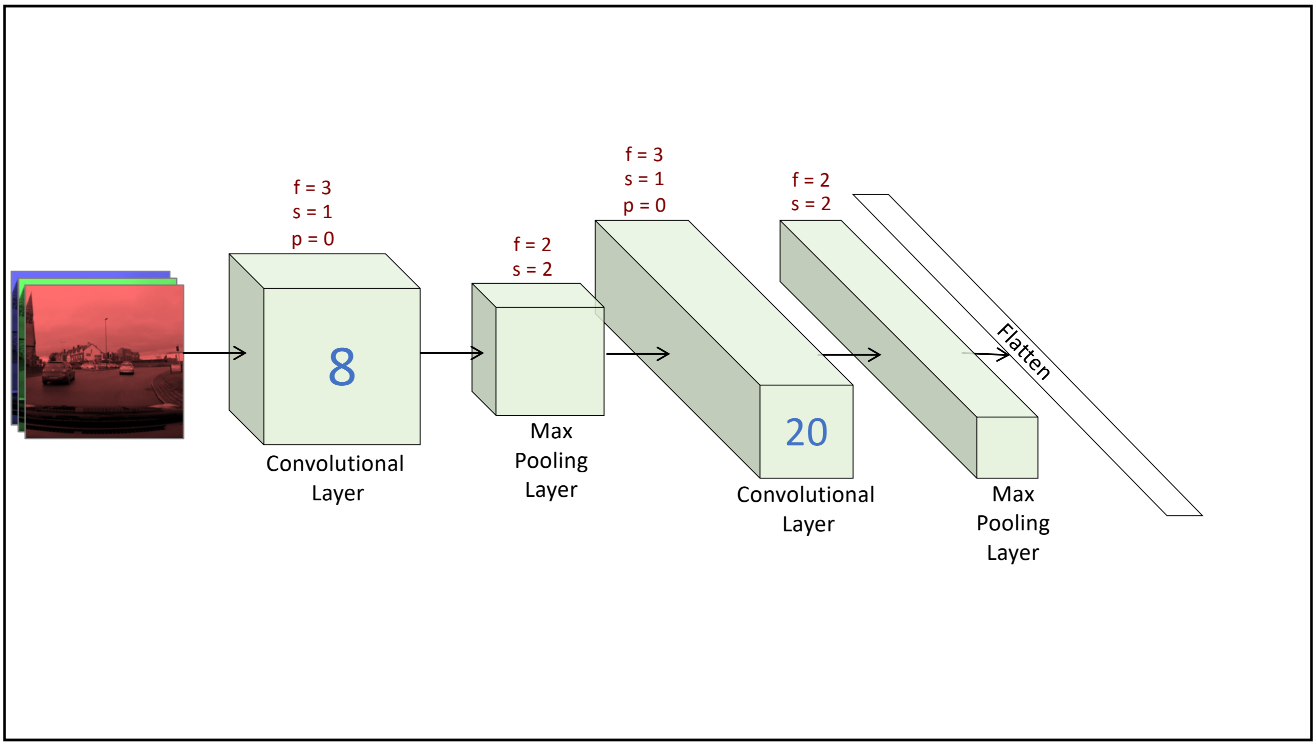

Convolution
f = 3
s = 1
p = 0
64 x 64 x 3
62 x 62 x 8
Max Pooling
f = 2
s = 2
31 x 31 x 8

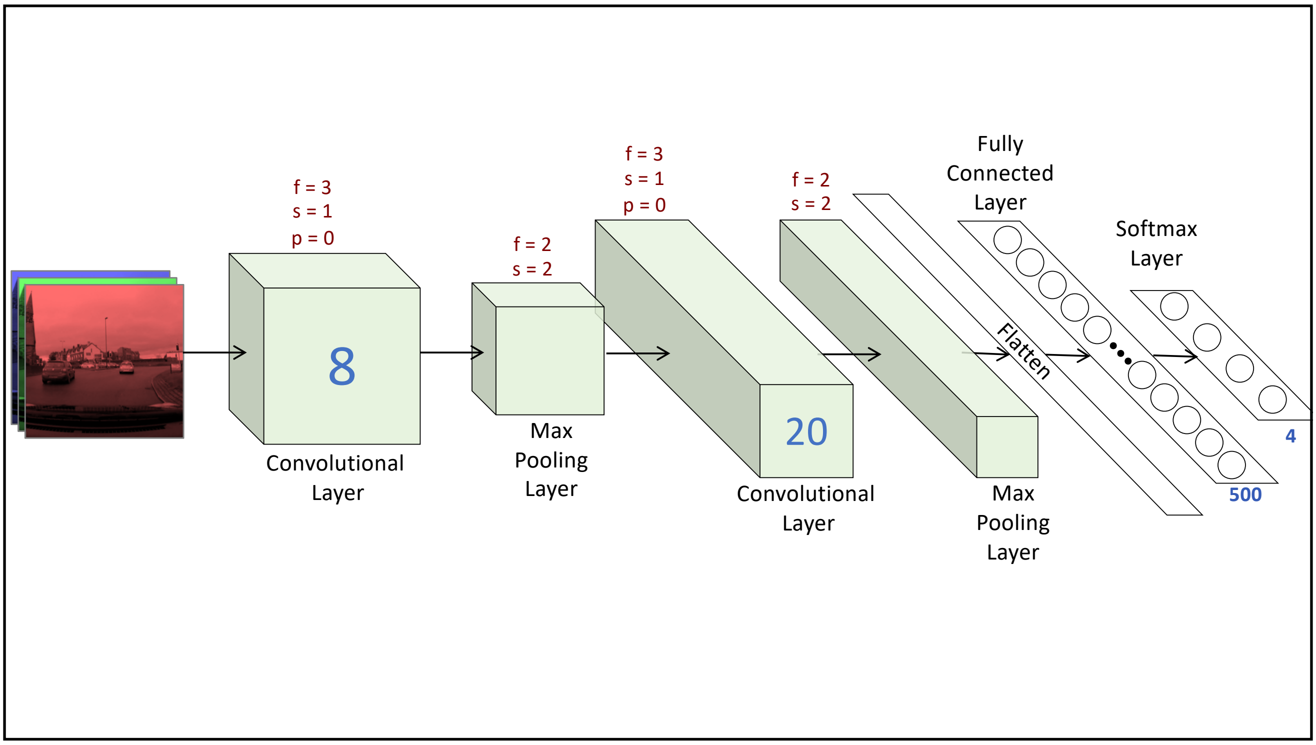

f = 3
s = 1
p = 0
8
Convolutional Layer
f = 2
s = 2
Max Pooling Layer

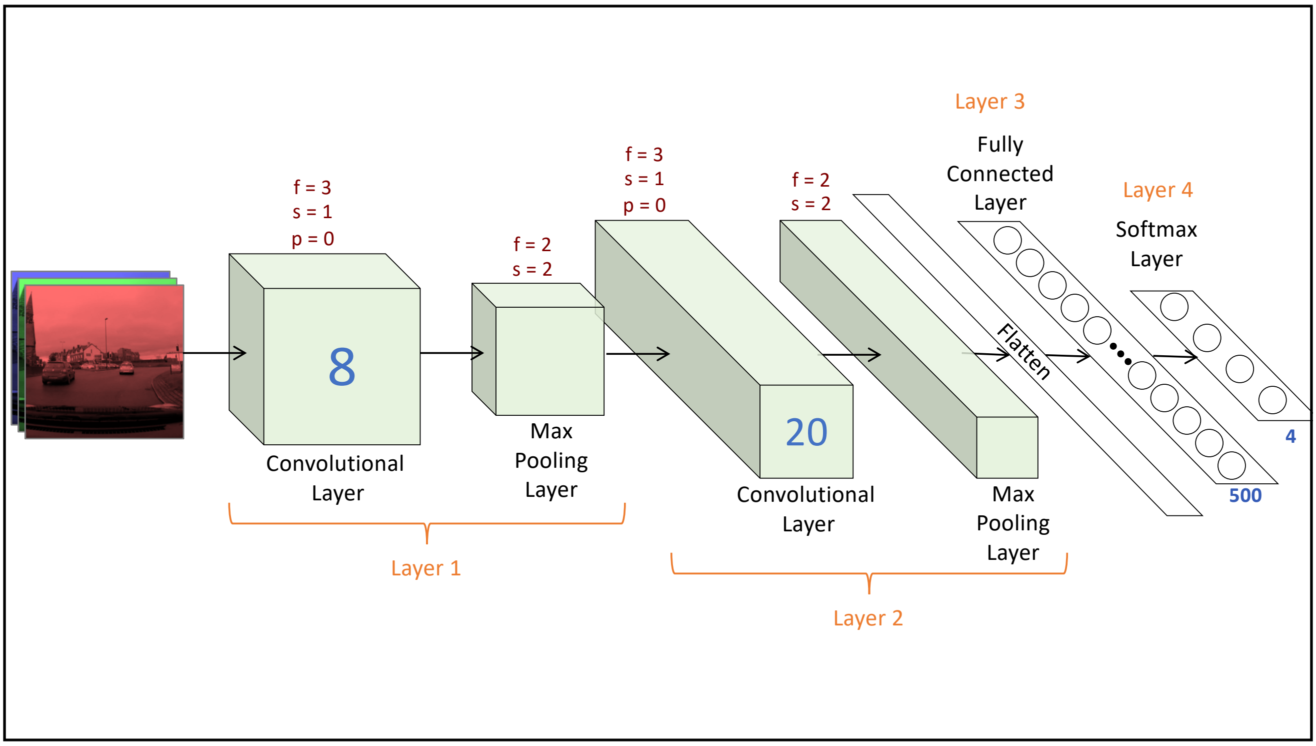

f = 3
s = 1
p = 0
8
Convolutional Layer
f = 2
s = 2
Max Pooling Layer

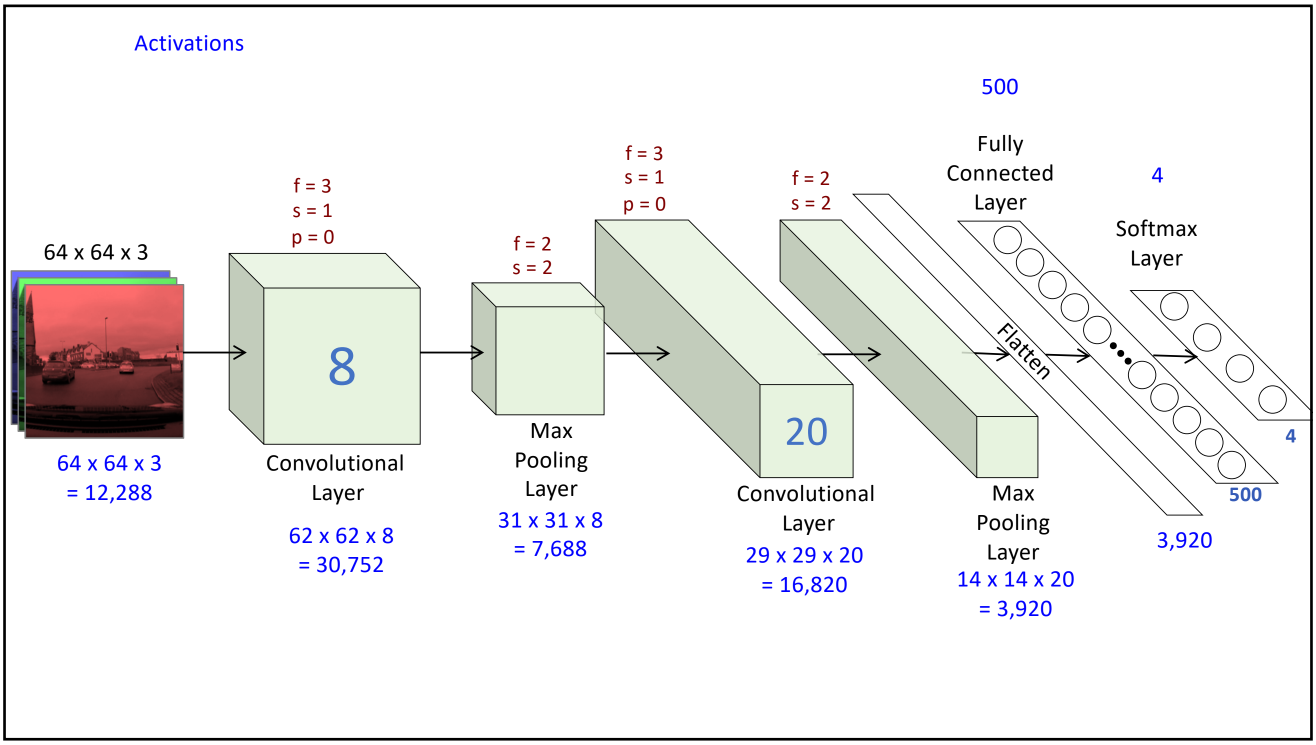

f = 3
s = 1
p = 0
8
Convolutional Layer
f = 2
s = 2
Max Pooling Layer
f = 3
s = 1
p = 0
20
Convolutional Layer
f = 2
s = 2
Max Pooling Layer

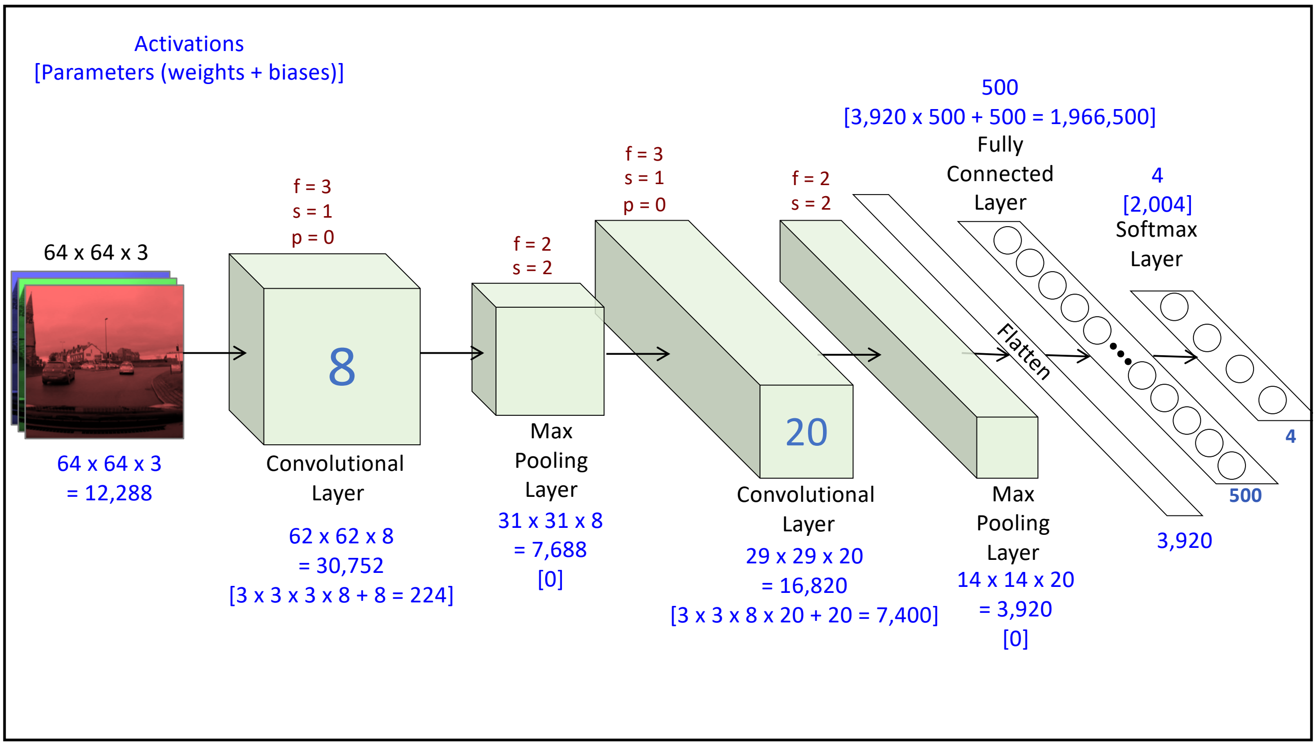

f = 3
s = 1
p = 0
8
Convolutional Layer
f = 2
s = 2
Max Pooling Layer
f = 3
s = 1
p = 0
20
Convolutional Layer
f = 2
s = 2
Max Pooling Layer
Flatten
Fully Connected Layer
500


f = 3
s = 1
p = 0
8
Convolutional Layer
f = 2
s = 2
Max Pooling Layer
f = 3
s = 1
p = 0
20
Convolutional Layer
f = 2
s = 2
Max Pooling Layer
Flatten


f = 3
s = 1
p = 0
8
Convolutional Layer
f = 2
s = 2
Max Pooling Layer
f = 3
s = 1
p = 0
20
Convolutional Layer
f = 2
s = 2
Max Pooling Layer
Flatten
Fully Connected Layer
500
Softmax Layer
4


f = 3
s = 1
p = 0
8
Convolutional Layer
f = 2
s = 2
Max Pooling Layer
Layer 1
f = 3
s = 1
p = 0
20
Convolutional Layer
f = 2
s = 2
Max Pooling Layer
Layer 2
Flatten
Layer 3
Fully Connected Layer
500
Layer 4
Softmax Layer
4


Activations
500
Fully
Connected
Layer
4
Softmax
Layer
64 x 64 x 3
f = 3
s = 1
p = 0
8
f = 2
s = 2
f = 3
s = 1
p = 0
f = 2
s = 2
20
Flatten
4
500
64 x 64 x 3
= 12,288
Convolutional
Layer
62 x 62 x 8
= 30,752
Max
Pooling
Layer
31 x 31 x 8
= 7,688
Convolutional
Layer
29 x 29 x 20
= 16,820
Max
Pooling
Layer
14 x 14 x 20
= 3,920
3,920


Activations
[Parameters (weights + biases)]
500
[3,920 x 500 + 500 = 1,966,500]
Fully
Connected
Layer
4
[2,004]
Softmax
Layer
64 x 64 x 3
f = 3
s = 1
p = 0
8
f = 2
s = 2
f = 3
s = 1
p = 0
f = 2
s = 2
20
Flatten
4
500
64 x 64 x 3
= 12,288
Convolutional
Layer
62 x 62 x 8
= 30,752
[3 x 3 x 3 x 8 + 8 = 224]
Max
Pooling
Layer
31 x 31 x 8
= 7,688
[0]
Convolutional
Layer
29 x 29 x 20
= 16,820
[3 x 3 x 8 x 20 + 20 = 7,400]
Max
Pooling
Layer
14 x 14 x 20
= 3,920
[0]
3,920

WHY CONVOLUTIONS?

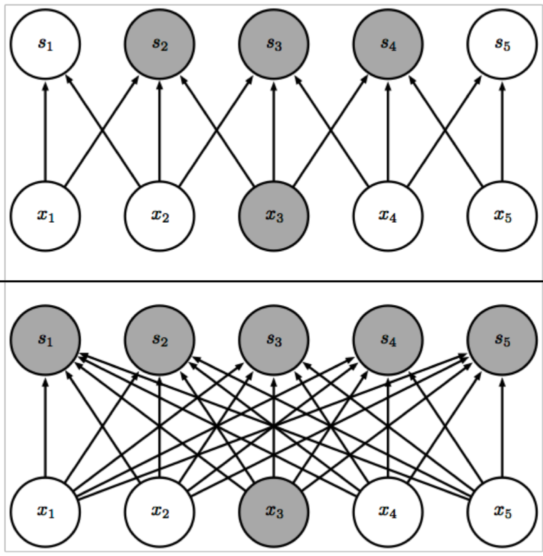
Why Convolutions?

There are two main ideas that are considered to be the main reasons that CNNS work so well

- Sparse connections
- Shared weights

Sparse Connections

Sparse connections due to small convolution kernel

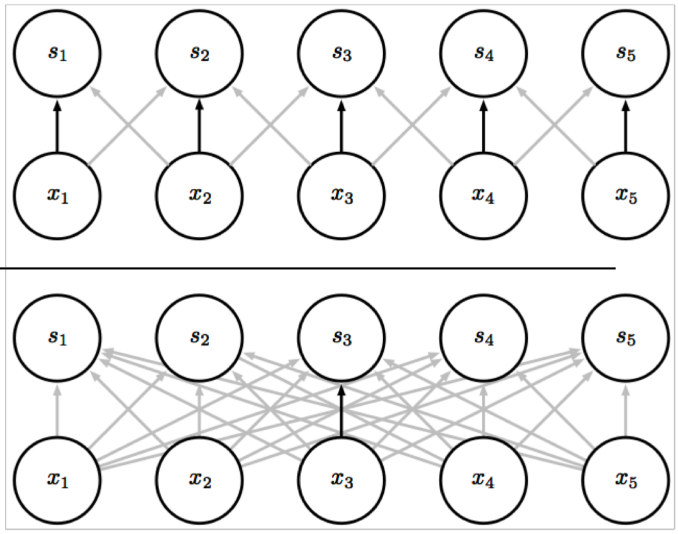


Dense connections

Goodfellow, Bengio, Courville, Deep Learning, MIT Press, 2016

Shared Weights

Convolution shares the same parameters across all spatial locations



Traditional matrix multiplication does not share any parameters

Goodfellow, Bengio, Courville, Deep Learning, MIT Press, 2016

Original Image

123	127	128	119	115	130
140	145	148	153	167	172
133	154	183	192	194	191
194	199	207	210	198	195
164	170	175	162	173	151

*

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

=

Filtered Image

SUMMARY

Summary

- Image processing is now dominated by the application of convolutional neural networks
- Introduction of convolutional and pooling units to our network structure allows us reduce the number of parameters in a large network and take advantage of data structure
- We can apply to the same ideas to lots of other data types

Questions

?