

# Class 10

## H Academy

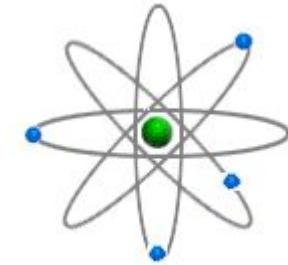
March 18th, 2021 - By Nathan Landman



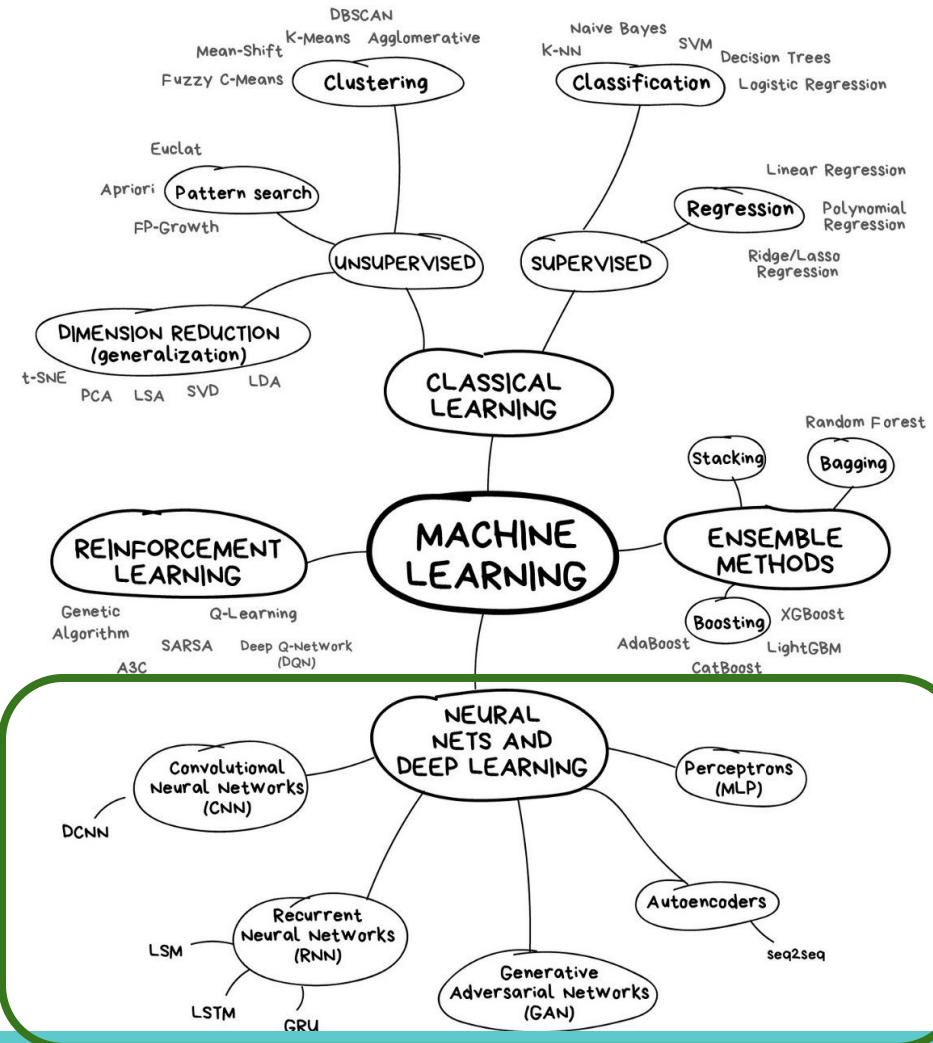
Academy

# Class Agenda

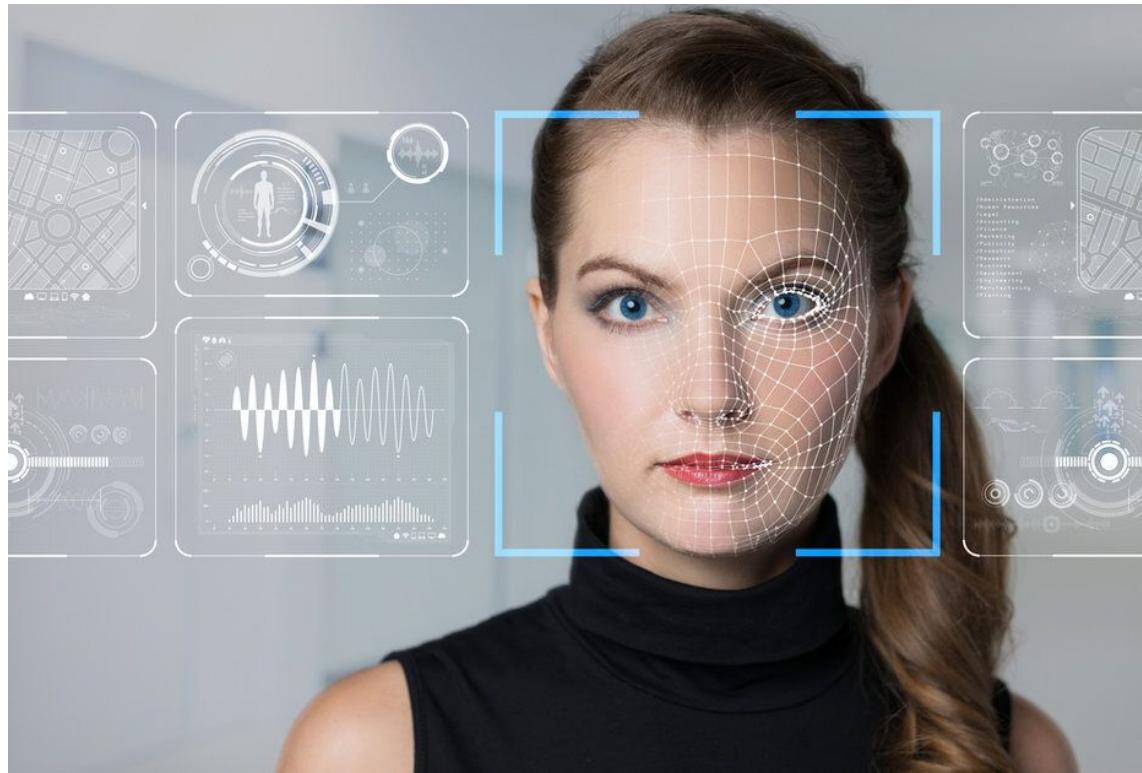
1. Neural Networks
  - a. Basics of Neural Networks
  - b. Neural Network Architectures
  - c. Neural Networks in Practice



# Machine Learning Landscape



# Let's talk about machine vision...



# How does a machine interpret an 8?



A machine processes an image by breaking it down into **pixels**, where each pixel contains a set of numbers representative of the color.

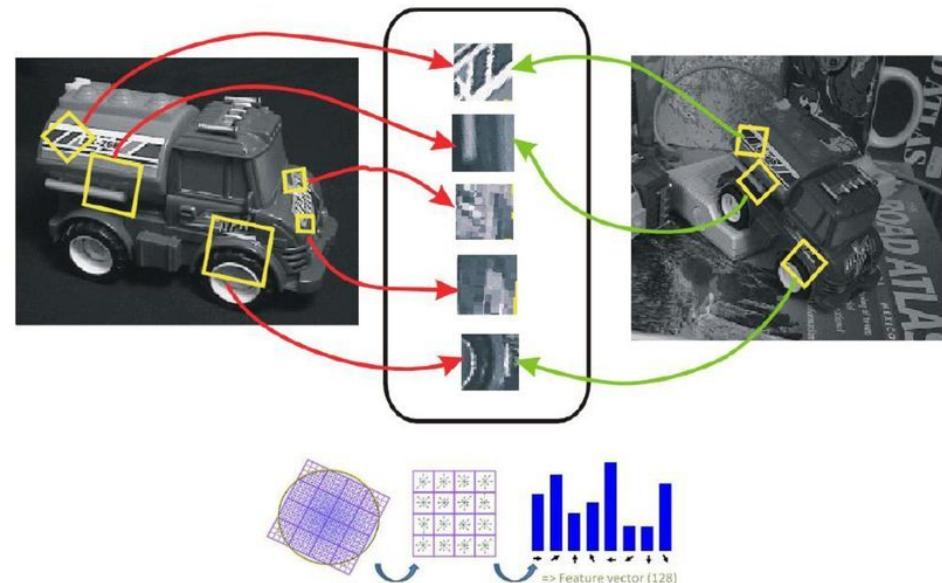
In large images, there's a bunch of noise, when we only need to see a fraction of the image to know what it is.

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29	
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0	
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1	
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49	
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36	
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62	
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0	
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0	
0	13	111	255	255	245	255	182	181	248	252	242	208	36	0	19	
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0	
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0	
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4	
0	22	205	252	246	251	241	100	24	113	255	245	255	194	9	0	
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0	
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3	
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0	
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4	
0	18	146	250	255	247	255	255	255	249	255	240	255	126	0	5	
0	0	23	115	215	255	250	248	255	255	248	245	118	14	12	0	
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1	
0	0	5	5	0	0	0	0	0	0	14	1	0	6	6	0	

# Computer Vision – Invariant Features

How do you obtain  
an accurate  
representation of  
an image?

Early computer vision systems  
were very fine-tuned to  
detecting edges and other  
salient features in specific  
images.



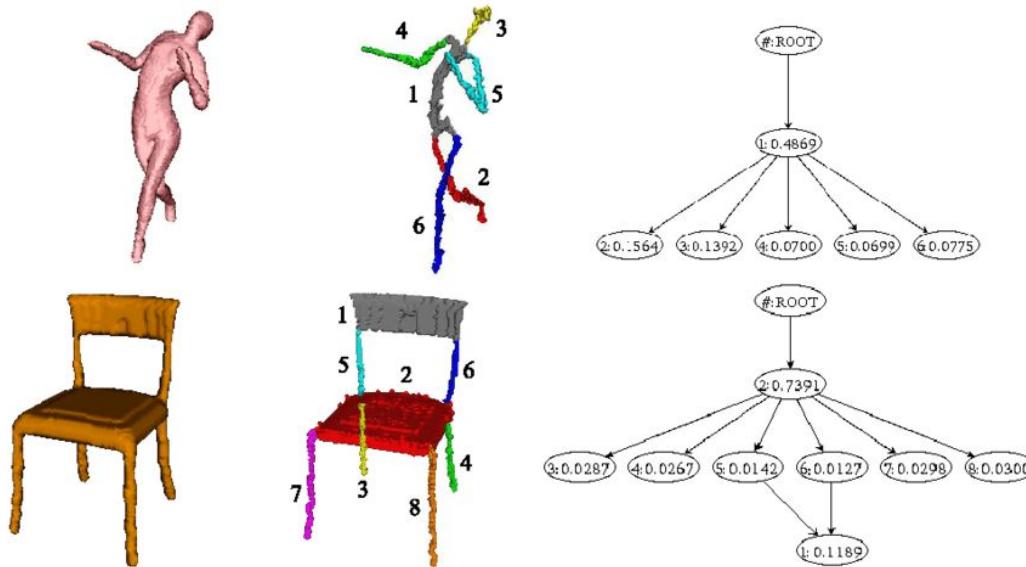
"SIFT" & Object Recognition, David Lowe, 1999

Slides based on cs231n by Fei-Fei Li & Andrej Karpathy & Justin Johnson

# So, how would we find representations of a chair?



# So, how would we find representations of a chair?



# But, would these classify as chairs?



# How about something you sit on?

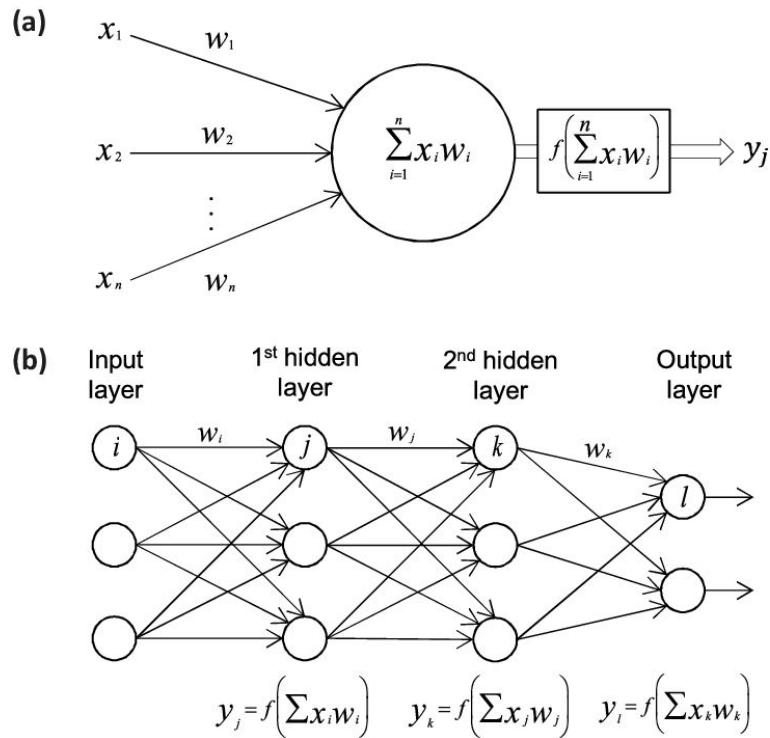


# How about something you sit on?

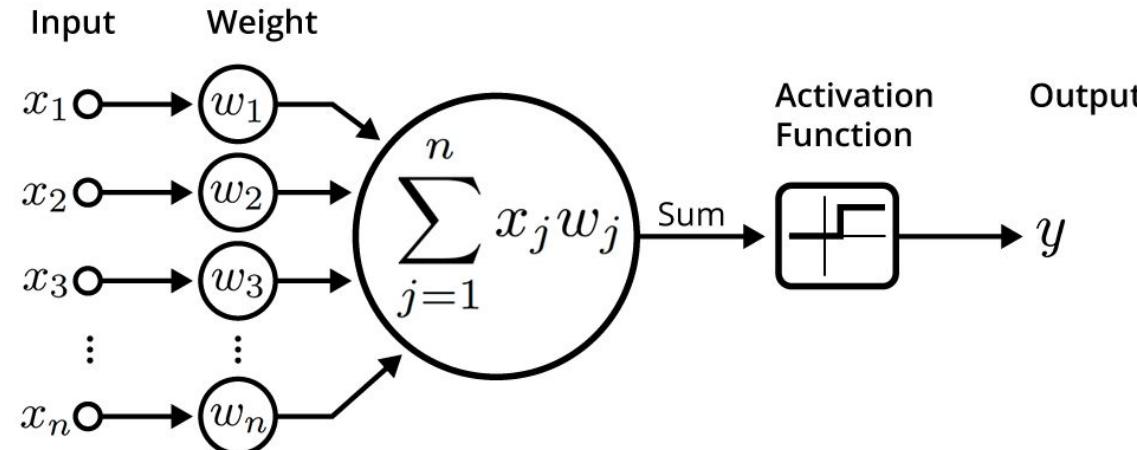


# Enter Deep Learning!

By intelligently trying a bunch of mathematical operations using fast computers, a machine is able to detect patterns in the data.



# What are these operations? An Introduction to the Neuron



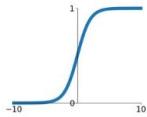
An illustration of an artificial neuron. Source: Becoming Human.

# Activation Functions

## Activation Functions

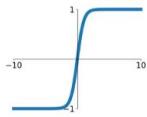
**Sigmoid**

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



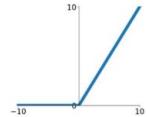
**tanh**

$$\tanh(x)$$



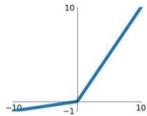
**ReLU**

$$\max(0, x)$$



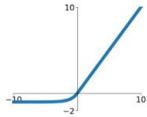
**Leaky ReLU**

$$\max(0.1x, x)$$



**Maxout**

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$



**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

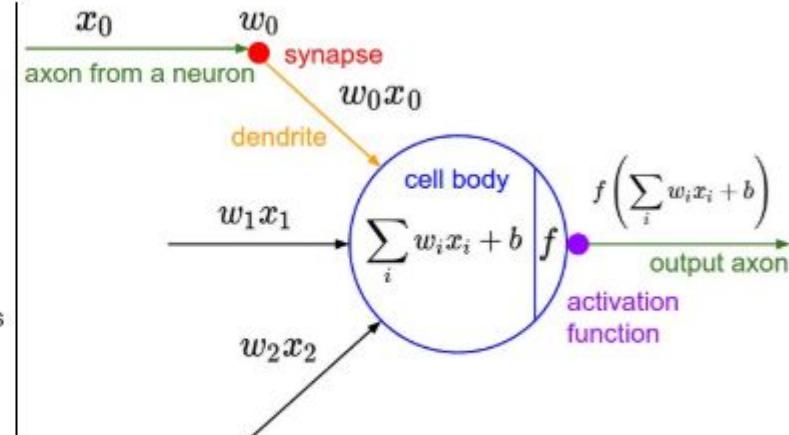
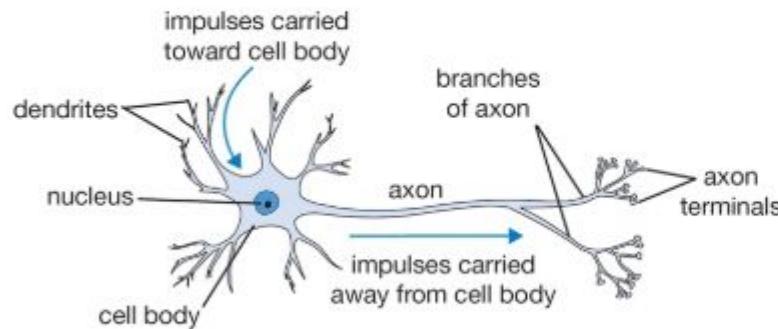
**Activation Function**



**Output**

# Sidenote.. Why is it called a Neural Network?

---

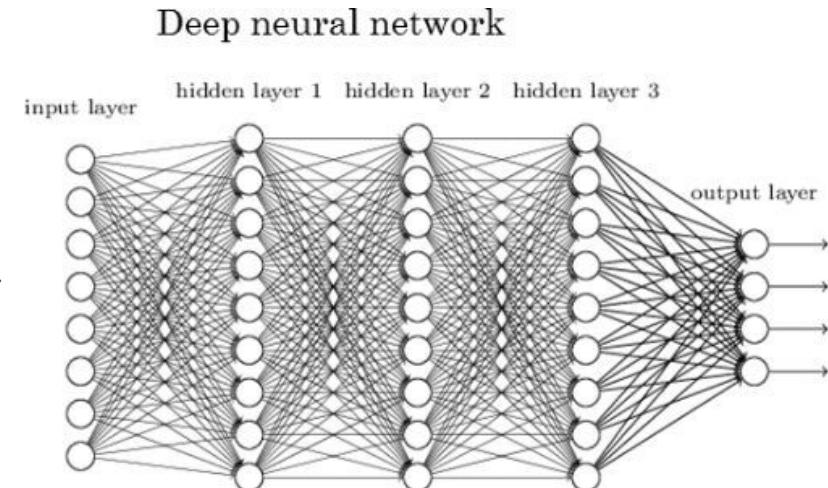
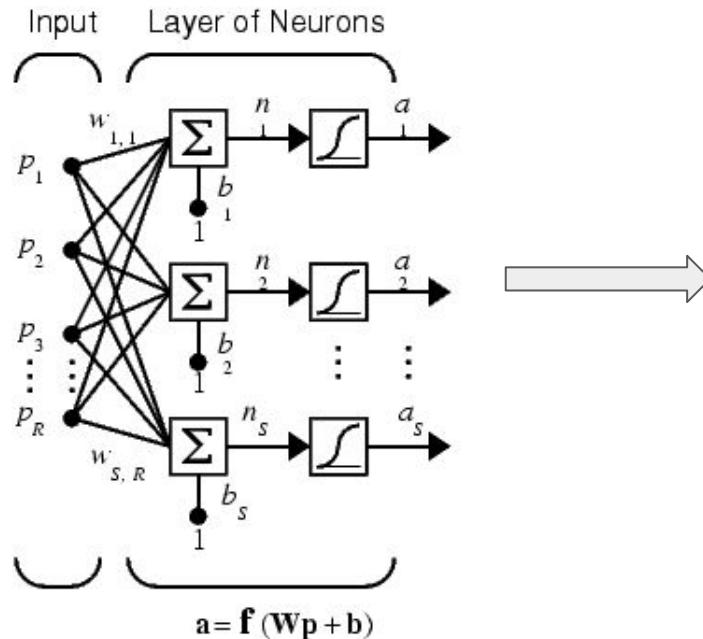


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

---

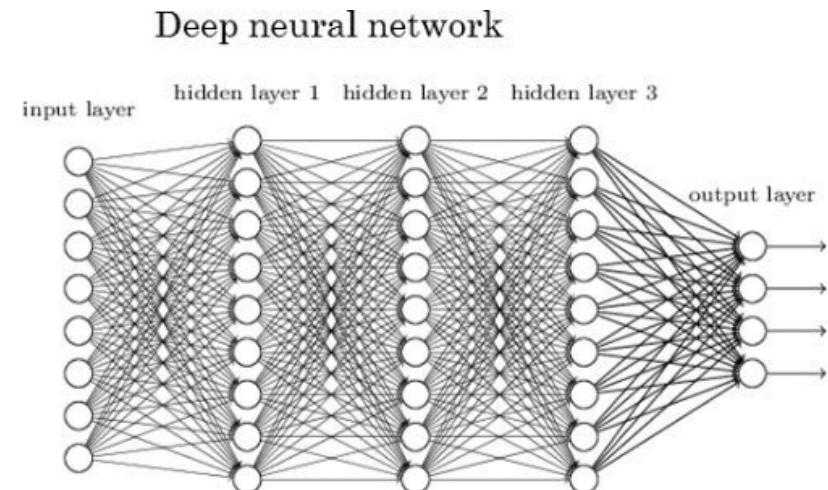
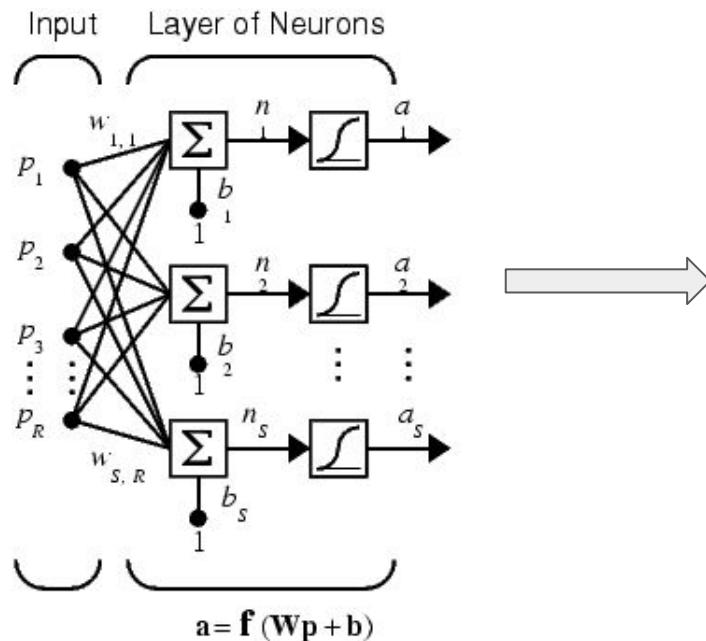
# How do we build complex neurons?

9	0	0
32	0	0
222	103	10
253	251	124
251	238	255
232	255	255
237	252	235
243	255	137
248	144	6
208	35	0
17	0	7
11	0	1

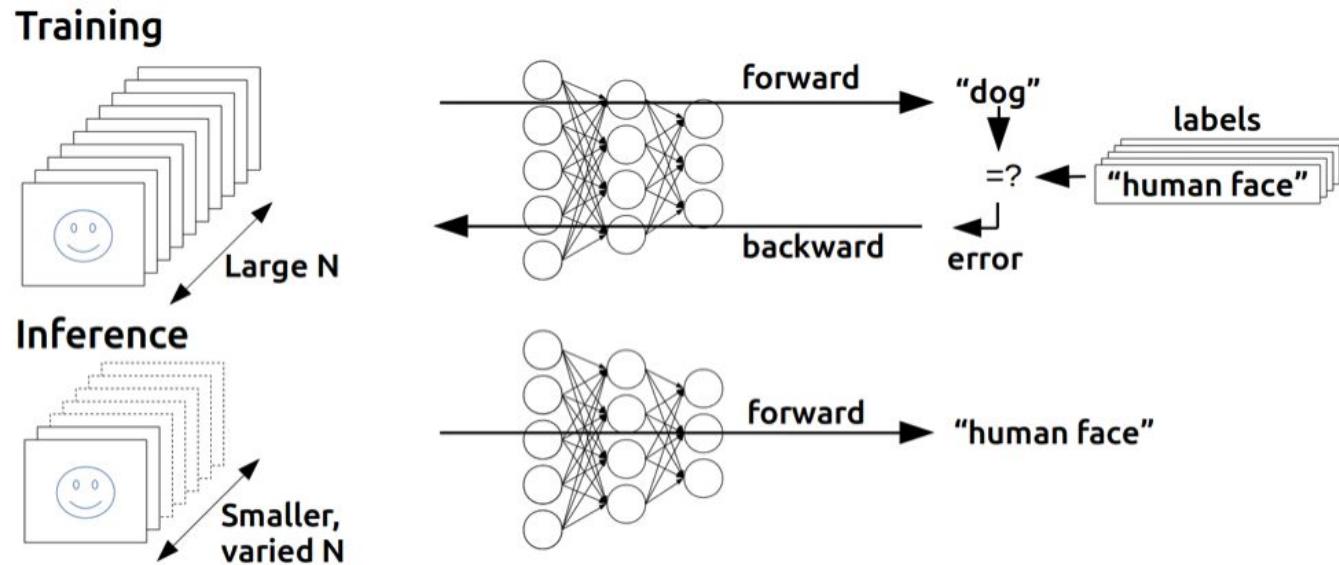


# We call this a Feed Forward Neural Network

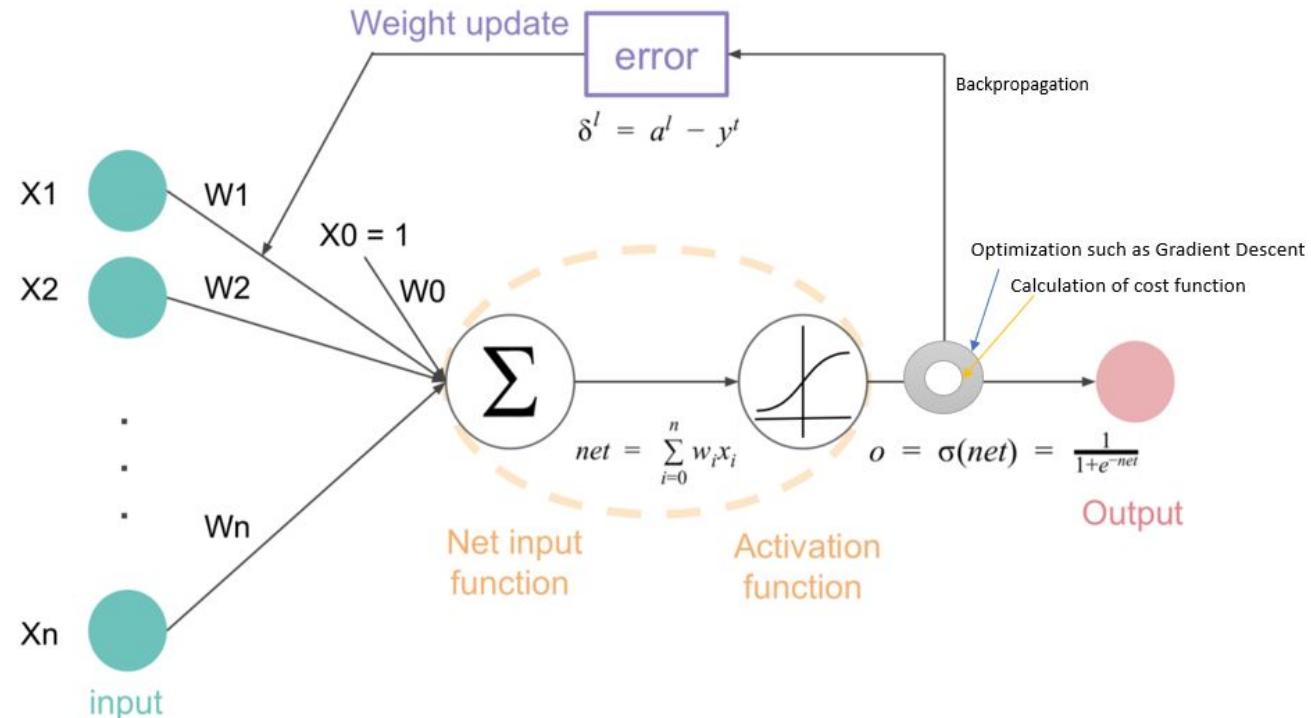
9	0	0
32	0	0
222	103	10
253	251	124
251	238	255
232	255	255
237	252	235
243	255	137
248	144	6
208	35	0
17	0	7
11	0	1



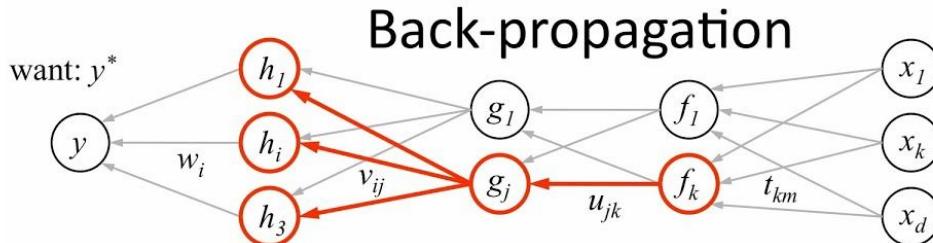
# How does it actually learn from its mistakes?



# How does it actually learn from its mistakes?



# How does it actually learn from its mistakes?



1. receive new observation  $\mathbf{x} = [x_1 \dots x_d]$  and target  $y^*$
2. **feed forward:** for each unit  $g_j$  in each layer  $1 \dots L$   
compute  $g_j$  based on units  $f_k$  from previous layer:  $g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right)$
3. get prediction  $y$  and error  $(y - y^*)$
4. **back-propagate error:** for each unit  $g_j$  in each layer  $L \dots 1$

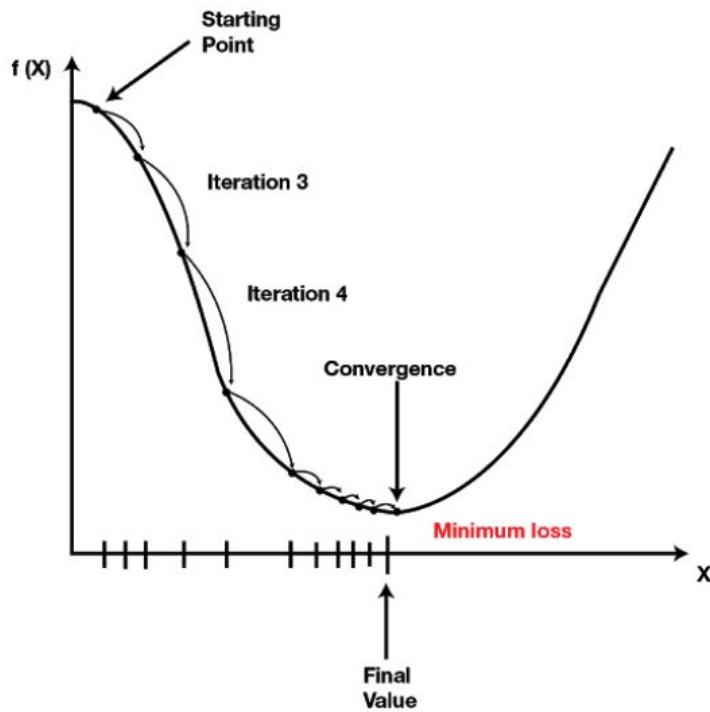
(a) compute error on  $g_j$

$$\frac{\partial E}{\partial g_j} = \sum_i \underbrace{\sigma'(h_i)}_{\substack{\text{should } g_j \\ \text{be higher or lower?}}} \underbrace{v_{ij}}_{\substack{\text{how } h_i \text{ will} \\ \text{change as } g_j \text{ changes}}} \underbrace{\frac{\partial E}{\partial h_i}}_{\substack{\text{was } h_i \text{ too} \\ \text{high or too low?}}}$$

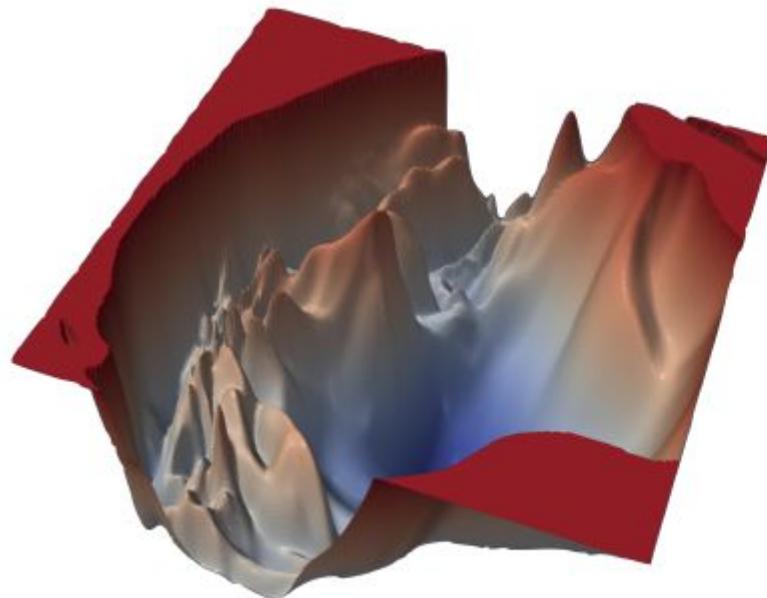
(b) for each  $u_{jk}$  that affects  $g_j$

(i) compute error on $u_{jk}$	(ii) update the weight
$\frac{\partial E}{\partial u_{jk}} = \frac{\partial E}{\partial g_j} \underbrace{\sigma'(g_j)}_{\substack{\text{do we want } g_j \text{ to} \\ \text{be higher/lower?}}} \underbrace{f_k}_{\substack{\text{how } g_j \text{ will change} \\ \text{if } u_{jk} \text{ is higher/lower?}}}$	$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$

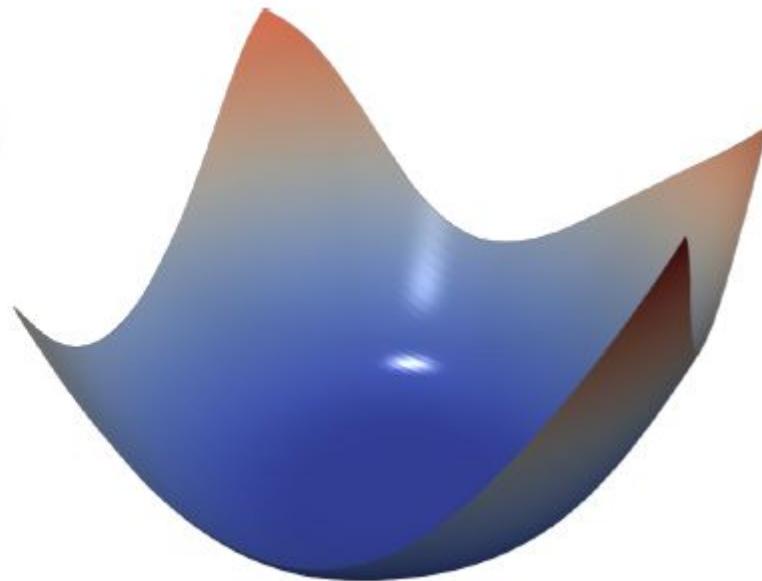
# Convergence to a minimum loss via backpropagation



# Beware of a local minimum!

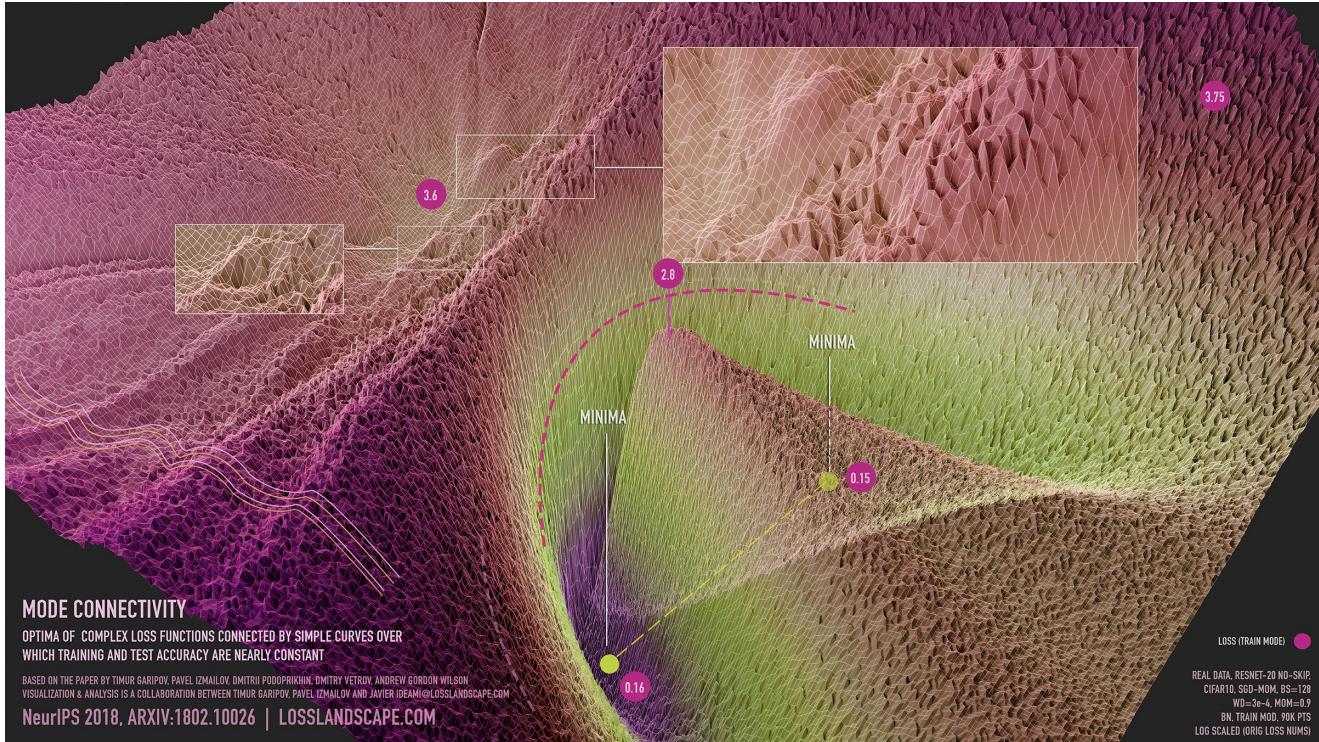


(a) ResNet-110, no skip connections

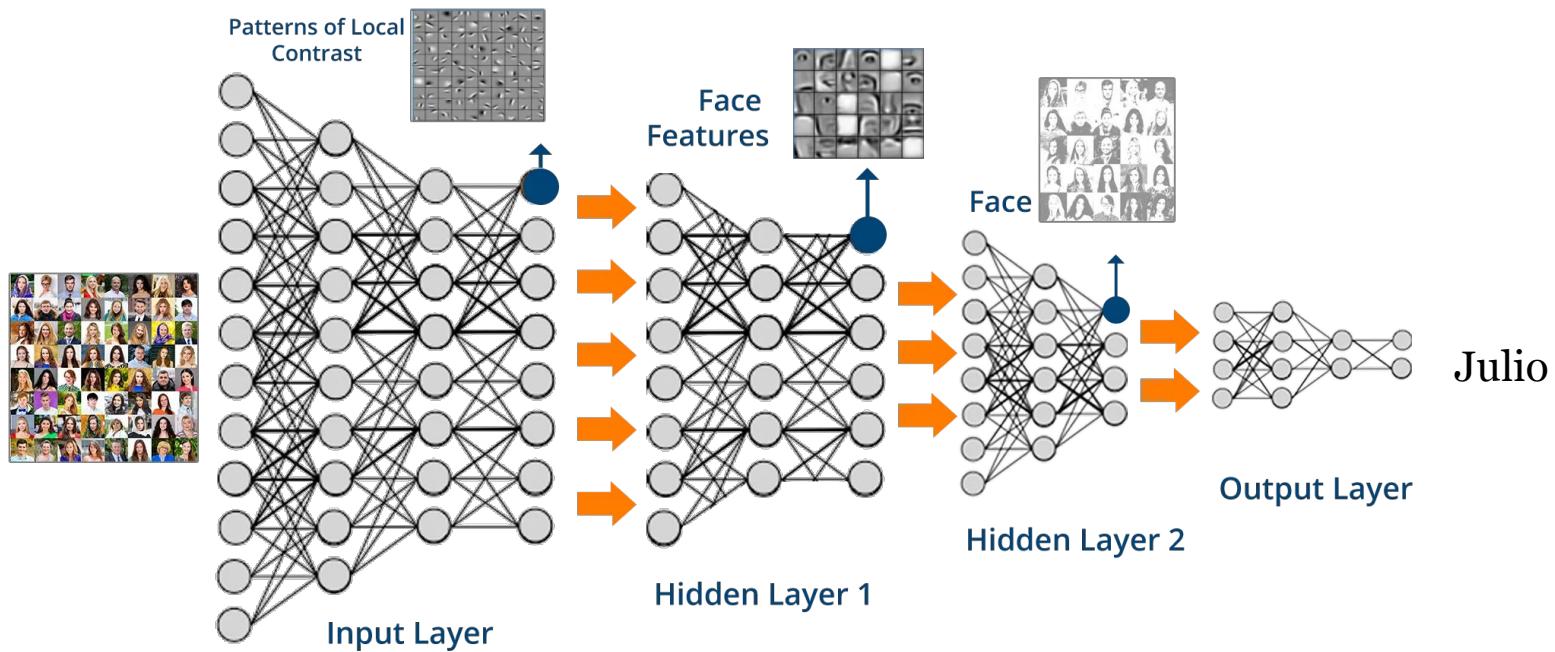


(b) DenseNet, 121 layers

# Beware of a local minimum!

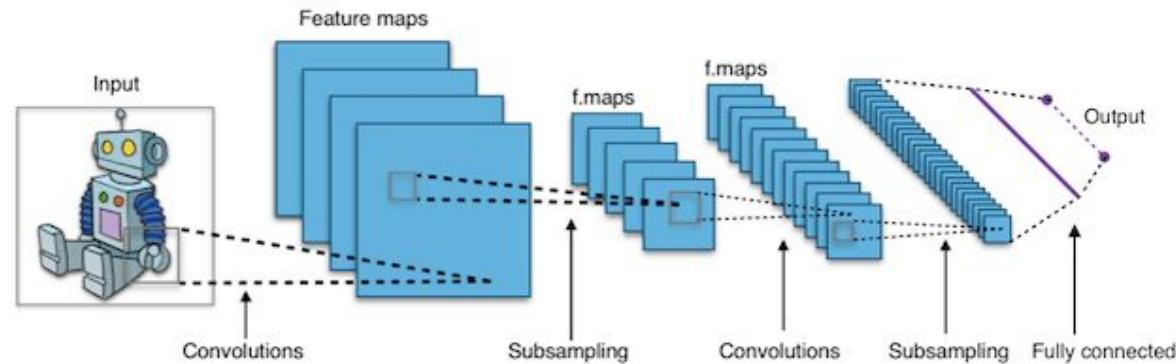


# Back to computer vision...



# Convolutional Neural Network (CNNs)

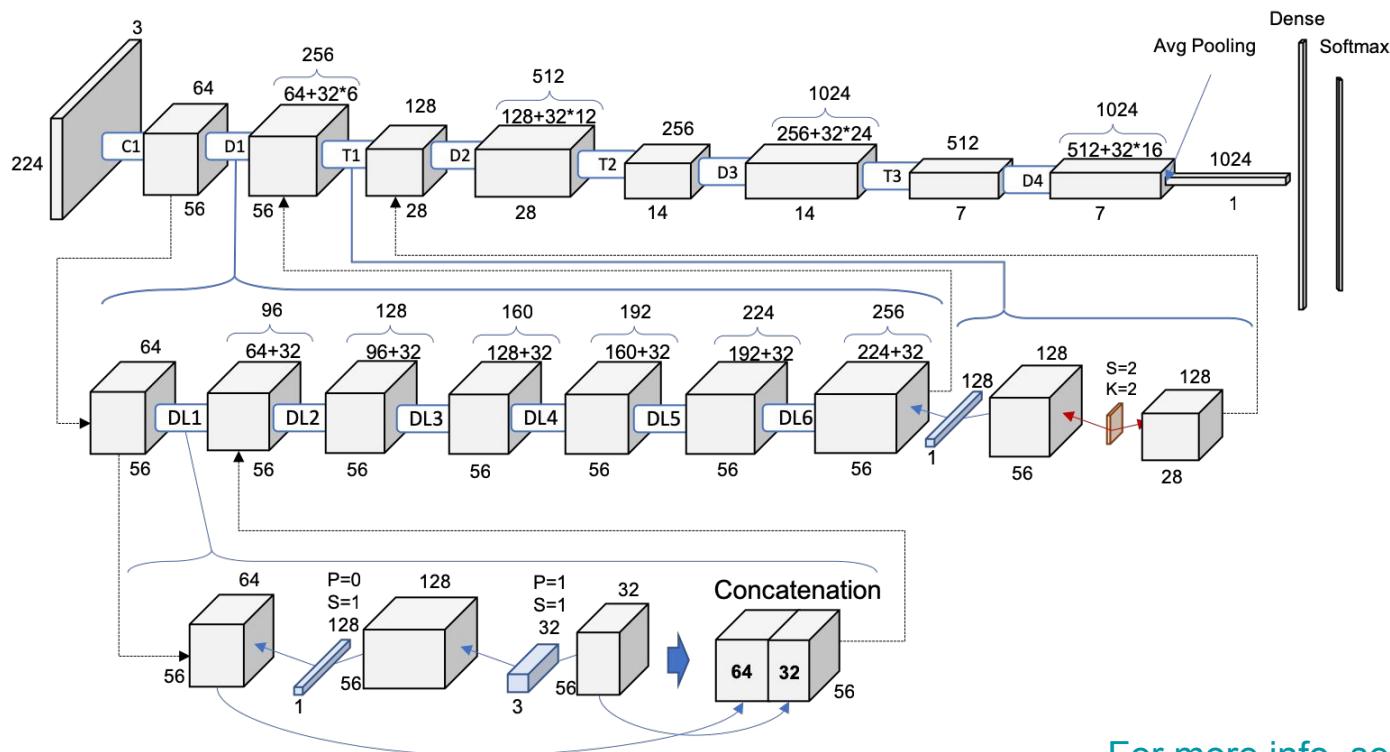
A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other



# Convolutional Neural Network (CNNs)

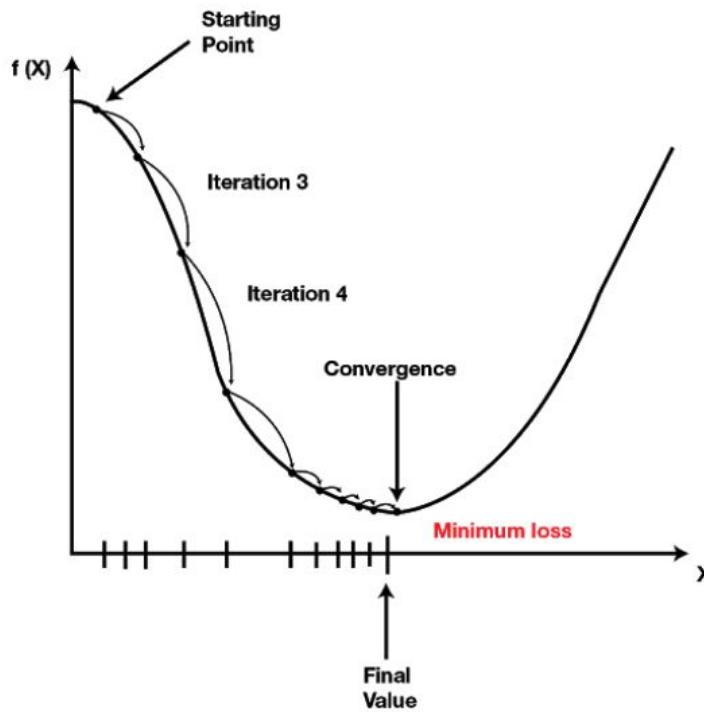
A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other

# CNNs gone wild



[For more info, see this note.](#)

# Remember this graph?

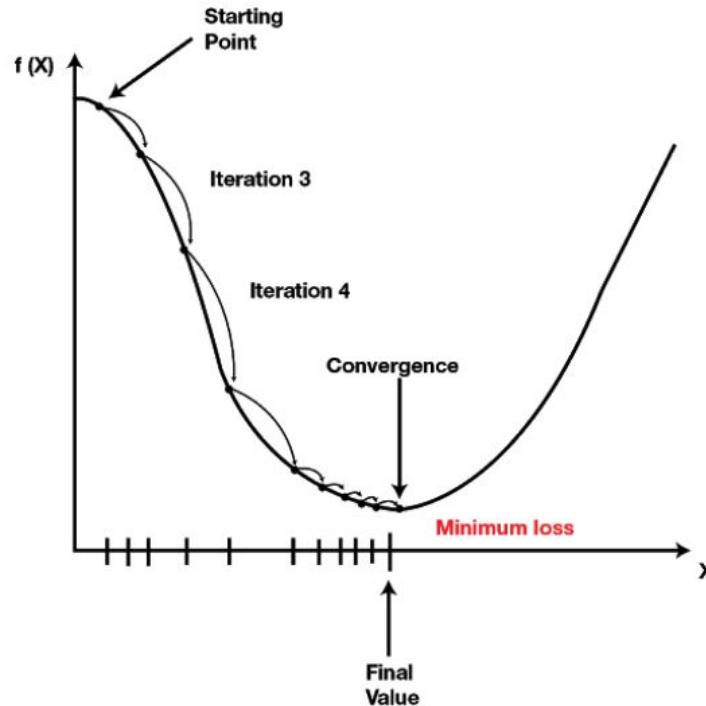


# Training Technicalities

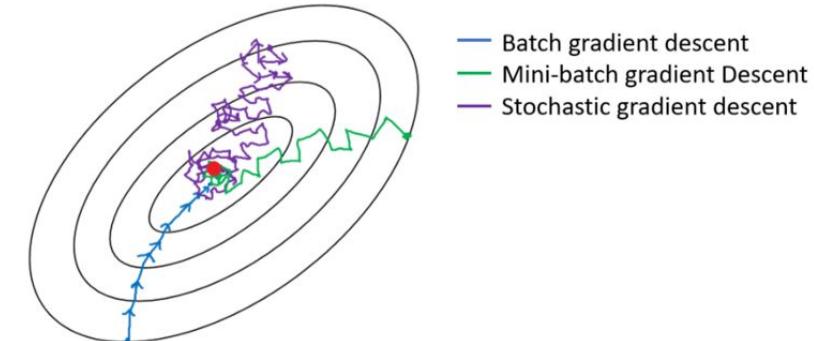
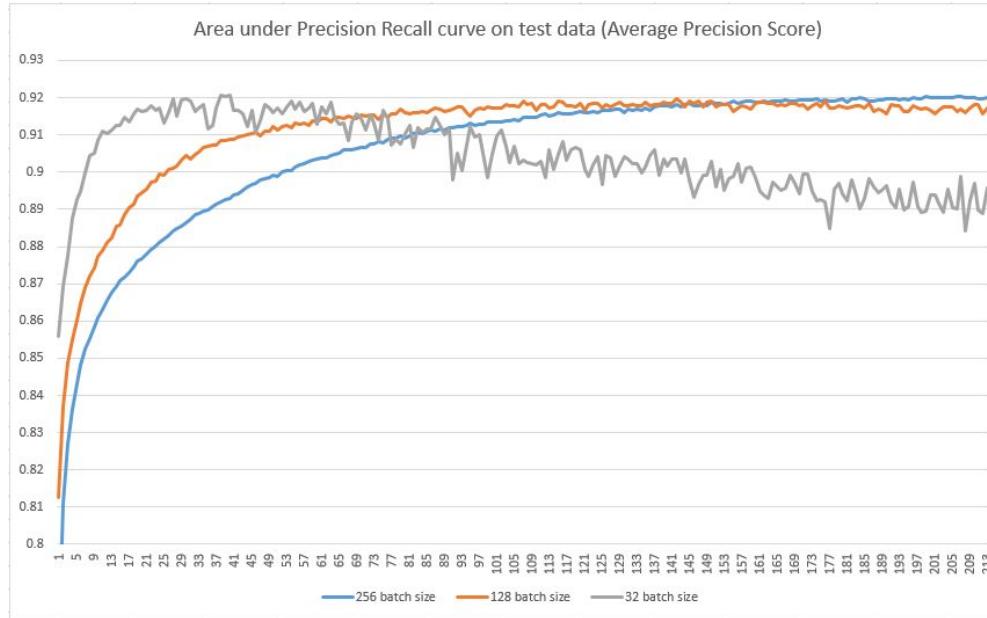
You wouldn't want to overfit your model to a single sample.

For this reason, we use **batches!** A batch is a number of samples whose errors are aggregated and shown to the model altogether.

If a batch is of size one, we call that **stochastic gradient descent**.



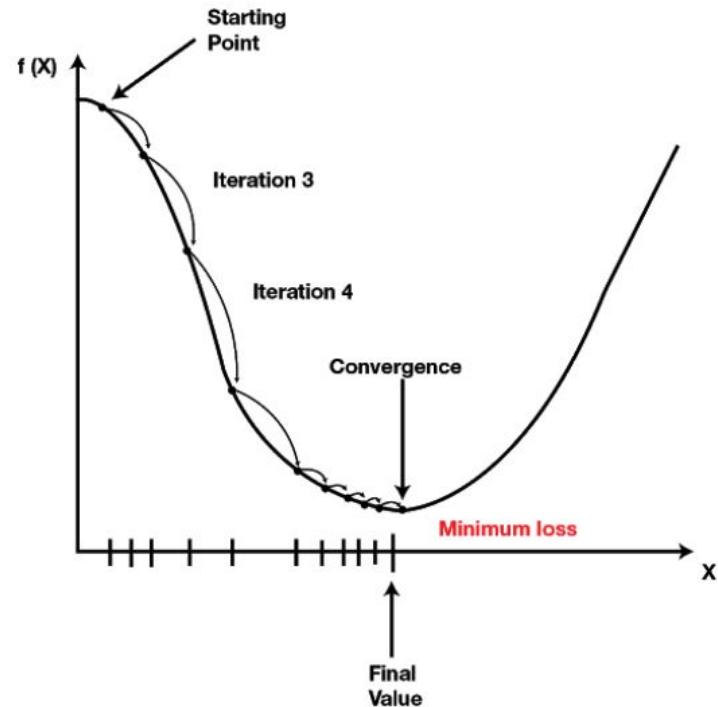
# How do we set these parameters?



# DNNs train via multiple epochs

The number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset.

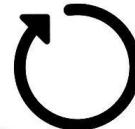
Because batches can be different across epochs, multiple epochs are useful when training.



# Epochs, Batches, and Iterations

## Epoch :

An Epoch represent one iteration over the entire dataset.



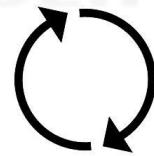
## Batch :

We cannot pass the entire dataset into the Neural Network at once. So, we divide the dataset into number of batches.



## Iteration :

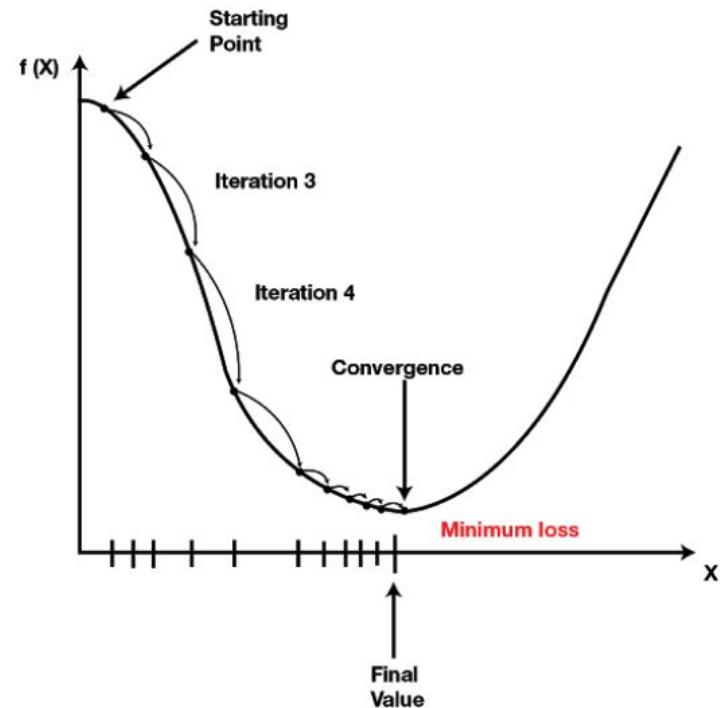
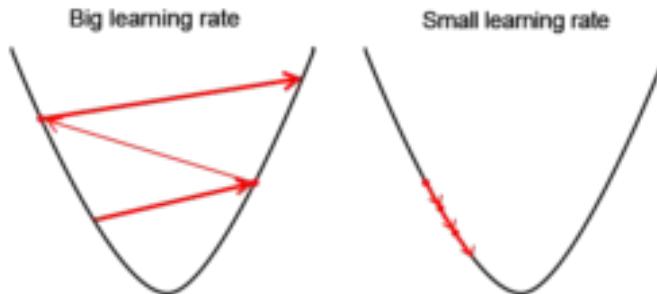
If we have 1000 images as Data and a batch size of 20, then an Epoch should run  $1000/20 = 50$  iteration.



# What determines the jumps per iteration?

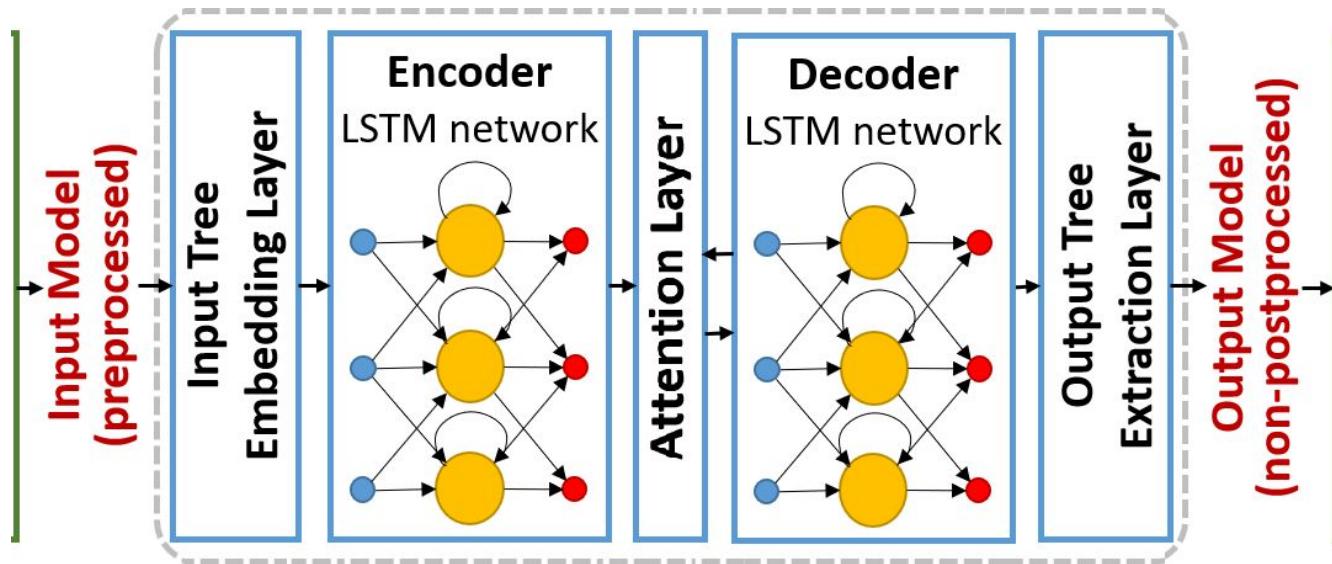
A **learning rate** is used to modulate the amount of corrections your algorithm has per training iteration.

Many algorithms exist to help you set the right learning rate.



# A quick word on NLP

Natural Language Processing (NLP) prediction problems can also be solved with Neural Networks. Popular architectures look like the one below.

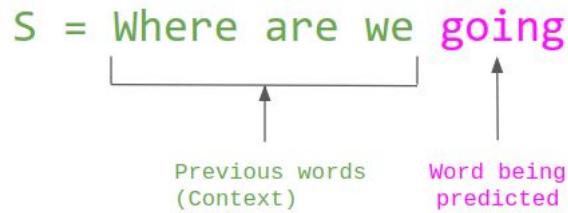


# How do they work?

They work via a Language Model - an assigned probability for words that fill in the blank.

Large language models like GTP-3 scan the entire internet to have an accurate and up-to-date language model.

Can do impressive information retrieval.



$$P(S) = P(\text{Where}) \times P(\text{are} \mid \text{Where}) \times P(\text{we} \mid \text{Where are}) \times P(\text{going} \mid \text{Where are we})$$

# Exercises!

# Resources

<https://cs231n.github.io/convolutional-networks/>

[https://github.com/Spandan-Madan/DeepLearningProject/blob/master/notebooks/Deep\\_Learning\\_Project.ipynb](https://github.com/Spandan-Madan/DeepLearningProject/blob/master/notebooks/Deep_Learning_Project.ipynb)

<https://keras.io/examples/>

<https://blog.statsbot.co/neural-networks-for-beginners-d99f2235efca>