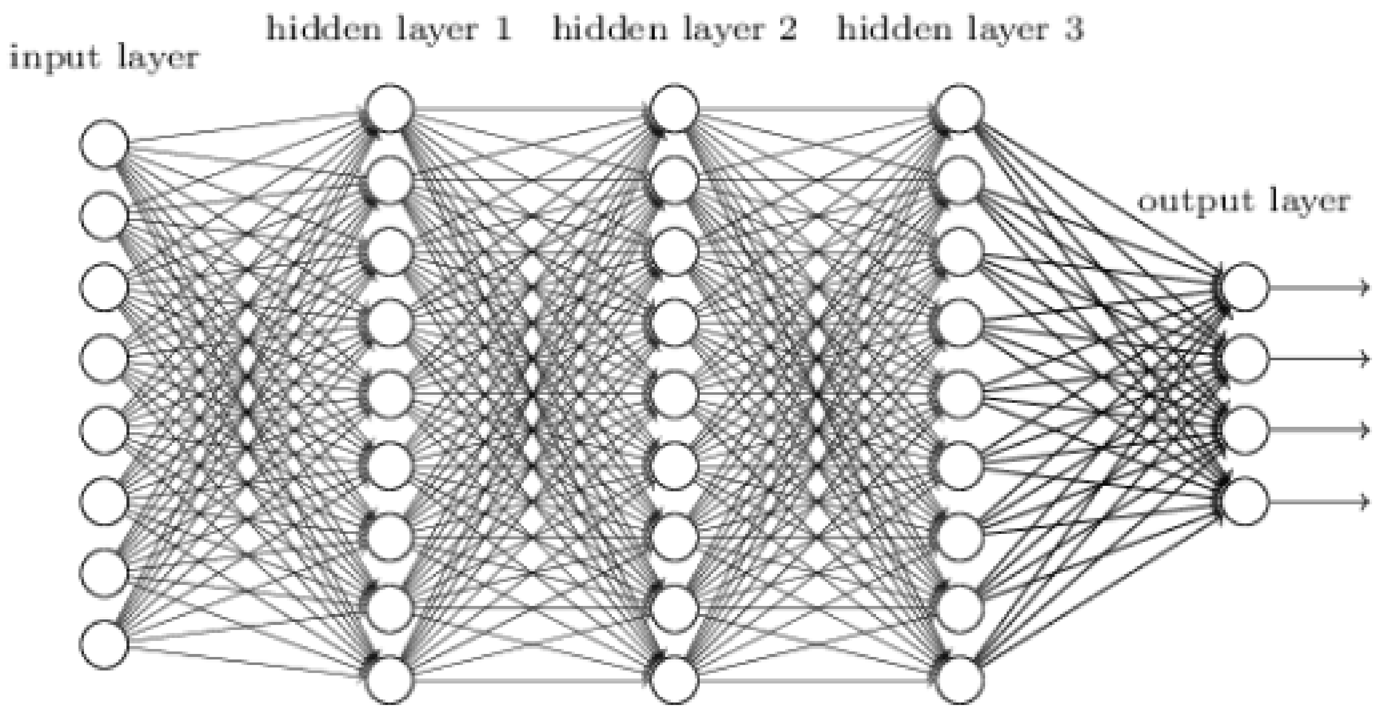
CNN

- We know it is good to learn a small model.
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?

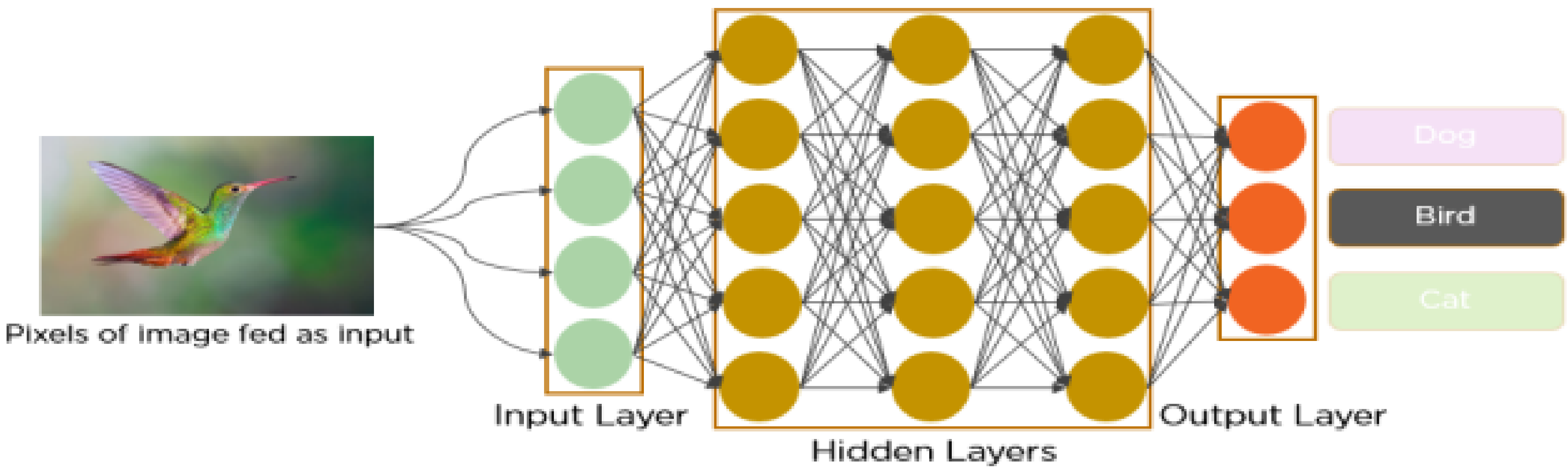


Introduction

In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks.

What exactly is a CNN?

In deep learning, a **convolutional neural network** (**CNN/ConvNet**) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics **convolution** is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

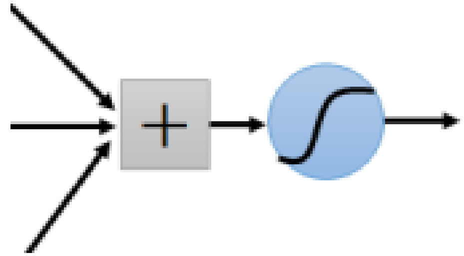


Consider learning an image:

- Some patterns are much smaller than the whole image

Can represent a small region

with fewer parameters



"beak" detector

Same pattern appears in different places:

They can be compressed!

What about training a lot of such "small" detectors and each detector must "move around".

"upper-left beak" detector

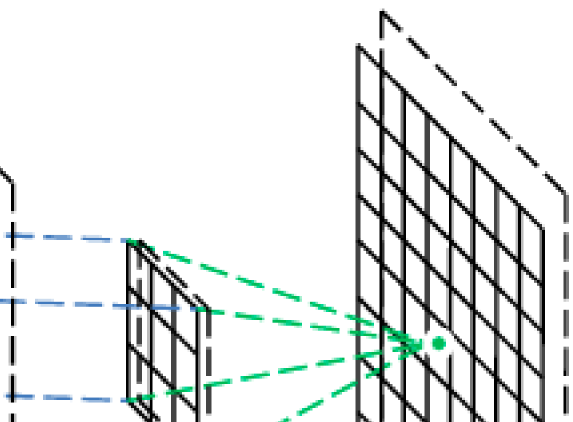
They can be compressed
to the same parameters.

“middle beak”
detector

A convolutional layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number

s convolutional operation.



Beak detector

Convolution

A filter

These are the network parameters to be learned.

1	-1	-1
-1	1	-1
-1	-1	1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0

1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

6 x 6 image

Each filter detects a small pattern (3 x 3).

Convolution

0	0	1	1
1	0	0	0
0	1	0	0
0	0	1	0

-1	-1	1

3 3

-1

stride=1

1	0	0	0
0	1	0	0

6 x 6 image

Dot product

1	-1
-1	1

Filter 1

Convolution

0	0	1	1	0
1	0	0	0	1
0	1	0	0	1
0	0	1	0	1

1	-1
-1	1
-1	-1

If stride=2

1	0	0	0	0
0	1	0	0	1

6 x 6 image

3

-3

Convolution

stride=1

Filter 1

1
-
-

Filter 1

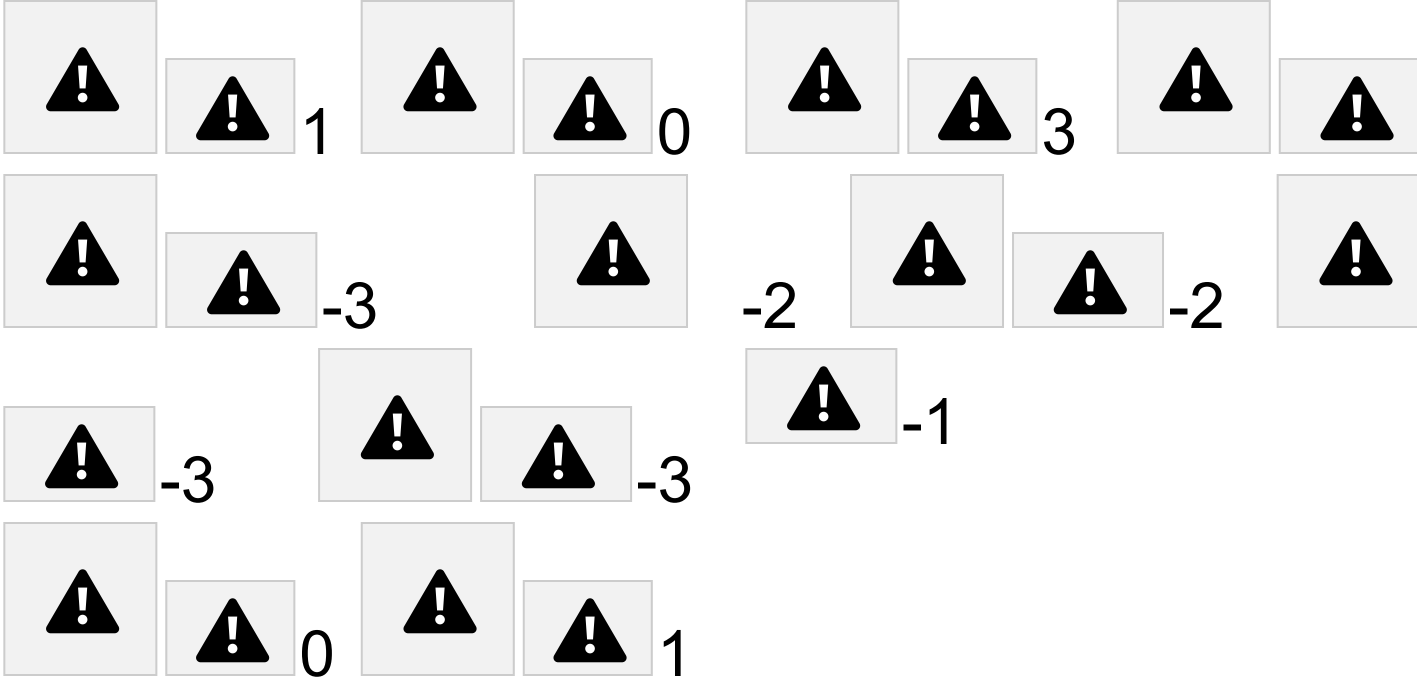
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3

-1 -3

-1 -3



-1	1	-1
-1	1	-1
-1	1	-1

Convolution

Filter 2

stride=1

Repeat this for each filter

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

3 -1 -3 -1
-1 -1 -1 -1

-3 1 0 -3
-1 -1 -2 1

-3 -3 0 1 -1 -1 -2 1

6 x 6 image

3 -2 -2 -1 -1 0 -4 3

Feature

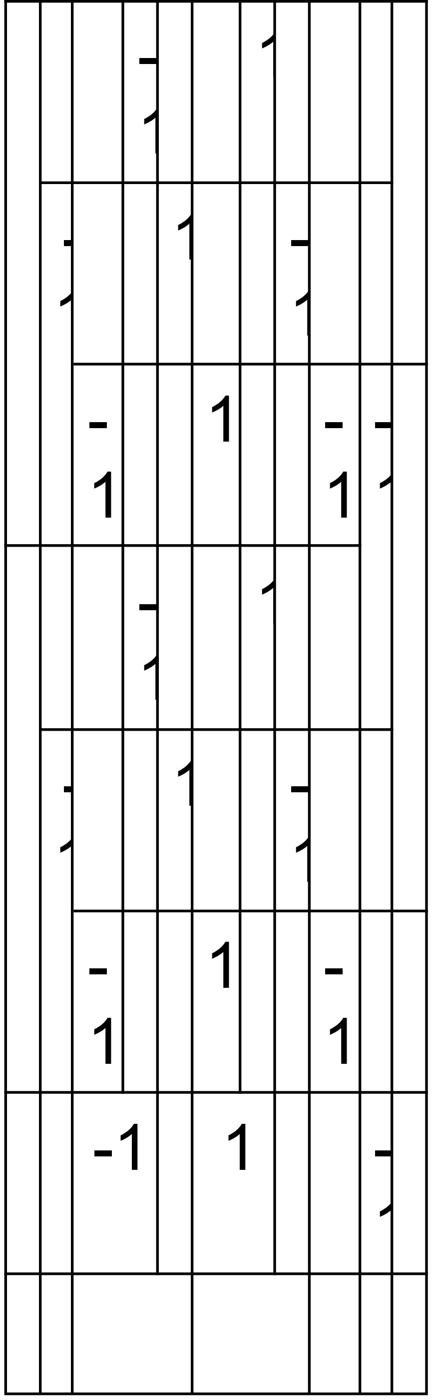
Map

Two 4 x 4 images

Forming 2 x 4 x 4 matrix

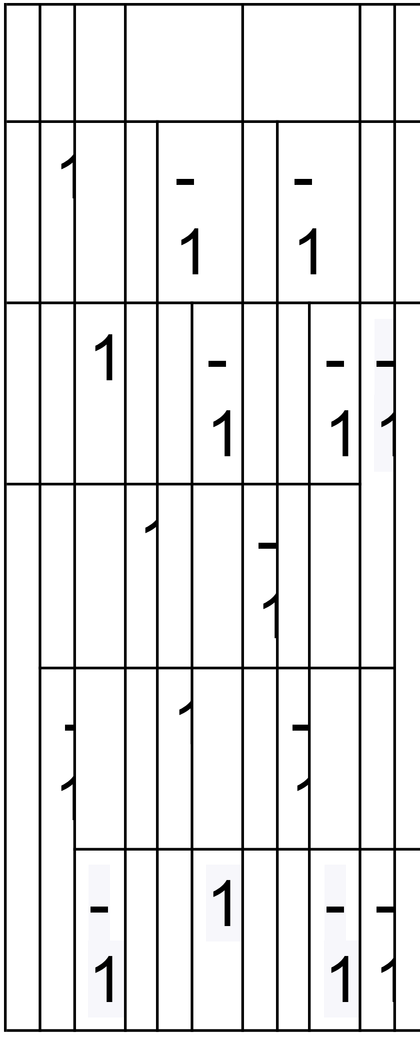
Color image: RGB 3 channels

-1	1	-1	
-1	1	-1	-1



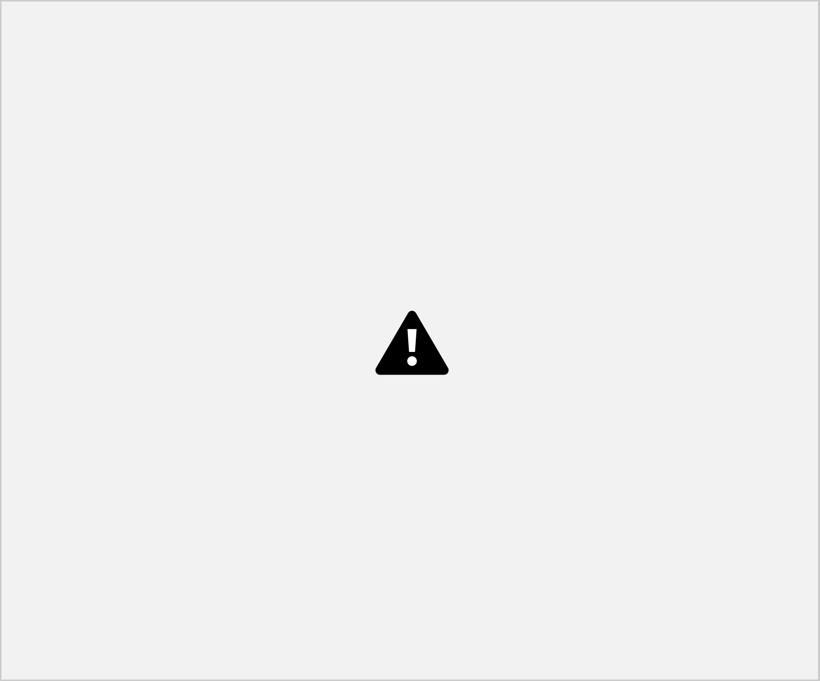
Filter 1 Filter 2

Color image

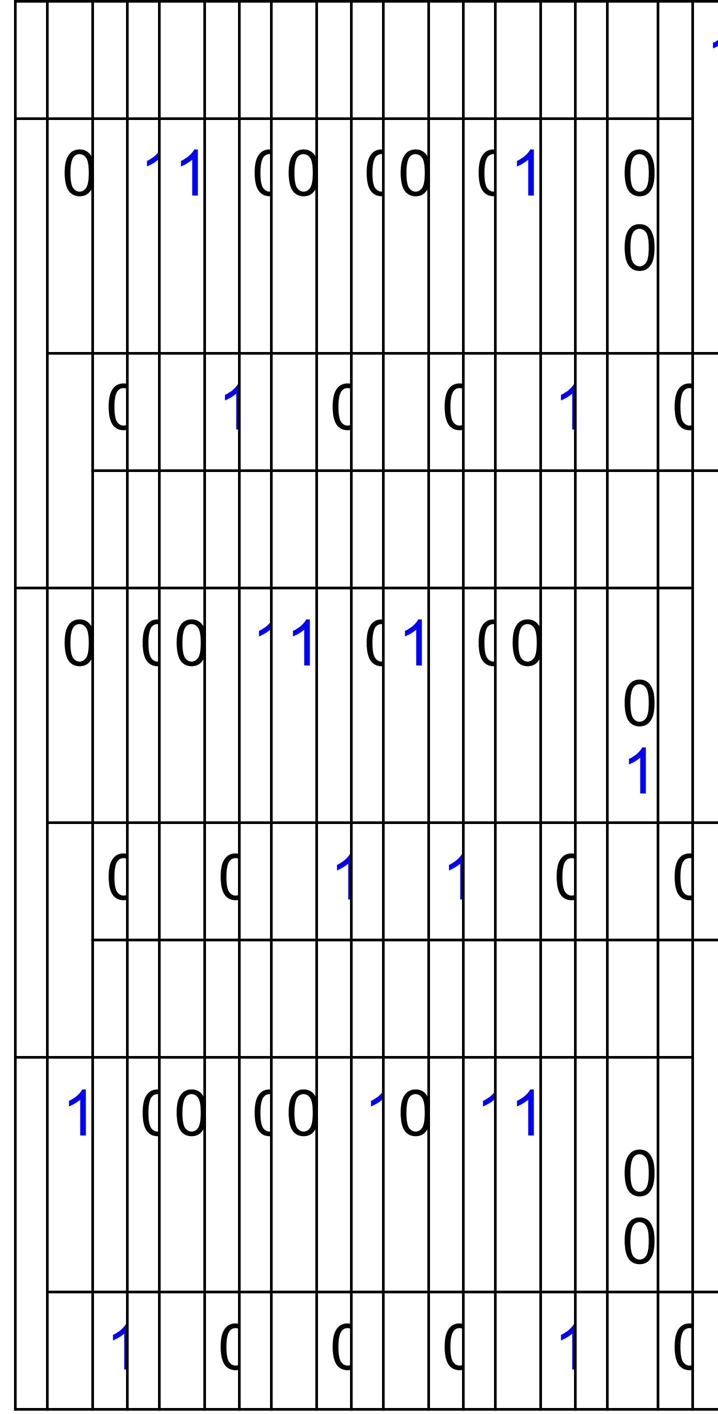


-1	1			
-1	-1	1		
-	-	1		

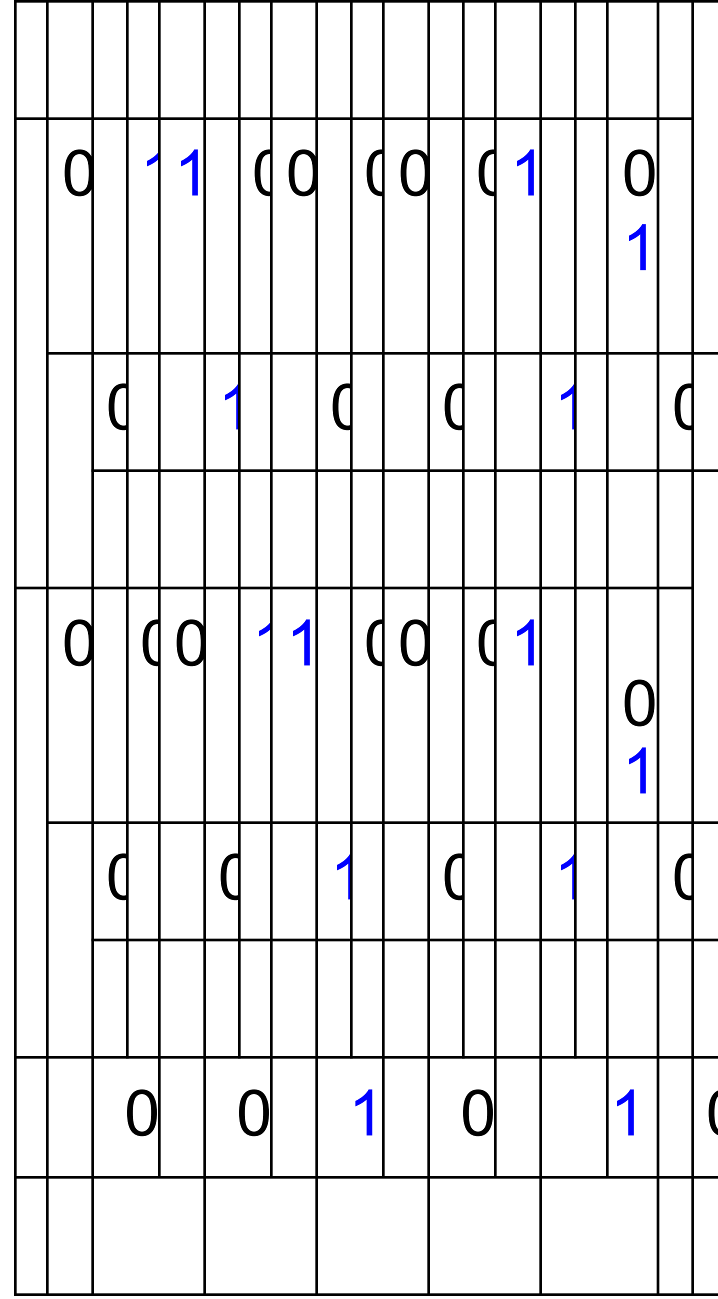
1	1			
-1	-1	1		



1	0	0	0	0	1
1	0	0	0	0	1



0 1 1 0 0 0 0 0 1 0
0
0 1 0 0 1 0
0 0 0 1 1 0 1 0 0
0
1
0 0 1 1 0 0
1 0 0 0 0 1 0 1 1
0
0
1 0 0 0 1 0


0 1 1 0 0 0 0 0 1 0 1
0 1 0 0 1 0
0 0 0 1 1 0 0 0 1 0 1
0 0 1 0 1 0
0 0 1 0 1 0

Convolution v.s. Fully Connected

0	1	0	0	1	0
0	0	1	0	1	0

-1	1	-1
-1	1	-1
-1	1	-1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0

1	-1	-1
-1	1	-1
-1	-1	1

image convolution

1	0	0	0
0	1	0	0
0	0	1	1
1	0	0	0
0	1	0	0
0	0	1	0

x_1

x_2

Fully connected

$_{36}\,x$

1

0

3

0

3

4:

0

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1 1 2

0	1	0	0
0	0	1	0

fewer parameters!

1	0	0	0
0	1	0	0
0	0	1	1
1	0	0	0

6 x 6 image

8

9

10:

1
3
14

15

16

01

00

001

1

9 inputs, not fully connected

Only connect to

1:

-1	-1	1

1	-1	-1
-1	1	-1

Filter 1

2: 3:4:

1

000

3

1	0	0	0	0
0	1	0	0	1
0	0	1	1	0
1	0	0	0	1
0	1	0	0	1
0	0	1	0	1

6 x 6 image

Fewer parameters

Even fewer parameters

7:

8:

9:

10:

1

3:

14:

15:

16:

01

00

001

1

-1

Shared weights

The whole CNN

cat dog ……

Convolution

Max Pooling

Fully Connected
Feedforward network

Flattened

Convolution

Can
repeat many times

Max Pooling

Max Pooling: Max Pooling is **a pooling operation that calculates the maximum value for patches of a feature map, and uses it to create a downsampled (pooled) feature map**. It is usually used after a convolutional layer.

1	-1	-1
-1	1	-1

-1	-1	1

Filter 1

-1	1
-1	1

-1	1	-1

Filter 2

-3 -3

3 0 1

-1 -3 3

-1 -3 -2 -2

-1

1 0

-3 -1

-1 -1 -1

-1 -2 1

-1 -1 -1

-2 0 -4

1 -1 3

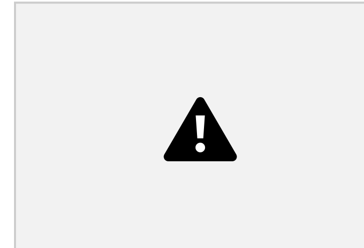
Why Pooling

- Subsampling pixels will not change the object

bird

bird

Subsampling



We can subsample the pixels to make image smaller

fewer parameters to characterize the image

- Convolutional layers in a convolutional neural network systematically apply learned filters to input images in order to create feature maps that summarize the presence of those features in the input.
- Convolutional layers prove very effective, and stacking convolutional layers in deep models allows layers close to the input to learn low-level features (e.g. lines) and layers deeper in the model to learn high-order or more abstract features, like shapes or specific objects.
- A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. This means that small movements in the position of the feature in the input image will result in a

different feature map. This can happen with re-cropping, rotation, shifting, and other minor changes to the input image.

- A common approach to addressing this problem from signal processing is called down sampling. This is where a lower resolution version of an input signal is created that still contains the large or important structural elements, without the fine detail that may not be as useful to the task.

A CNN compresses a fully connected network in two ways:

- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

Max Pooling

0	0	1	1	0	0	0	0	1	0	1	0

1	0	0	0	0	1
0	1	0	0	1	0

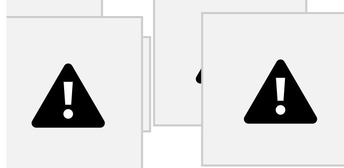
1	0	0	0	1	0
0	1	0	0	1	0

6 x 6 image

Conv

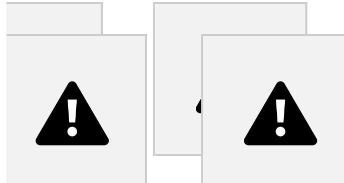
Max Pooling

New image but smaller



3 0

-1 1



3 1

0 3

2 x 2 image

Each filter is a channel

The whole CNN

3 0

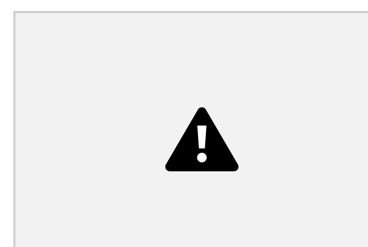
-1 1

3 1

0 3

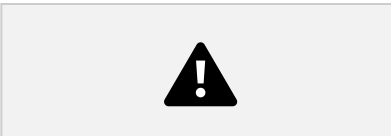
Smaller than the original image

The number of channels is the number of filters



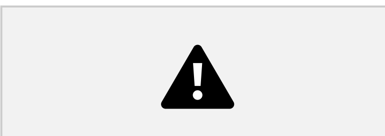
Pooling

Convolution

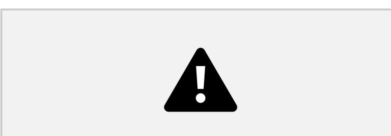


Max

Pooling



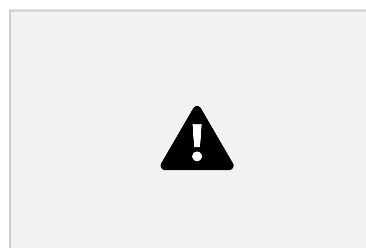
Convolution



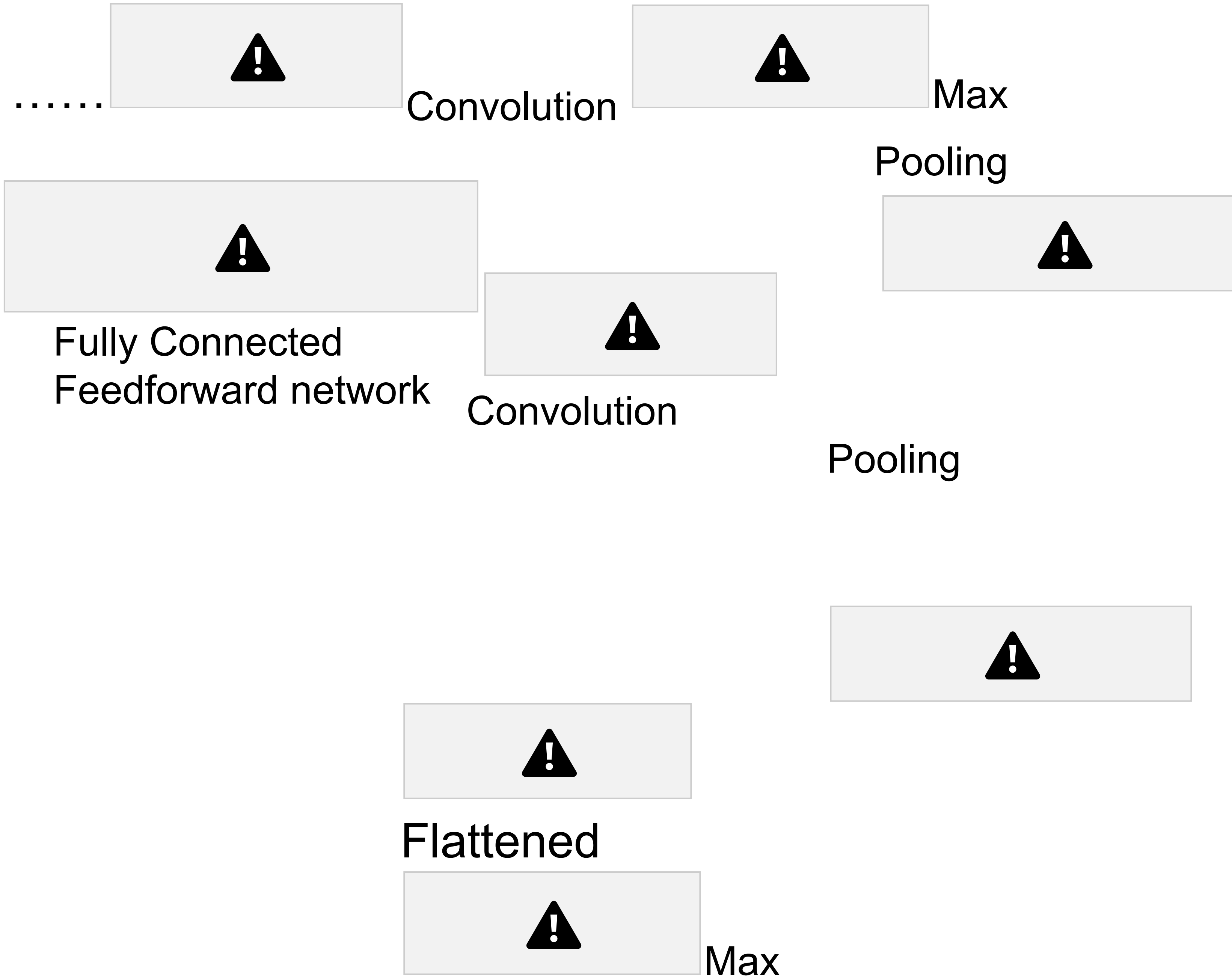
Max

Can
repeat many times

The whole CNN



cat dog



……
Convolution
Max
Pooling
Fully Connected
Feedforward network
Convolution
Pooling
Flattened
Max

Flattening

1 3

-1 1

0 3

1 1

0

3 Flattened

3 0

Fully Connected
Feedforward network

Only modified the

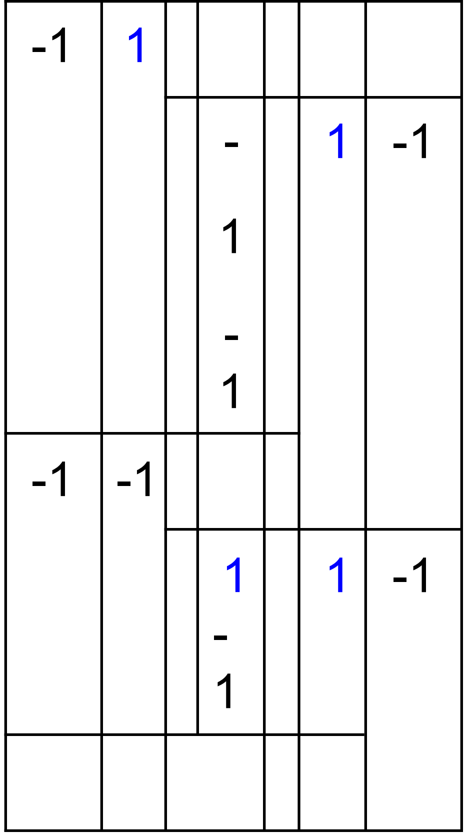
network structure and

CNN in Keras ***input format (vector -> 3-D tensor)***

input

Convolution

1						
	-1		- 1 - 1		1	-1



There are

...

25 3x3

…

Max Pooling

filters.

Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

Convolution

Max Pooling

3 -1

3

-3

1

Only modified the

network

structure and

CNN in Keras

input format (vector -> 3-D array)

Input

1 x 28 x 28

Convolution

How many parameters for each filter?

25 x 26

x 26

25 x 13

x 13

Convolution

Max

Pooling

How many parameters for each filter?

50 x

Max Pooling

225=
25x9

50 x 11 x 11

5 x 5

Only modified

the ***network***

structure and

<u>CNN in</u>

Keras ***input format (vector -> 3-D array)***

Input

1 x 28 x 28

Output

Fully connected feedforward network

1250

Convolution

25 x 26 x 26

Max Pooling

x 11 x 11

Max Pooling

50 x 5 x 5
Flattened

25

x 13 x 13

Convolution

50

AlphaGo

Neural

Next move

19 x 19 matrix

Black: 1

white: -1

none: 0

Network (19 x 19 positions)

Fully-connected feedforward network can be used

But CNN performs much better

AlphaGo's policy network

The following is quotation from their Nature article:

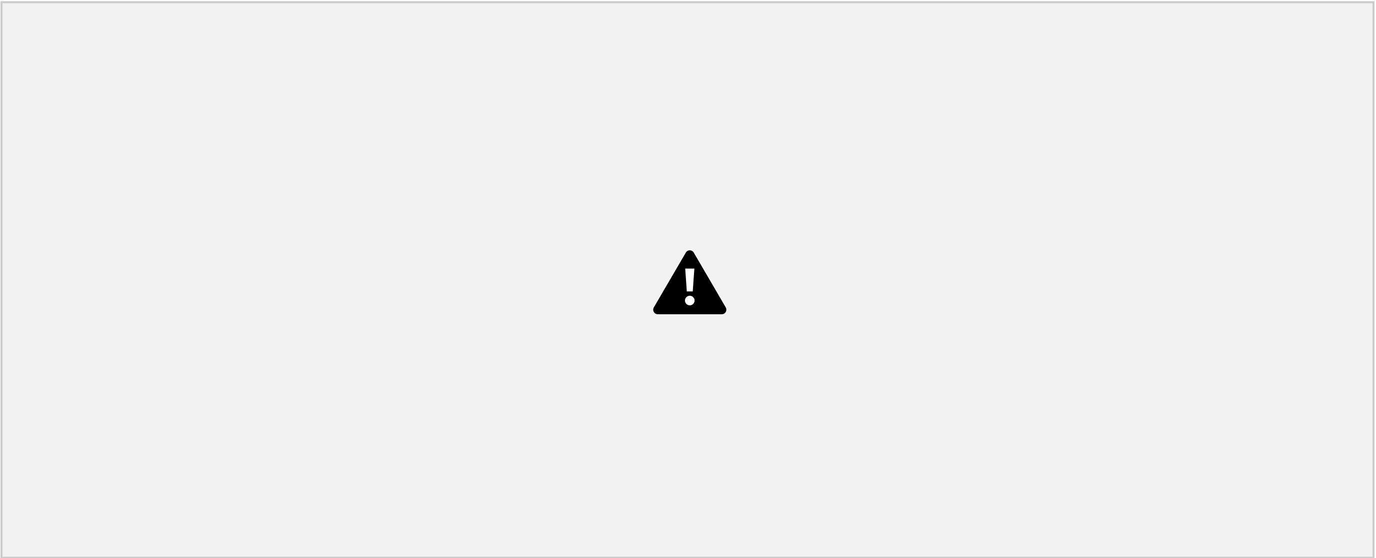
Note: AlphaGo does not use Max Pooling.

CNN in speech recognition

Frequency

CNN Image

The filters move in the frequency direction.



Time

Spectrogram

CNN in text classification

?

Source of image: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.6858&rep=rep1&type=pdf