Changing random stuff until your program works is “hacky” and “bad coding practice”.
But if you do it fast enough, it’s “Machine Learning” and pays 4x your current salary

- Unknown

Convolutional Neural Networks

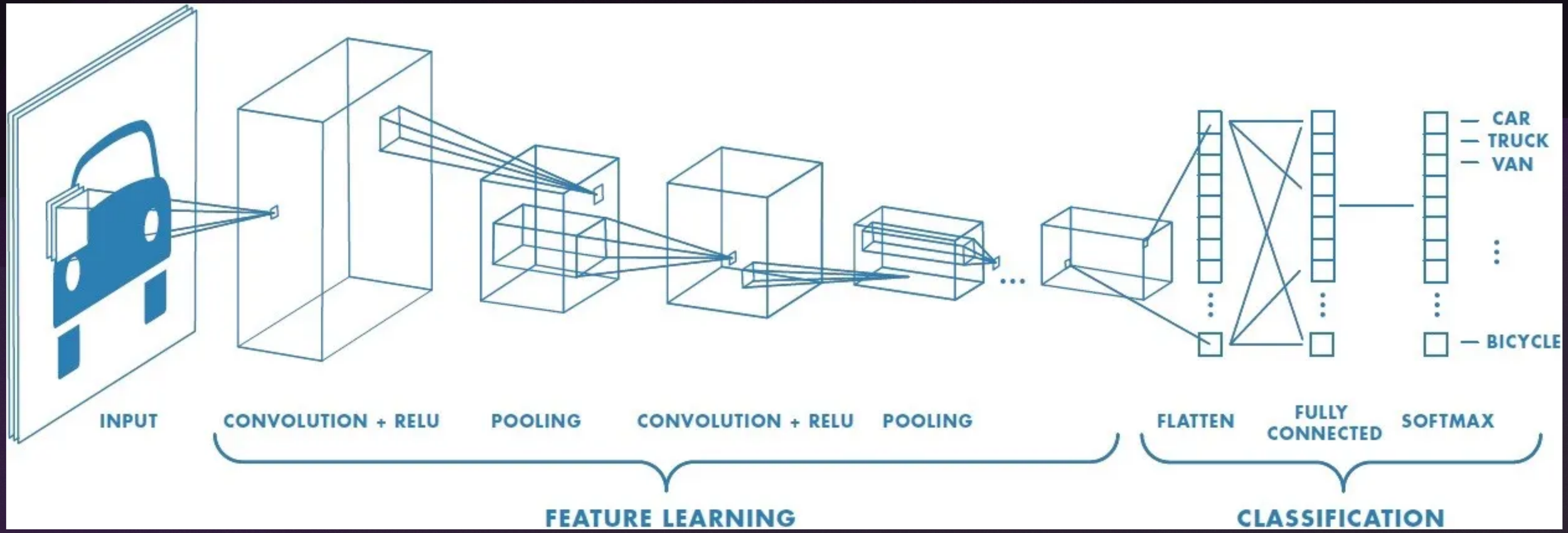
AI Systems Implementation

George Langroudi
Based on slides by Gareth Howells

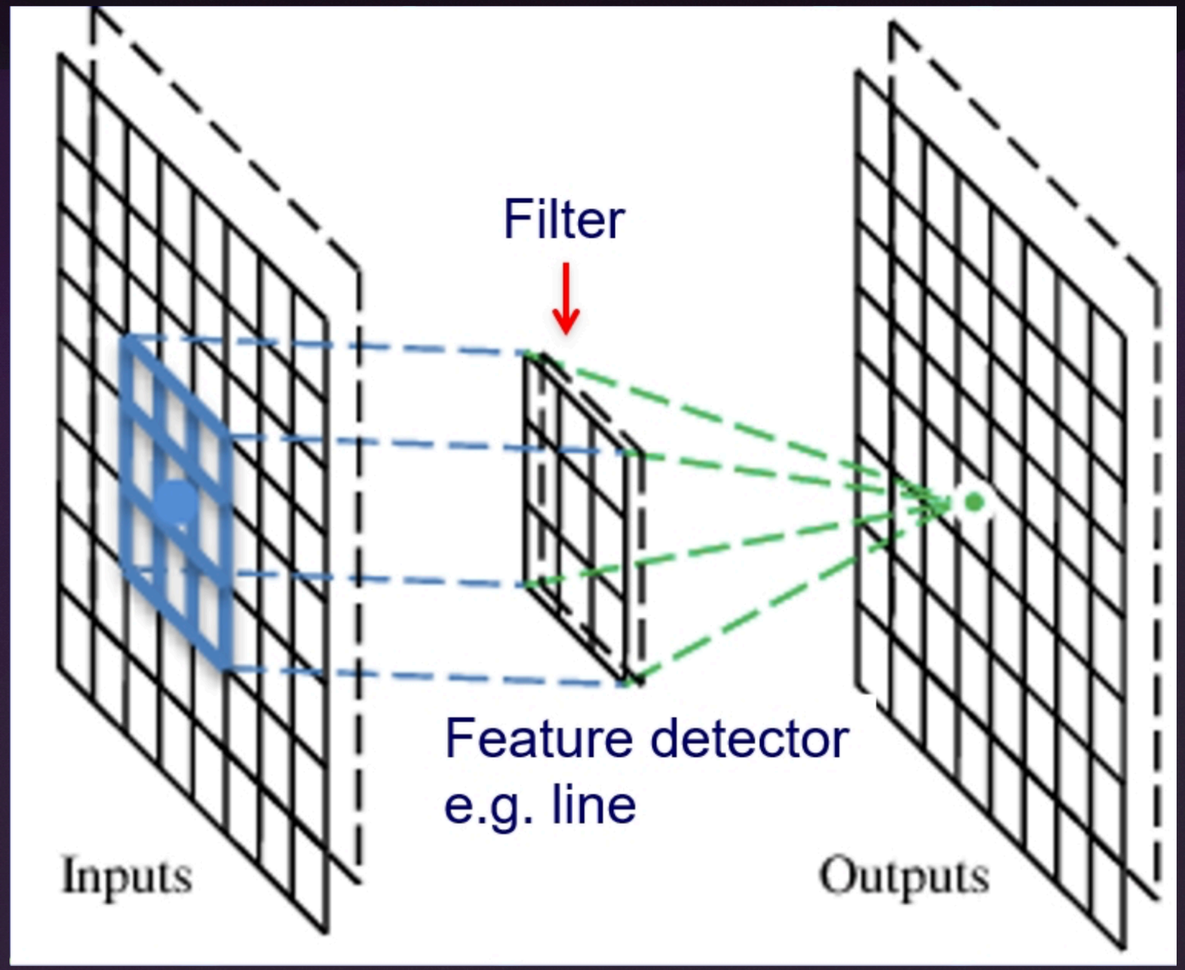
Overview

- Convolutional Neural Networks
  - Architecture
  - Filters / Convolution
  - Pooling
  - Flattening
- Fully Connected Layers

Convolutional Neural Networks

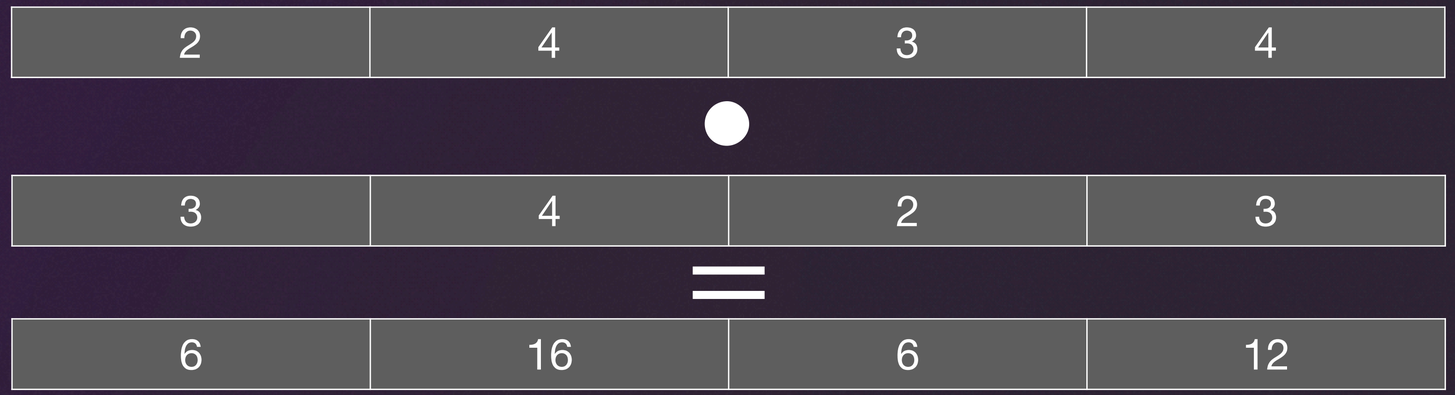


A Convolutional layer



Wait, what is convolution?

- First, we need to understand *dot product*
- $$dotproduct_{ab} = \sum_{i=1}^{n} a_i b_i$$
- *Essentially,* multiple every point in a by it's corresponding point in b



Wait, what is convolution?

- $(a * b)_k = \sum_{i=1}^{n} a_i b_{k-i}$
- $(a * b)_k$ = The convolution of a and b, at point k
- $\sum_{i=1}^{n} a_i b_{k-i}$ = Calculating the dot product at point k
  - OK… what?

1-Dimensional Convolution

$$(1 \times 4) + (2 \times 3) + (3 \times 2) + (2 \times 4) + (3 \times 1)$$

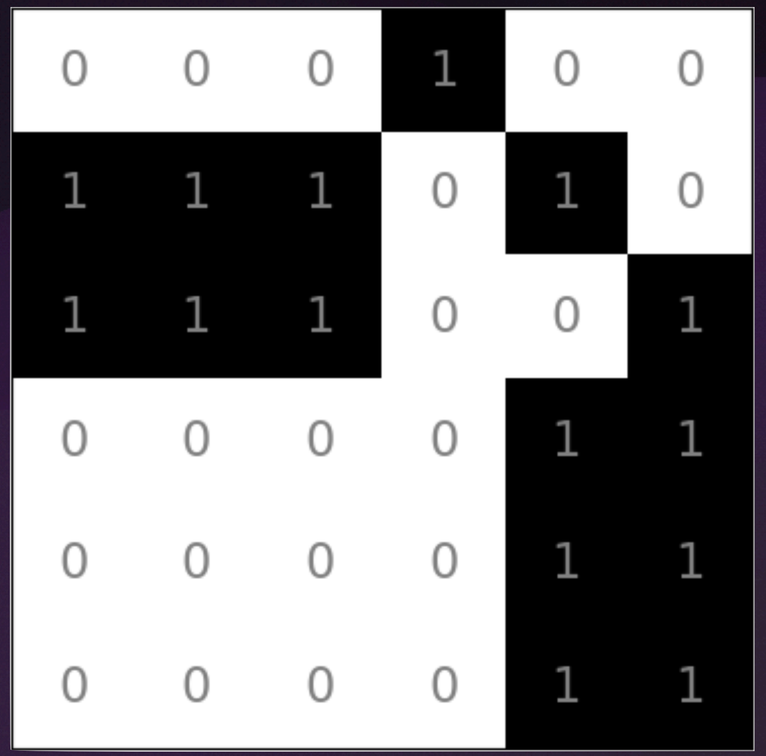
1	2	3

4	3	2	4	1	5	3	5

16	19	13	21	20	26

2 Dimensional Convolution

Filters



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

Vertical lines

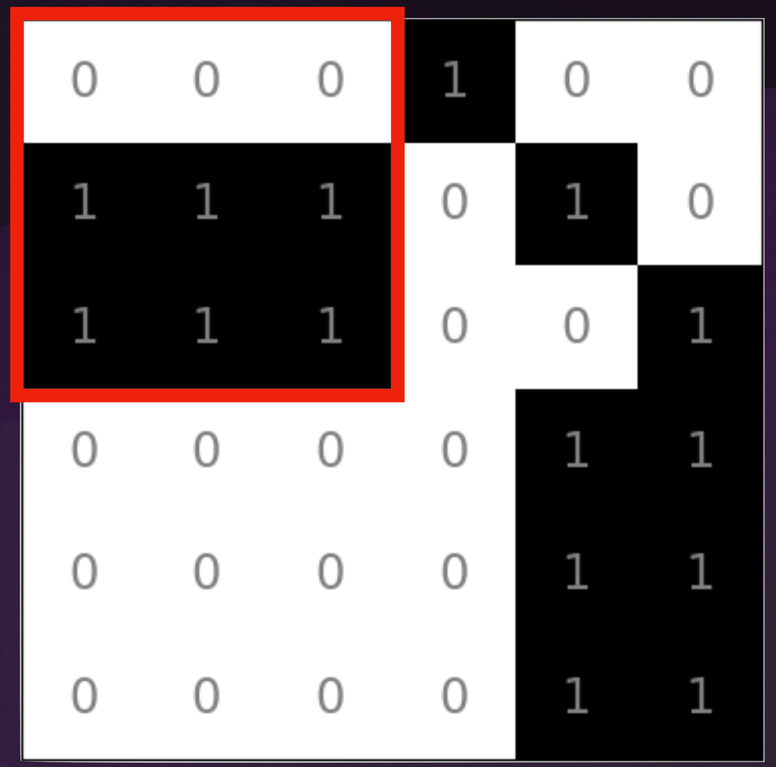
-1	-2	-1
0	0	0
1	2	1

Horizontal lines

2 Dimensional Convolution

Filters

Stride = 1



6×6 image

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

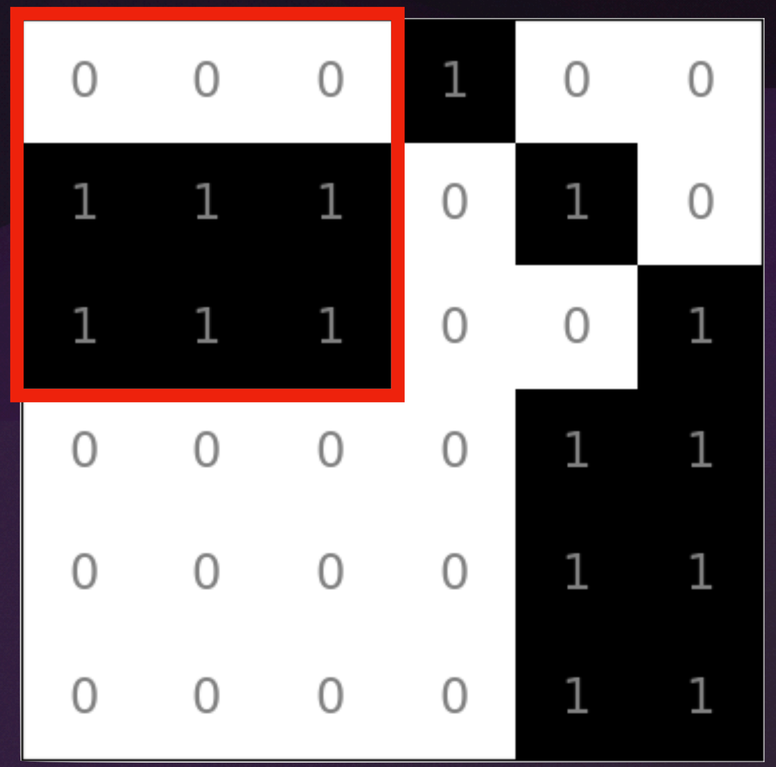
-1	-2	-1
0	0	0
1	2	1

Sobel Horizontal filter

2 Dimensional Convolution

Filters

Stride = 1



6×6 image

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

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

Vert Horizontal filter

Example

The shape of the filter allows us to extract spatial features from our data

Lets use Sobel filters to do edge detection

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

Vertical Sobel filter

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

Horizontal Sobel filter

Example

Original Image: Black and White

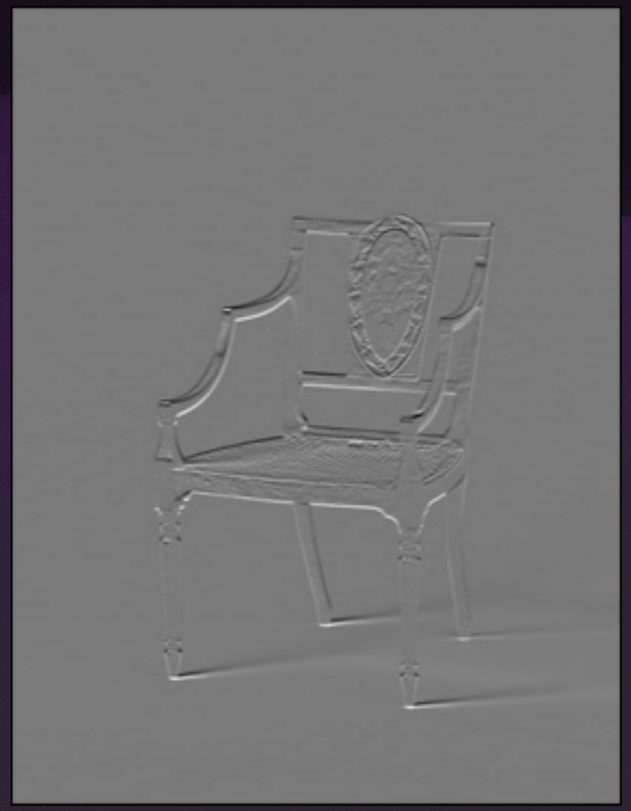


Image after convolution with a horizontal Sobel Filter

Image after convolution with a vertical Sobel Filter

Example

```python
def conv2d(image: np.array, kernel: np.array) -> np.array:
    f_width, f_height = kernel.shape
    conv_array = np.zeros([image.shape[0]+1 - f_width, image.shape[1]+1 - f_height])
    c_width, c_height = conv_array.shape

    for x in range(c_width):
        for y in range(c_height):
            target = image[x:x+f_width, y:y+f_height]
            conv_array[x,y] = np.sum(conv(target, kernel))
    return conv_array

def conv(target, filter):
    output = np.zeros(target.shape)
    for x in range(output.shape[1]):
        for y in range(output.shape[1]):
            output[x,y] = target[x,y] * filter[x,y]
    return output
```

Pooling Layer

Max Pooling

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

Sobel Horizontal filter

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

Sobel Vertical filter

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

Horizontal filtered image

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

Vertical filtered image

Pooling Layer

Max Pooling

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

Sobel Horizontal filter

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

Sobel Vertical filter

4	2
-4	-3

-1	0
-1	1

-4	-3
0	0

0	2
0	0

Horizontal filtered image

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

Vertical filtered image

Pooling Layer

Max Pooling

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

Sobel Horizontal filter

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

Sobel Vertical filter

4	2
-4	-3

-1	0
-1	1

-4	-3
0	0

0	2
0	0

Horizontal filtered image

0	-2
0	-3

-1	0
-1	3

0	-1
0	0

2	4
4	4

Vertical filtered image

Pooling Layer

Max Pooling

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

Sobel Horizontal filter

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

Sobel Vertical filter

4	

	1

-4	-3
0	0

0	2
0	0

Horizontal filtered image

0	-2
0	-3

-1	0
-1	3

0	-1
0	0

2	4
4	4

Vertical filtered image

Pooling Layer

Max Pooling

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

Sobel Horizontal filter

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

Sobel Vertical filter

4	1
0	

0	2
0	0

Horizontal filtered image

0	-2
0	-3

-1	0
-1	3

0	-1
0	0

2	4
4	4

Vertical filtered image

Pooling Layer

Max Pooling

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

Sobel Horizontal filter

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

Sobel Vertical filter

4	1
0	2

Horizontal filtered image

0	-2
0	-3

-1	0
-1	3

0	-1
0	0

2	4
4	4

Vertical filtered image

Pooling Layer

Max Pooling

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

Sobel Horizontal filter

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

Sobel Vertical filter

4	1
0	2

Horizontal filtered image

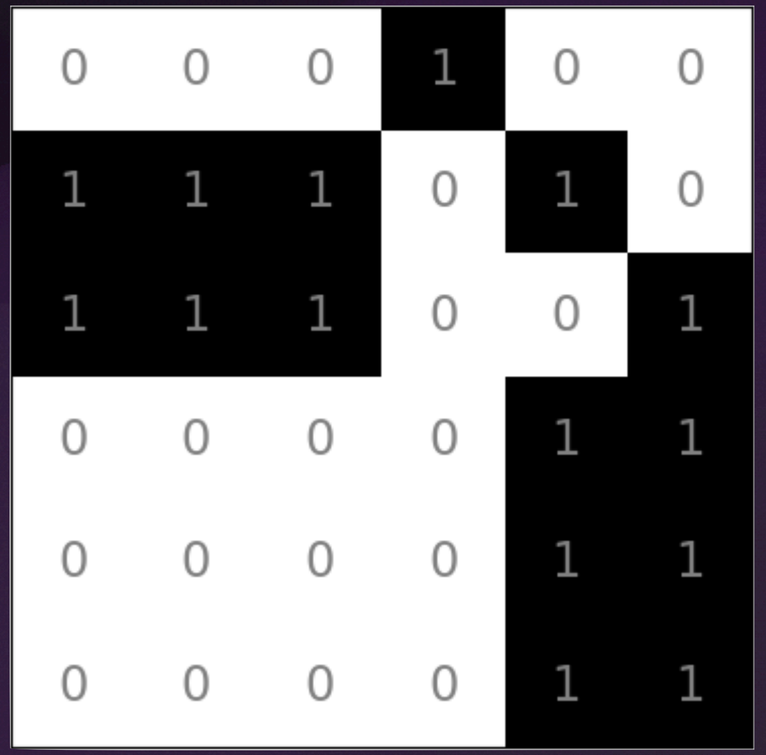
0	
0	

	3

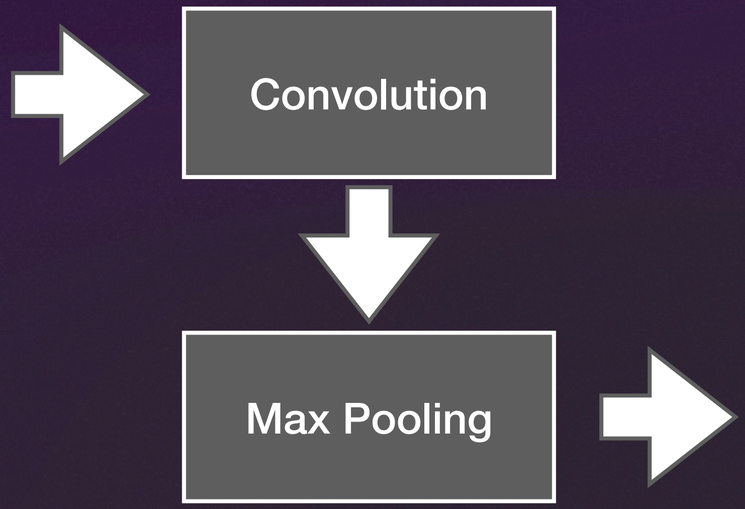
	4

Vertical filtered image

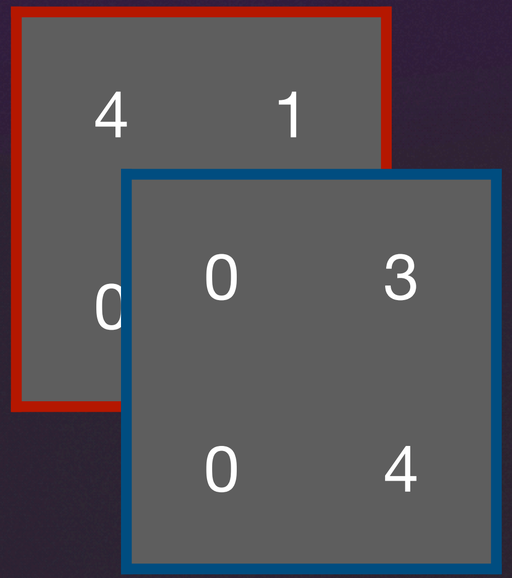
Pooling Layer



6 × 6 image



New image, but smaller



2 × 2 image

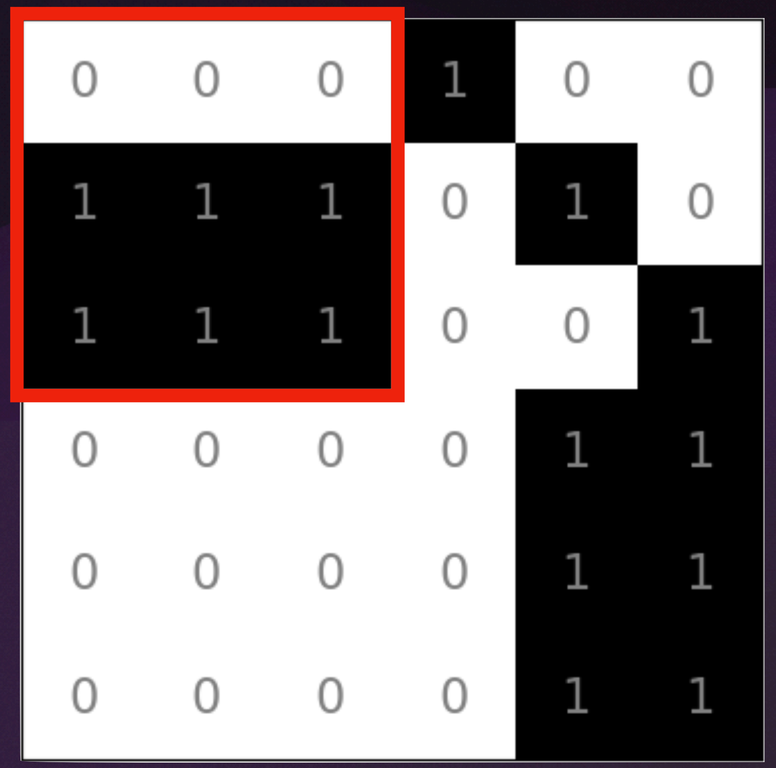
Each filter is a channel

2 Dimensional Convolution

QUIZ TIME

2 Dimensional Convolution

QUIZ TIME



1	0	0
0	1	0
0	0	1

A. 0

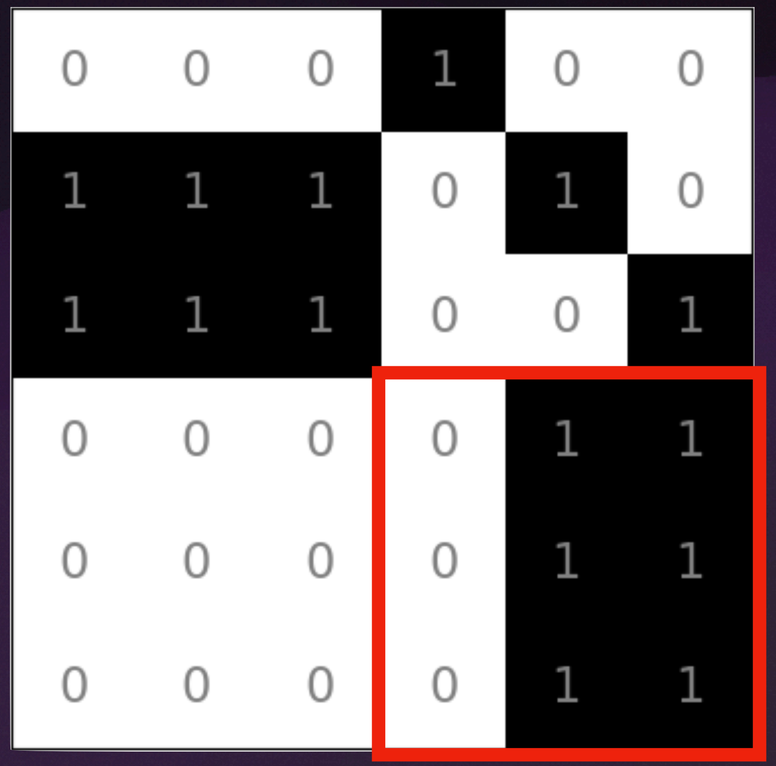
B. 1

C. **2**

D. 3

2 Dimensional Convolution

QUIZ TIME



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

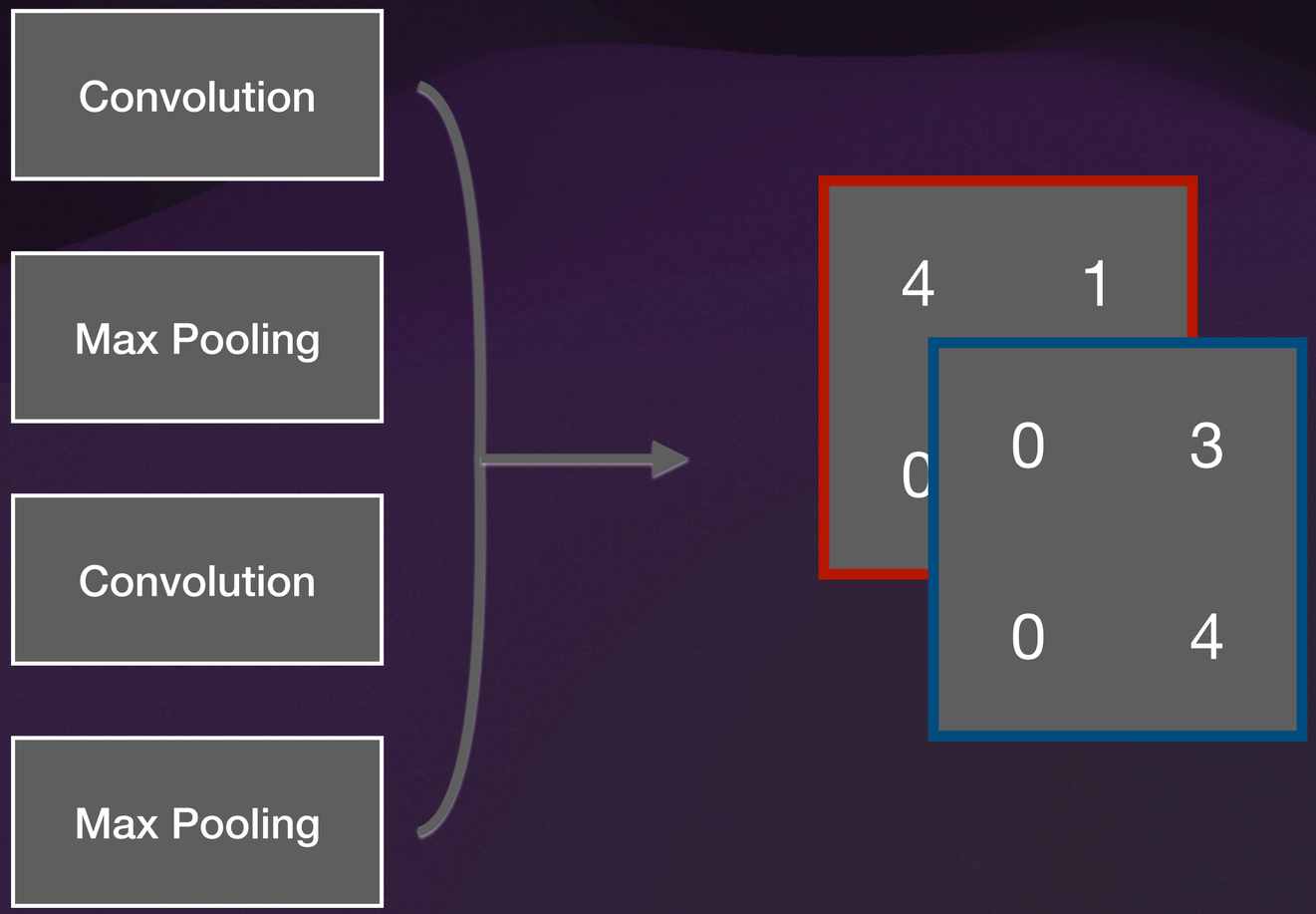
A. **0**

B. 1

C. 2

D. 3

Multiple Convolution Layers



Smaller than the original image

The number of channels is the number of filters

Subsequent layers look for combinations of features from previous layers!

The Whole CNN

Source Image

Convolution

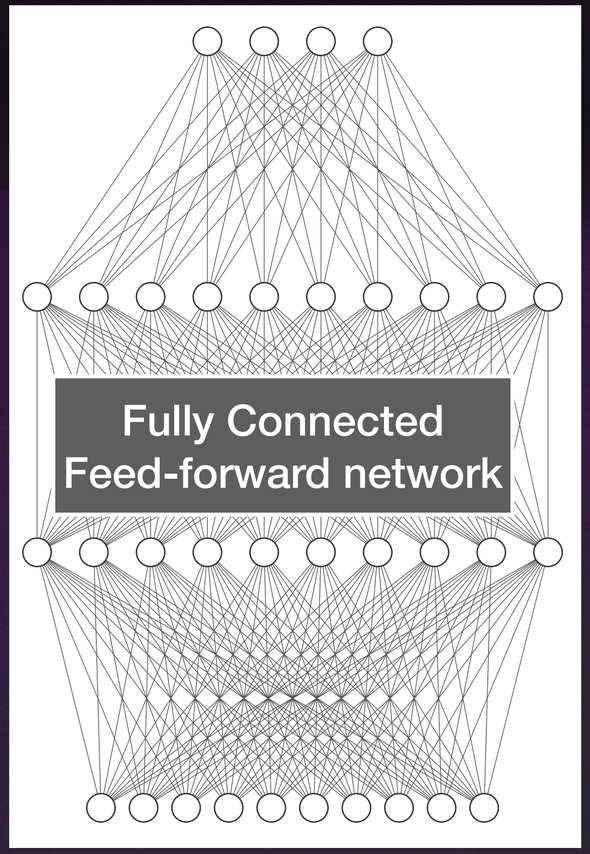
Max Pooling

Convolution

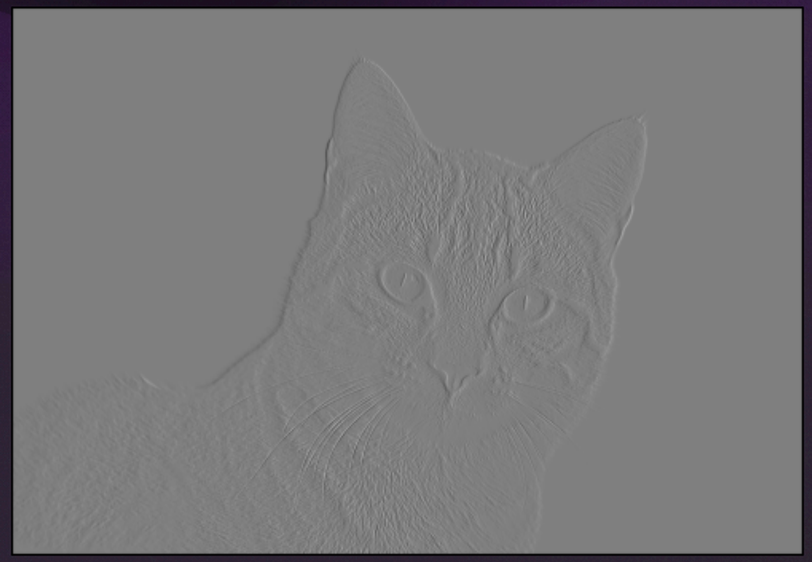
Max Pooling

Max Pooling

Flattened



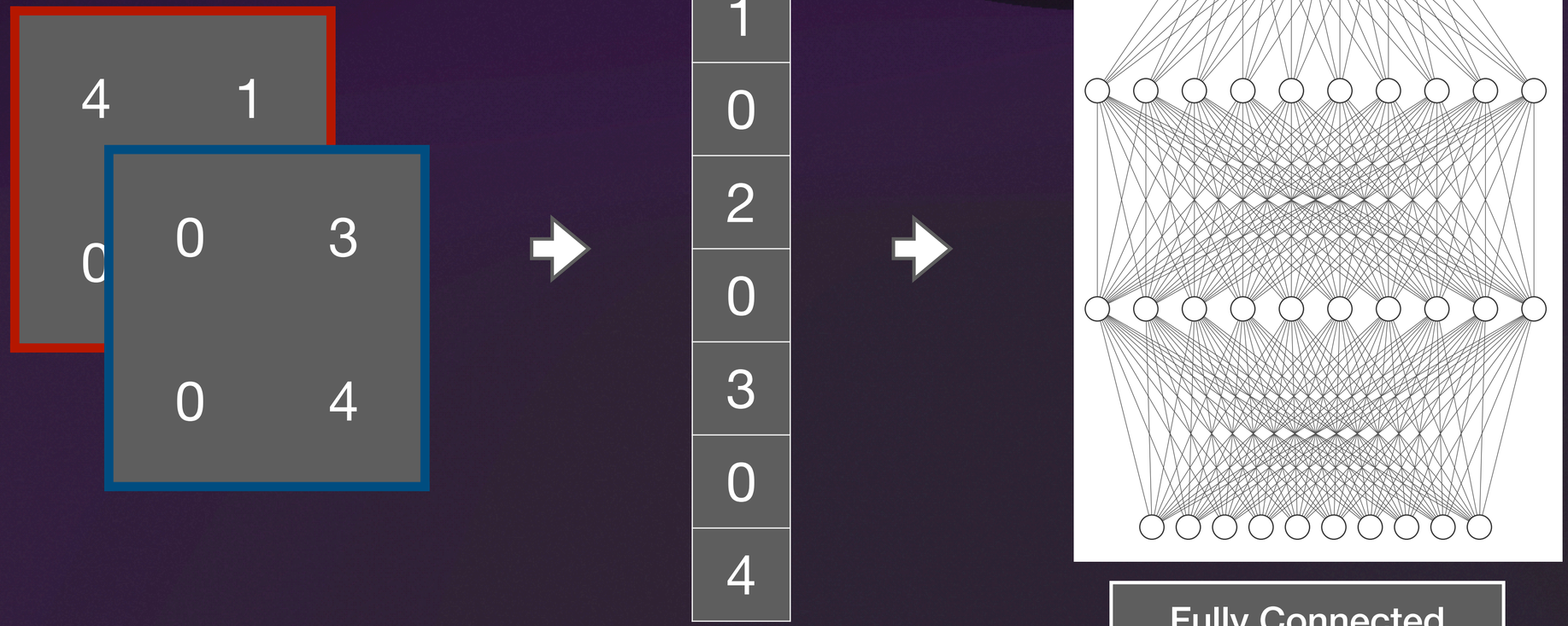
Cat
Dog
Llama
...



Cat!

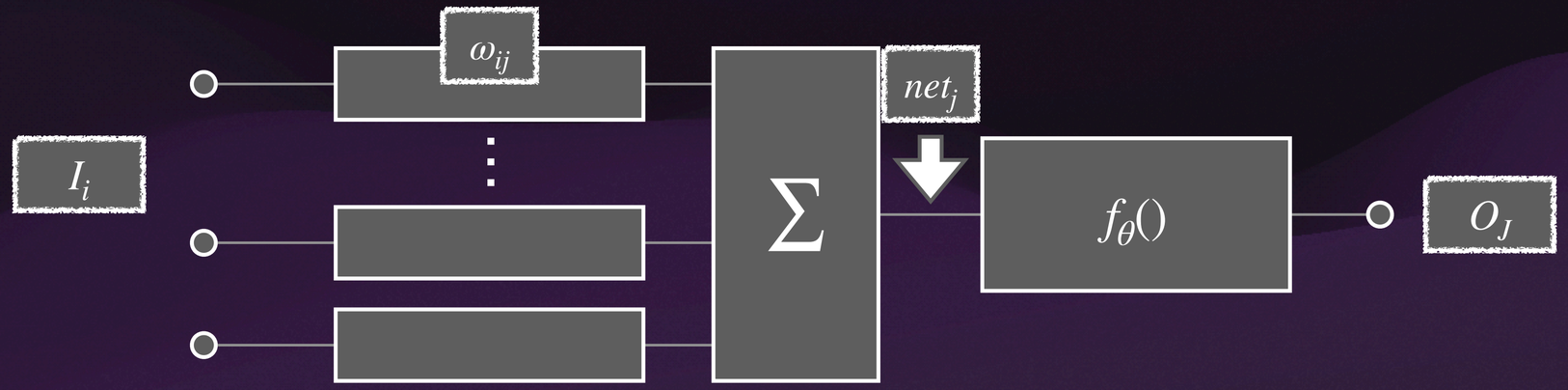
Cat image by Alvesgaspar, Wikimedia Commons

Flattening



Neurons

The McCulloch and Pitts Neuron (1943)



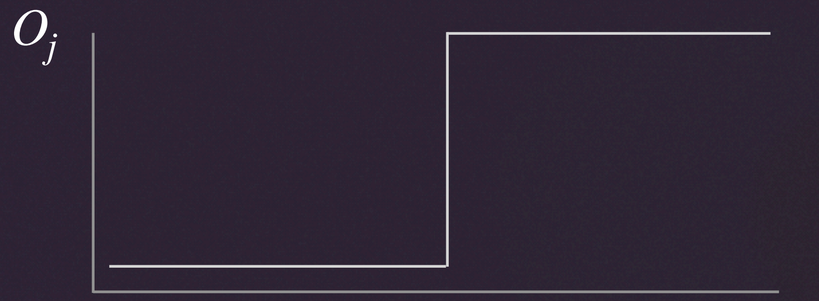
$$O_j = f_\theta(net_j)$$

$$net_j = \sum_i \omega_{ij} i_i$$

Where ω_{ij} is the weight, θ is the threshold, and:

$f_\theta(x) = 1$ if $x \geq \theta$

$f_\theta(x) = 0$ if $x < \theta$



Example step activation function $f_\theta(x)$

Single Input Neuron

NB: i_1 not Boolean in this example

$$net = i_1 \times w_1$$

$$o = 1, \text{ if } net \geq \theta$$

$$o = 0, \text{ if } net < \theta$$

For threshold θ, may rewrite as:

$$net = i_1 \times w_1 - \theta$$

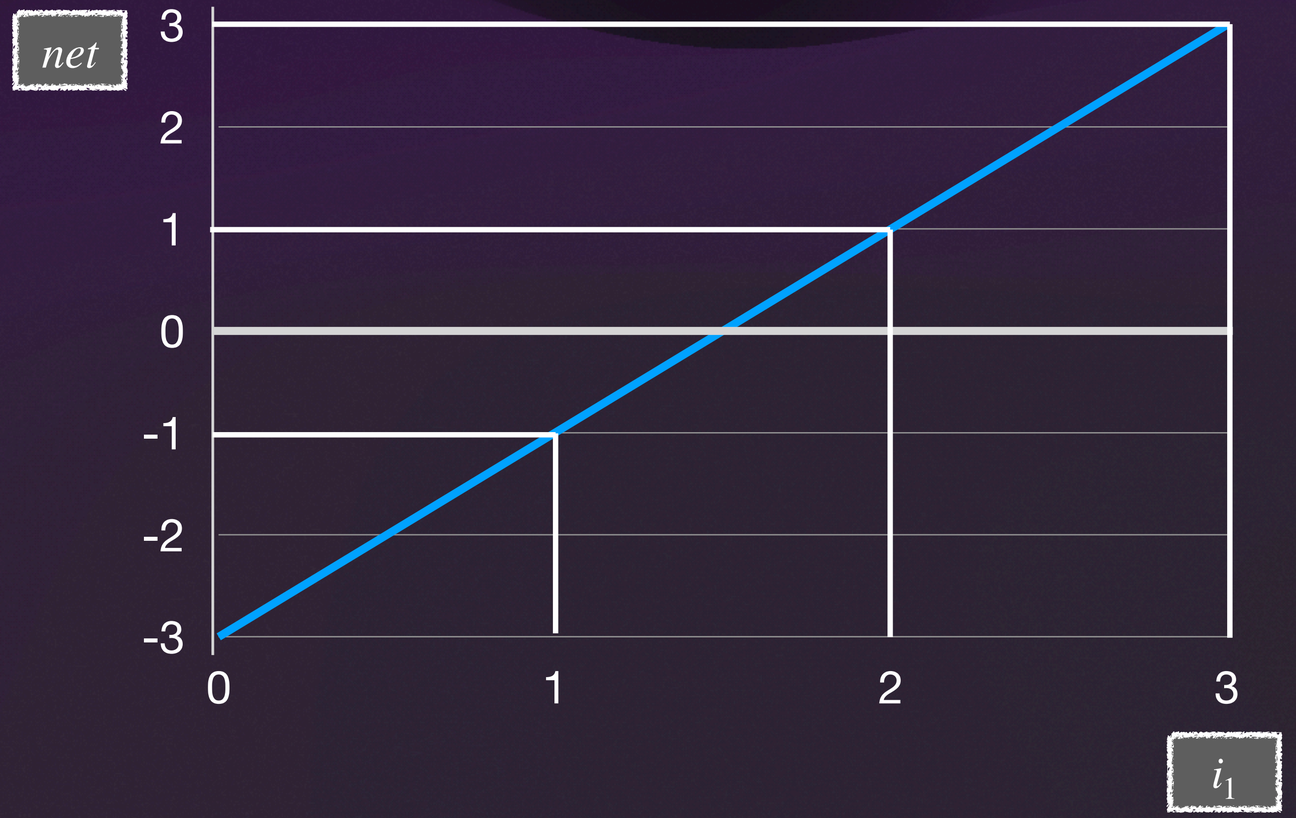
$$o = 1, \text{ if } net \geq 0$$

$$o = 0, \text{ if } net < 0$$

Analysis

- Assume $w_1 = 2$ and $\theta = 3$

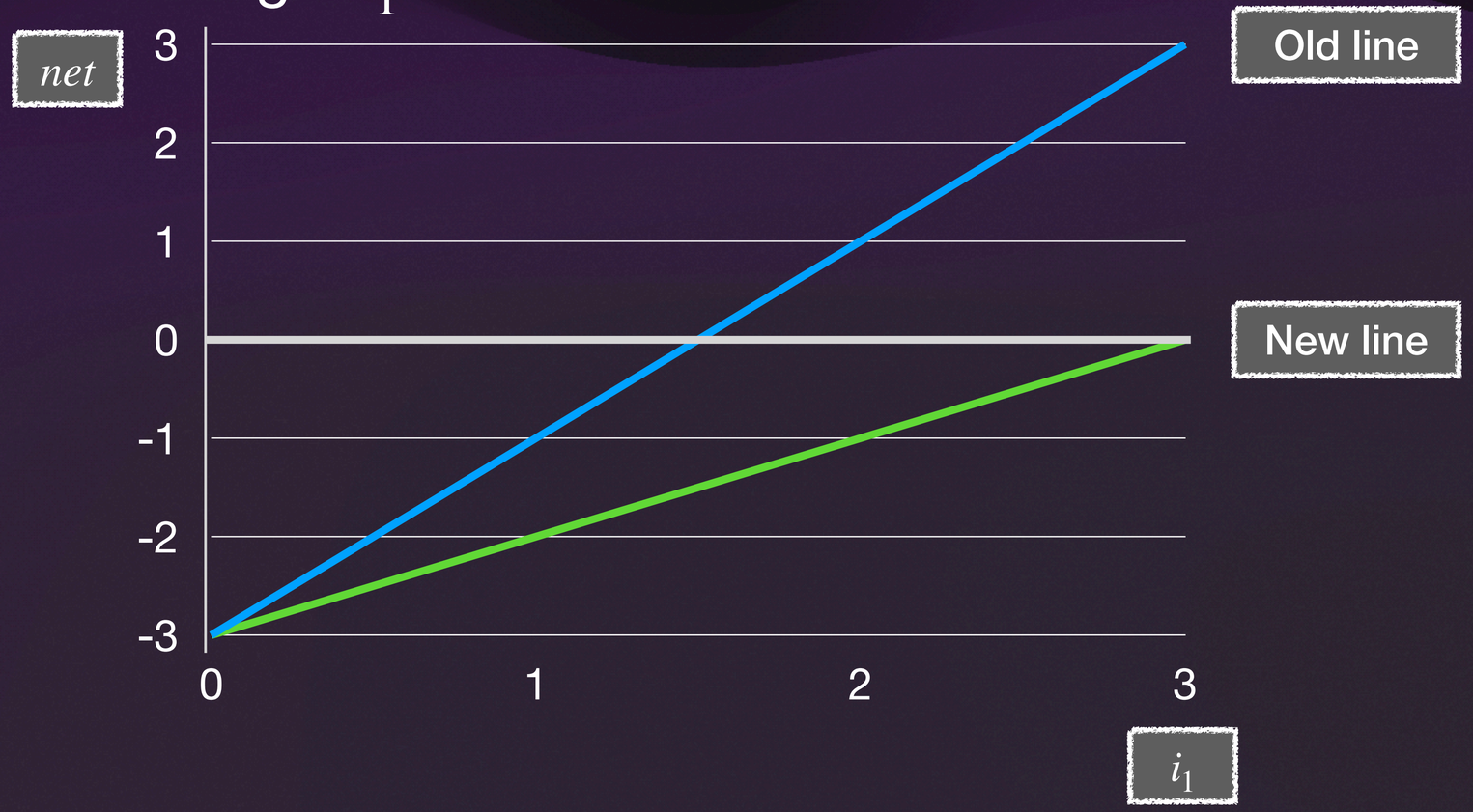
i_1	net
0	-3
1	-1
2	1
3	3



Analysis

- Changing w_1 changes gradient e.g. $w_1 = 1$

i_1	net
0	-3
1	-2
2	-1
3	0



Analysis

- Changing θ changes the net axis intercept e.g. $\theta = 1$

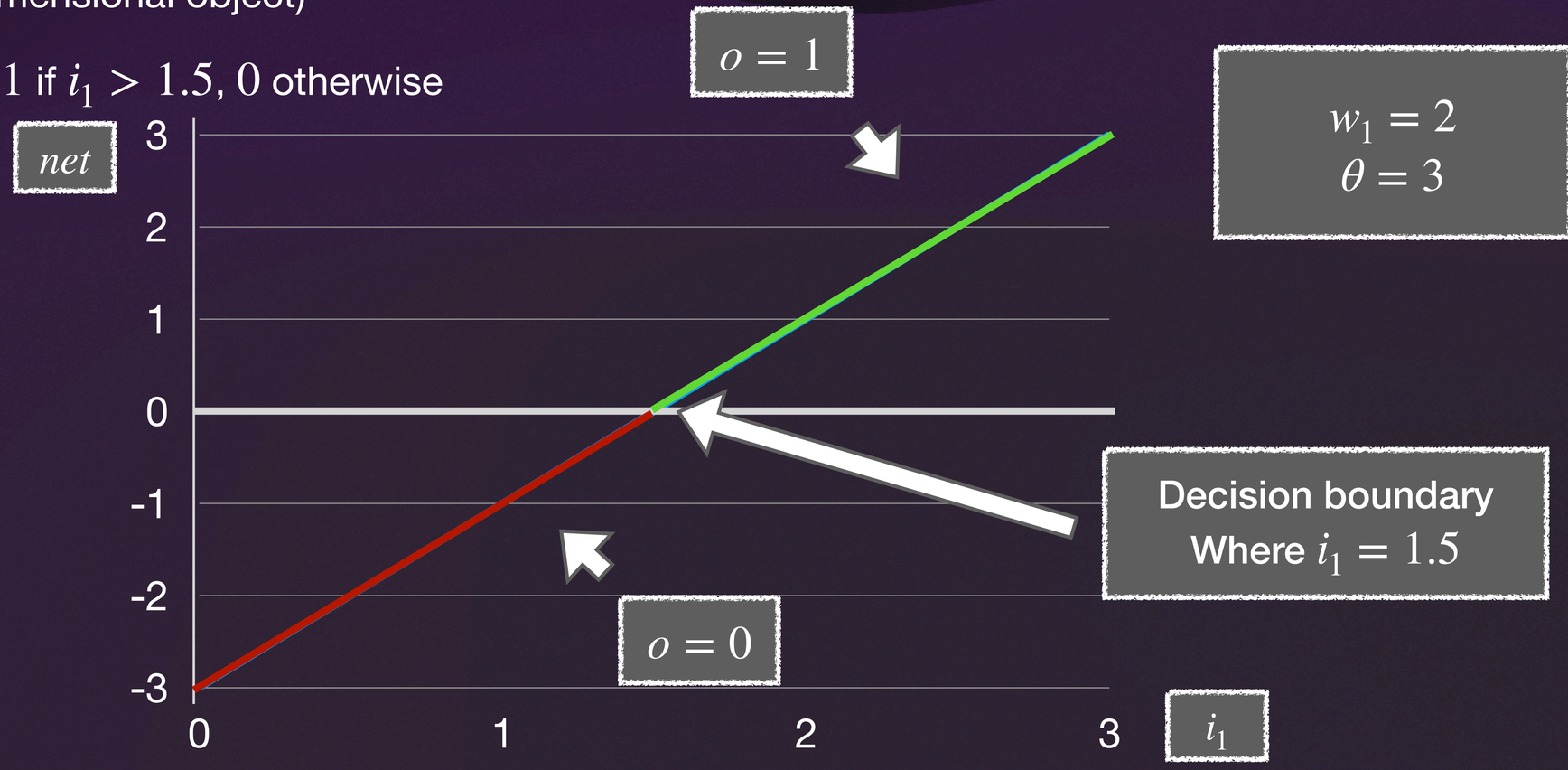
i_1	net
0	-1
1	1
2	3
3	5

- Old and New lines are parallel

Analysis

- Decision Boundary: Where straight line formed by net equations meets straight line formed by i_1 axis
  - It is a single point (0 dimensional object)
  - Original example: $o = 1$ if $i_1 > 1.5$, 0 otherwise

i_1	net
0	-3
1	-1
2	1
3	3



Limitations

- Consider again a neutron with one input. It take input x, multiplies it by a weight m, and tests if it exceeds threshold c.

$$z = mx - c$$

- It outputs 1 if $z > 0$, otherwise 0
- It is a one-dimensional object that divides a 2 dimensional space
- Although a neuron can divide a space, it can *only divide any space by a straight line*

Summary

- Convolution: Allows the extraction of spatial features in data
- Pooling: Shrink the data
- Flattening: Make the data convenient for a network
- Individual Neurons: Allow splitting a decision space in 2