# Mathematics of Deep Learning - II

Amir Hajian

July 2019

aisc


SCRIBD

# Read without limits

# Product **Overview**

The best of the **written word,** all in a **single app**, **personalized** for every reader.


**Books**

**Audiobooks**

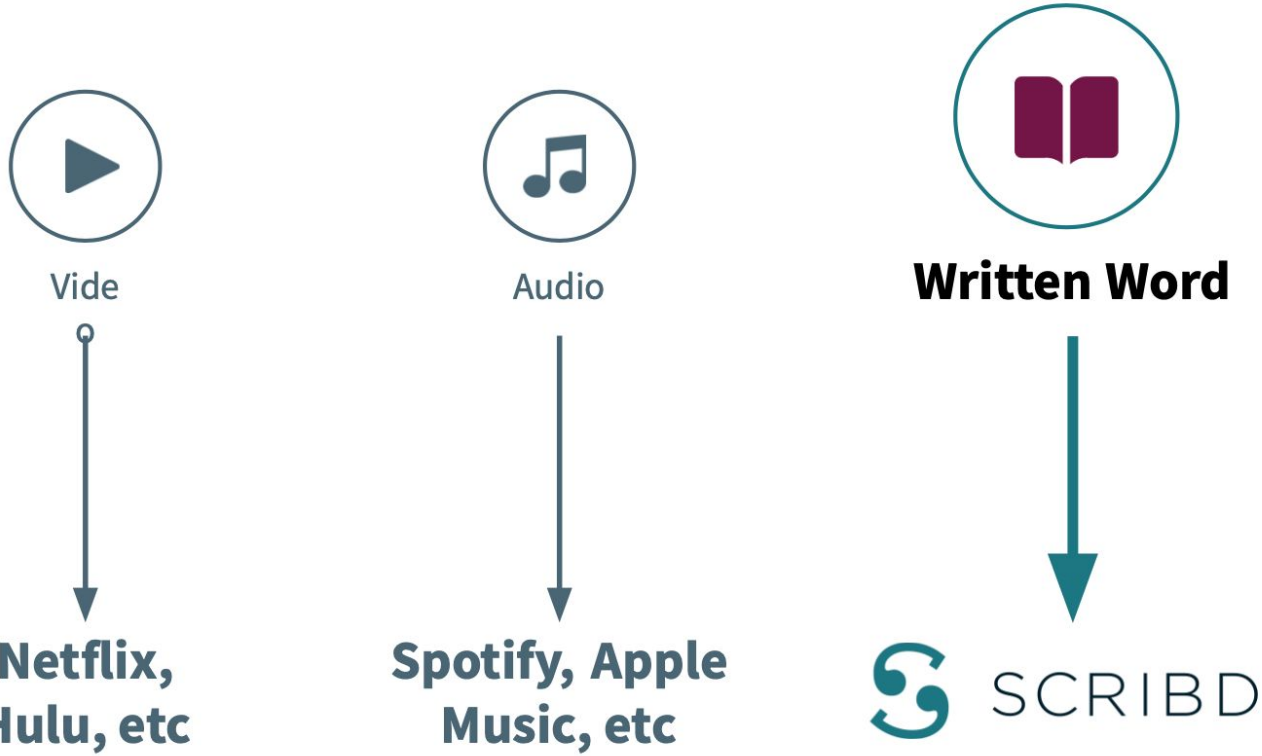
**Magazines**

**Sheet Music**

**News Articles**

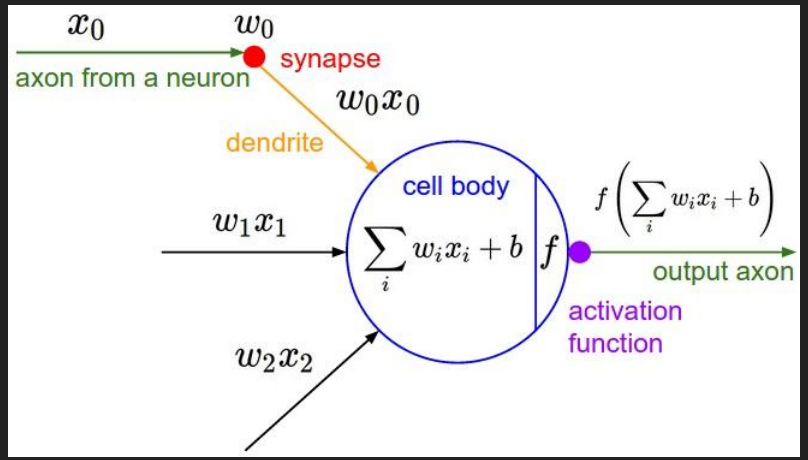
**Documents**

# The Written Word is **the Next Big Transformation** in Media



# Recap

- It is all about matrices, vectors and scalars.

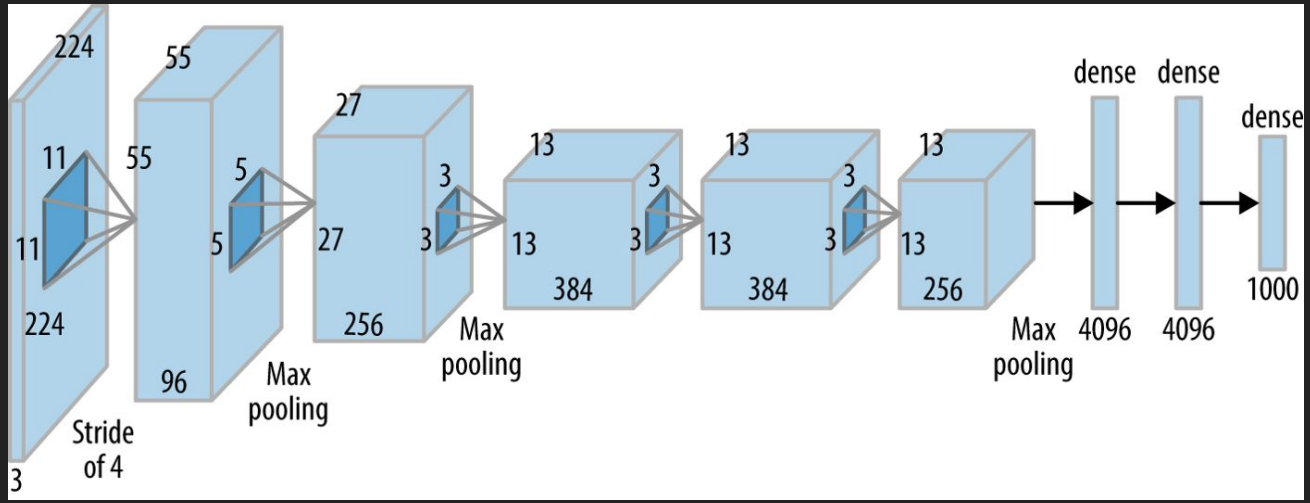


# Outline

- Convolution:
  - Why we need more than MLP?
  - What is convolution? What is a kernel?
  - What does it do?
  - 1D convolution
  - 2D convolution
  - Hands-on experiments with convolutions in python
  - Efficient convolution algorithms
  - ConvNets: a lightning fast introduction to their structure (conv layers, pooling, etc) and their applications.
- Dropout:
  - How we prevent overfitting in neural networks?
  - What is the math behind it?
  - How do you use it in PyTorch?

# ConvNets

- Today we'll learn the math behind convolutional neural networks



# What is a convolution?

- Formal definition: convolution is a mathematical operation on two functions (f and g) to obtain a third function that expresses how  the shape of one is modified by the other.

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau.$$

# What is a convolution?

- Formal definition: convolution is a mathematical operation on two functions (f and g) to obtain a third function that expresses how the shape of one is modified by the other.

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau.$$

- Practical definition:
  a. Take a function f
  b. Take a function g
  c. Shift g by a finite amount T
  d. Multiply f with the shifted g: f(t) g(t-T)
  e. sum over the whole range to get the value of f*g at point T
  f. Go to step c and repeat for all T values.

# What is a convolution?

- A simple example:
  - What is the result of convolving a delta function with a Gaussian kernel?

$$\delta(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ \text{undefined} & \text{at } x = 0 \end{cases}$$

$$\int_{-\infty}^{\infty} f(t)\delta(t-a)dt =$$

$$\int_{-\infty}^{\infty} \delta(x)dx = 1,$$

# What is a convolution?

- A simple example:
  - What is the result of convolving a delta function with a Gaussian?

$$\delta(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ \text{undefined} & \text{at } x = 0 \end{cases}$$

$$\int_{-\infty}^{\infty} f(t)\delta(t-a)dt =$$

$$\int_{-\infty}^{\infty} \delta(x)dx = 1,$$

$$f(a)\int_{-\infty}^{\infty} \delta(t-a)dt =$$

# What is a convolution?

- A simple example:
  - What is the result of convolving a delta function with a Gaussian kernel?

$$\delta(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ \text{undefined} & \text{at } x = 0 \end{cases}$$

$$\int_{-\infty}^{\infty} f(t)\delta(t-a)dt =$$

$$\int_{-\infty}^{\infty} \delta(x)dx = 1,$$

$$f(a)\int_{-\infty}^{\infty} \delta(t-a)dt =$$

$$f(a)$$

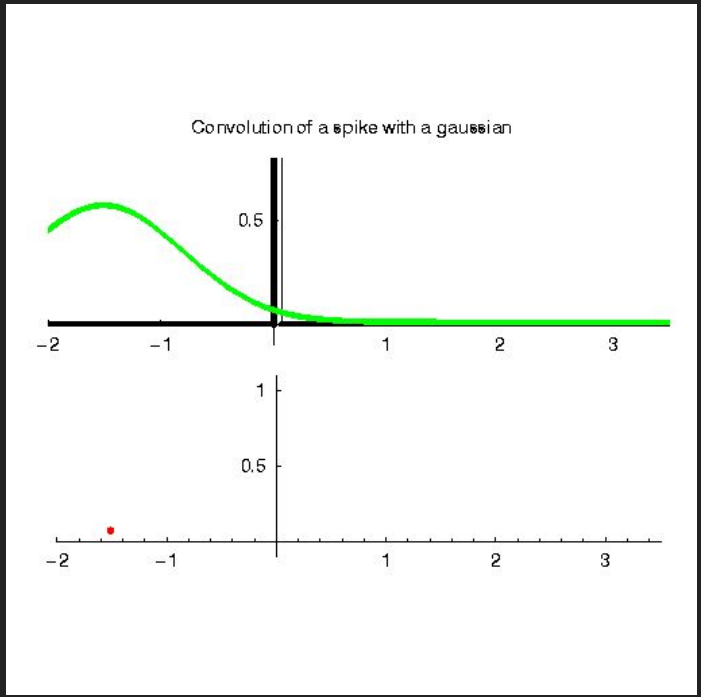
Read more

# What is a convolution?

- A simple example:
  - What is the result of convolving a delta function with a Gaussian kernel?

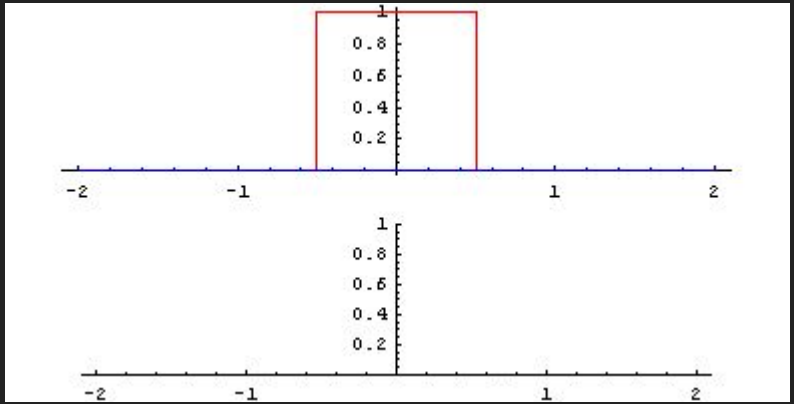
$$\delta(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ \text{undefined} & \text{at } x = 0 \end{cases}$$

$$\int_{-\infty}^{\infty} \delta(x)dx = 1,$$



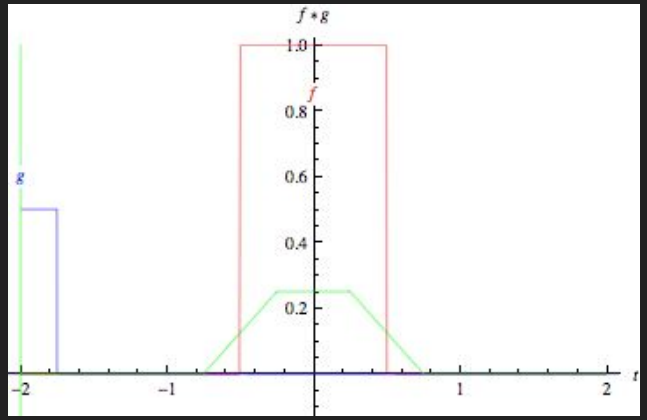
# What is a convolution?

- A visual example:



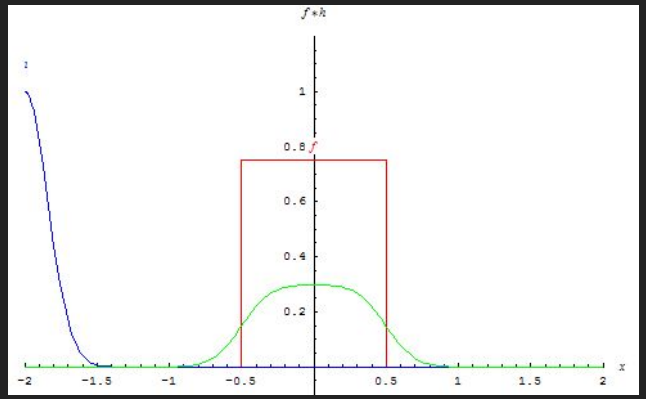
# What is a convolution?

- A visual example:



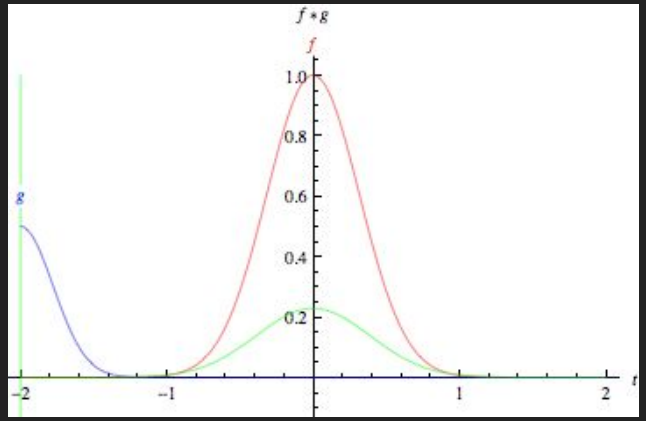
# What is a convolution?

- A visual example:



# What is a convolution?

- A visual example:



# What is a convolution?

- How to code it up?

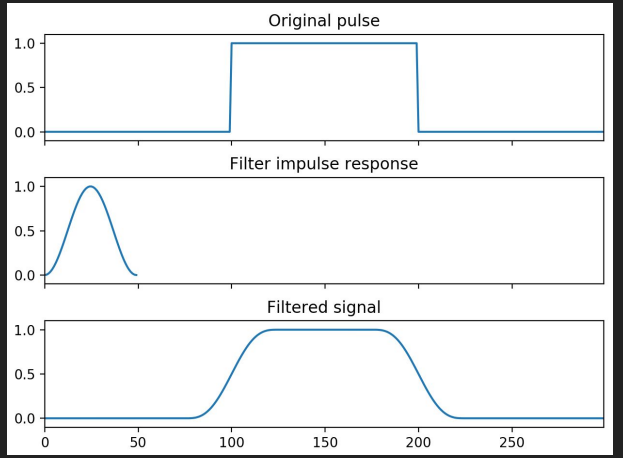
$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

- In Python:
  - `scipy.signal.convolve` for 1D conv
  - `scipy.signal.conv2d` for 2D

# Experiments: Hands-on

Experiment with convolutions in 1D by smoothing a top-hat function with a Hann function

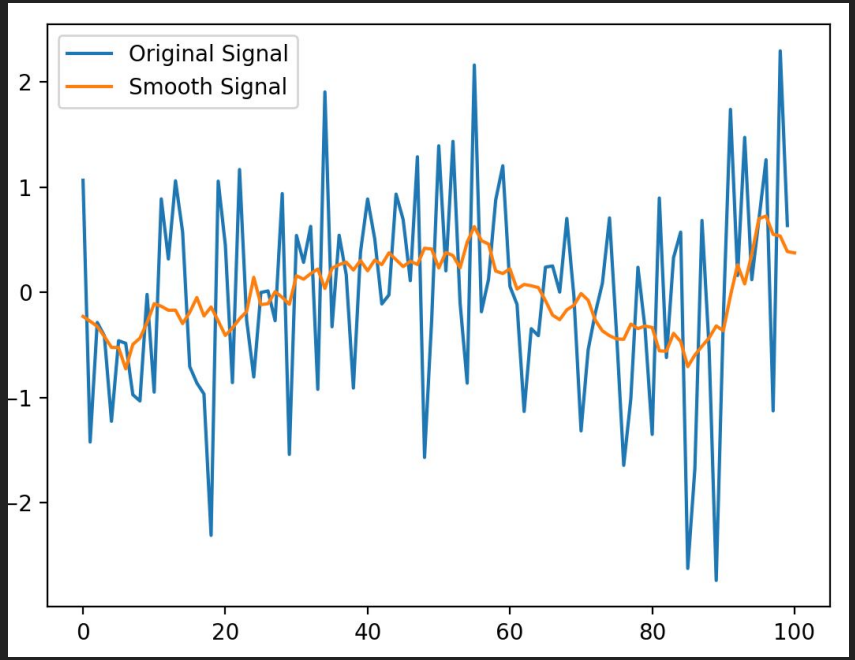
- Define a top-hat function that is non-zero in the range of [100:200]
- Define a Hann function between 0 and 50 - Hint: use scipy.signal.hann
- Apply the Hann function to the top-hat - Hint: use signal.convolve
- Plot the signal before and after smoothing to see the result.
- Discuss with your teammates to make sure you understand the results.




Repeat it with PyTorch Conv function at home.

# Application: smoothing/binning noisy functions

- Pick the kernel to be a top-hat function of length L
- Convolve a noisy function with the kernel
- Observe how the function is binned using this operation.



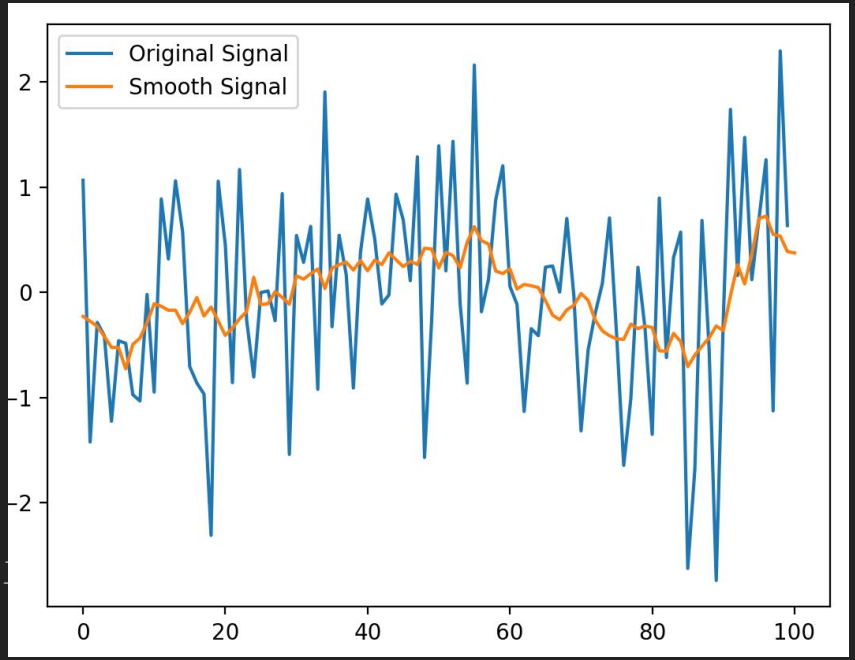

In PyTorch:

```
torch.conv1d(original_signal, kernel, padding=5)
```

# Application: smoothing/binning noisy functions

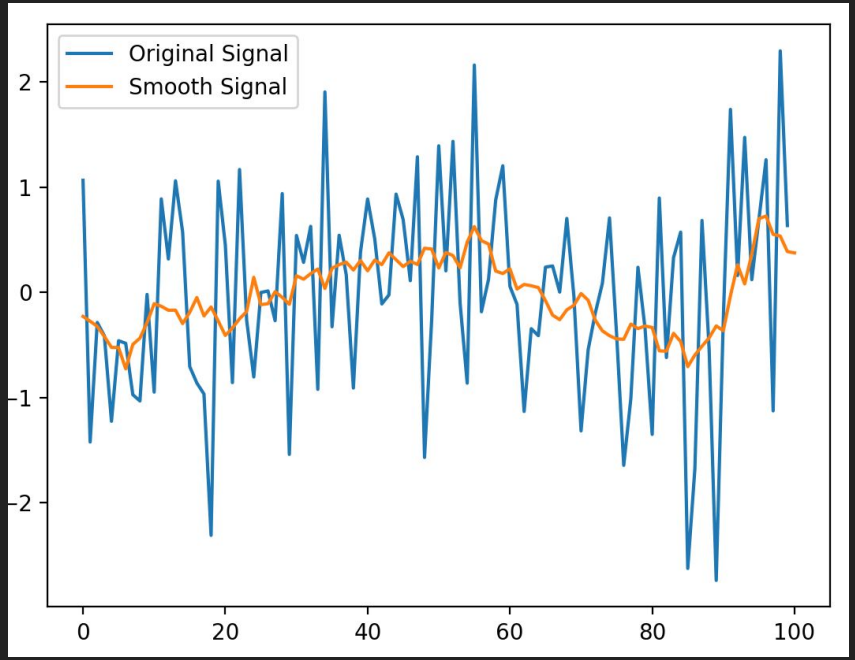
- Pick the kernel to be a top-hat function of length L
- Convolve a noisy function with the kernel
- Observe how the function is binned using this operation.

```
original_signal = torch.randn([1,1,100])

kernel = torch.ones([1,1,10])

smooth_signal = torch.conv1d(original_signal,
kernel, padding=5)/kernel.sum()
```



# Hands-On: 1D Convolution in PyTorch

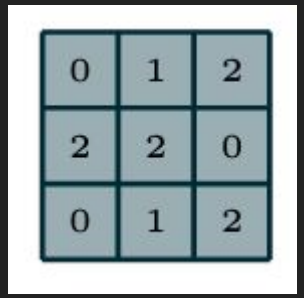
- Experiment with 1D conv in PyTorch by recreating this plot to bin a noisy function.
  - Get creative. Pick your own function.
  - Add noise to it.
  - Pick different kernels, experiment with the width and shape of the kernels.



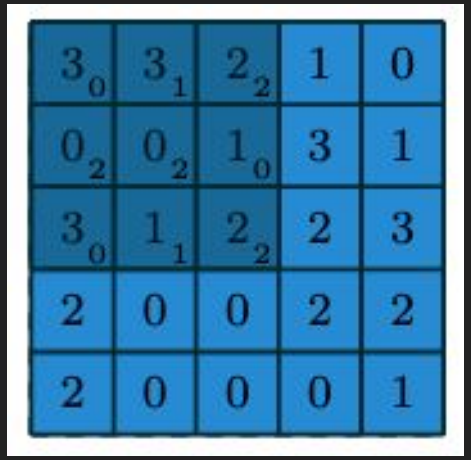

```
original_signal = torch.randn([1,1,100])

kernel = torch.ones([1,1,10])

smooth_signal =
torch.conv1d(original_signal, kernel,
padding=5)/kernel.sum()
```

# Convolutions in 2D: A step towards ConvNets

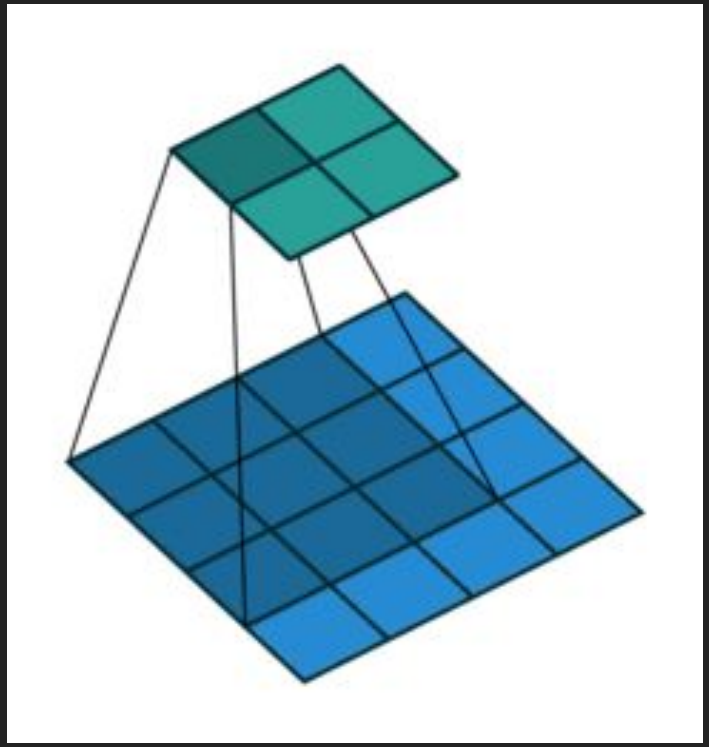
Kernel



# Convolutions in 2D: A step towards ConvNets



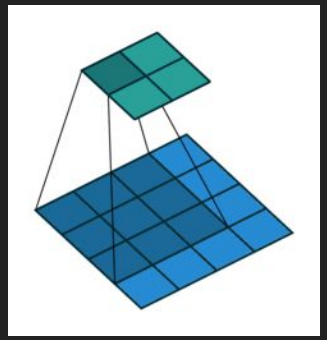

# Convolutions in 2D



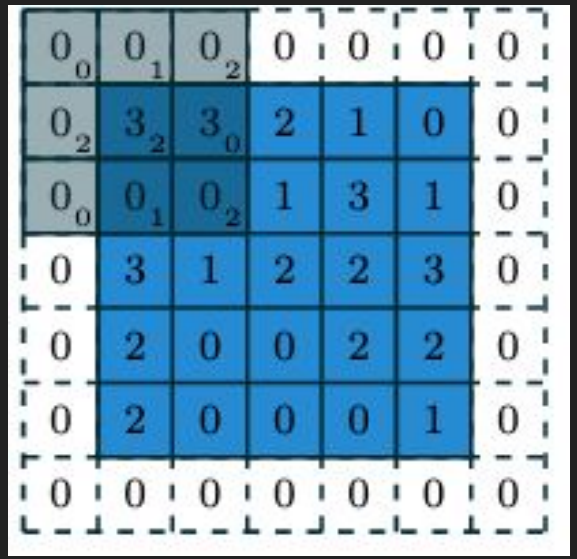
# Convolutions in 2D:

## It is actually a massive matrix multiplication

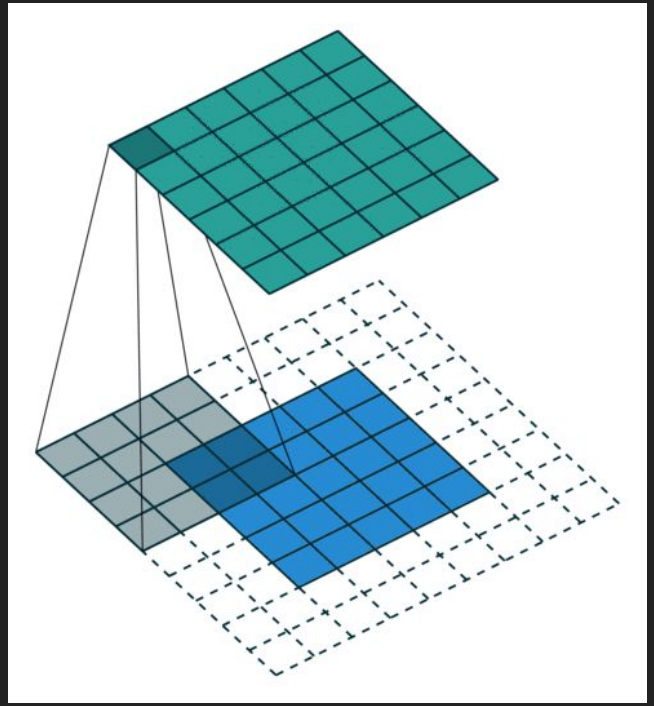
$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$



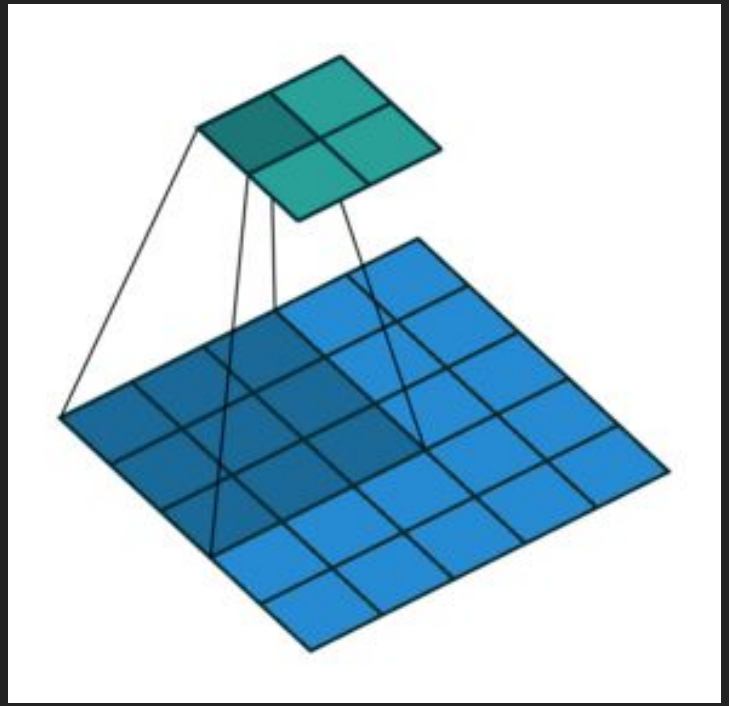
# Convolutions in 2D: Padding



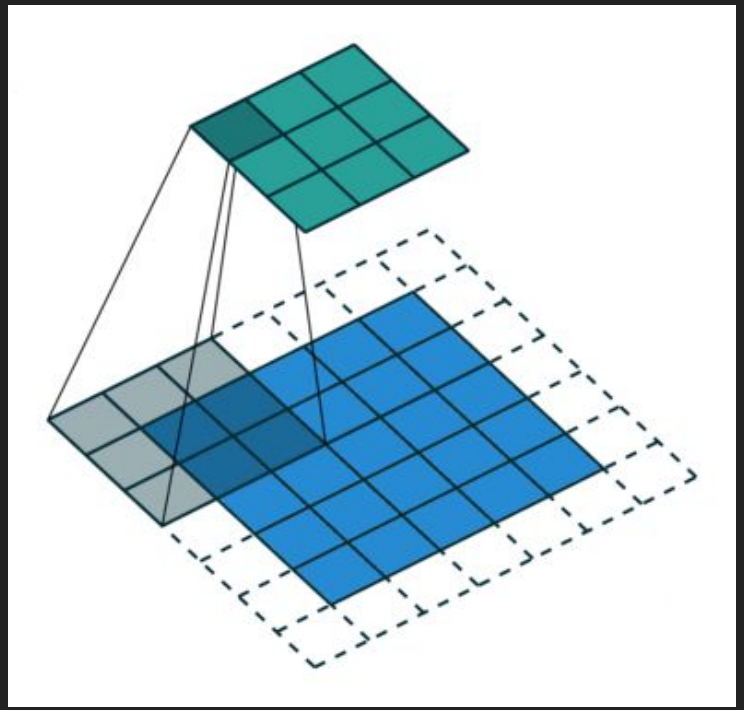
# Convolutions in 2D: Padding



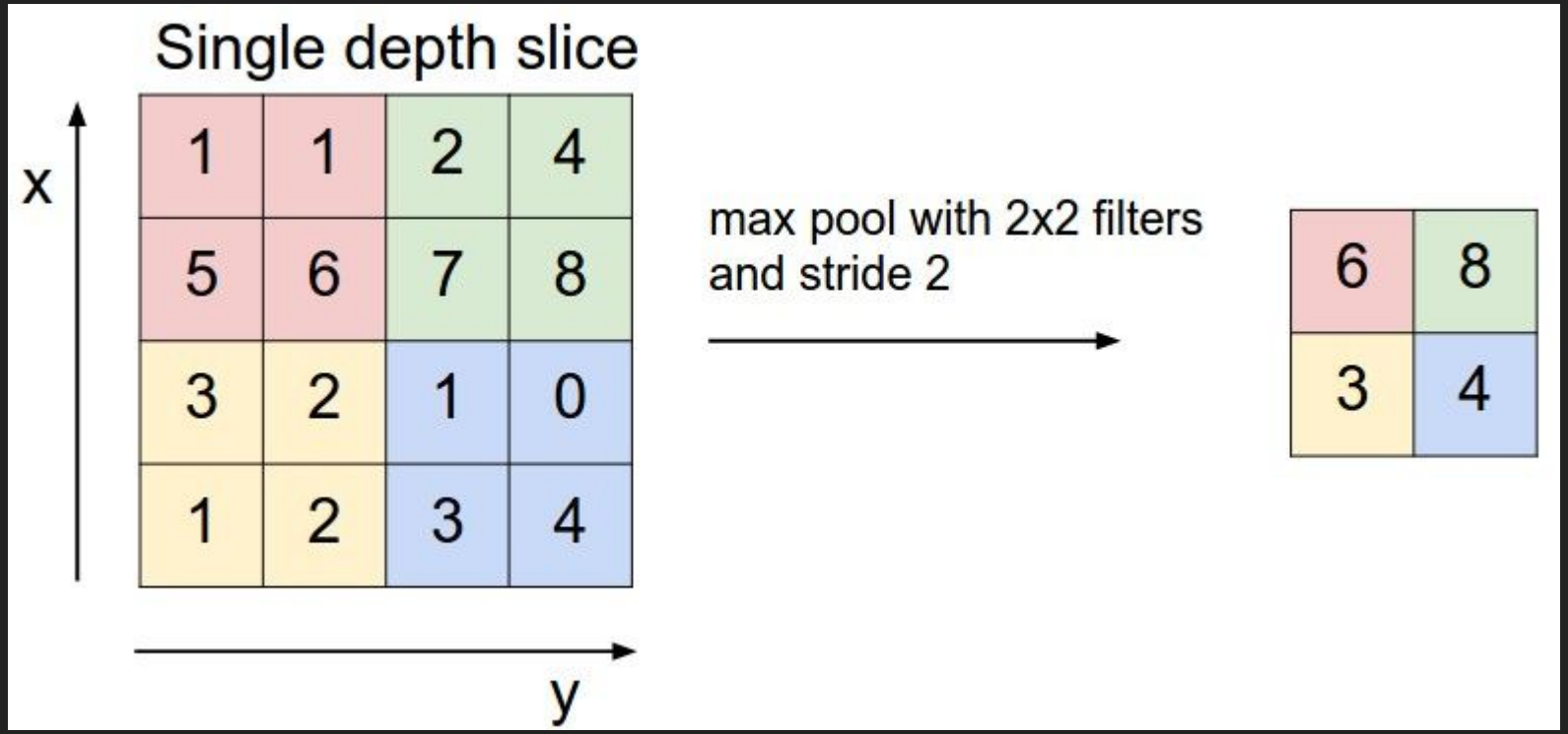
# Convolutions in 2D: Strides



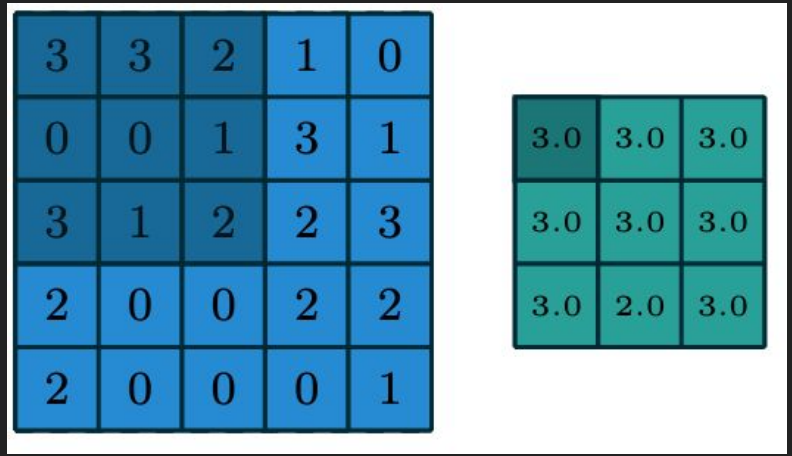
# Convolutions in 2D: Padding & Strides



# Convolutions in 2D: Pooling



# Convolutions in 2D: Pooling



# Pooling in PyTorch

```
Examples::

        >>> # pool of size=3, stride=2

        >>> m = nn.MaxPool1d(3, stride=2)

        >>> input = torch.randn(1,1,20)

        >>> output = m(input)
```

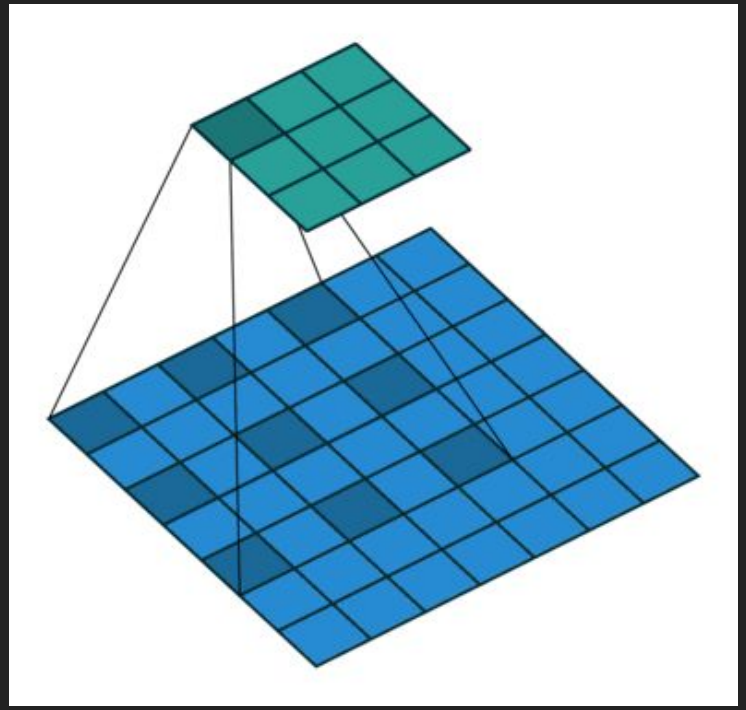
# Pooling in PyTorch

```
Examples::

        >>> # pool of size=3, stride=2

        >>> m = nn.MaxPool1d(3, stride=2)

        >>> input = torch.randn(1,1,20)

tensor([[[ 0.8724, -1.1383,  0.0582,

0.3185,  1.9933, -0.6547,

-0.8365, -0.0783,  0.5539, -0.5272]]])

         >>> output = m(input)
```

# Pooling in PyTorch

```
Examples::

        >>> # pool of size=3, stride=2

        >>> m = nn.MaxPool1d(3, stride=2)

        >>> input = torch.randn(1,1,20)

tensor([[[ 0.8724, -1.1383,  0.0582,

0.3185,  1.9933, -0.6547,

-0.8365, -0.0783,  0.5539, -0.5272]]])

         >>> output = m(input)

tensor([[[0.8724, 1.9933, 1.9933, 0.5539]]])
```

# Convolutions in 2D: Dilated Convolution

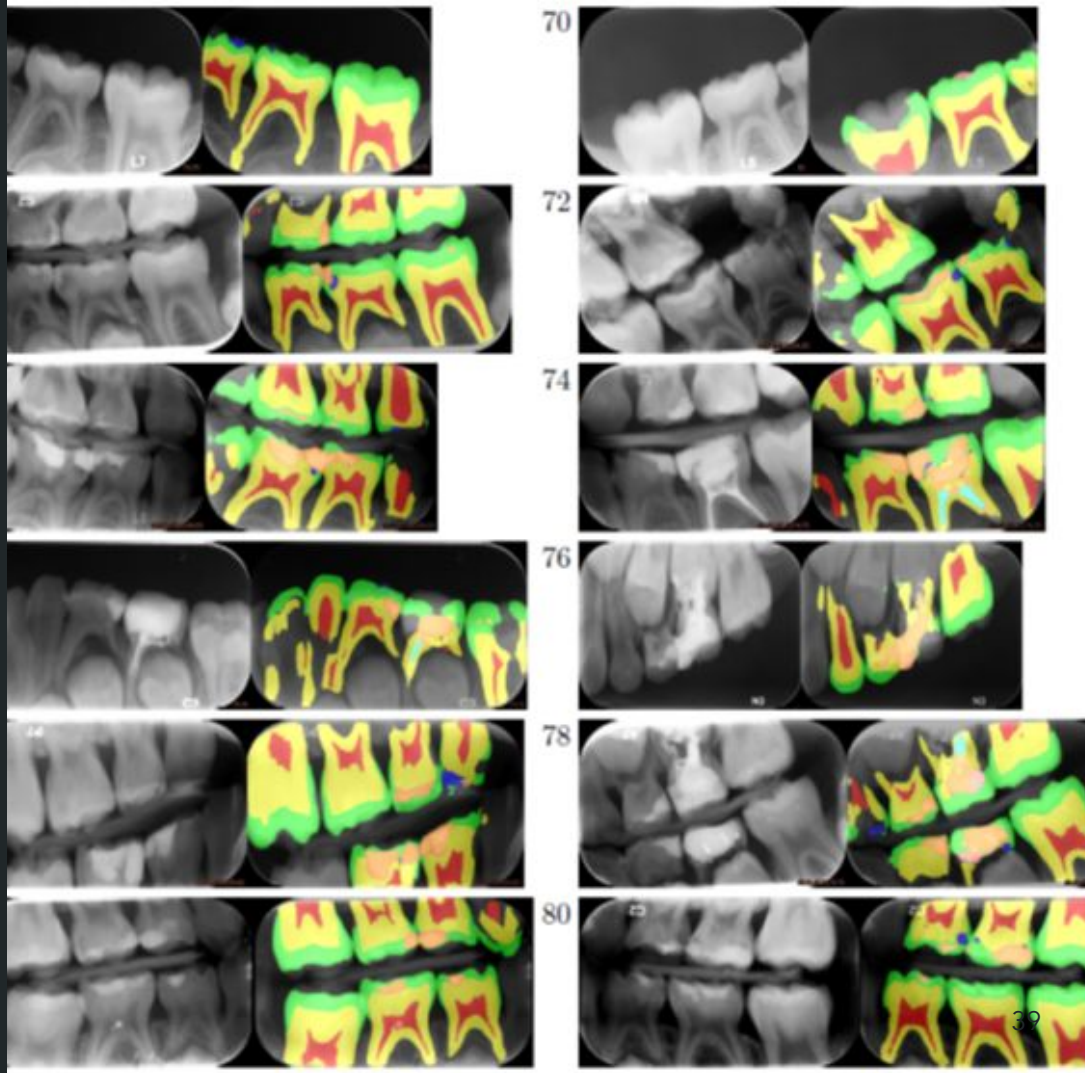


$$(F *_l k)(\mathbf{p}) = \sum_{\mathbf{s}+l\mathbf{t}=\mathbf{p}} F(\mathbf{s})\, k(\mathbf{t}).$$

Read more

# What is it good for?

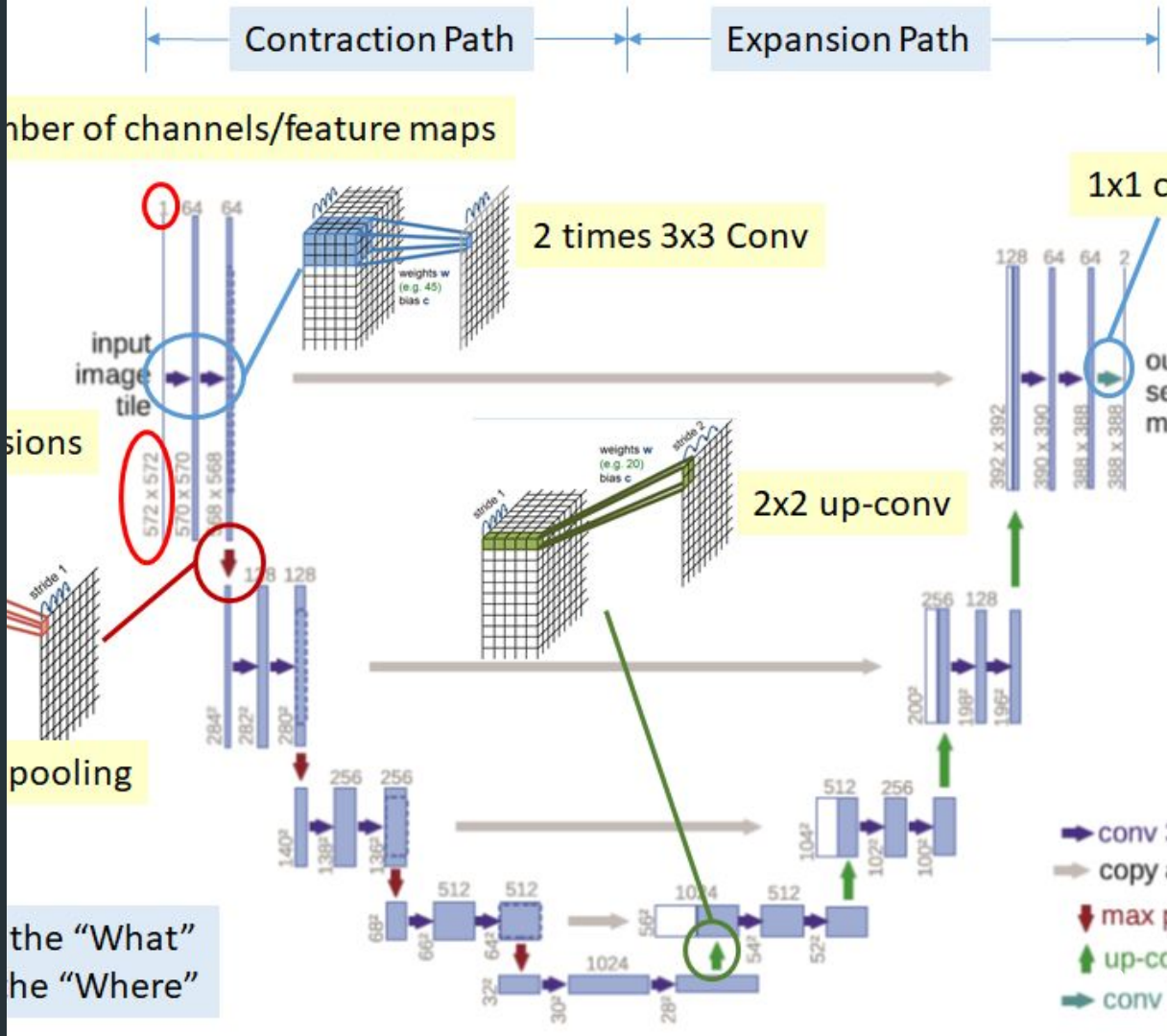
# Image Segmentation




reference

# How is it done?

U-Net: Based on Convolutional Neural Networks



# What does convolution really do?

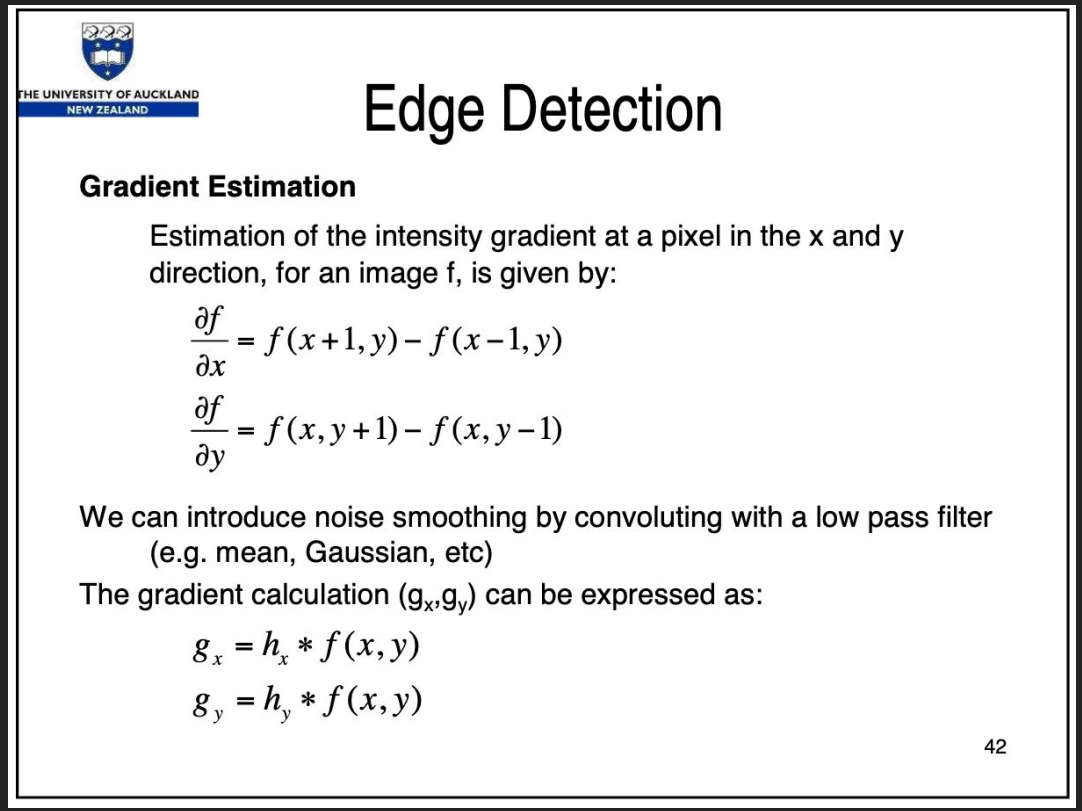
Let's look at a simple example: edge detection

Edge detection is a major application for convolution.

What is an edge:

- A location in the image where is a sudden change in the intensity/colour of pixels.
- A transition between objects or object and background.
- From a human visual perception perspective it attracts attention.

# What does convolution really do?




Read more

# Sobel Operator

The Sobel filter is used for edge detection.
Calculates the gradient of image intensity at each pixel within the image.

Finds the direction of the largest increase from light to dark and the rate of change in that direction.

The result shows how abruptly or smoothly the image changes at each pixel, and therefore how likely it is that that pixel represents an edge.
It also shows how that edge is likely to be oriented.

The result of applying the filter to a pixel in a region of constant intensity is a zero vector.

The result of applying it to a pixel on an edge is a vector that points across the edge from darker to brighter values.

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

# Sobel Filter

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

# Sobel Filter

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

# Sobel Filter

Combined

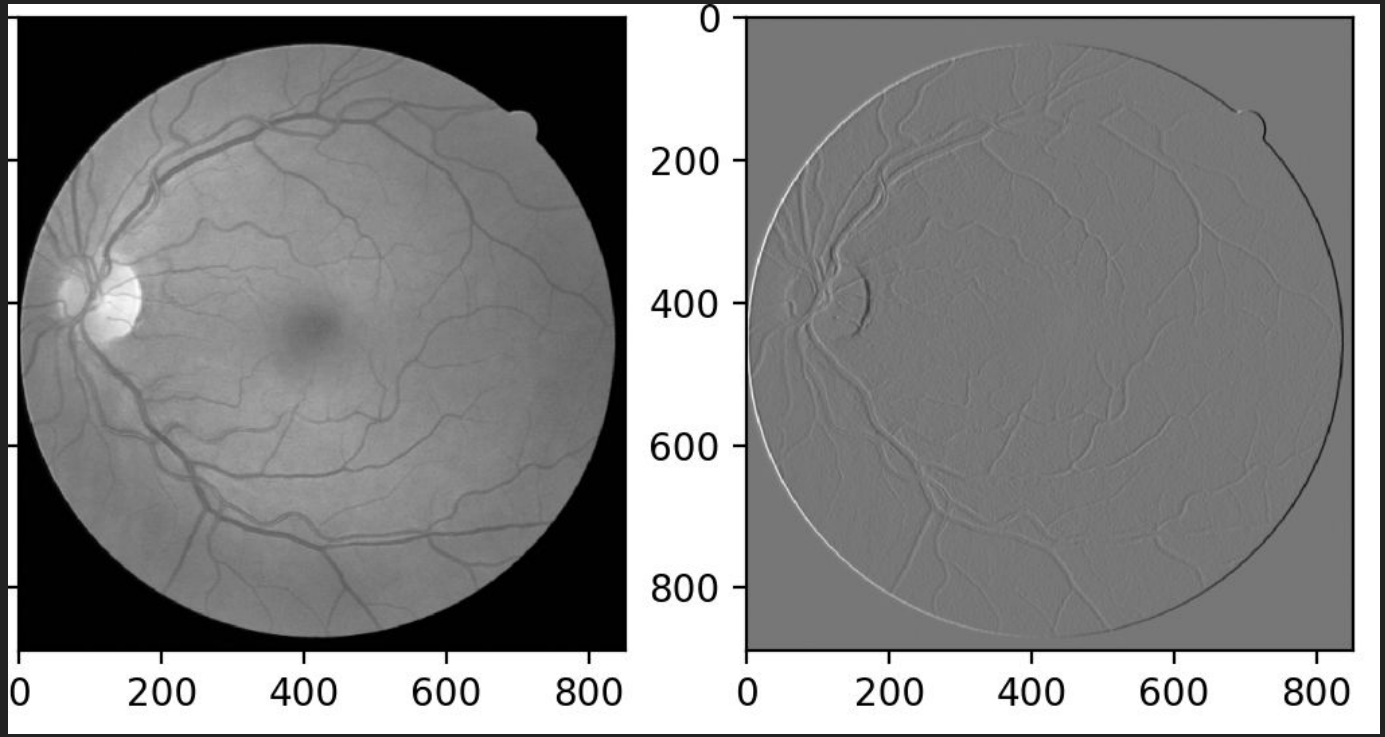
# Scharr Operator

$$\begin{bmatrix} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{bmatrix} \qquad \begin{bmatrix} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$$

# 2D Convolution: Detect Edges with Sobel Operator

Do it yourself:

```
from scipy import ndimage

ndimage.sobel(image)
```



# 2D Convolution: Detect Edges with Sobel Operator

Let's really do it ourselves:

```
sobel_x =
np.array([[ -1, 0,  +1],
          [-2, 0, +2],
           [ -1, 0,  +1]])
```

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

# 2D Convolution: Detect Edges with Sobel Operator

Let's really do it ourselves:

```
sobel_x =
np.array([[ -1, 0,  +1],
          [-2, 0, +2],
           [ -1, 0,  +1]])
```

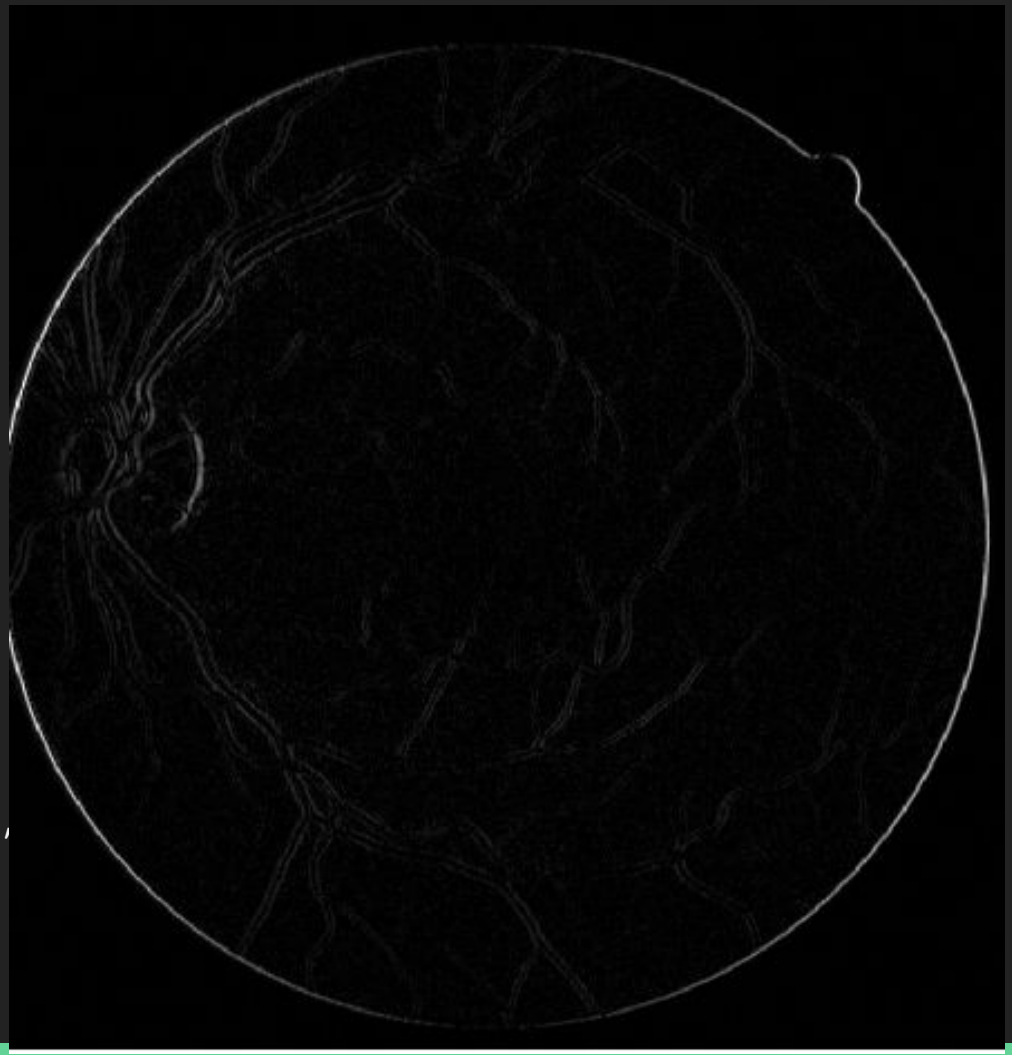
```
from scipy import signal
sobel = signal.convolve2d(image, sobel_x, boundary='symm', mode='same')
```

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

# 2D Convolution: Detect

Let's really do it ourselves:

```
sobel_x =
np.array([[ -1, 0,  +1],
          [-2, 0, +2],
           [ -1, 0,  +1]])


from scipy import signal

after_sobel = signal.convolve2d(image,
sobel_x)
```



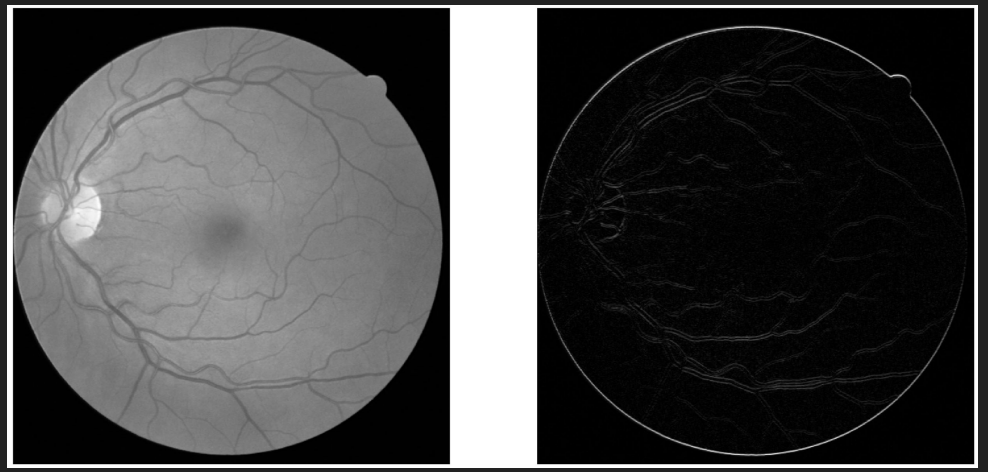
# Experiments: Hands-on time

Experiment with convolutions in 2D to detect edges in an image

1) Read the image and convert it to grey.
2) Define the kernel
3) Apply the kernel to the image using `scipy.signal.convolve2d`
4) Plot the results

Try Sobel Kernel as well as Scharr Kernel.

See the difference in the results?

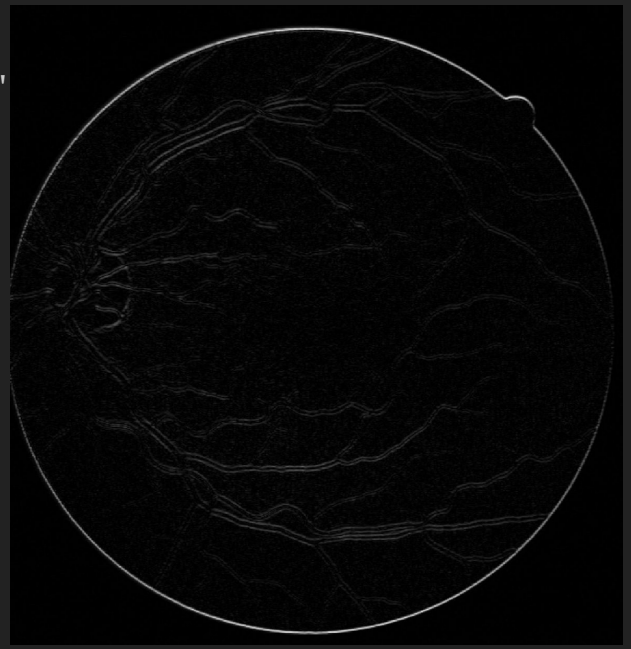


Note:

```
scharr = np.array([[ -3-3j, 0-10j,  +3 -3j],
                   [-10+0j, 0+ 0j, +10 +0j],
                   [ -3+3j, 0+10j,  +3 +3j]])
      # Gx + j*Gy
```

# 2D Convolution: Detect Edges with Sobel Operator

```
import imageio
from scipy import signal
from scipy import misc
image = imageio.imread("/Users/amirh/Downloads/Veins.png", as_gray=True)
sobel_y = np.array([[ -1, -2,  -1],
[0, 0, 0],
[ 1, 2,  1]])
sobel = signal.convolve2d(image, sobel_y, boundary='symm', mode='same'

import matplotlib.pyplot as plt
fig, (ax_orig, ax_mag) = plt.subplots(1, 2)
ax_orig.imshow(image, cmap='gray')
ax_orig.set_title('Original')
ax_orig.set_axis_off()
ax_mag.imshow(np.absolute(sobel_y), cmap='gray')
ax_mag.set_title('Sobel Applied')
ax_mag.set_axis_off()
fig.show()
```



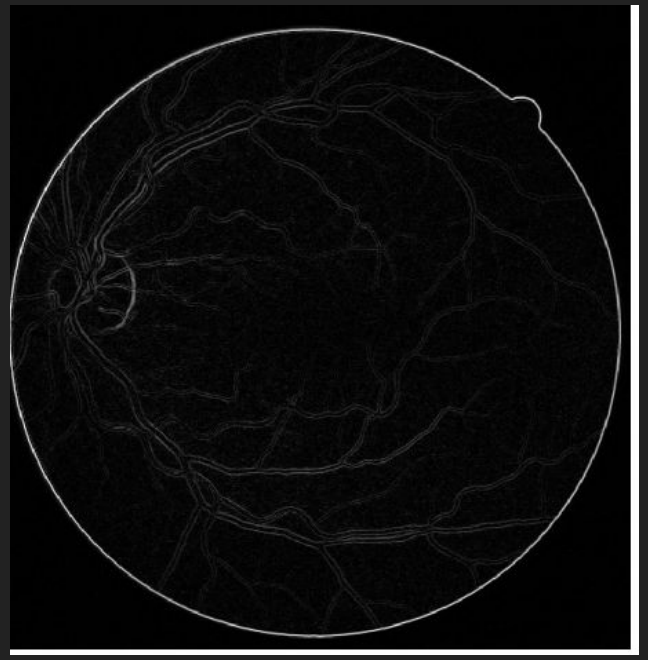
Exercise: 1) find edges in the x-direction using

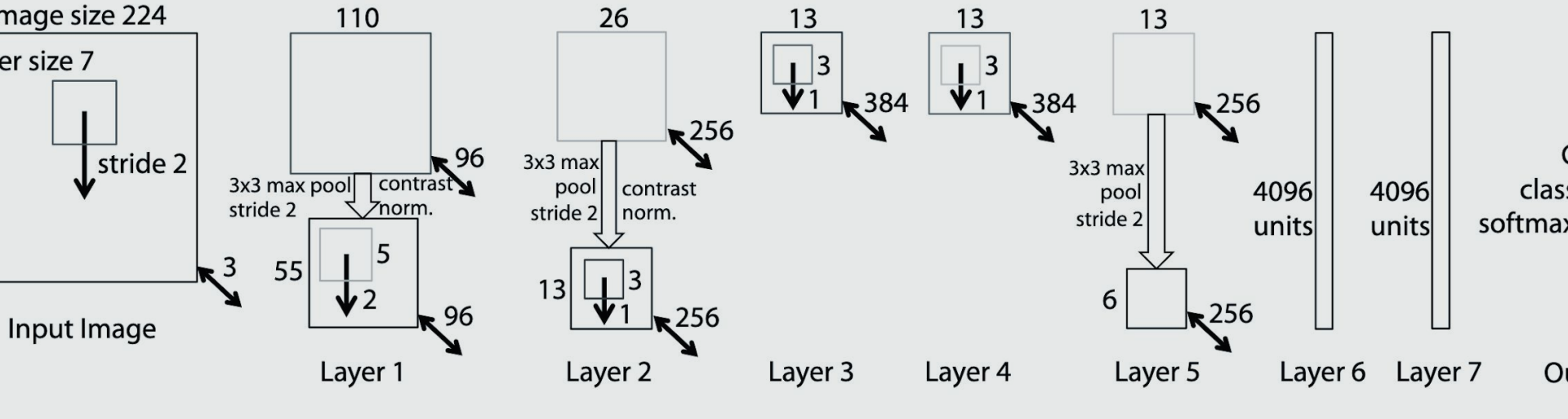
```
sobel_x = np.array([[ -1, 0,  +1],
[-2, 0, +2],
[ -1, 0,  +1]])
```

Exercise: 2) combine x and y results to get a final result

# 2D Convolution: Detect Edges with Scharr Operator

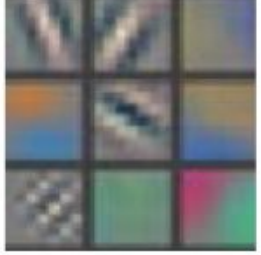
```
import imageio
from scipy import signal
from scipy import misc
image = imageio.imread("/Users/amirh/Downloads/Veins.png", as_gray=True)
scharr = np.array([[ -3-3j, 0-10j,  +3 -3j],
[-10+0j, 0+ 0j, +10 +0j],
[ -3+3j, 0+10j,  +3 +3j]]) # Gx + j*Gy
grad = signal.convolve2d(image, scharr, boundary='symm', mode='same')

import matplotlib.pyplot as plt
fig, (ax_orig, ax_mag) = plt.subplots(1, 2)
ax_orig.imshow(image, cmap='gray')
ax_orig.set_title('Original')
ax_orig.set_axis_off()
ax_mag.imshow(np.absolute(grad), cmap='gray')
ax_mag.set_title('Gradient magnitude')
ax_mag.set_axis_off()
fig.show()
```



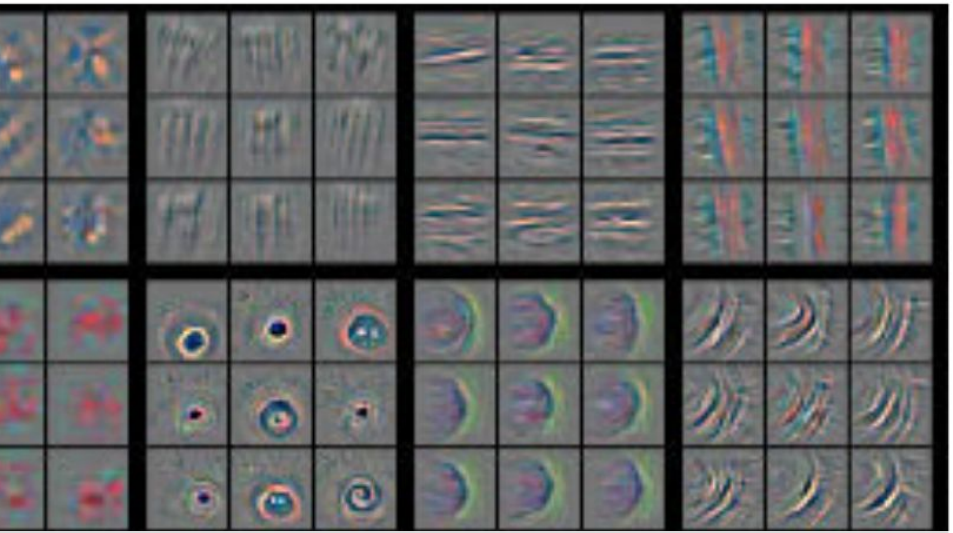

mage size 224
er size 7
stride 2
3
Input Image
110
96
3x3 max pool
stride 2
contrast
norm.
55
5
2
96
Layer 1
26
256
3x3 max
pool
stride 2
contrast
norm.
13
3
1
256
Layer 2
13
3
1
384
Layer 3
13
3
1
384
Layer 4
13
256
3x3 max
pool
stride 2
6
256
Layer 5
4096
units
Layer 6
4096
units
Layer 7
clas
softmax

# ConvNets

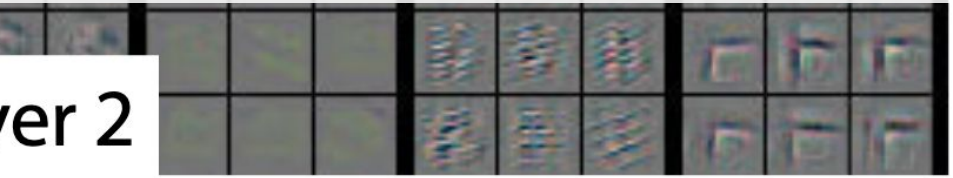
# Understanding ConvNets



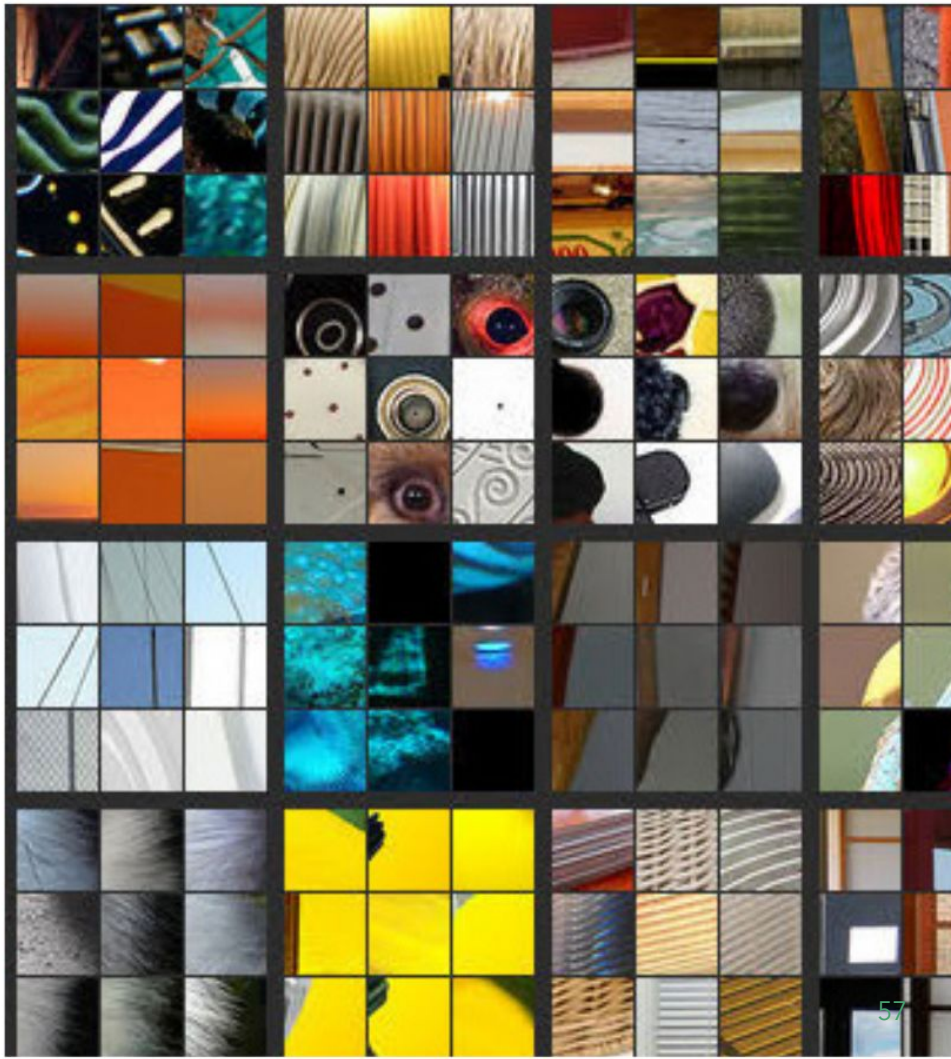
Layer 1



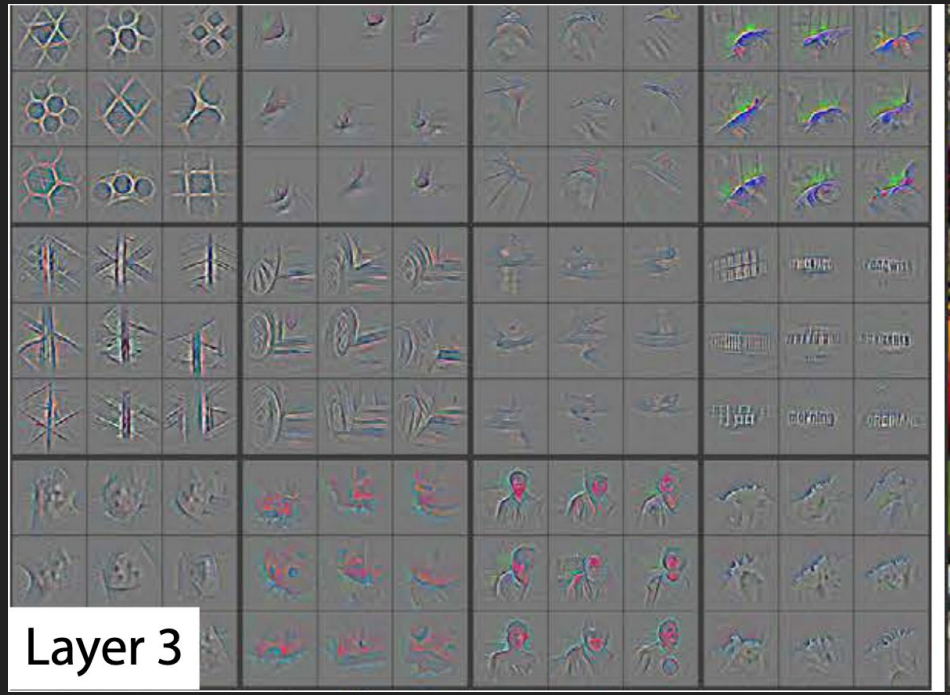
Understanding
ConvNets



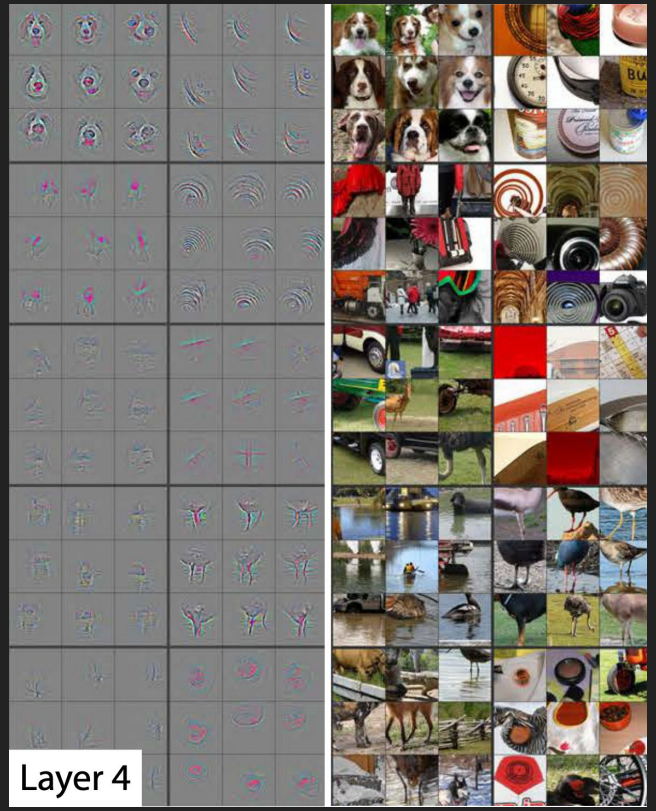
ver 2



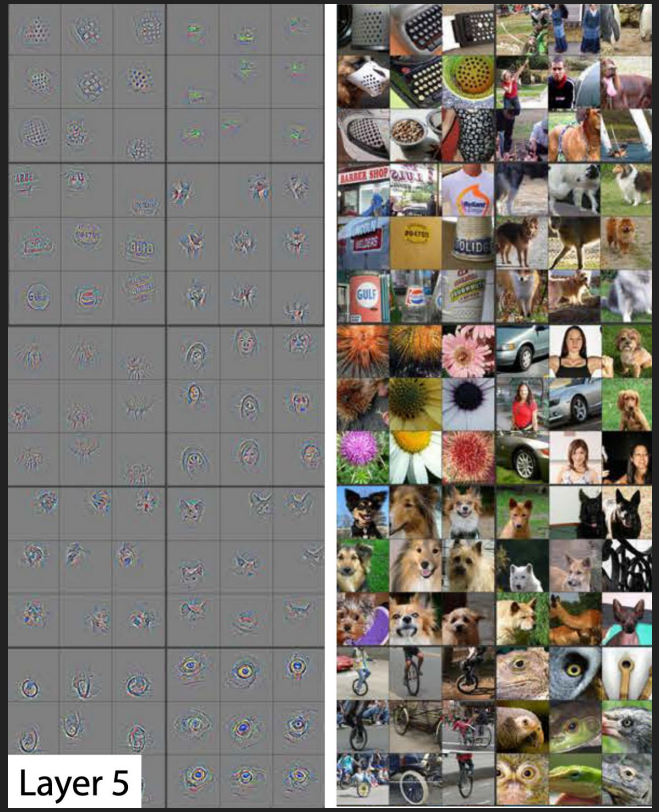
# Understanding ConvNets




# Understanding ConvNets

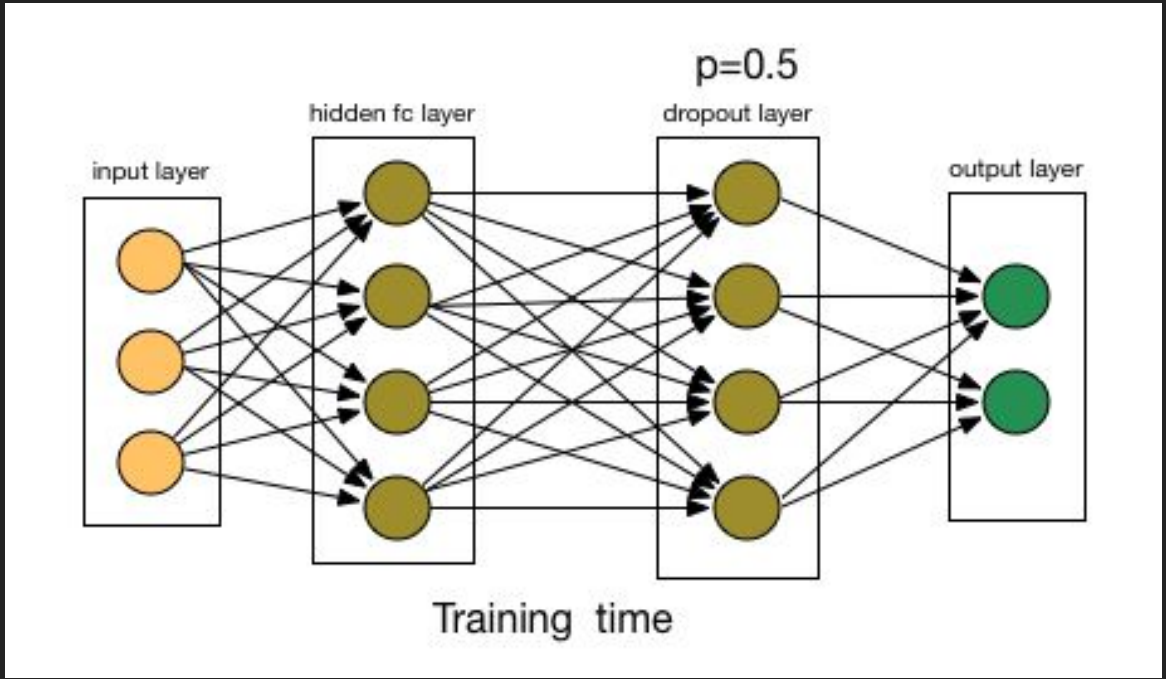


# Understanding ConvNets

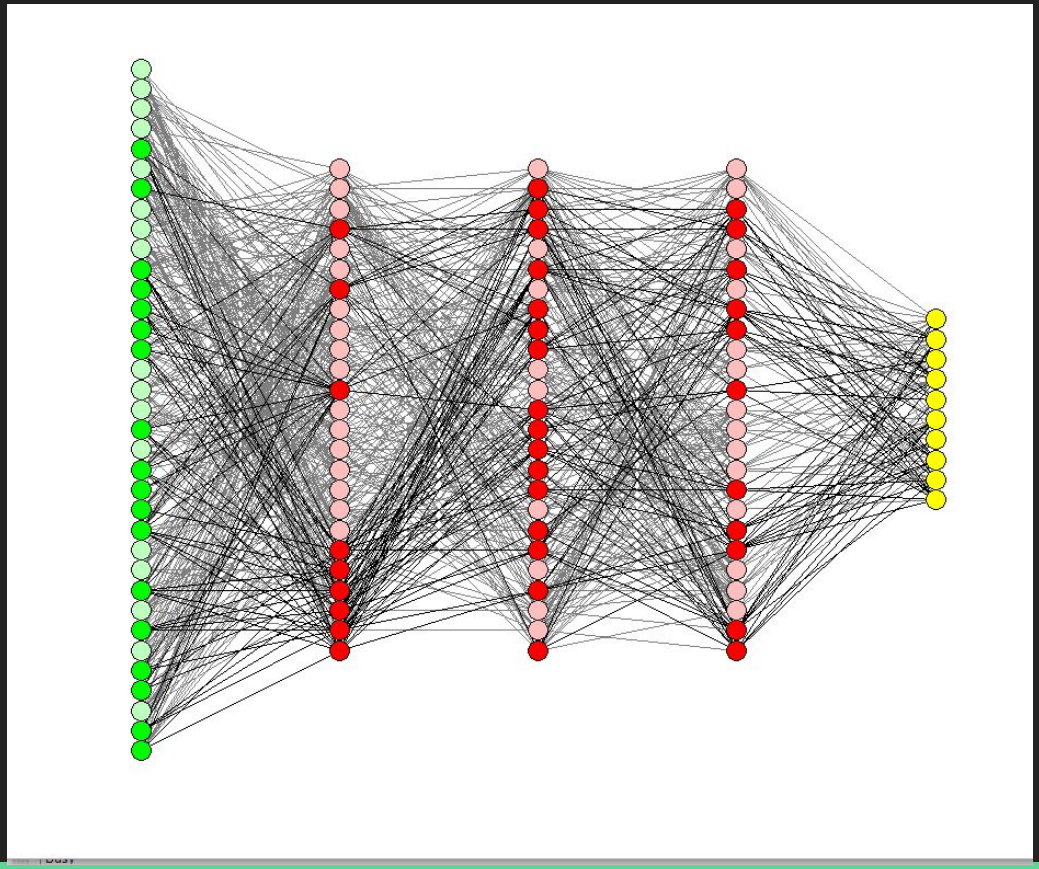


# Dropout

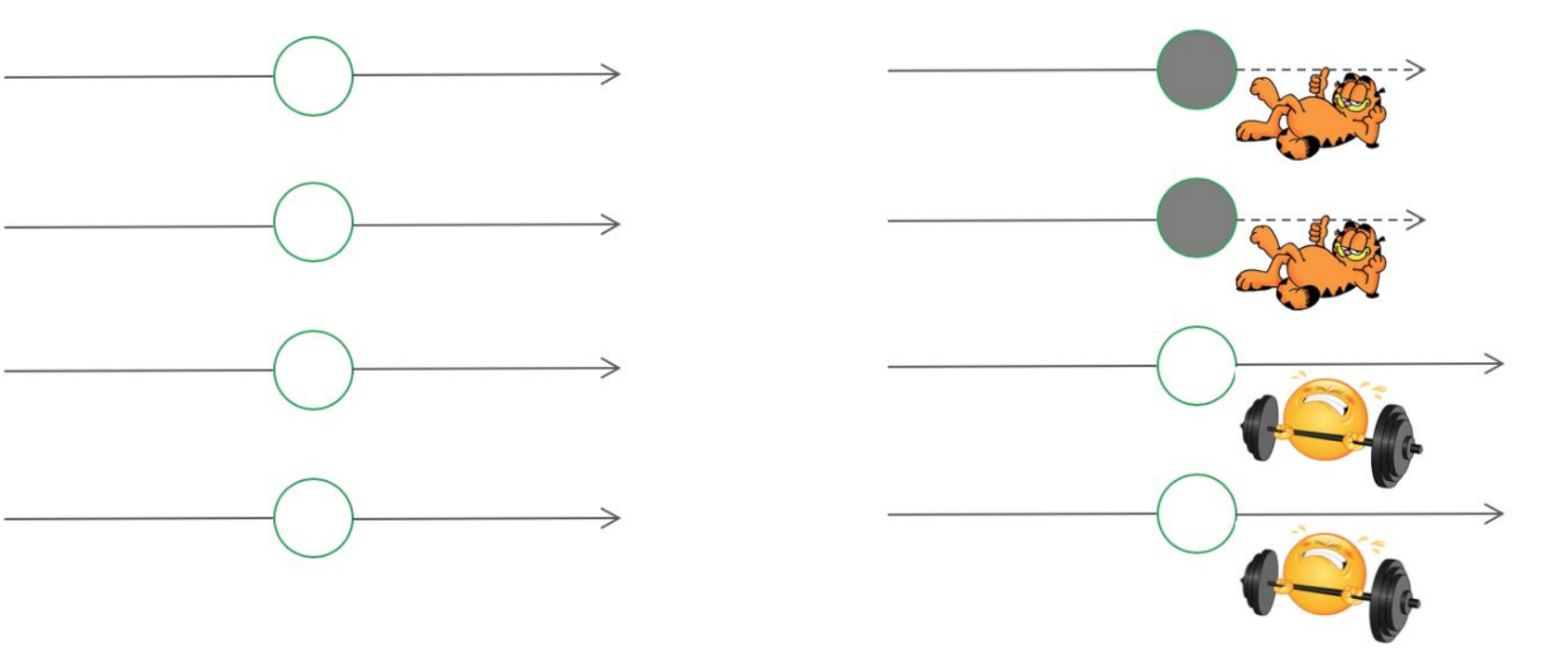
# Dropouts or how not to overfit



# Dropout: learn less to learn better

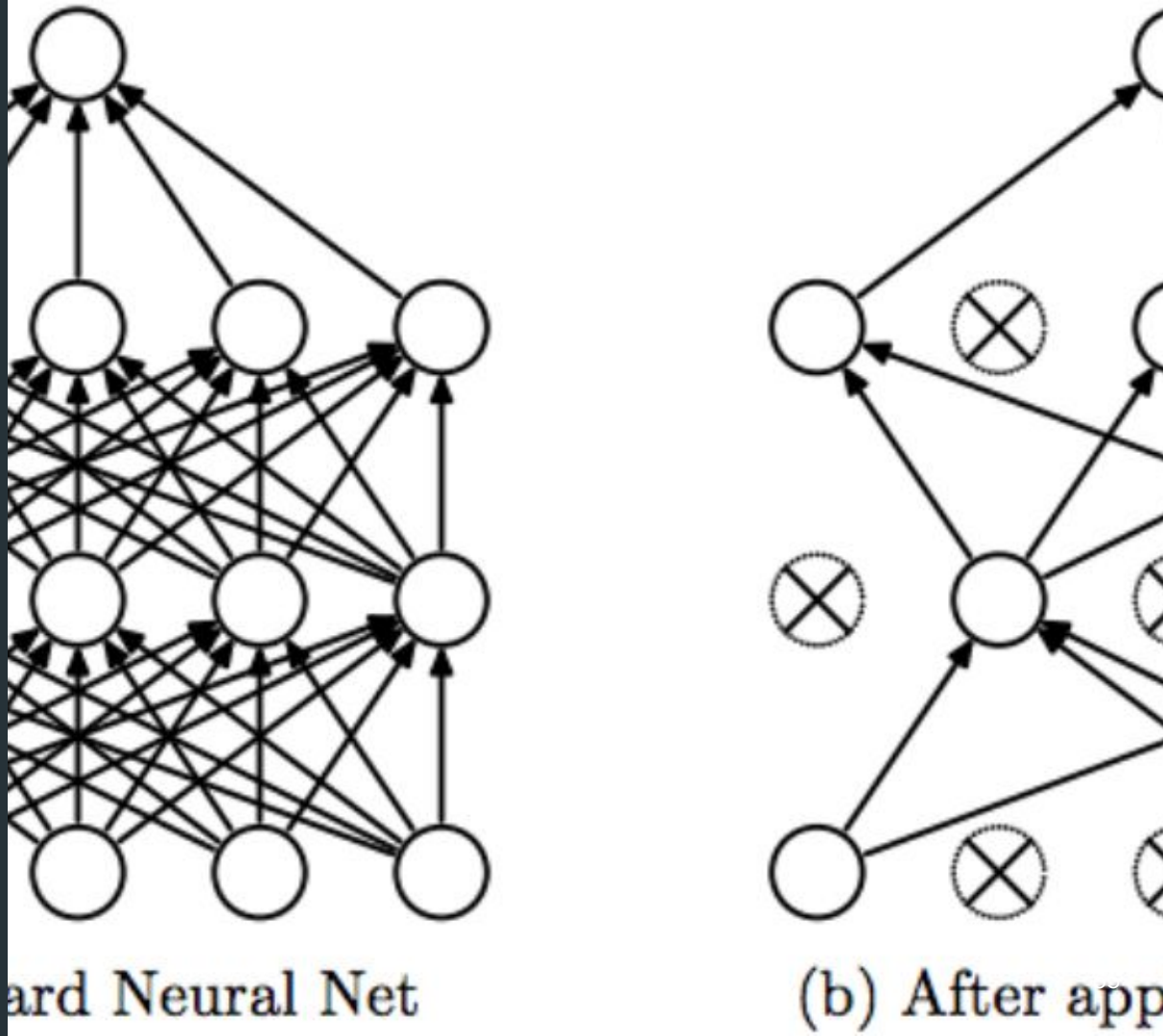


# The problem of co-adaptation

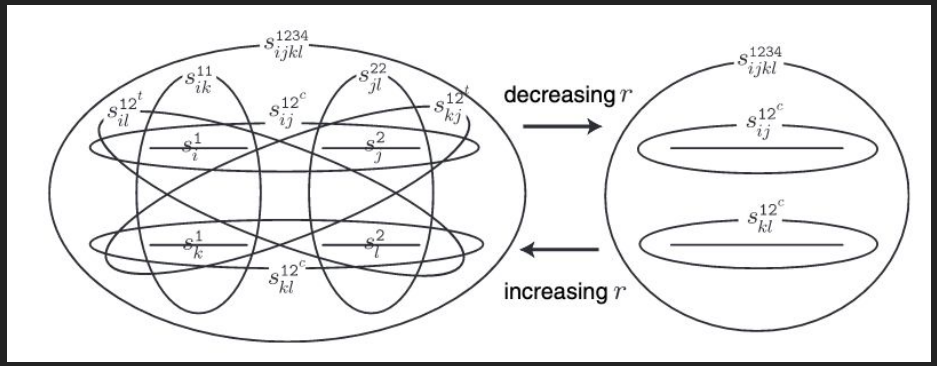


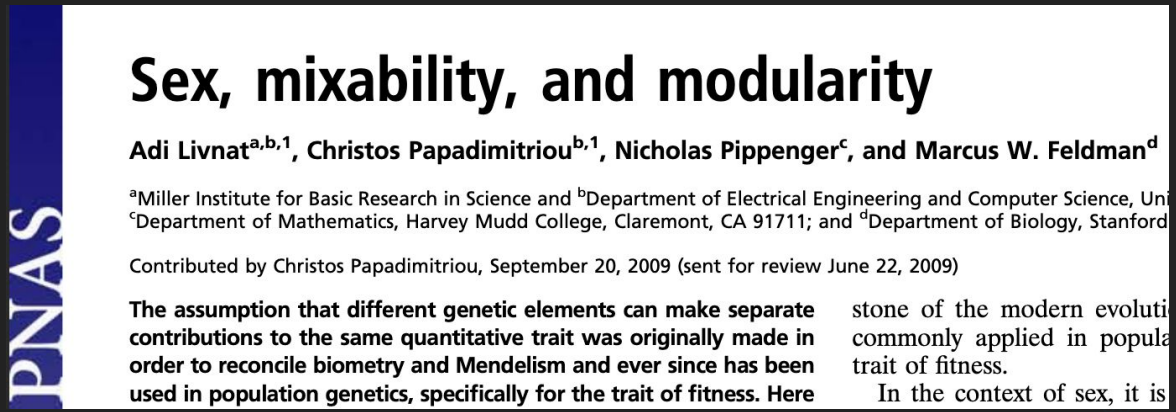
# Solution

Dropout: A neuron cannot rely on the presence of other neurons => it is forced to learn features that are not dependent on the presence of other neurons. Thus network learns robust features, and are less susceptible to noise.



# Motivation: Evolutionary Biology



PNAS

## Sex, mixability, and modularity

**Adi Livnat[a,b,1], Christos Papadimitriou[b,1], Nicholas Pippenger[c], and Marcus W. Feldman[d]**

[a]Miller Institute for Basic Research in Science and [b]Department of Electrical Engineering and Computer Science, Uni
[c]Department of Mathematics, Harvey Mudd College, Claremont, CA 91711; and [d]Department of Biology, Stanford

Contributed by Christos Papadimitriou, September 20, 2009 (sent for review June 22, 2009)

**The assumption that different genetic elements can make separate contributions to the same quantitative trait was originally made in order to reconcile biometry and Mendelism and ever since has been used in population genetics, specifically for the trait of fitness. Here**

stone of the modern evoluti
commonly applied in popula
trait of fitness.

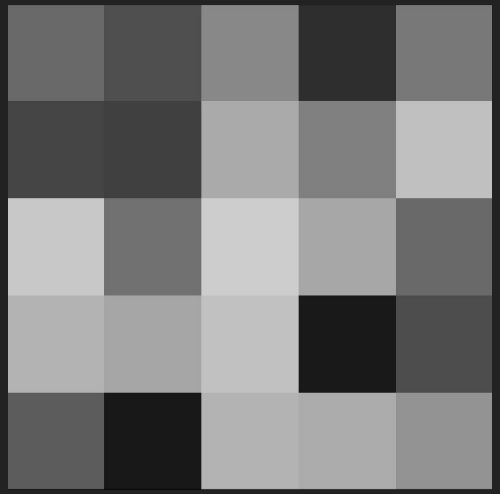
In the context of sex, it is

# Dropout in PyTorch

```
>> m = nn.Dropout(p=0.5)
>> input = torch.randn(5,5)
>> output = m(input)
```

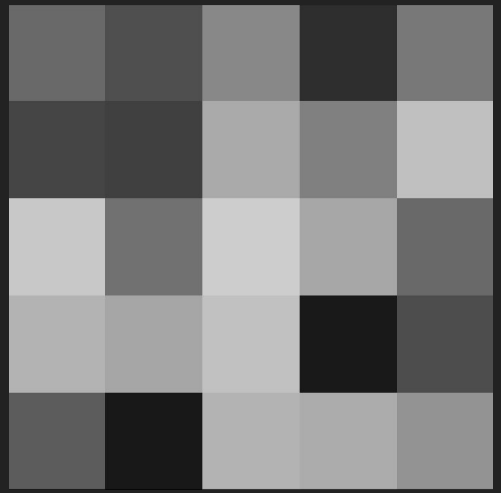
# Dropout in PyTorch

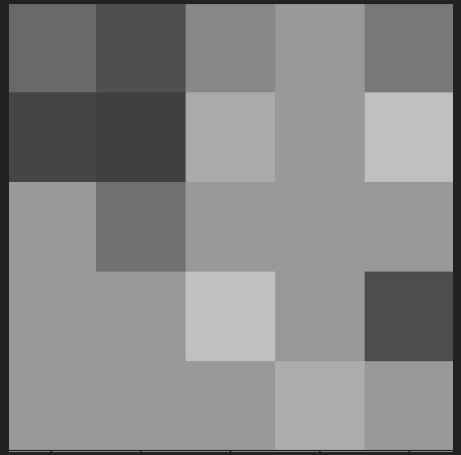
```
>> m = nn.Dropout(p=0.5)
>> input = torch.randn(5,5)
>> output = m(input)
```



# Dropout in PyTorch

```
>> m = nn.Dropout(p=0.5)
>> input = torch.randn(5,5)
>> output = m(input)
```





# Assignments

Reading:

- Chapter 9 of Deep Learning Book
- ConvNets Tutorial in PyTorch
- Understanding Convnets (blogpost, paper)
- Understanding Dropouts (blogpost, paper, a bit more advanced)

Explainability:

- How ConvNets characterize images
- Understanding ConvNets

Optional Reading:

- Evolutionary biology motivation behind dropout (Motivation section here and the paper here).
- U-Net and image segmentation (paper, review, fastai)

Programming

- Learn to work with ConvNets. Follow these tutorials to learn how to use ConvNets for various tasks in PyTorch
  - Beginner: Training a classifier
  - Advanced: Gated ConvNets for Neural NLP

Optional:

- Experiment with fast.ai image segmentation library:
  - Learn about it here and here
  - Try it as is
  - Build your own image segmentation either by using annotated data or by using pixel annotation tool.

# Discussion Points