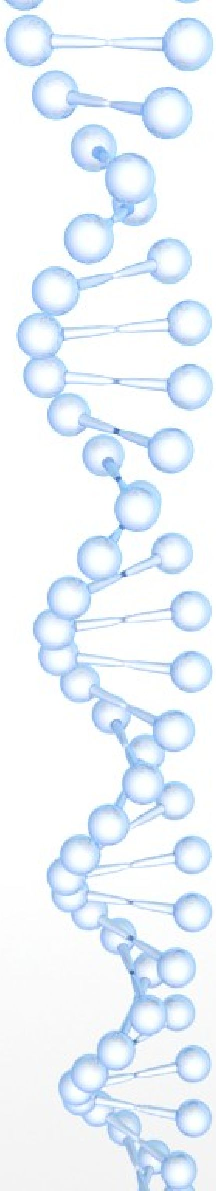
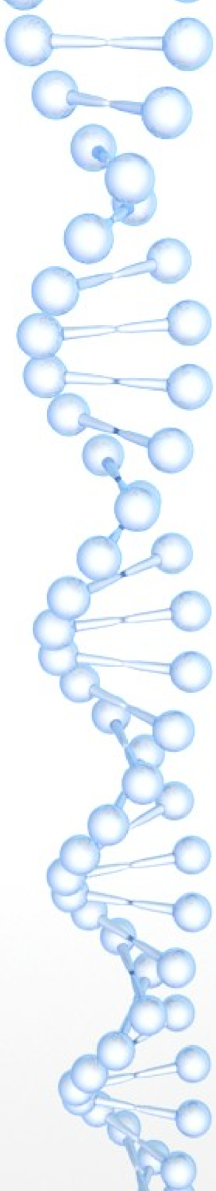


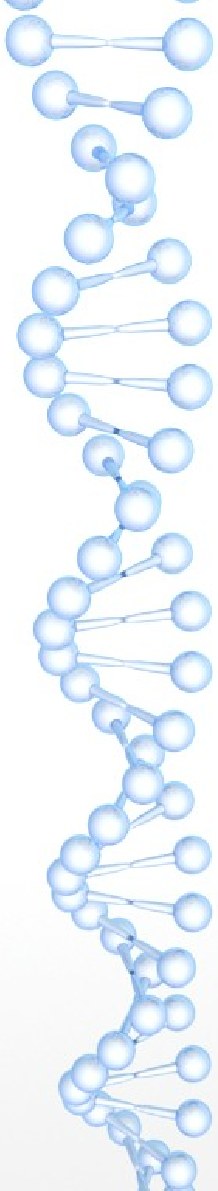
Data Science Workshop Weeks 5 & 6

... (and maybe intro to project) ...



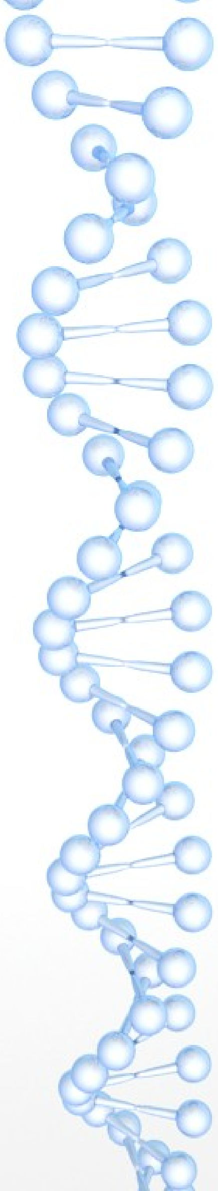
Week 5

- Dropout
- Initialization
- Gradient Descent with Momentum
- Batch Normalization



Dropout

- Form of regularization
- Similar function to L1, L2, etc., very different implementation
- Different behaviour when training, evaluating



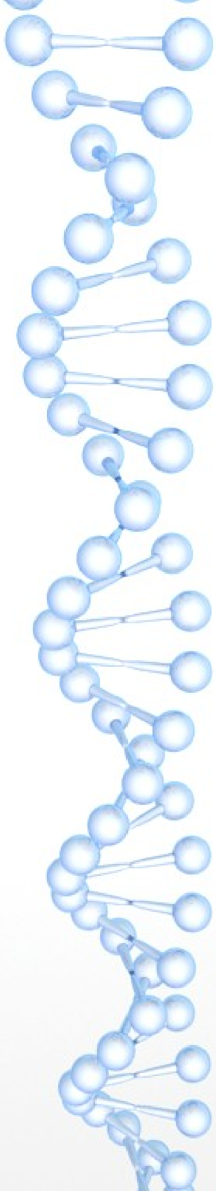
Dropout (Con’t)

- During training:
  - Randomly weights output of neuron to zero with probability p (or 1-p, in some literature)
  - This has the effect of reducing the networks reliance on that neuron
  - “Normalizes” neuron output to 1/(1-p) to account*
- During evaluation:
  - Does nothing
  - * Some implementations “normalize” during evaluation instead
- This is considered an incredible tool*
- More info here:
  - https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

*Some argue it is becoming obselete (more on this later)

Initialization

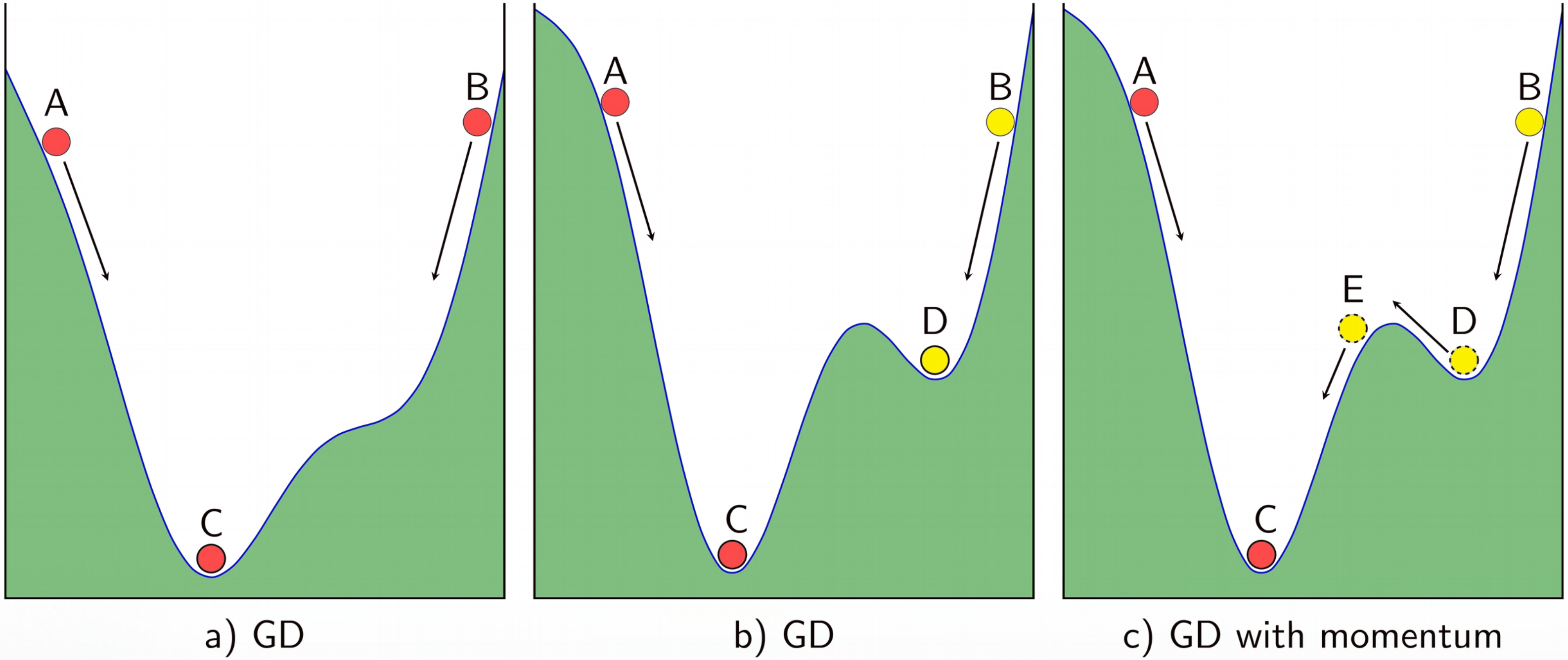
- Recall from Coursera that one cannot have uniform initialization of weights
- Many forms of initialization exist
  - Many are some form of random initialization within some bounds
- One can randomly initialize and test same ANN several times to estimate bounds of convergence



Gradient Descent with Momentum

- Normal GD can get stuck in local minima
- GD with momentum takes into account the “speed” at which GD occurs along with a “weight”

GD /w Momentum (con't)

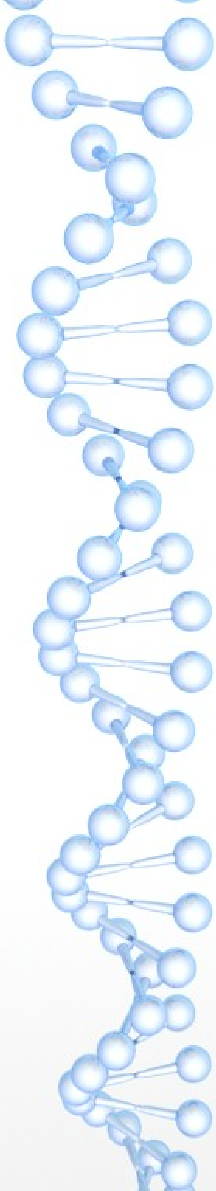


a) GD

b) GD

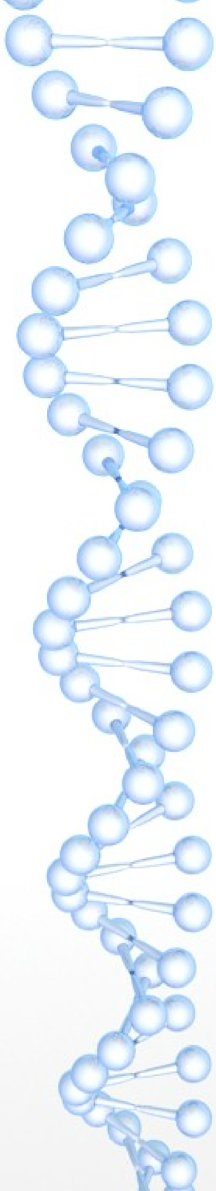
c) GD with momentum

https://machinelearningcoban.com/2017/01/16/gradientdescent2/



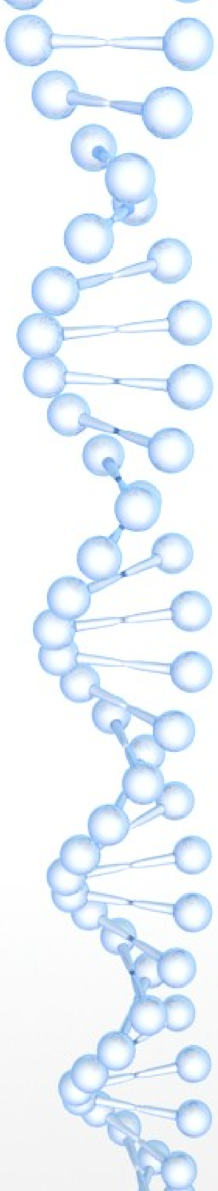
GD /w Momentum (con’t)

- Momentum parameter ranges:
  - Too low: won’t push past local minima
  - Too High: may bounce around chaotically



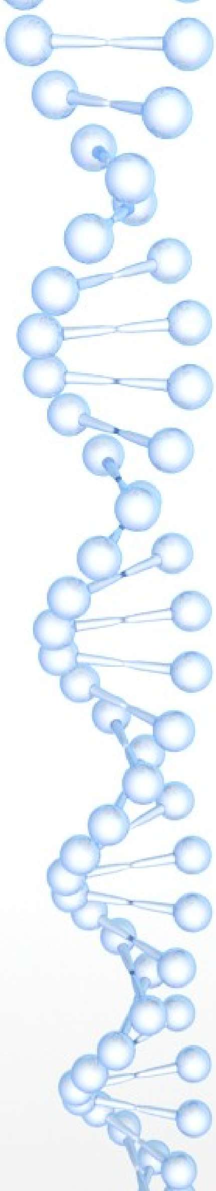
Batch Normalization

- Remember batches?
- For each batch:
  - Scale values such that mean is 0 and standard deviation is 1



Week 6

- What is convolution, and why use them?
- Convolutional Neural Network
- Advantages of **CNN** vs *ANN*
- Advantages of *ANN* vs **CNN**
- Dropout, Batch Norm, and CNN?



Convolutions

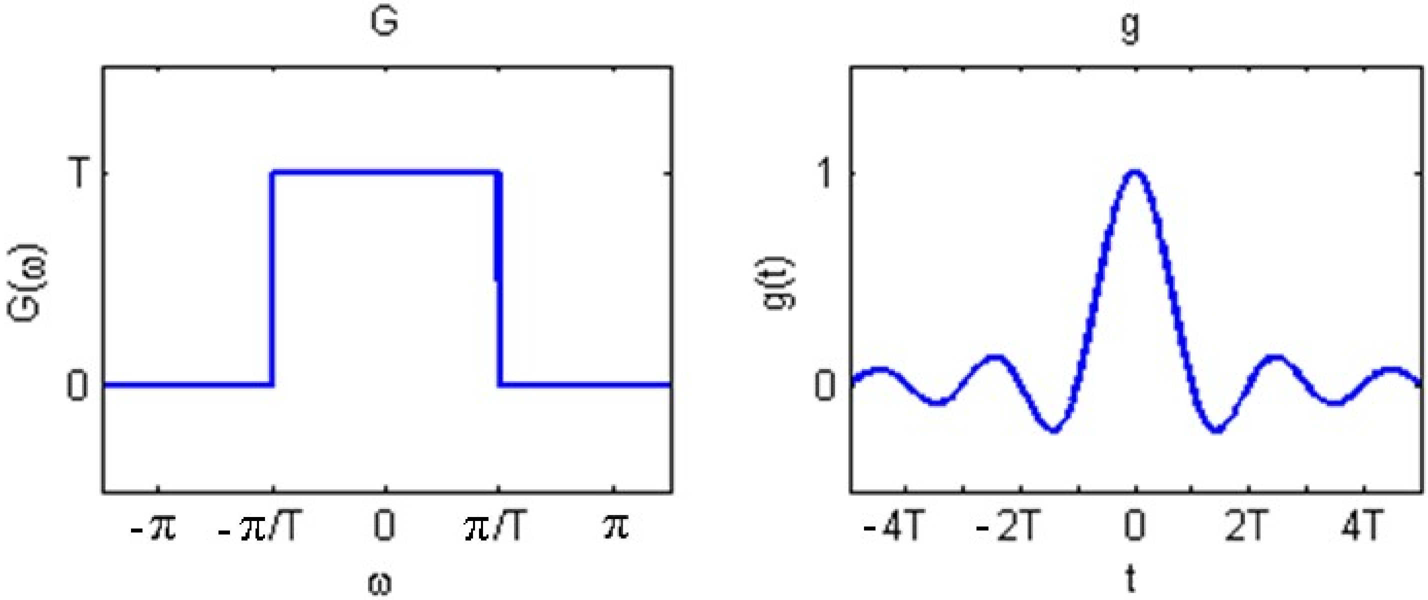
$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \, d\tau.$$

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

- Multiplication in frequency domain can be related to convolution in time domain*
- Can use convolution to filter (select) features of interest

https://en.wikipedia.org/wiki/Convolution_theorem

Convolutions (Con't)

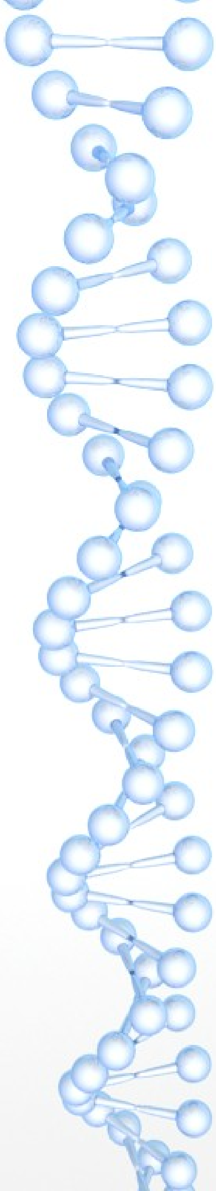


http://pilot.cnxproject.org/content/collection/col10064/latest/module/m10790/latest

Convolutions (Con’t)

- In practice, signals/filters have finite width
  - Can partially “solve” this with padding
- Signals/filters are also digital
  - See Nyquist(-Shannon) Theorem
- We can add many filters together in lieu of making good filters
- We can use these filters to select specific traits

https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem



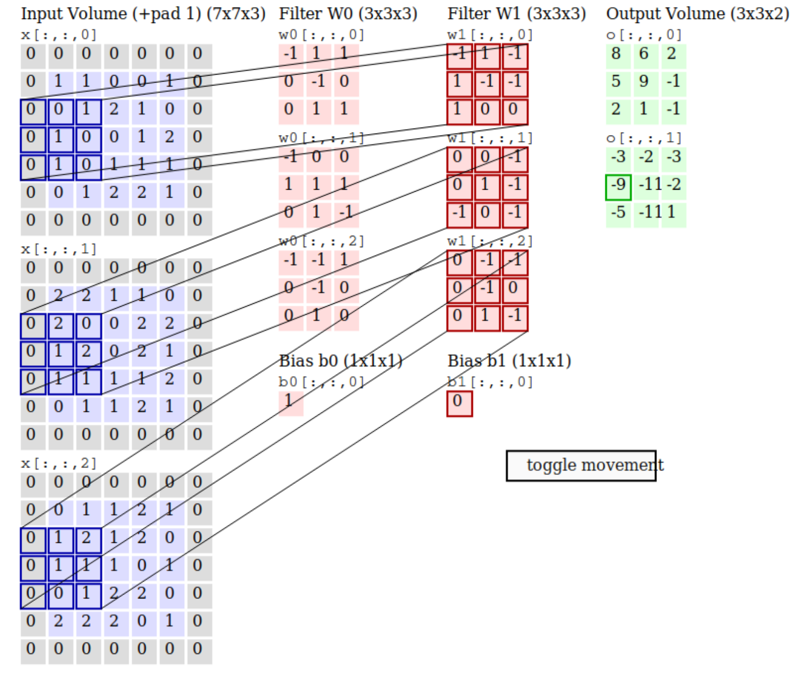
Convolutional Neural Network

- CNN is a layer that executes a convolution operation
- Parameters are the size/shape of the filter, as well as the stride of the filter (and bias)

$$Z = W \star x + b$$

- Values of the filter (and bias) are what gets trained
- Typically several filters are chained, and pooled

Convolutional Neural Network (Con’t)



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	1	1	0	0	1	0
0	0	1	2	1	0	0
0	1	0	0	1	2	0
0	1	0	1	1	1	0
0	0	1	2	2	1	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	2	1	1	0	0
0	2	0	0	2	2	0
0	1	2	0	2	1	0
0	1	1	1	1	2	0
0	0	1	1	2	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	1	1	2	1	0
0	1	2	1	2	0	0
0	1	1	1	0	1	0
0	0	1	2	2	0	0
0	2	2	2	0	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	1	1
0	-1	0
0	1	1

w0[:,:,1]

-1	0	0
1	1	1
0	1	-1

w0[:,:,2]

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

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

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

w1[:,:,1]

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

w1[:,:,2]

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

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

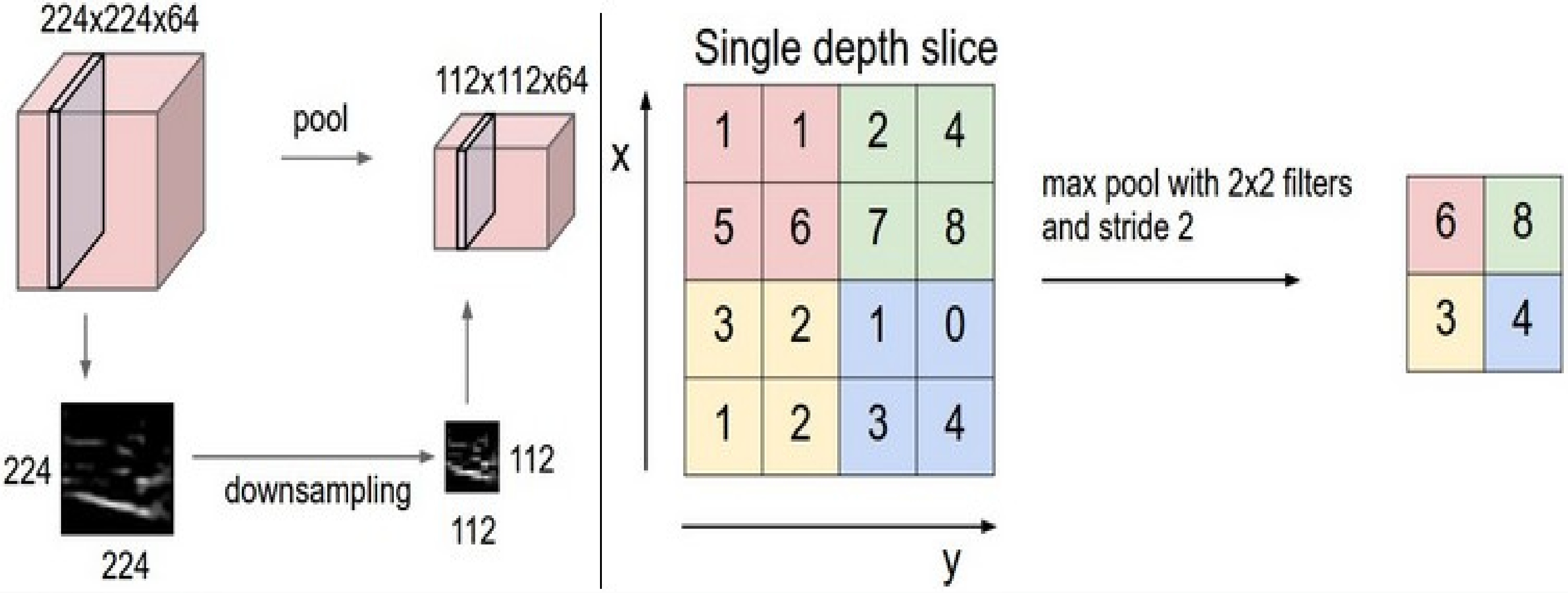
8	6	2
5	9	-1
2	1	-1

o[:,:,1]

-3	-2	-3
-9	-11	-2
-5	-11	1

http://cs231n.github.io/convolutional-networks/

Convolutional Neural Network (Con’t)

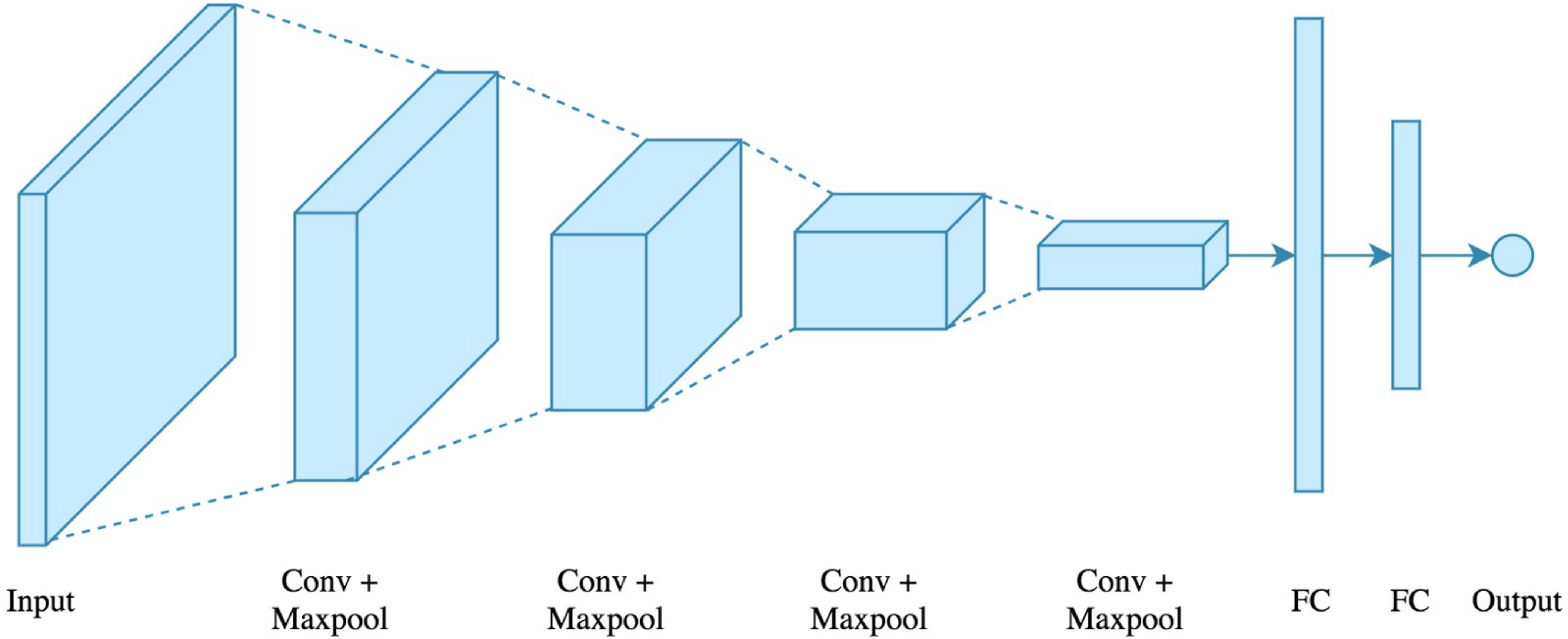


1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

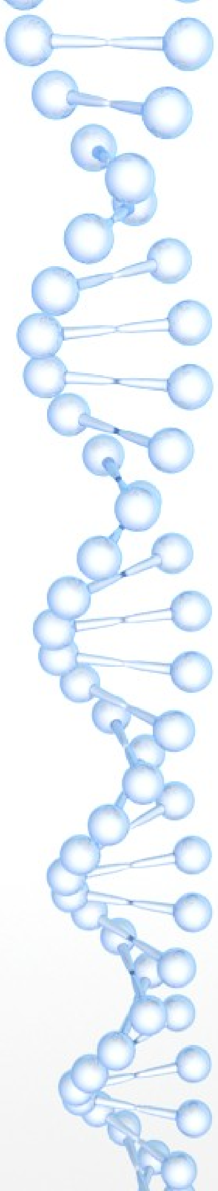
6	8
3	4

http://cs231n.github.io/convolutional-networks/

Convolutional Neural Network (Con’t)



https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

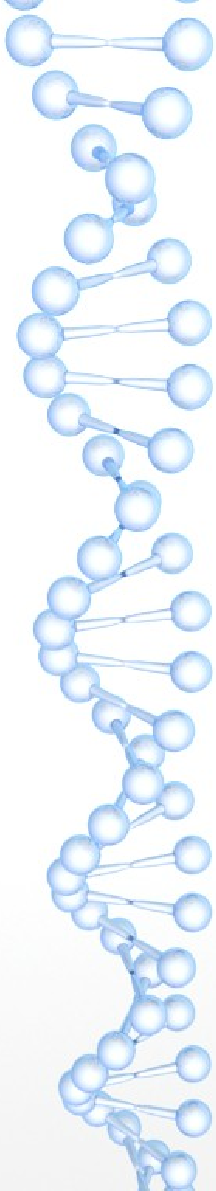


Advantages of **CNN** vs *ANN*

- Number of parameters per CNN layer can *(sort of)* be independant of input size
- CNN preserves relative position of data
  - Ie, translation, rotation, magnification less impactful

Advantages of *ANN* vs **CNN**

- CNN often require many, many layers to accomplish goal
- CNN usually must be paired with ANN anyway



Dropout, Batch Normalize, and CNN?

- You can, but according to [1], you shouldn’t
  - Dropout is good if you have many parameters
  - CNN already excellent at ignoring/filtering data
- Batch normalization and CNN pair nicely

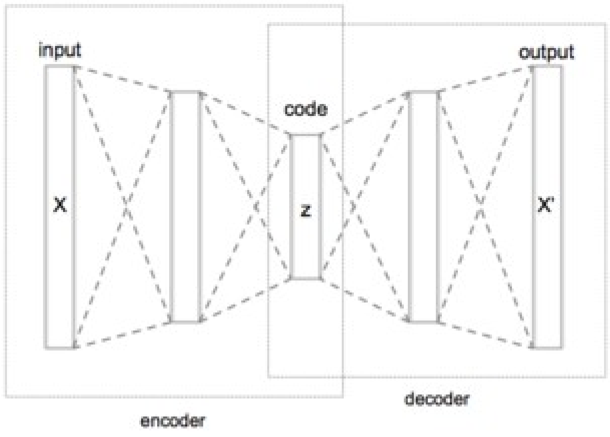
[1] https://towardsdatascience.com/dont-use-dropout-in-convolutional-networks-81486c823c16

Project/future Ideas

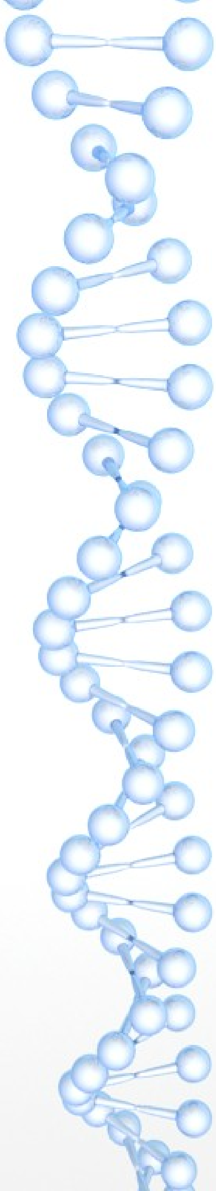
- Delay on HyperK
- Other ideas:
  - We can do our own project(s):
    - Classifier
    - Autoencoder
    - Generative adversarial network (GAN)
    - Etc.
  - Take turns presenting papers/articles on ML or applied ML:
    - Medium
    - TowardsDataScience
    - TwoMinutePapers

Autoencoder

- Unsupervised learning
- Useful for lossy compression
- Can generate cool art



https://en.wikipedia.org/wiki/Autoencoder



Generative Adversarial Network (GAN)

- Train a network to produce structured data from random noise
- A painting created by a GAN recently sold for USD432000