

Deep learning methods in population genomics and phylogeography

Manolo Fernandez Perez
Department of Life Sciences, Imperial College London

@ManoloLearning 
manolofperez@gmail.com 
sites.google.com/site/manolofperez 

Goals

- Conceive and simulate genetic data under competing demographic scenarios 
- Understand deep learning background and how a CNN works
- How to use deep learning to compare demographic scenarios

Program

Program

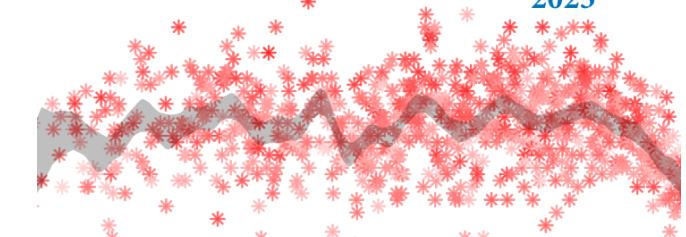
- *Part I: A gentle introduction to supervised deep learning and how to apply it to genomic data.*
- *Part II: Understanding the basic CNN architecture for image recognition.*
- *Part III: Practical on Image classification with CNN.*

Part I: A gentle introduction to supervised deep learning.

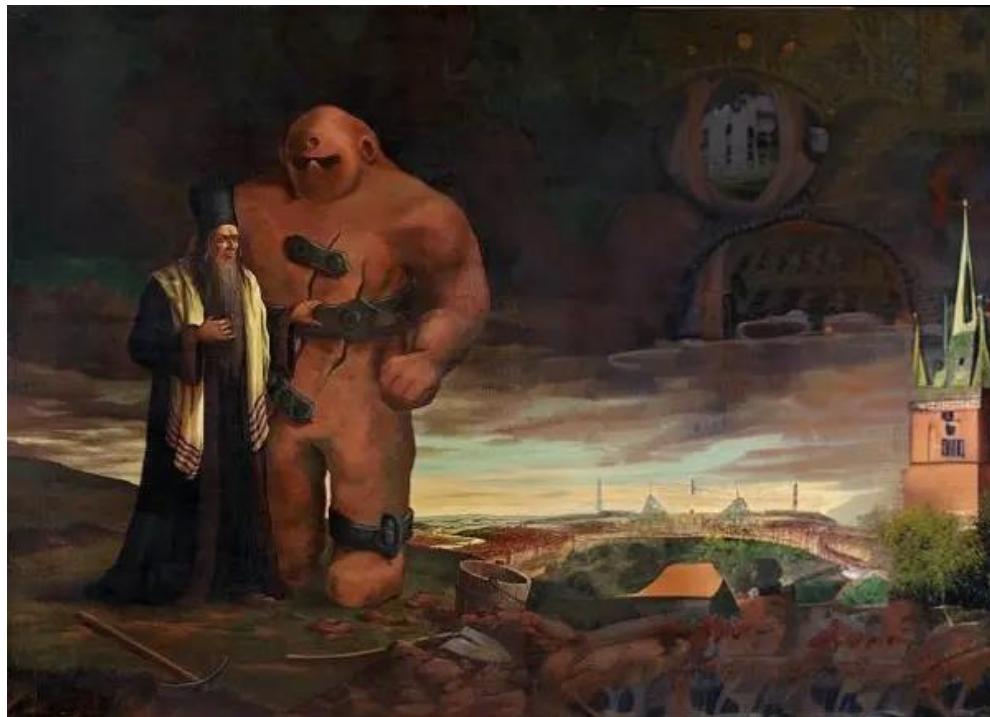
Credit for some slides:
Matteo Fumagalli and Flora Jay

The Golem of Prague

Statistical Rethinking
2023



[https://github.com/rmcelreath/
stat_rethinking_2023](https://github.com/rmcelreath/stat_rethinking_2023)



https://trips-tickets.com/the_golem_of_prague/



[https://www.atlantajewishtimes.com/
the-golem-of-prague/](https://www.atlantajewishtimes.com/the-golem-of-prague/)

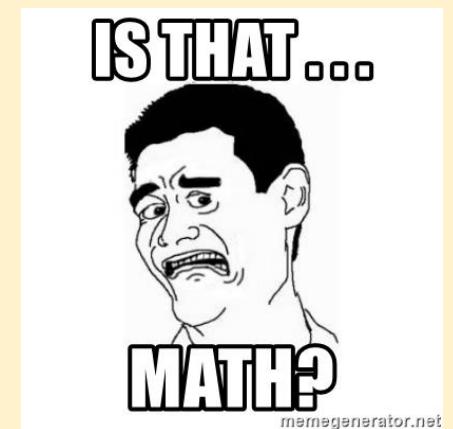
Likelihood-free methods

Complex evolutionary processes -> too many possible models and parameters to estimate the likelihood function (ML and Bayesian methods).

Post. Prob.

Likelihood Prior

$$\Pr(M|data) = \frac{\Pr(data|M) * \Pr(M)}{\Pr(data)}$$



Likelihood-free methods

Possible solutions:

Use a few (simple) models for all species/datasets
(IM; Migrate-n...)

Likelihood-free methods

- ABC;
- Approximate Likelihoods;
- Deep Learning



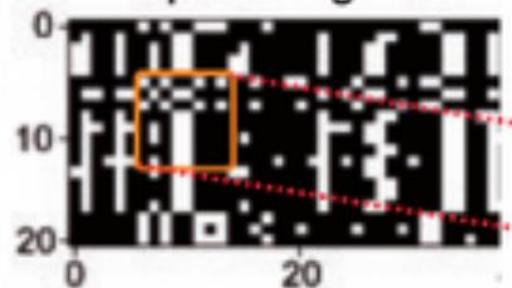
What is machine learning?

TASK:
predict y from x



Angermueller et al Mol Syst Biol. (2016) 12: 878

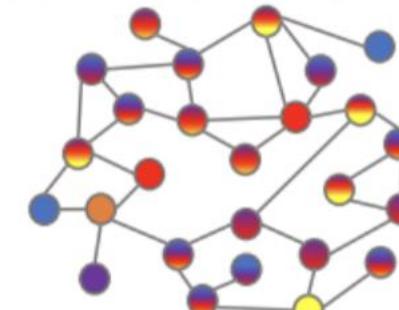
What is the data?



Images

Flagel et al. (2019) *MBE*

Gene interaction network



Graphs

Li et al. (2022) *Nat Biom Eng*

123



[text]

Numerical
Data

Categorical
Data

Time Series
Data

Text
Data

What is the model?

Unsupervised / Supervised Tasks

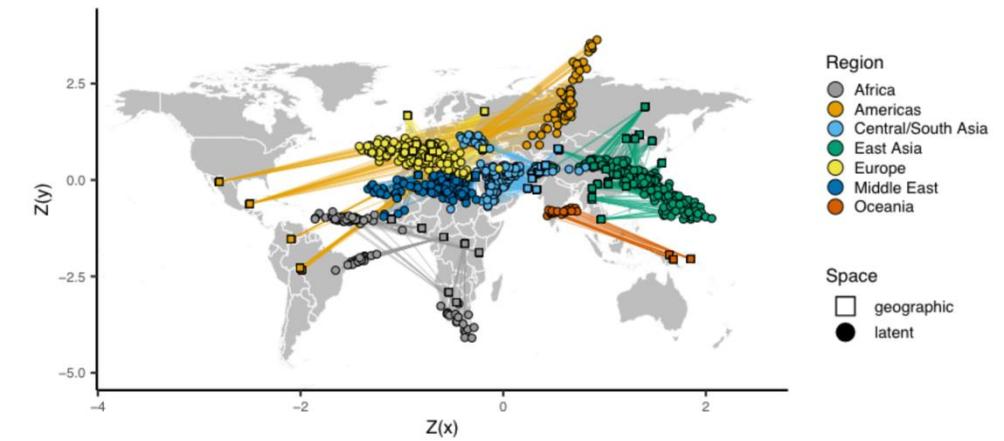
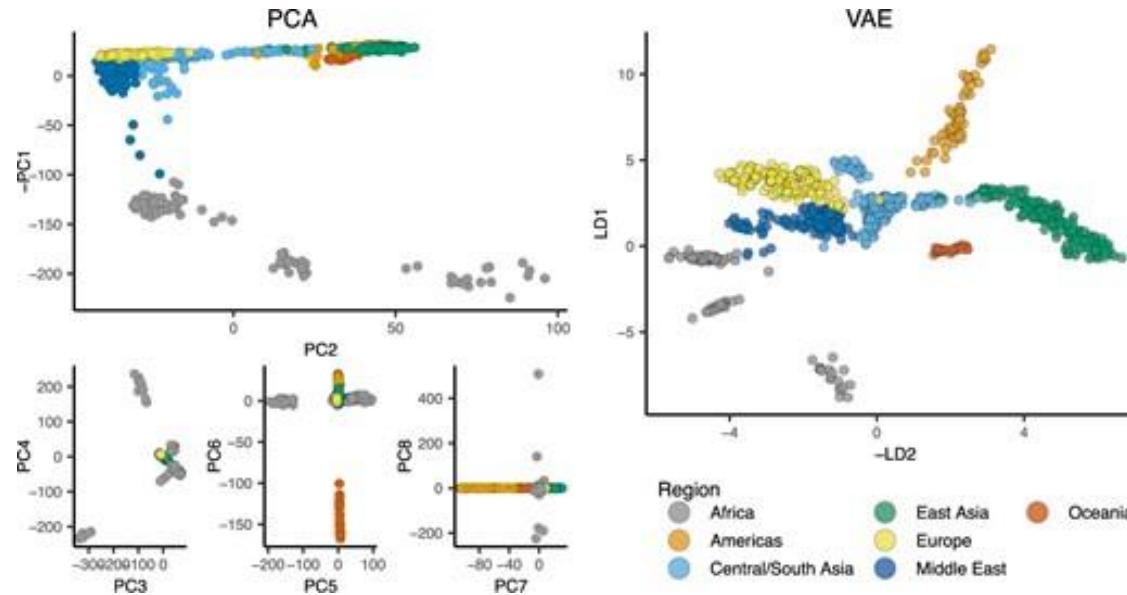
- Learning something from **data**
- Either **unsupervised** (no labels) or **supervised** (discrete or continuous labels)
- What's a label ? a target class (discrete) or a target value (continuous) observed for each sample
Data are not always labeled. They can also have multiclass labels
ex : pic of dog/person/car..., price of house, level of cholesterol

What is the model?

.Unsupervised = discovering patterns in data without prior knowledge

You do NOT have labels, or you do NOT use them

- Dimension reduction methods, e.g. PCA, Matrix factorization
- Clustering algorithms, e.g. K-means, hierarchical clustering, ...
- Outlier detection (can be then used for filtering, ..)
- ...



Battey et al. (2021) G3

What is the model?

- Supervised = Learn a relationship (a general model) linking input data (or features) to observed labels

Classification (predict a class)



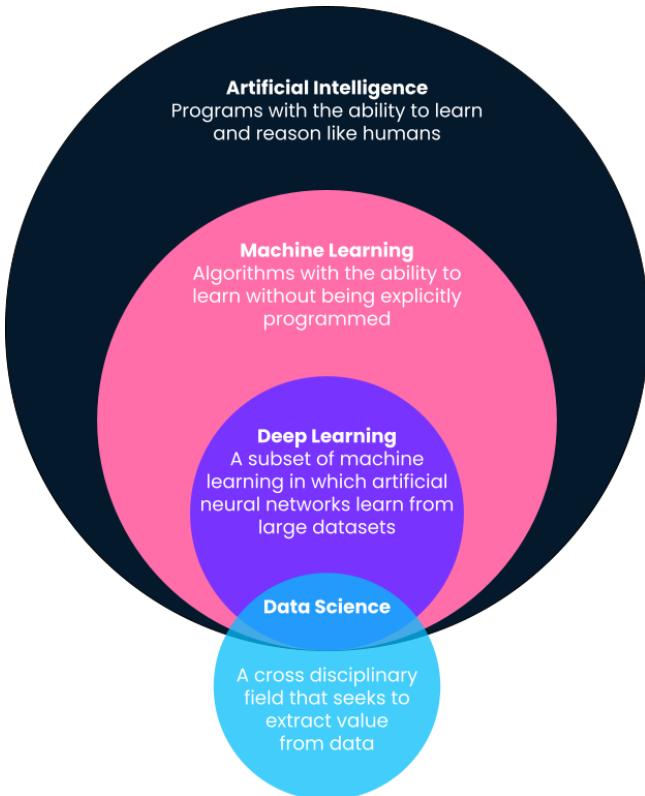
Regression (predict a variable)



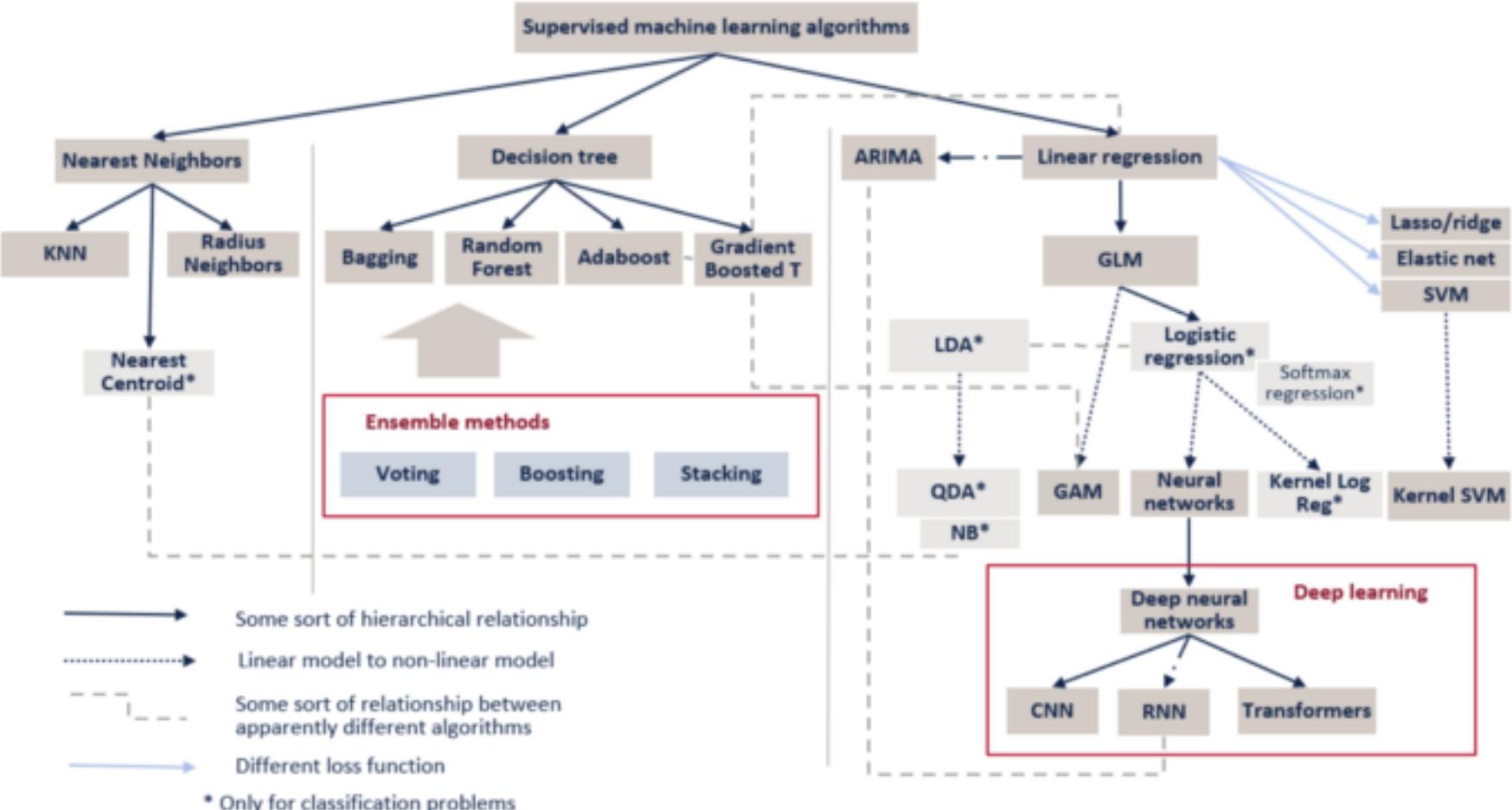
What for:

- Predict labels of new unlabeled samples (eg what's on an image?),
- Understand better the relationship between features and the label (eg understand which set of genes allow to predict a disease risk),
- ...

Deep Neural Networks



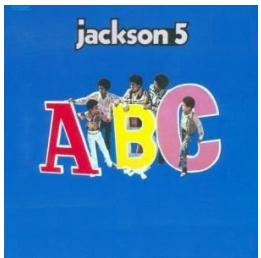
<https://www.datacamp.com/blog/what-is-machine-learning>



<https://towardsdatascience.com/overview-of-supervised-machine-learning-algorithms-a5107d036296>

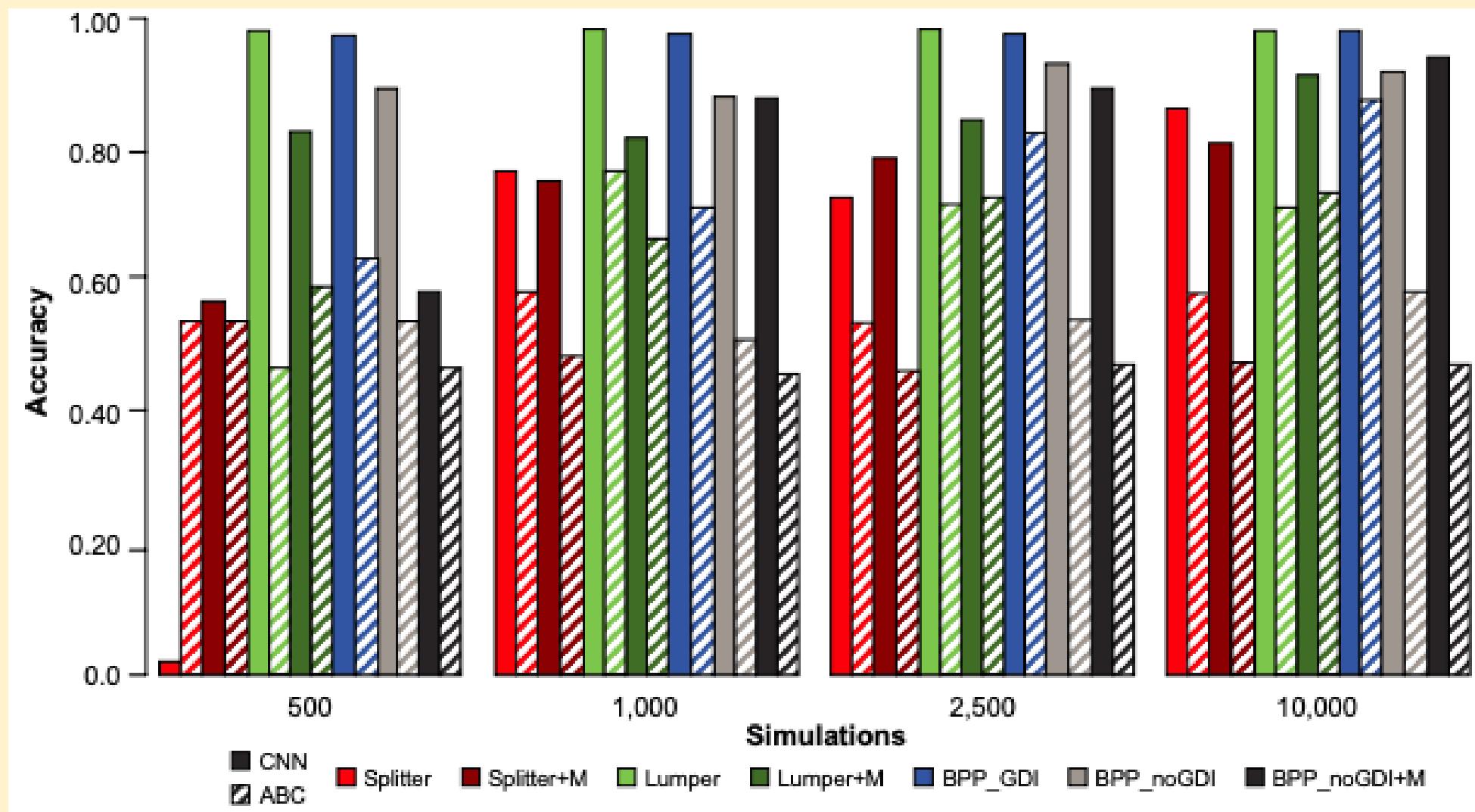
Why Deep Learning?

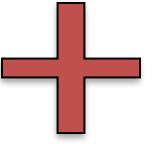
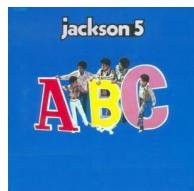
Deep Learning



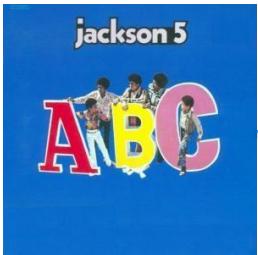
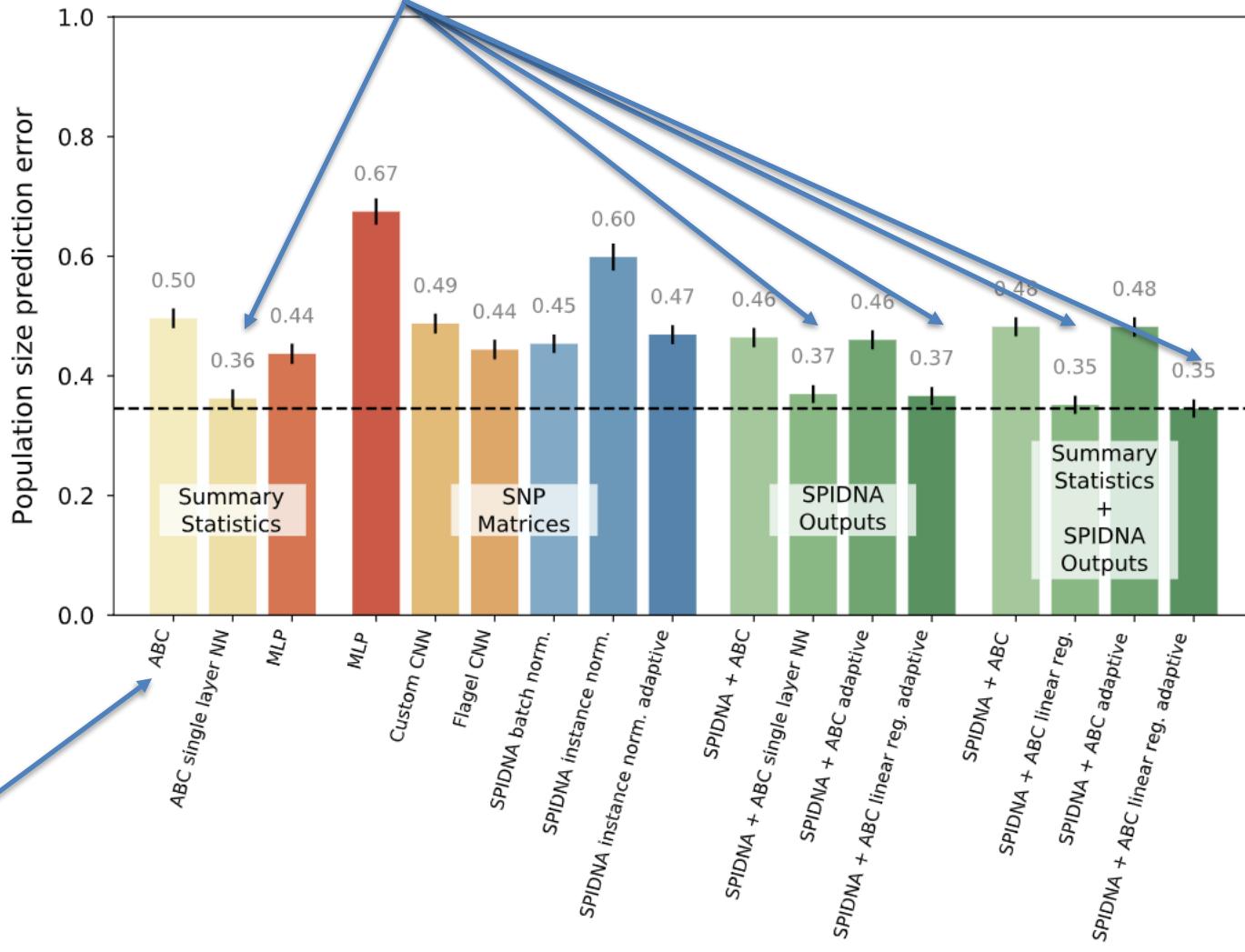
CJ Battey
Twitter 2018

Why Deep Learning?

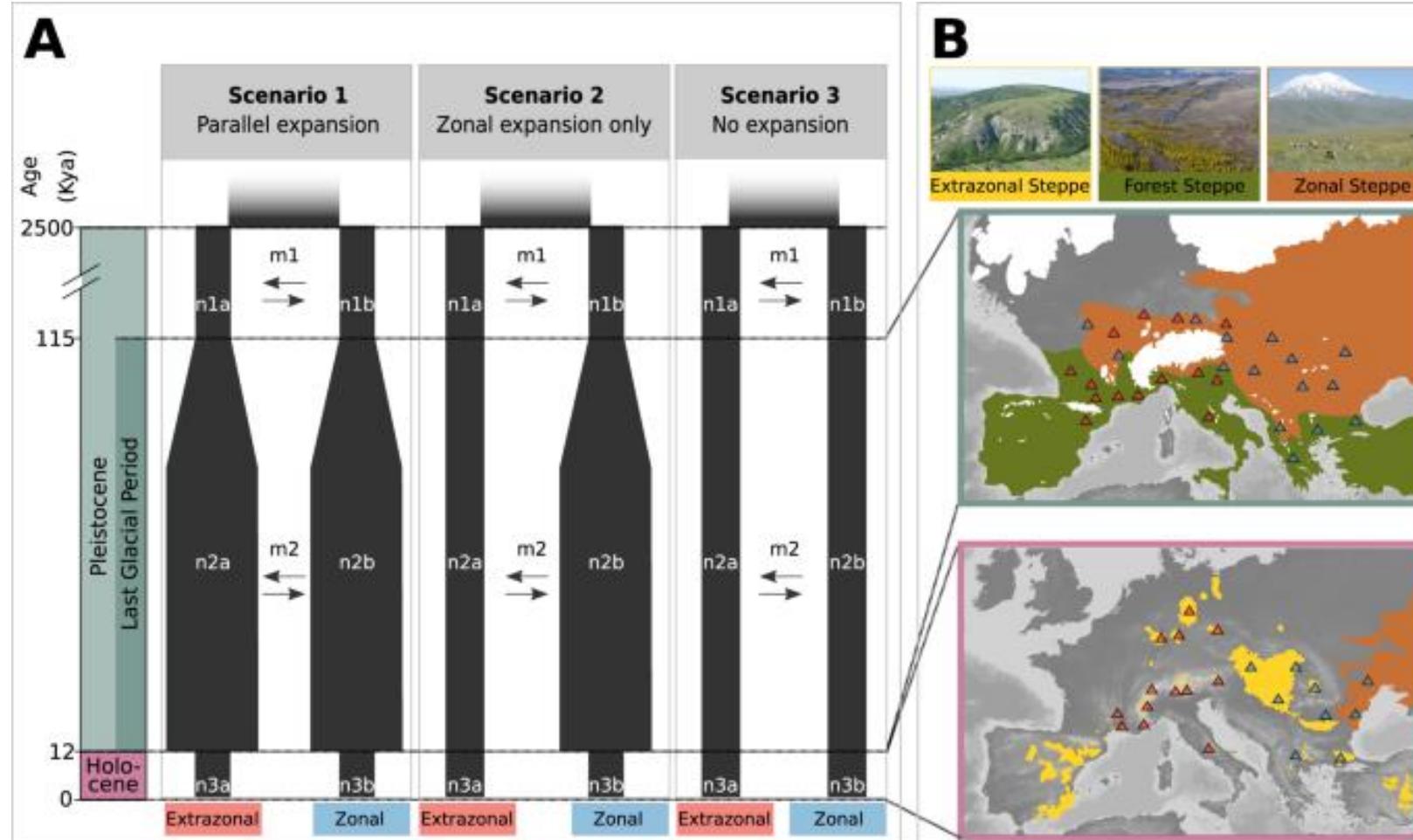




Why Deep Learning?



Supervised Deep Learning ~ ABC



Simulate Genetic Data – ms (Hudson 2002)

BIOINFORMATICS APPLICATIONS NOTE

Vol. 18 no. 2 2002
Pages 337–338



Generating samples under a Wright–Fisher neutral model of genetic variation

Richard R. Hudson

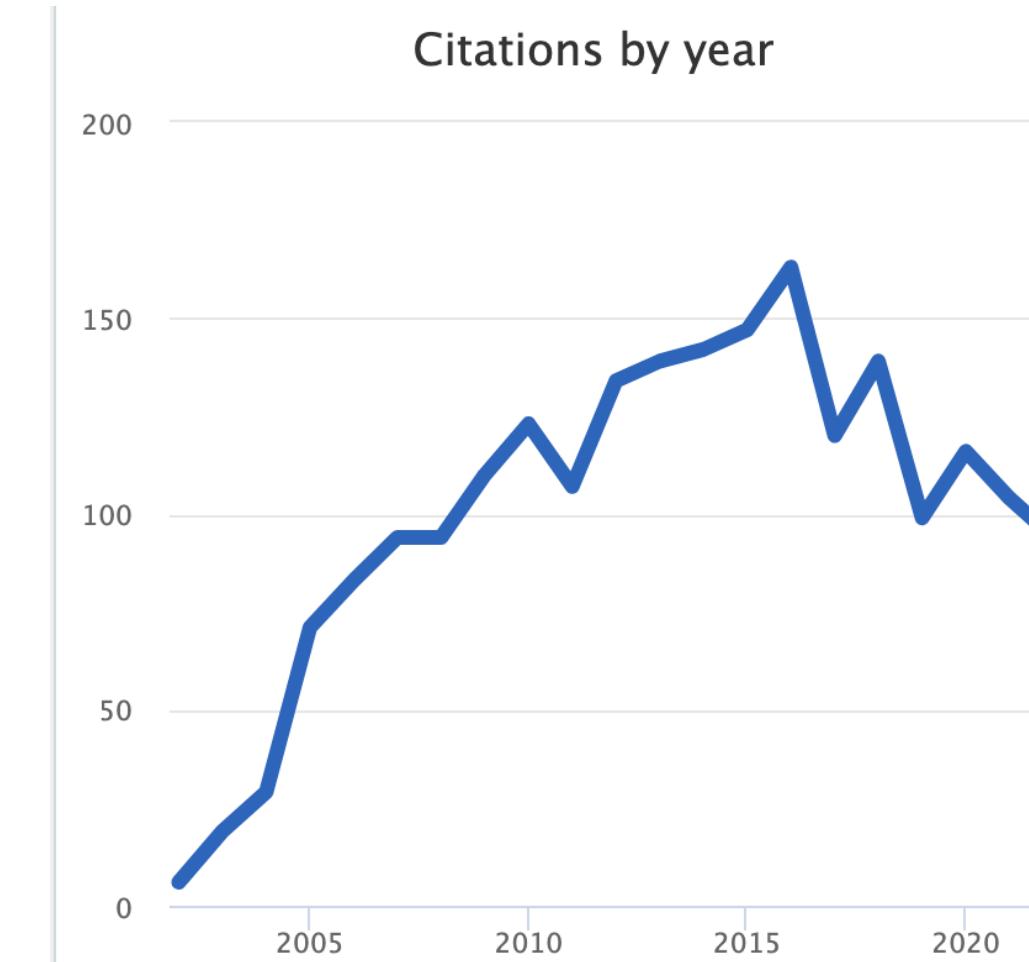
Department of Ecology and Evolution, University of Chicago, 1101 E. 57th Street,
Chicago, IL 60637, USA

Received on August 8, 2001; revised and accepted on September 13, 2001

- Seminal simulator.
- Allows to fix the number of segregating sites.

2,142 citations

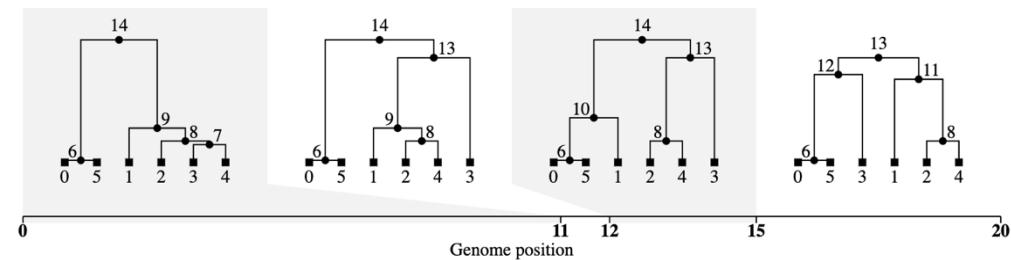
Citations by year



Simulate Genetic Data – other options

■ TABLE 6.1 Programs for simulating population samples of DNA sequences

PROGRAM	SOURCE
COALESCENT ("BACKWARD") SIMULATION SOFTWARE	
<i>ms</i>	Hudson 1990; Hudson 2002
<i>GENOME</i>	Liang et al. 2007
<i>SIMCOAL/SIMCOAL 2.0</i>	Excoffier et al. 2000; Laval and Excoffier 2004
<i>CoaSim</i>	Mailund et al. 2005
<i>Recodon</i>	Arenas and Posada 2007
<i>discoal</i>	Kern and Schrider 2016
<i>msprime</i>	Kelleher et al. 2016
WRIGHT-FISHER ("FORWARD") SIMULATION SOFTWARE:	
<i>EASYPOL</i>	Baloux 2001
<i>simuPop</i>	Peng and Kimmel 2005
<i>Nemo</i>	Guillaume and Rougemont 2006
<i>FREGENE</i>	Hoggart et al. 2007; Chadeau-Hyam et al. 2008
<i>ForSim</i>	Lambert et al. 2008
<i>FORWSIM</i>	Padhukasahasram et al. 2008
<i>GENOMEPOP</i>	Carvajal-Rodriguez 2008
<i>SFS_CODE</i>	Hernandez 2008
<i>FPopSim</i>	Zanini and Neher 2012
<i>fwdpp</i>	Thornton 2014
<i>SLiM/SLiM 2</i>	Messer 2013; Haller and Messer 2017
<i>ARGON</i>	Palamara 2016
APPROXIMATE COALESCENT ("SIDeways") SIMULATION SOFTWARE	
<i>FastCoal</i>	McVean and Cardin 2005; Marjoram and Wall 2006
<i>MaCS</i>	Chen et al. 2009
<i>fastsimcoal/fastsimcoal2</i>	(Excoffier and Foll 2011; Excoffier et al. 2013)



Simulate Genetic Data – other options

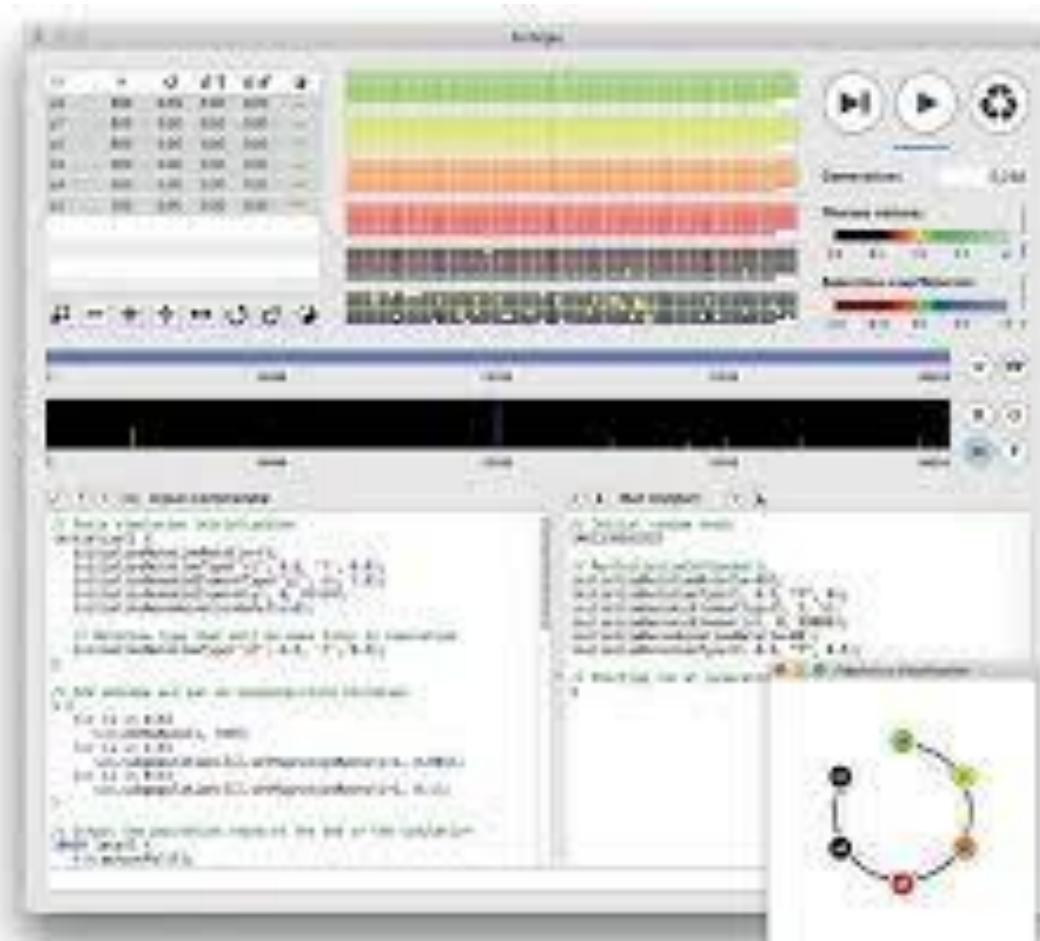
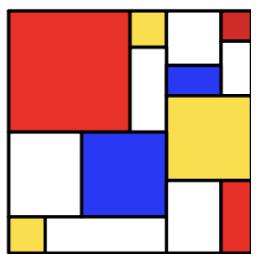
■ TABLE 6.1 Programs for simulating population samples of DNA sequences

PROGRAM	SOURCE
COALESCENT ("BACKWARD") SIMULATION SOFTWARE	
<i>ms</i>	Hudson 1990; Hudson 2002
<i>GENOME</i>	Liang et al. 2007
<i>SIMCOAL/SIMCOAL 2.0</i>	Excoffier et al. 2000; Laval and Excoffier 2004
<i>CoaSim</i>	Mailund et al. 2005
<i>Recodon</i>	Arenas and Posada 2007
<i>discoal</i>	Kern and Schrider 2016
<i>msprime</i>	Kelleher et al. 2016
WRIGHT-FISHER ("FORWARD") SIMULATION SOFTWARE:	
<i>EASYPOP</i>	Balloux 2001
<i>simuPop</i>	Peng and Kimmel 2005
<i>Nemo</i>	Guillaume and Rougemont 2006
<i>FREGENE</i>	Hoggart et al. 2007; Chadeau-Hyam et al. 2008
<i>ForSim</i>	Lambert et al. 2008
<i>FORWSIM</i>	Padhukasahasram et al. 2008
<i>GENOMEPOP</i>	Carvajal-Rodriguez 2008
<i>SFS_CODE</i>	Hernandez 2008
<i>FFPopSim</i>	Zanini and Neher 2012
<i>fwdpp</i>	Thornton 2014
<i>SLiM/SLiM 2</i>	Messer 2013; Haller and Messer 2017
<i>ARGON</i>	Palamara 2016
APPROXIMATE COALESCENT ("SIDeways") SIMULATION SOFTWARE	
<i>FastCoal</i>	McVean and Cardin 2005; Marjoram and Wall 2006
<i>MaCS</i>	Chen et al. 2009
<i>fastsimcoal/fastsimcoal2</i>	(Excoffier and Foll 2011; Excoffier et al. 2013)

SLiM: An Evolutionary Simulation Framework

Benjamin C. Haller and Philipp W. Messer

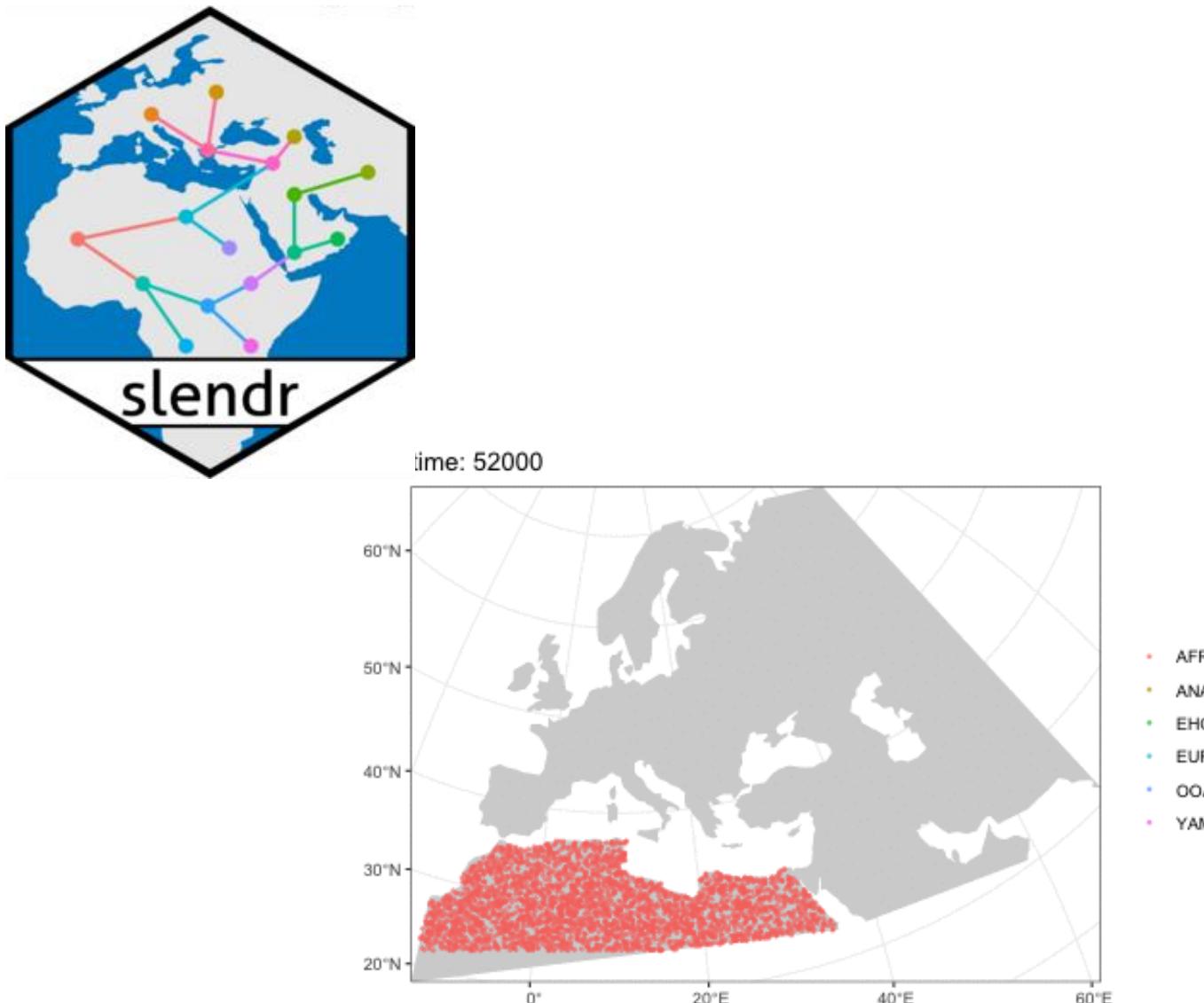
Dept. of Computational Biology
Cornell University, Ithaca, NY 14853
Correspondence: bhaller@benhaller.co



Simulate Genetic Data – other options

■ TABLE 6.1 Programs for simulating population samples of DNA sequences

PROGRAM	SOURCE
COALESCENT ("BACKWARD") SIMULATION SOFTWARE	
<i>ms</i>	Hudson 1990; Hudson 2002
<i>GENOME</i>	Liang et al. 2007
<i>SIMCOAL/SIMCOAL 2.0</i>	Excoffier et al. 2000; Laval and Excoffier 2004
<i>CoaSim</i>	Mailund et al. 2005
<i>Recodon</i>	Arenas and Posada 2007
<i>discoal</i>	Kern and Schrider 2016
<i>msprime</i>	Kelleher et al. 2016
WRIGHT-FISHER ("FORWARD") SIMULATION SOFTWARE:	
<i>EASYPop</i>	Balloux 2001
<i>simuPop</i>	Peng and Kimmel 2005
<i>Nemo</i>	Guillaume and Rougemont 2006
<i>FREGENE</i>	Hoggart et al. 2007; Chadeau-Hyam et al. 2008
<i>ForSim</i>	Lambert et al. 2008
<i>FORWSIM</i>	Padhukasahasram et al. 2008
<i>GENOMEPOP</i>	Carvajal-Rodriguez 2008
<i>SFS_CODE</i>	Hernandez 2008
<i>FFPopSim</i>	Zanini and Neher 2012
<i>fwdpp</i>	Thornton 2014
<i>SLiM/SLiM 2</i>	Messer 2013; Haller and Messer 2017
<i>ARGON</i>	Palamara 2016
APPROXIMATE COALESCENT ("SIDeways") SIMULATION SOFTWARE	
<i>FastCoal</i>	McVean and Cardin 2005; Marjoram and Wall 2006
<i>MaCS</i>	Chen et al. 2009
<i>fastsimcoal/fastsimcoal2</i>	(Excoffier and Foll 2011; Excoffier et al. 2013)





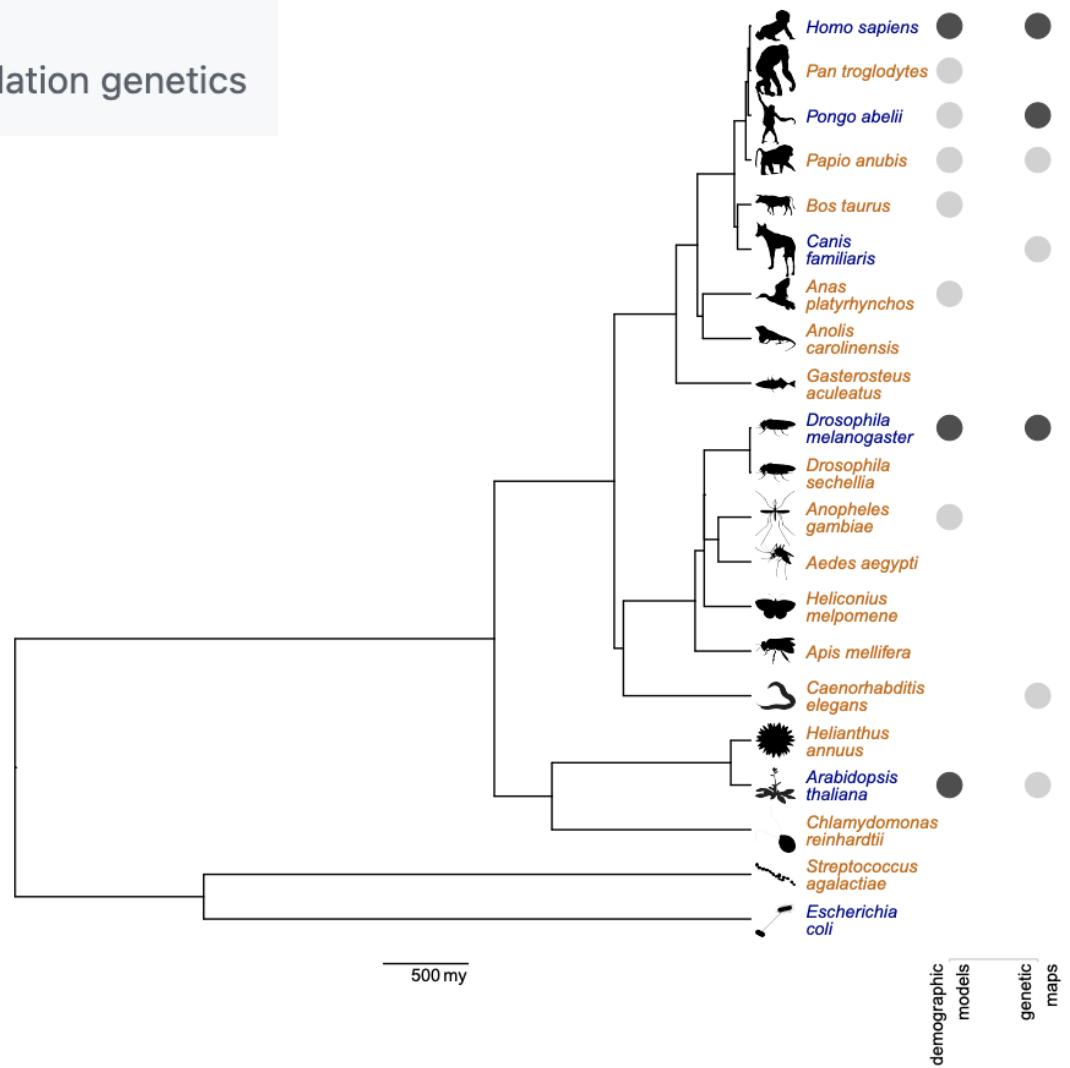
Simulate Genetic Data – other options

PopSim Consortium

A community-driven effort to standardize population genetics

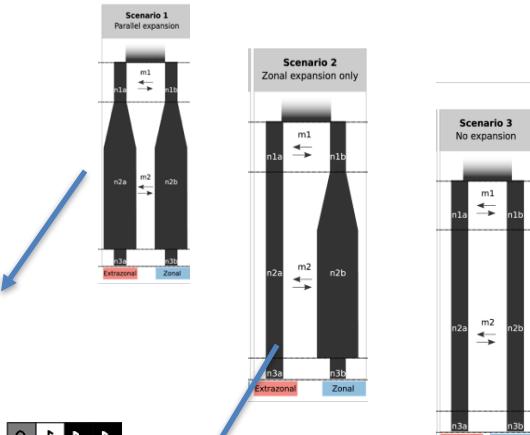
Expanding the `stdpopsim` species catalog, and lessons learned for realistic genome simulations

M. Elise Lauterbur^{1,+}, Maria Izabel A. Cavassim^{2,*}, Ariella L. Gladstein^{3,*}, Graham Gower^{4,*}, Nathaniel S. Pope^{5*}, Georgia Tsambos^{6,*}, Jeff Adrion^{5,7}, Saurabh Belsare⁵, Arjun Biddanda⁸, Victoria Caudill⁵, Jean Cury⁹, Ignacio Echevarria¹⁰, Benjamin C. Haller¹¹, Ahmed R. Hasan^{12,13}, Xin Huang^{14,15}, Leonardo Nicola Martin Iasi¹⁶, Ekaterina Noskova¹⁷, Jana Obšteter¹⁸, Vitor Antonio Corrêa Pavinato¹⁹, Alice Pearson^{20,21}, David Peede^{22,23}, Manolo F. Perez²⁴, Murillo F. Rodrigues⁵, Chris C. R. Smith⁵, Jeffrey P. Spence²⁵, Anastasia Teterina⁵, Silas Tittes⁵, Per Unneberg²⁶, Juan Manuel Vazquez²⁷, Ryan K. Waples²⁸, Anthony Wilder Wohns²⁹, Yan Wong³⁰, Franz Baumdicker³¹, Reed A. Cartwright³², Gregor Gorjanc³³, Ryan N. Gutenkunst³⁴, Jerome Kelleher³⁰, Andrew D. Kern⁵, Aaron P. Ragsdale³⁵, Peter L. Ralph^{5,36}, Daniel R. Schrider³⁷, and Ilan Gronau^{38,+}

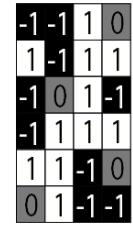
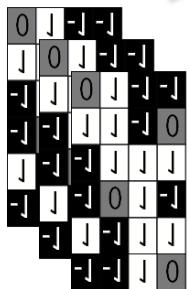
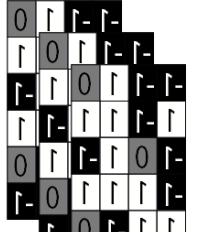


Supervised Deep learning ~ ABC

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



Scenario 1



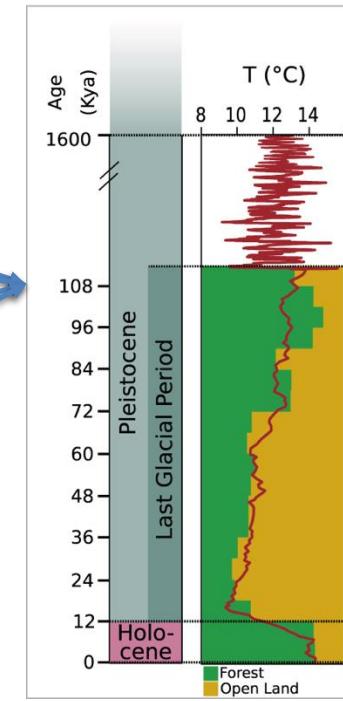
Data is labeled

Scenario 3

	θ	T1	T2	T3	Ne
Sim1					
Sim2					
Sim3					
Sim4					

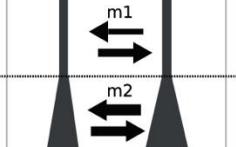
-1	-1	1	0
1	-1	1	1
-1	0	1	-1
-1	1	1	1
1	1	-1	0
0	1	-1	-1

0	J	-J	-J
J	0	J	-J
-J	J	0	J
J	-J	J	0
-J	J	-J	J
J	-J	J	-J
-J	J	-J	J



Omocestus petraeus

PP = 1.00



m_1

m_2

Extrazonal

Zonal

Parameters

1. Define Models

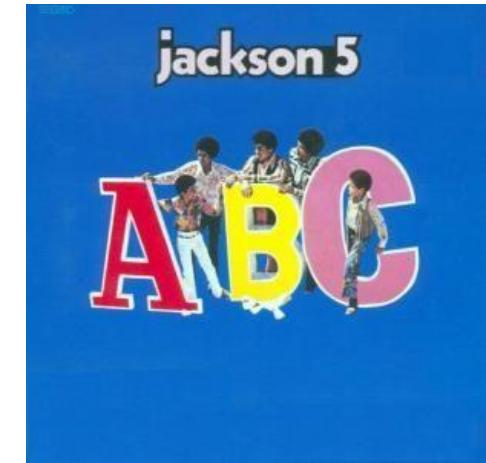
ABC

2. Sample parameters from a given distributions (prior)

3. Simulate data for each model using the sampled parameter values

4. Calculate Summary Statistics (SuSt) from simulations and from the empirical data

5. Compare simulated and empirical data retaining only simulations that are more similar.



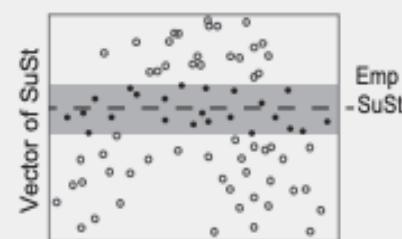
1) simulate each hypothesis



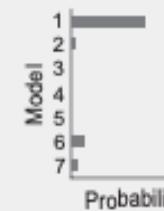
2b) extract SuSt from simulated data

SuSt		T
		S
		D
		θH
		H
		πW
		πB

3b) retain only 20% more similar simulations



4b) approximate the posterior



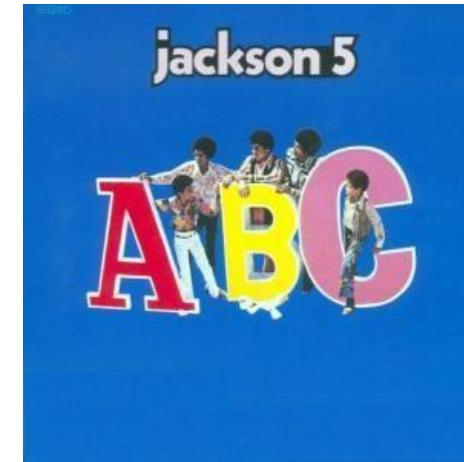
Modified from
Perez et al. (2022) *Mol Ecol Res*

ABC implies 3 approximations:

1. finite # simulations

2. informativeness of S

3. S don't match S^* exactly



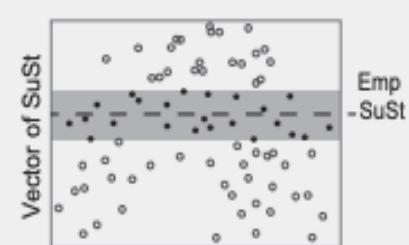
1) simulate each hypothesis



2b) extract SuSt from simulated data

SuSt		TT
		S
		D
		θH
		H
		πW
		πB

3b) retain only 20% more similar simulations



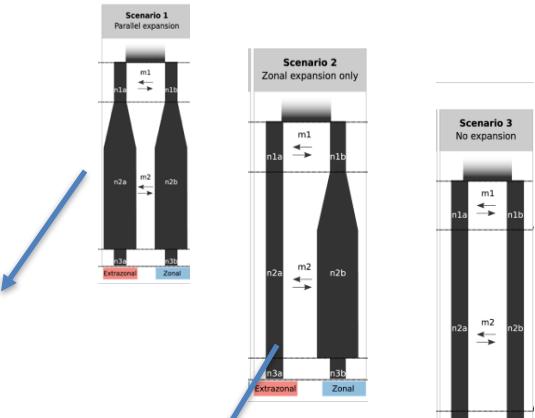
4b) approximate the posterior



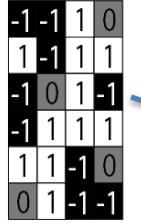
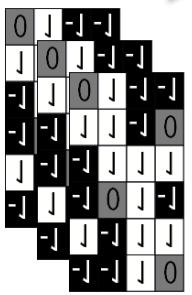
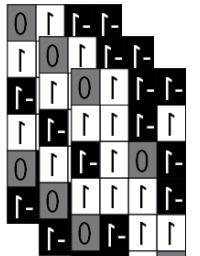
Modified from
Perez et al. (2022) *Mol Ecol Res*

Supervised Deep Learning ~ ABC

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

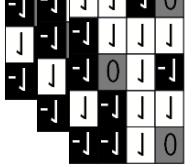


Scenario 1

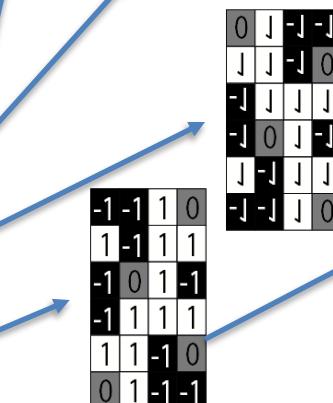
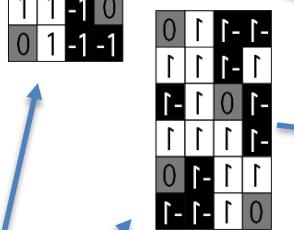


Data is labeled

Scenario 2



Scenario 3



Parameters

	θ	T1	T2	T3	Ne
Sim1					
Sim2					
Sim3					
Sim4					

DL witchcraft



Cuca - Brazilian Legend

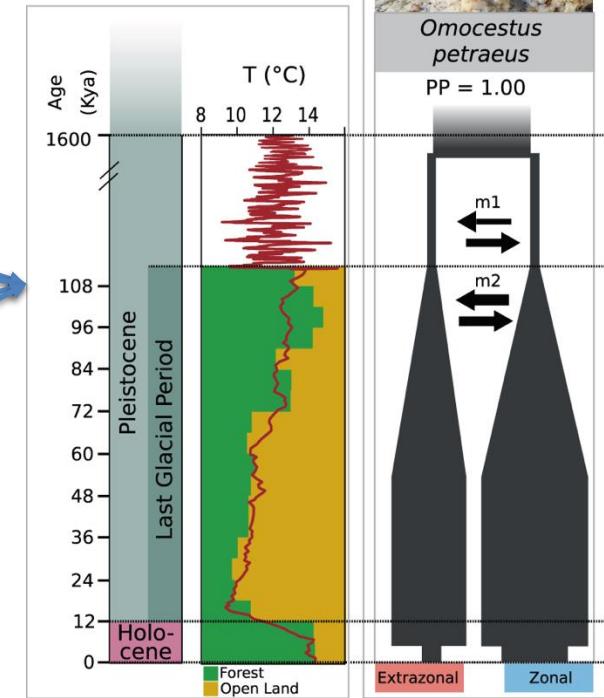


Image recognition

What the computer sees?

What do you see?



Challenges

Viewpoint variation



Scale variation



Deformation



Occlusion



Illumination conditions



Background clutter



Intra-class variation



Image recognition

Data-Driven approach (need a large training set)

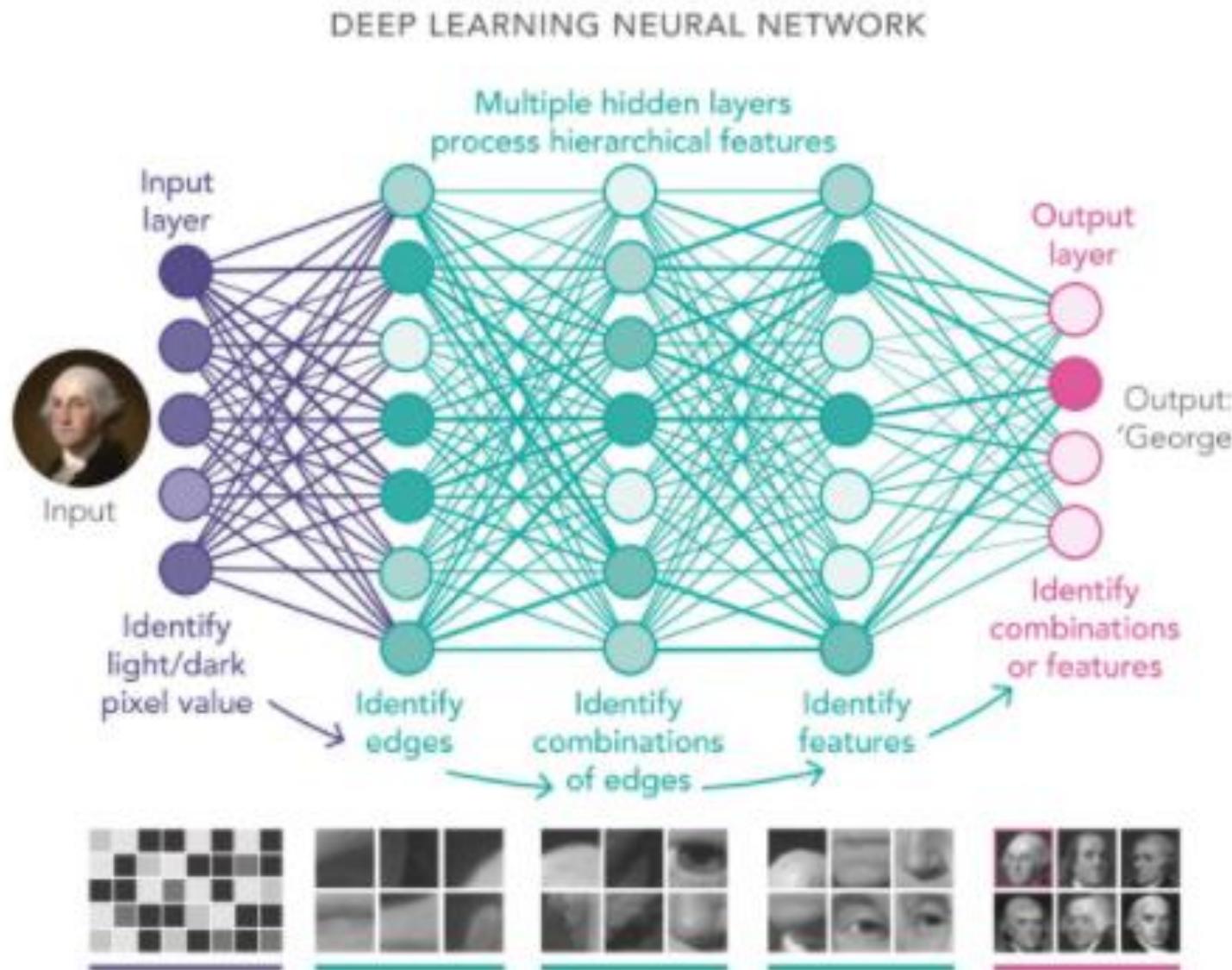
The MNIST dataset (70,000 hanwritten numbers)

0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9

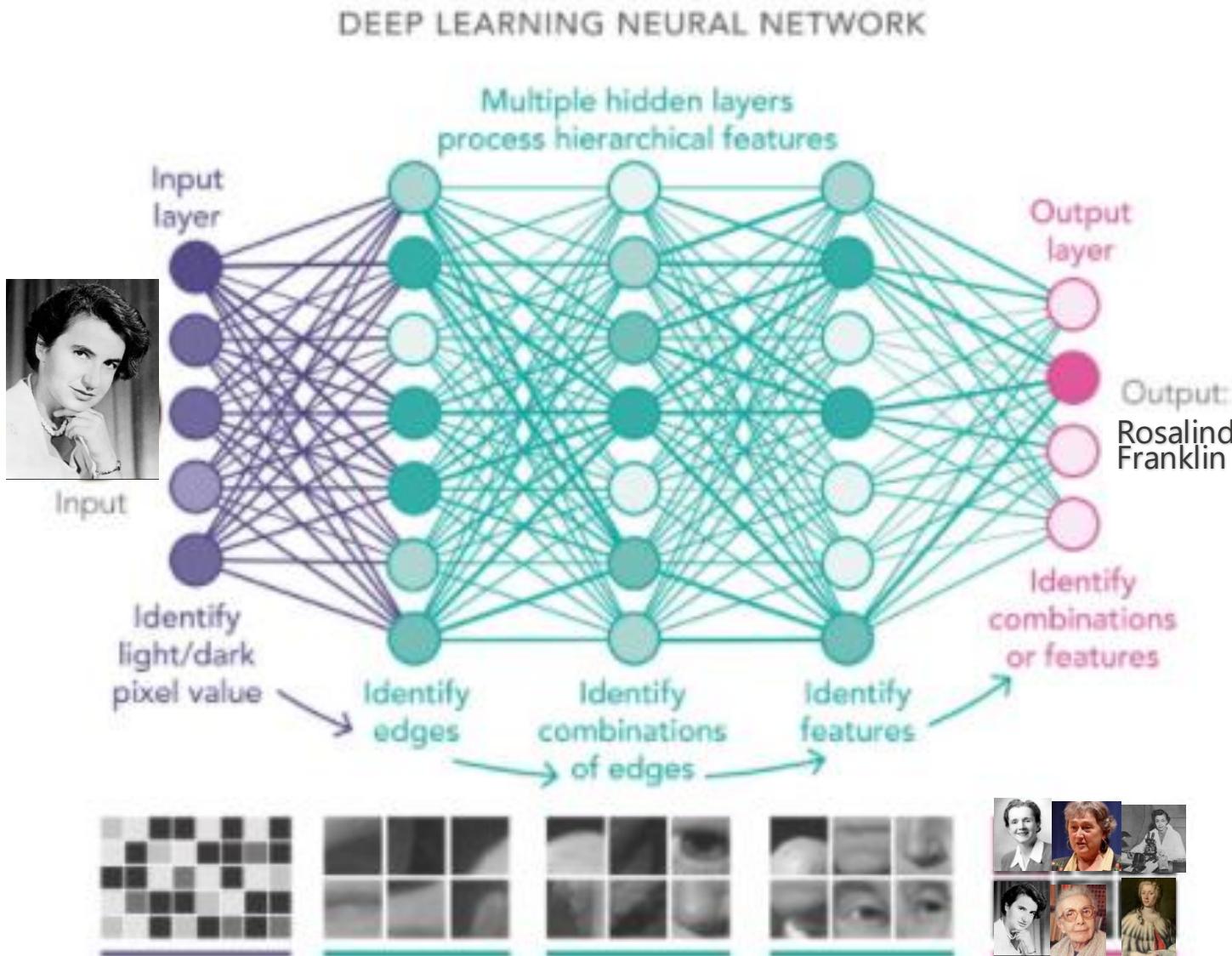


<https://blog.filestack.com/api/from-mnist-classification-to-intelligent-check-processing/>

Deep Neural Networks



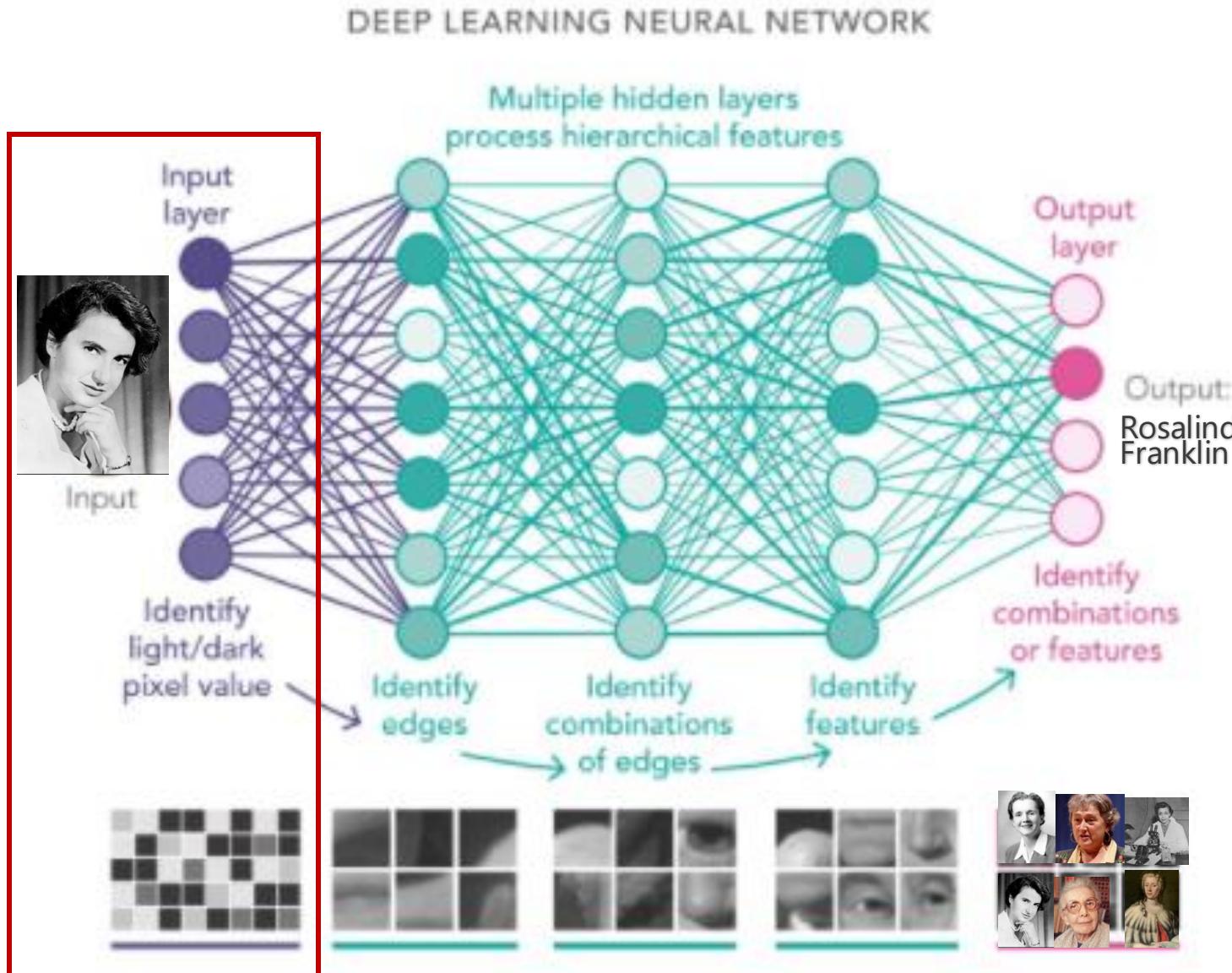
Deep Neural Networks



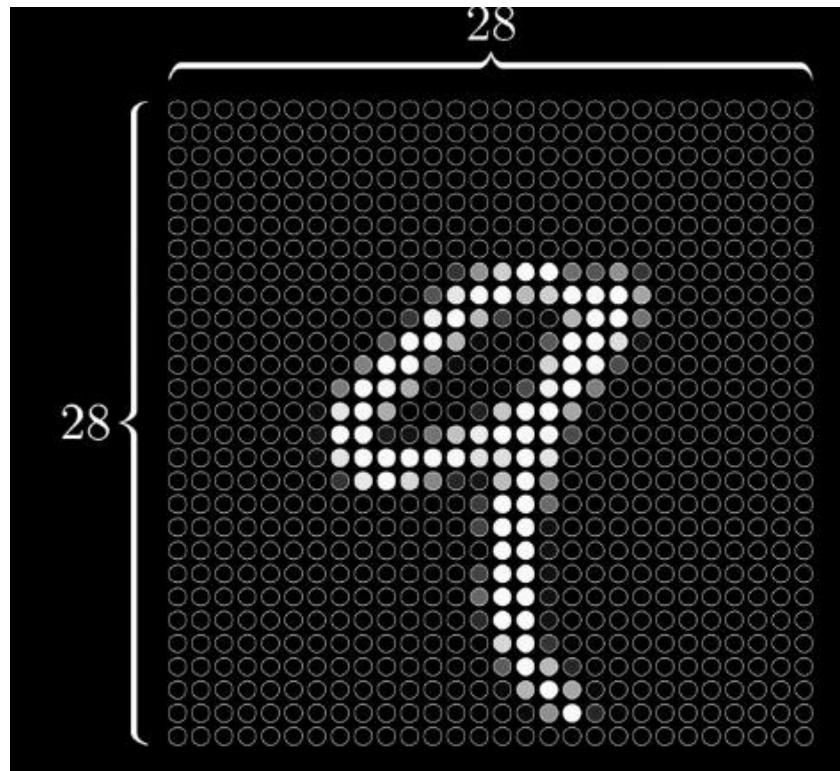
Multi-Layer Perceptron (MLP)

Adapted from Waldrop (2019) *PNAS*

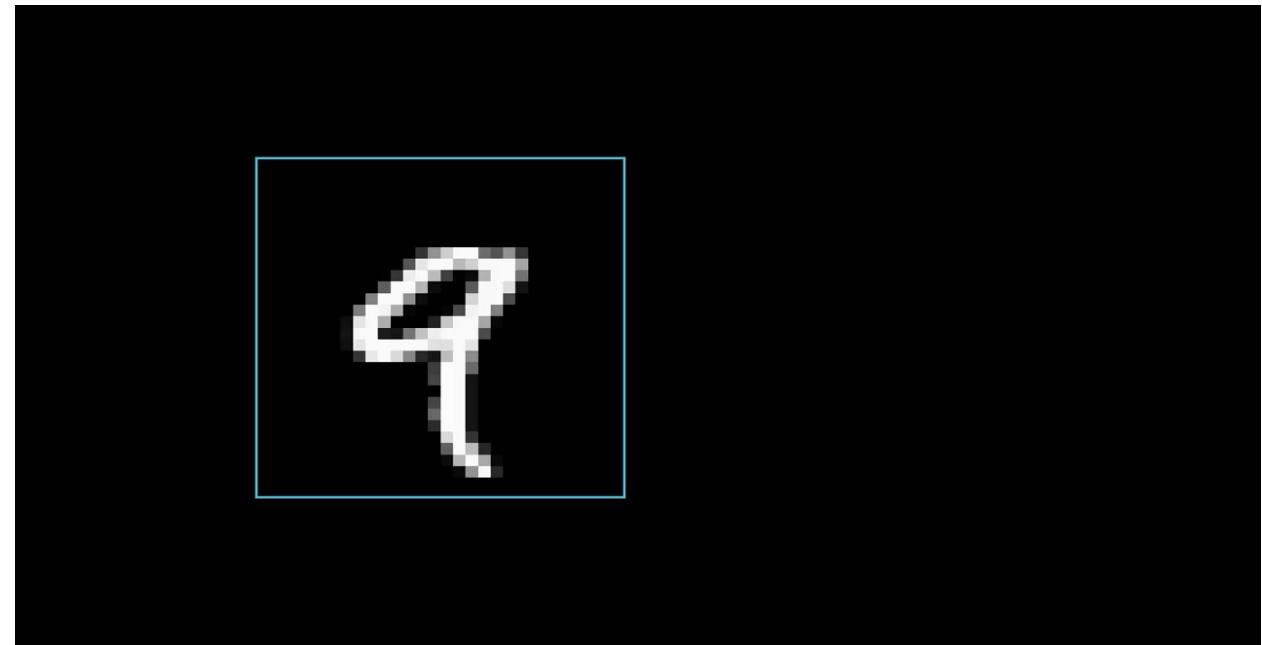
Deep Neural Networks



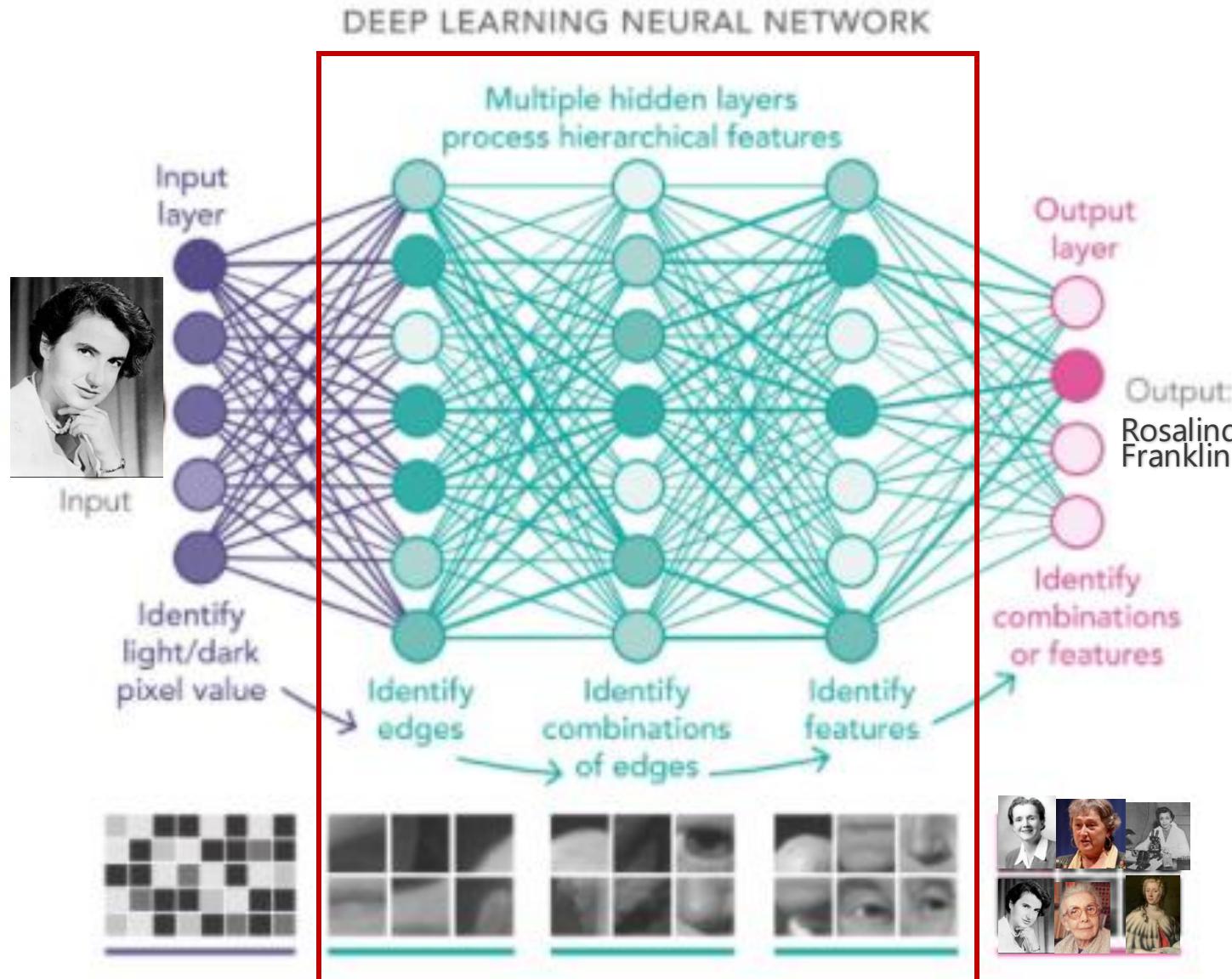
Adapted from Waldrop (2019)
PNAS



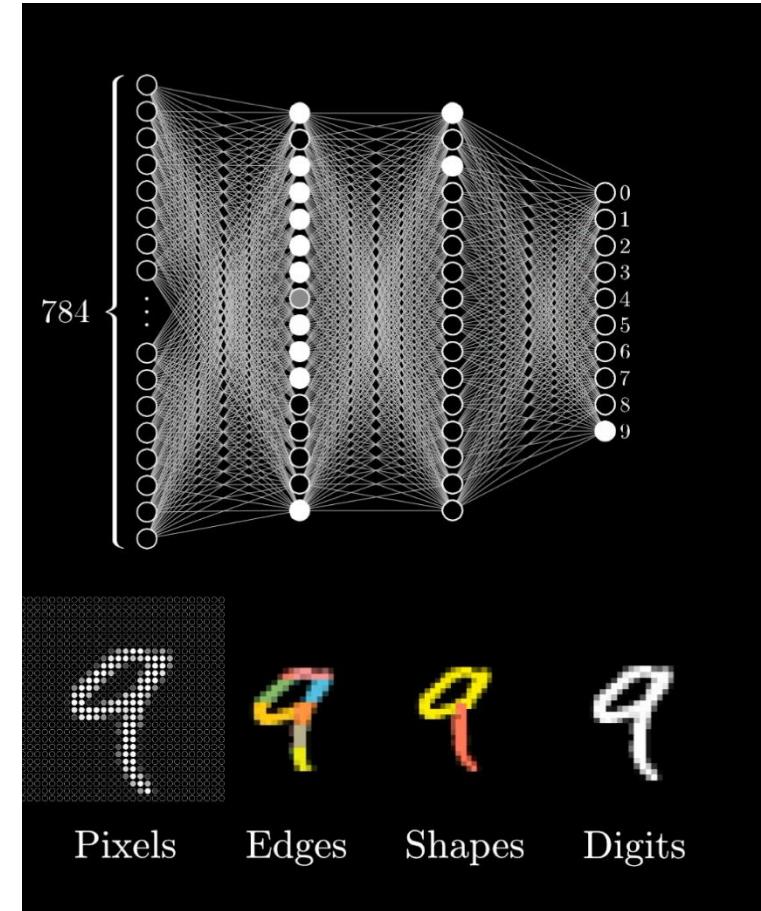
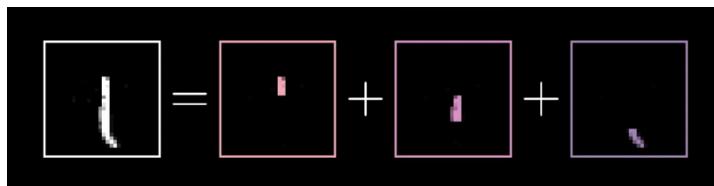
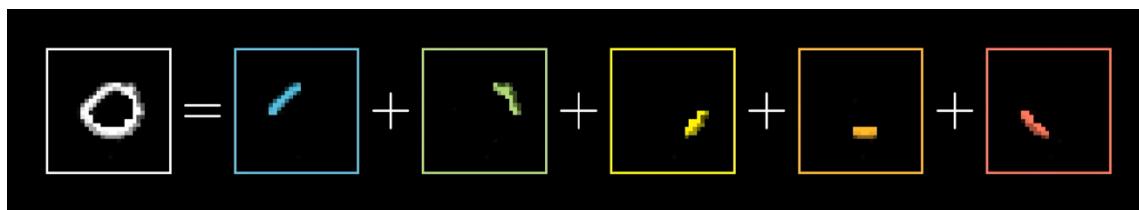
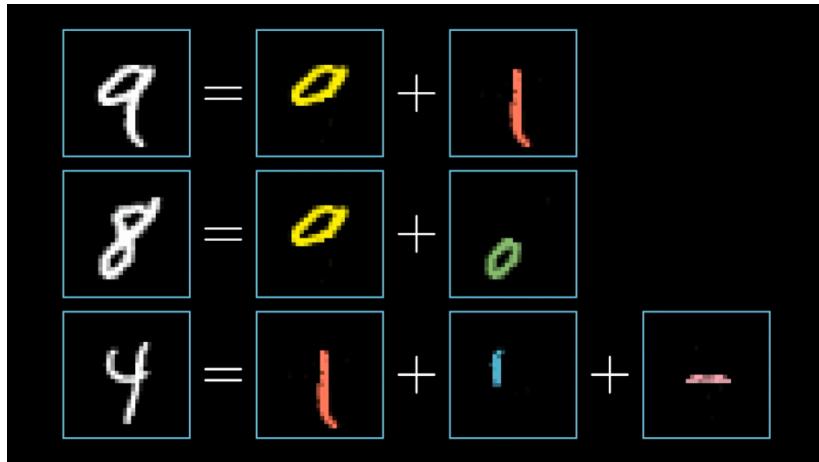
Input Layer



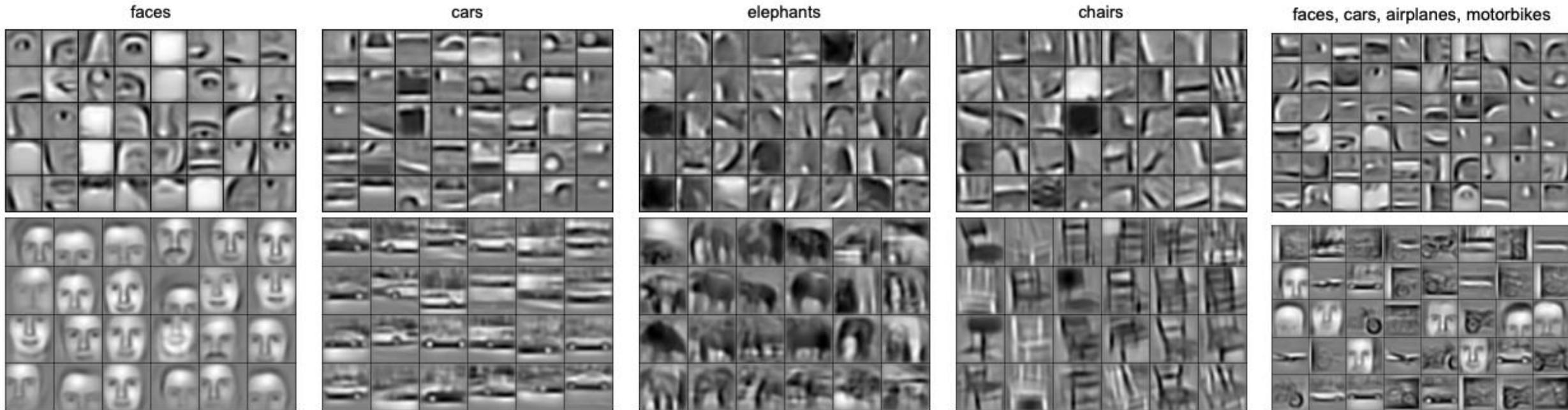
Deep Neural Networks



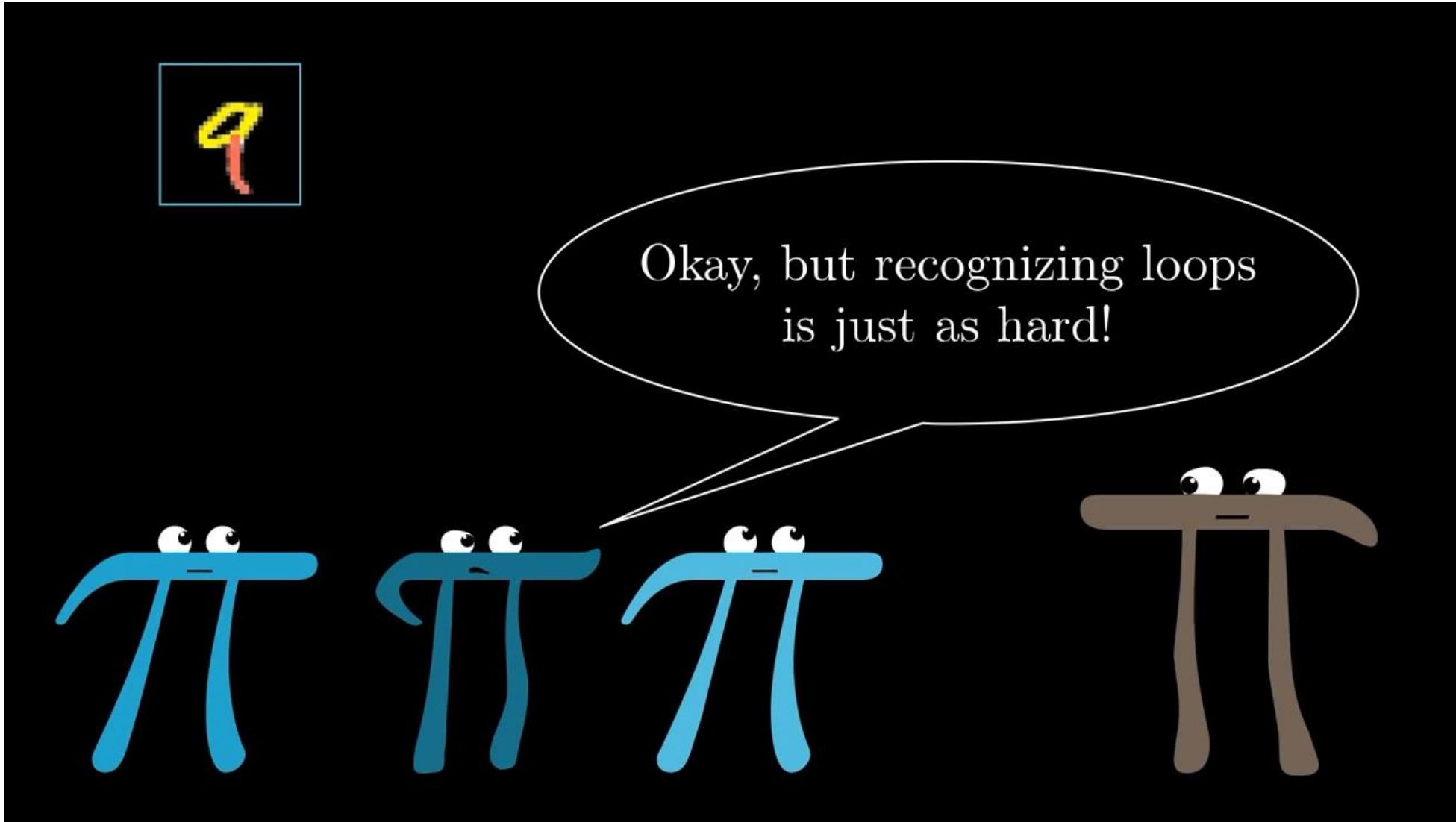
Hidden Layers process hierarchical features



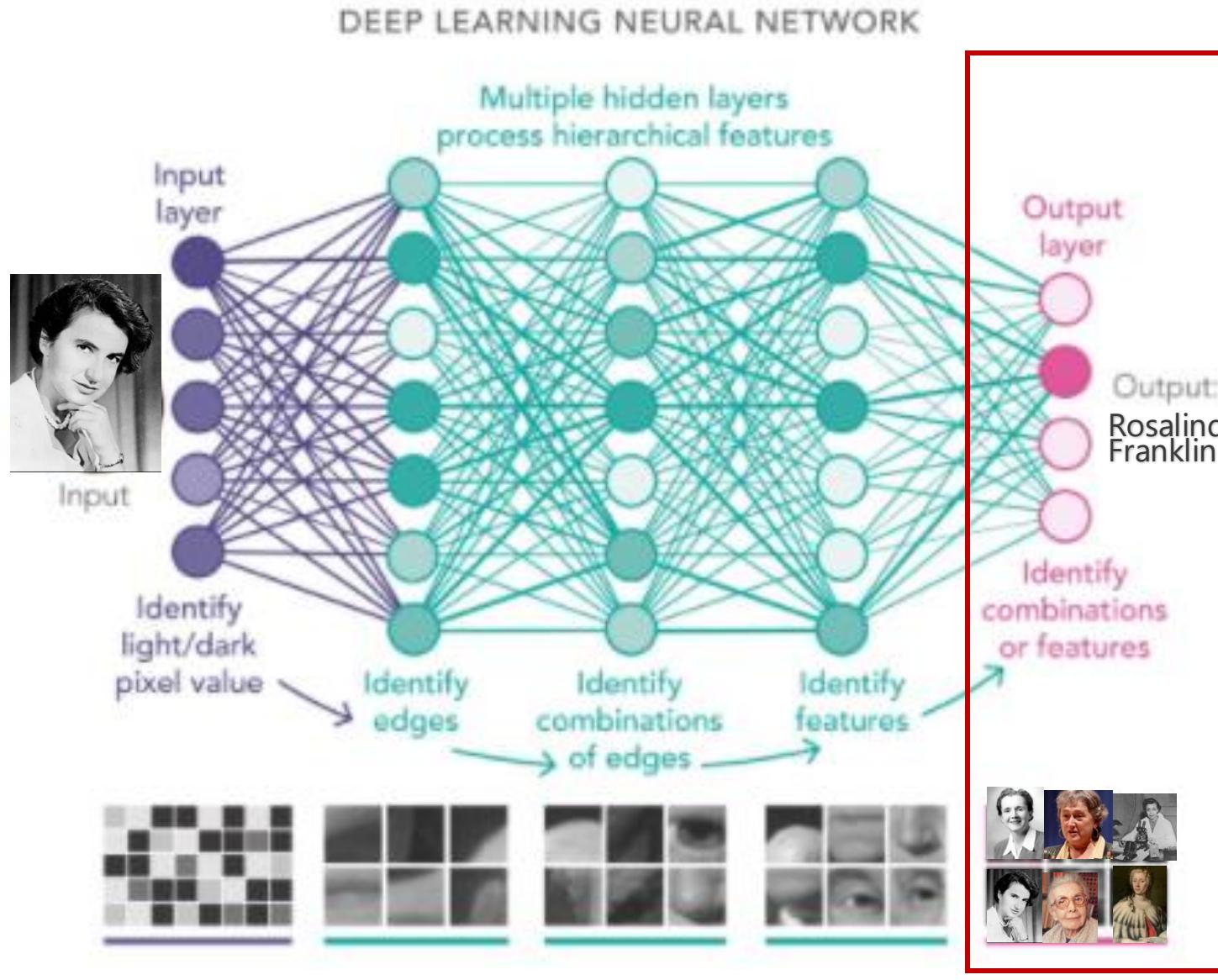
Hidden Layers process hierarchical features



Hidden Layers process hierarchical features



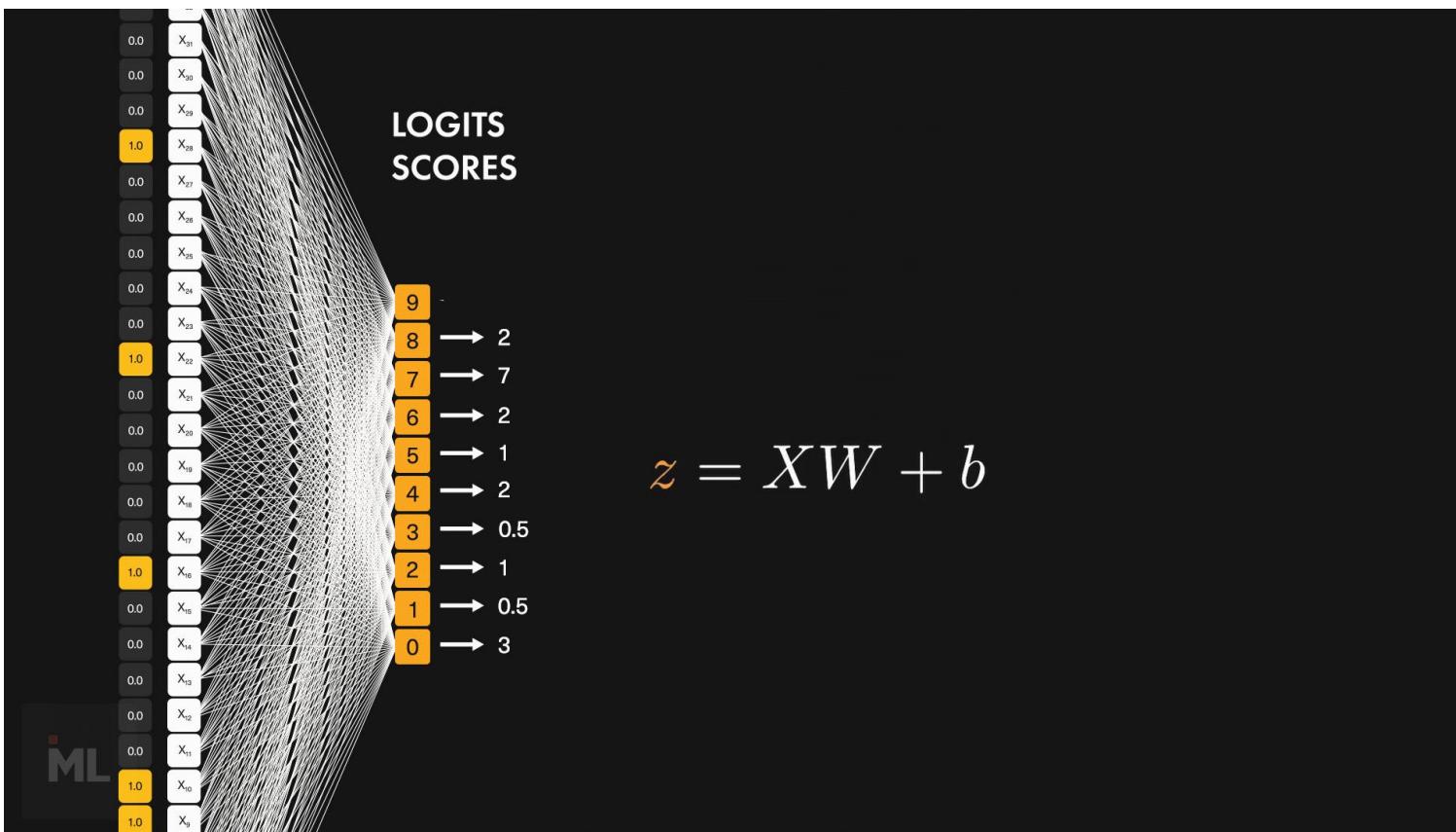
Deep Neural Networks



Output activation function

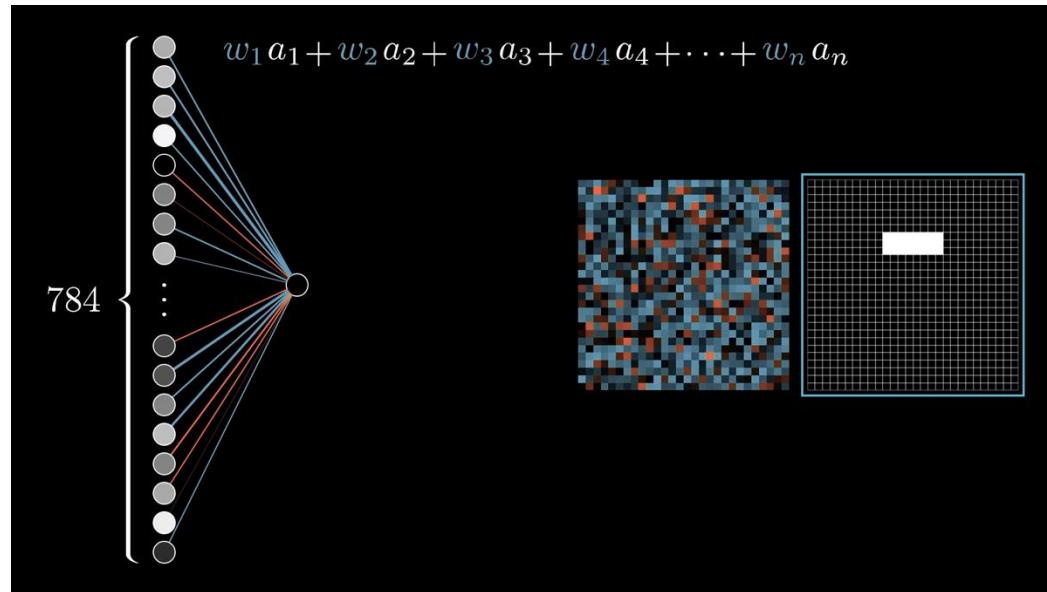
Softmax for classification tasks

- Outputs the “probability” of each class

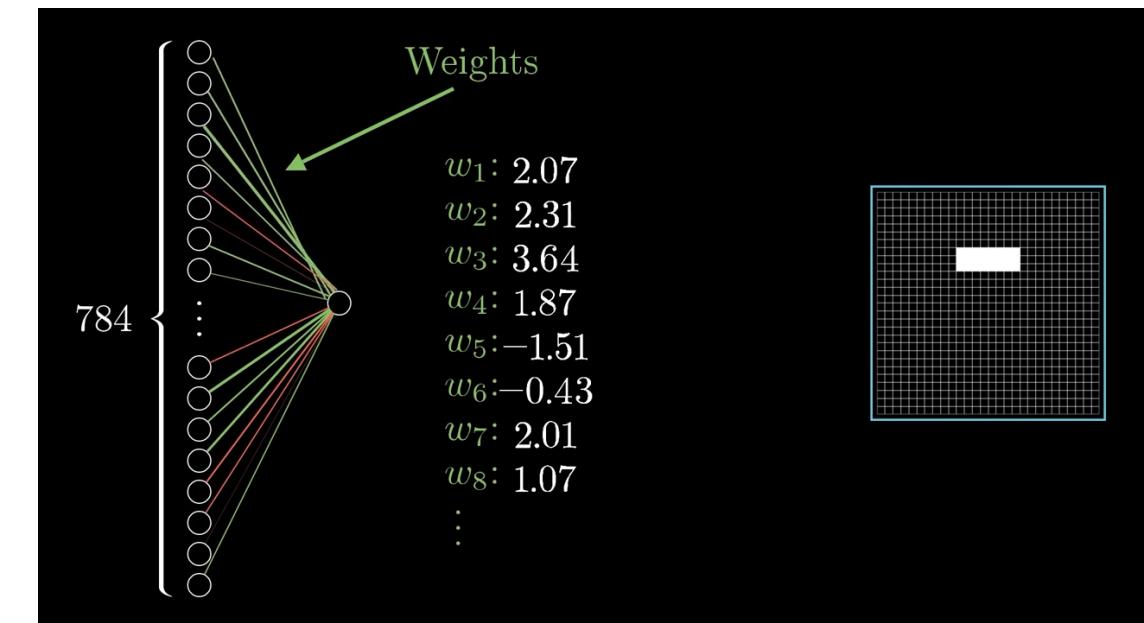


- **Practical Exercise 1:**
- Access <https://www.3blue1brown.com/lessons/neural-networks>
- Draw different number in the first interative exercise and look at how the hidden layers and the output layers get activated.

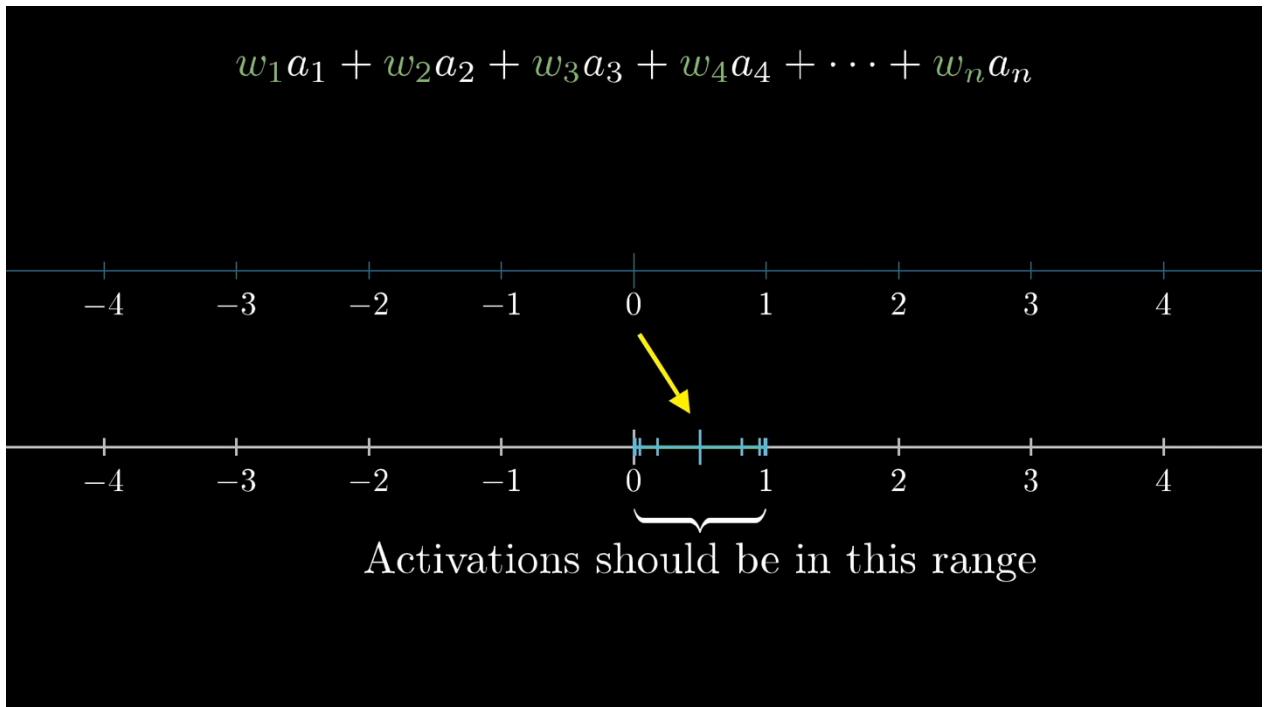
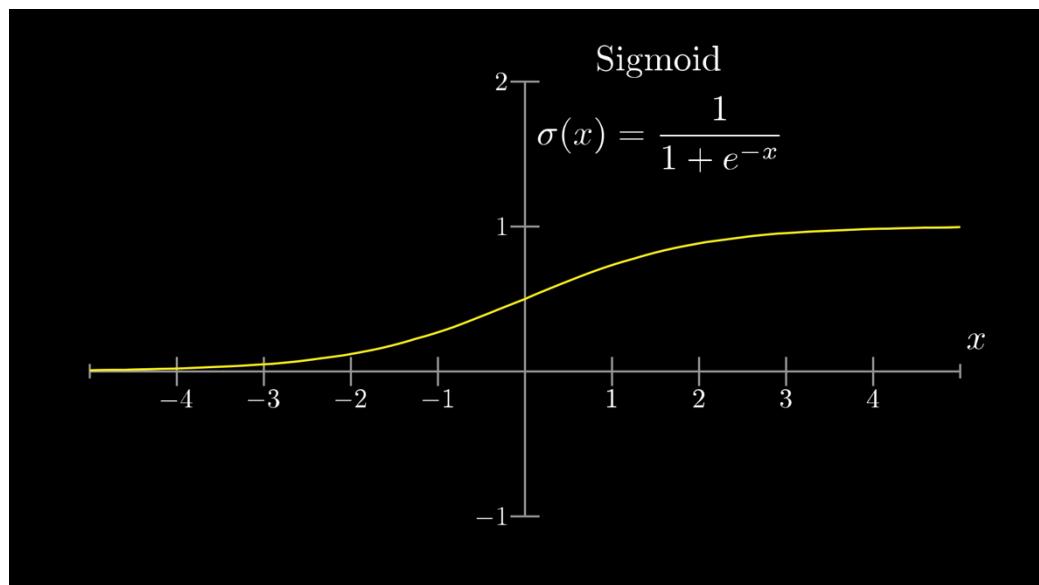
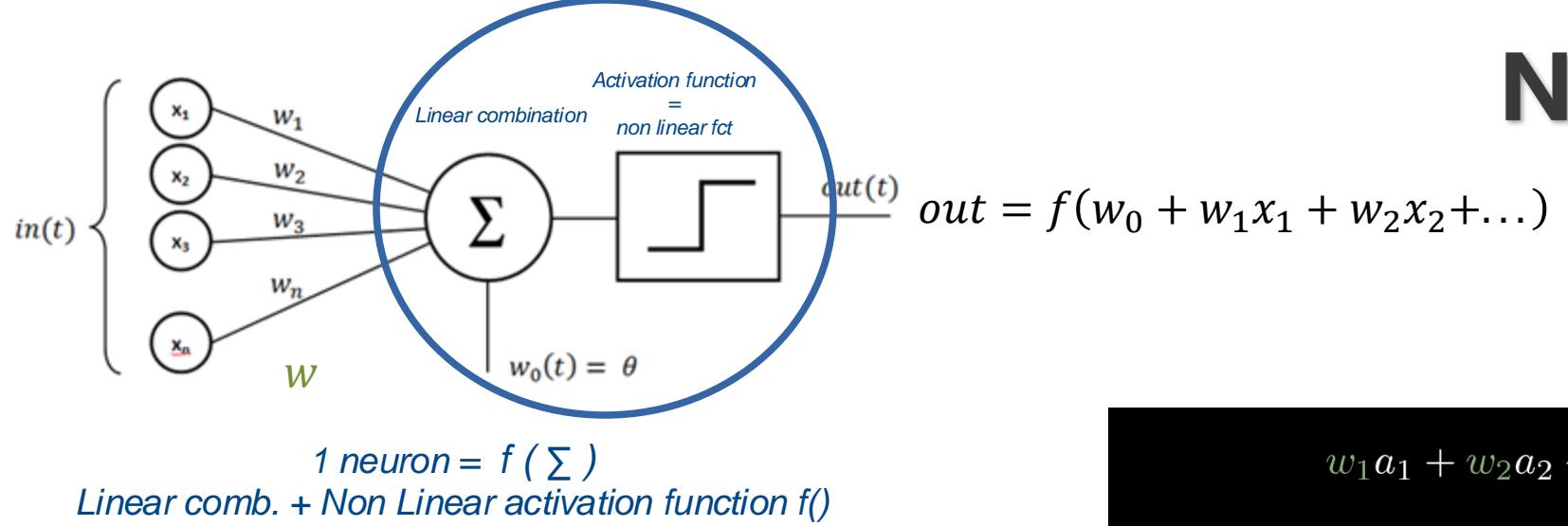
Network Weights



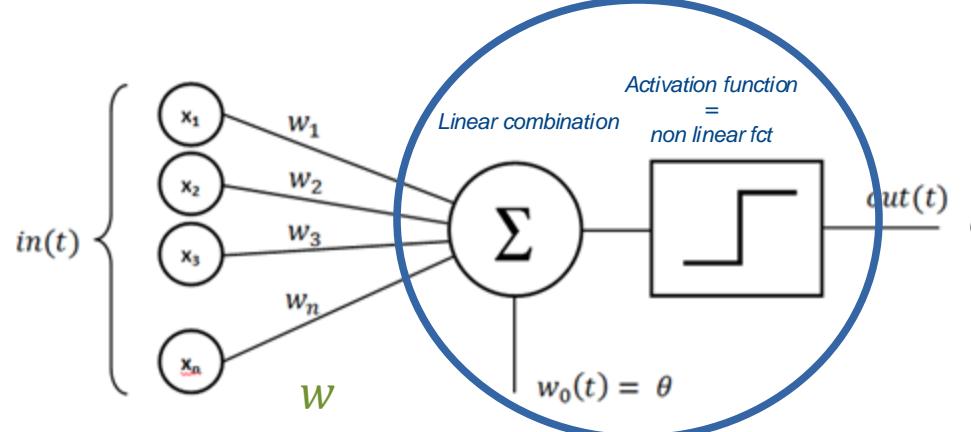
Neuron in the second layer to check whether the image contains this one, specific edge.



Network Weights

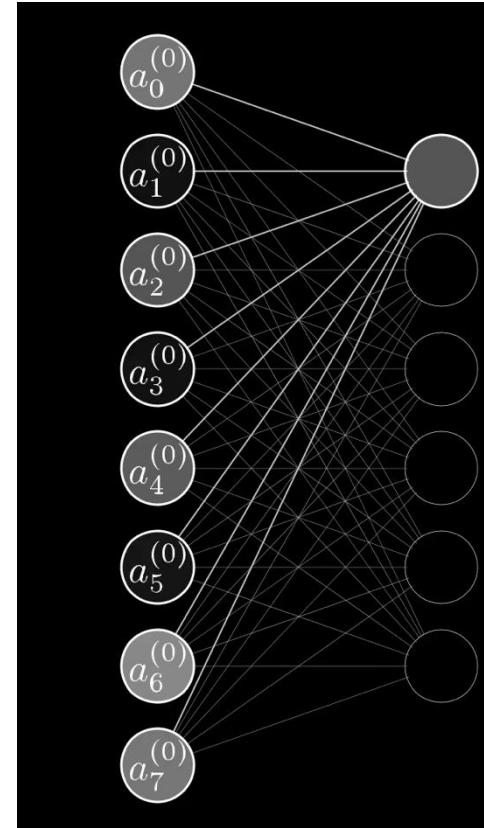


Network Weights



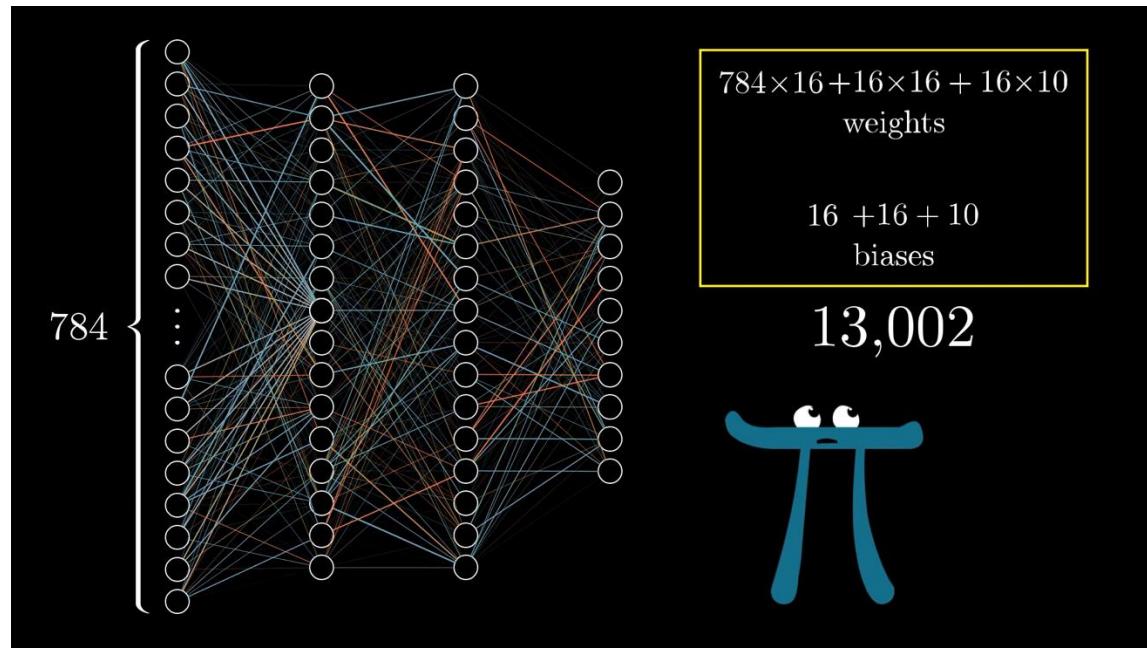
*1 neuron = $f(\sum)$
Linear comb. + Non Linear activation function $f()$*

$$out = f(w_0 + w_1x_1 + w_2x_2 + \dots)$$

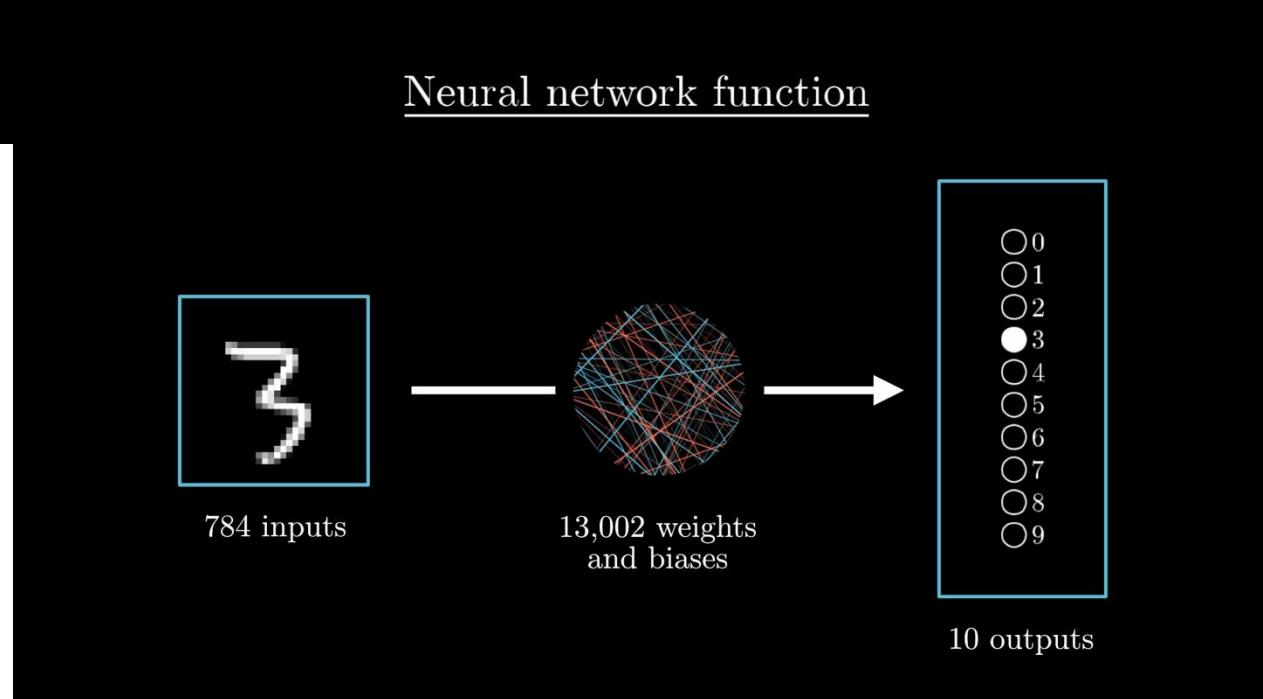


$$\sigma \left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$$

Score function

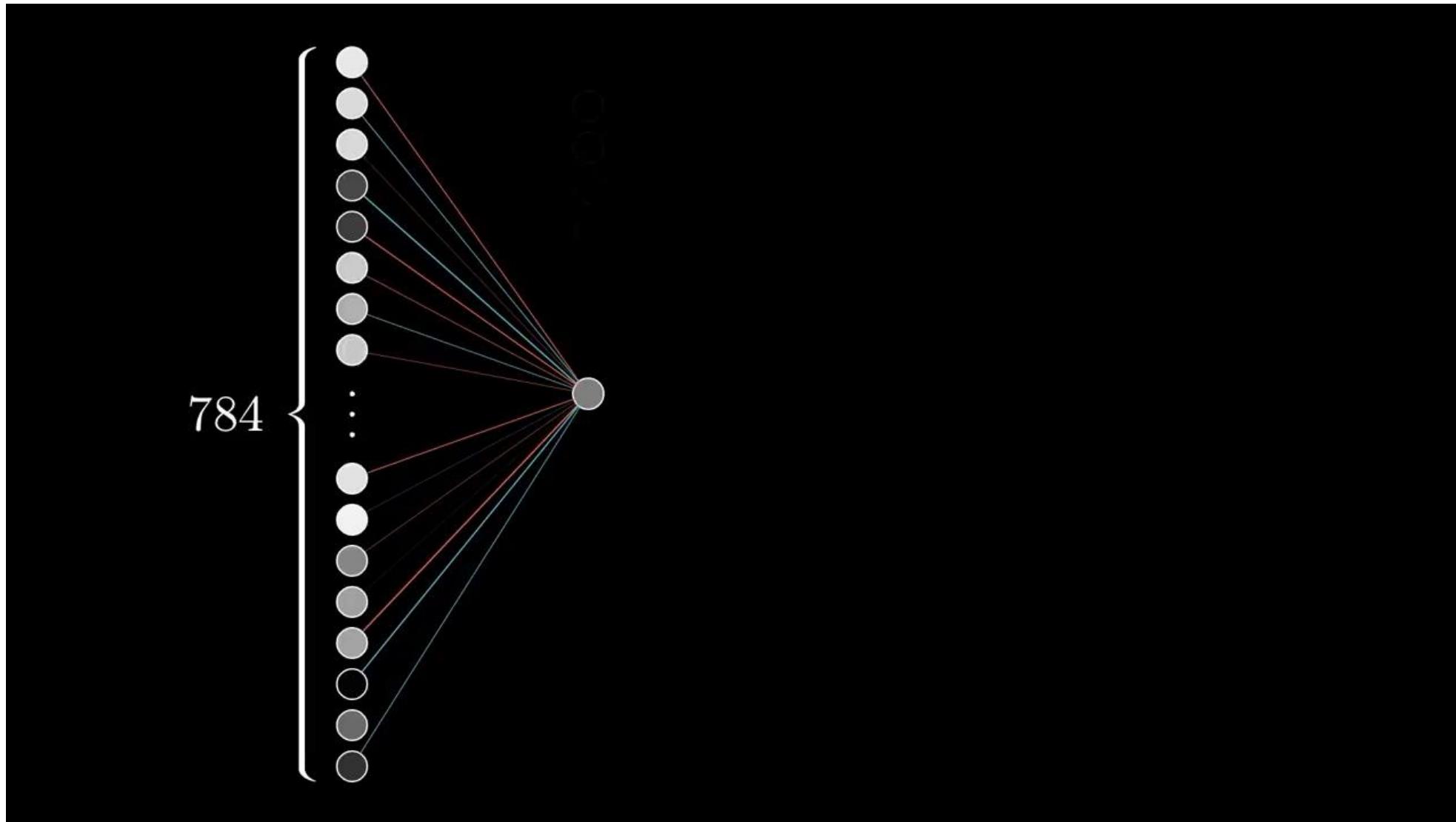


Neural network function



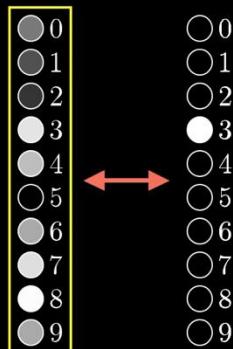
How the network learns?

Cost function



Cost Function

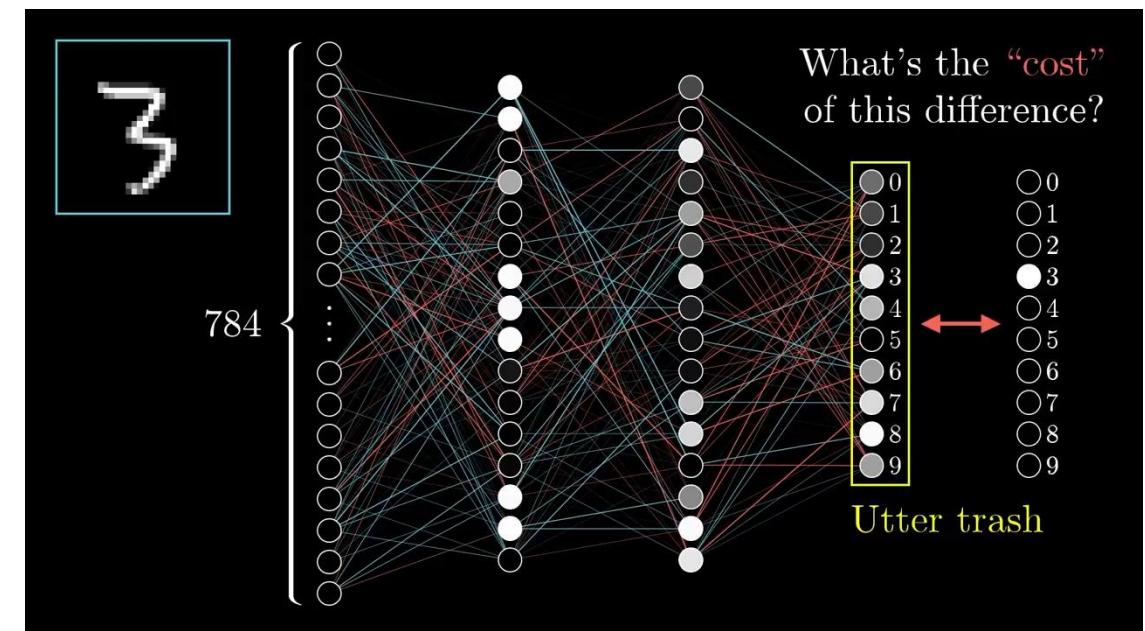
What's the “cost”
of this difference?



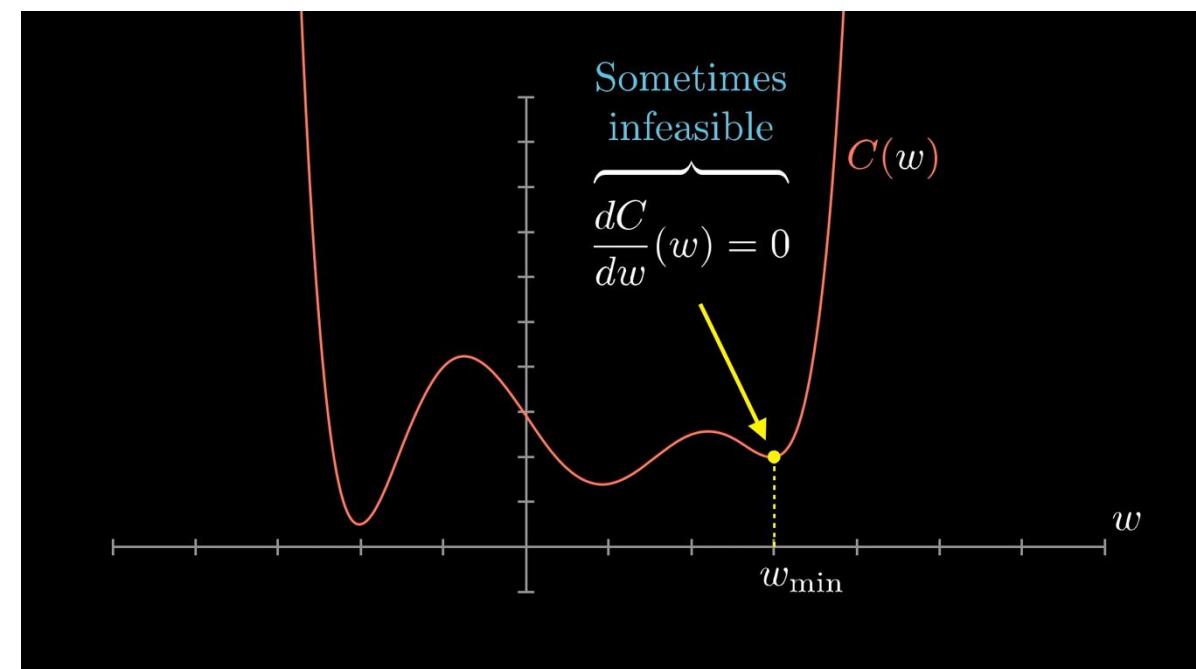
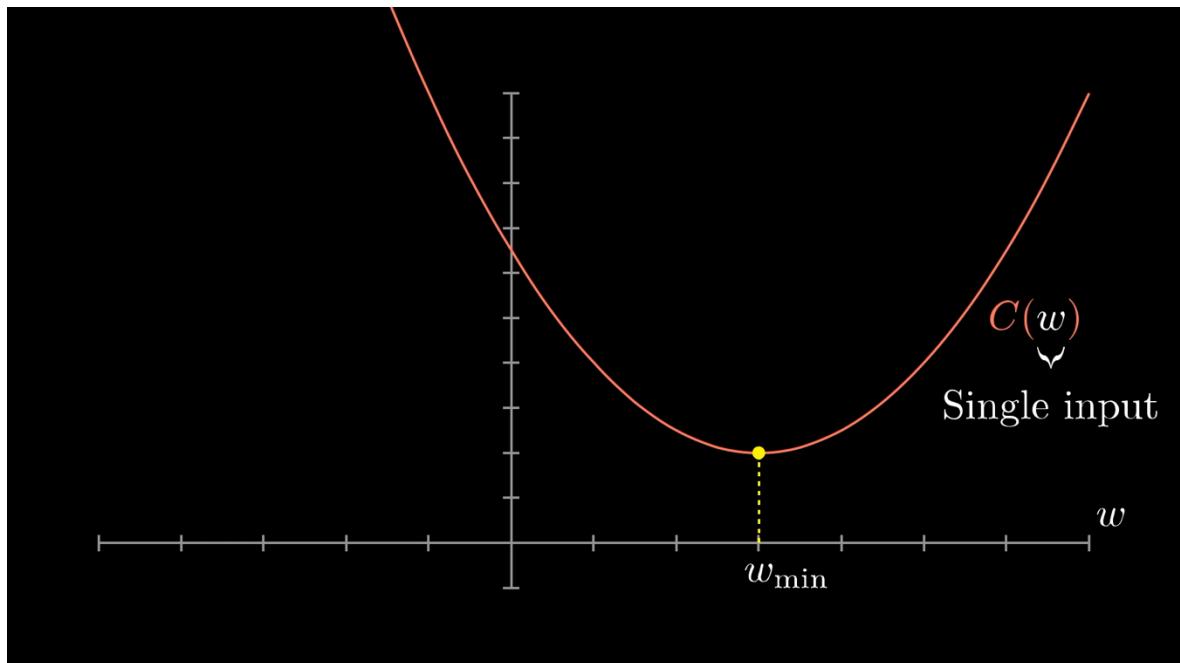
Utter trash

Cost of 

$$\left\{ \begin{array}{l} (0.43 - 0.00)^2 + \\ (0.28 - 0.00)^2 + \\ (0.19 - 0.00)^2 + \\ (0.88 - 1.00)^2 + \\ (0.72 - 0.00)^2 + \\ (0.01 - 0.00)^2 + \\ (0.64 - 0.00)^2 + \\ (0.86 - 0.00)^2 + \\ (0.99 - 0.00)^2 + \\ (0.63 - 0.00)^2 \end{array} \right.$$

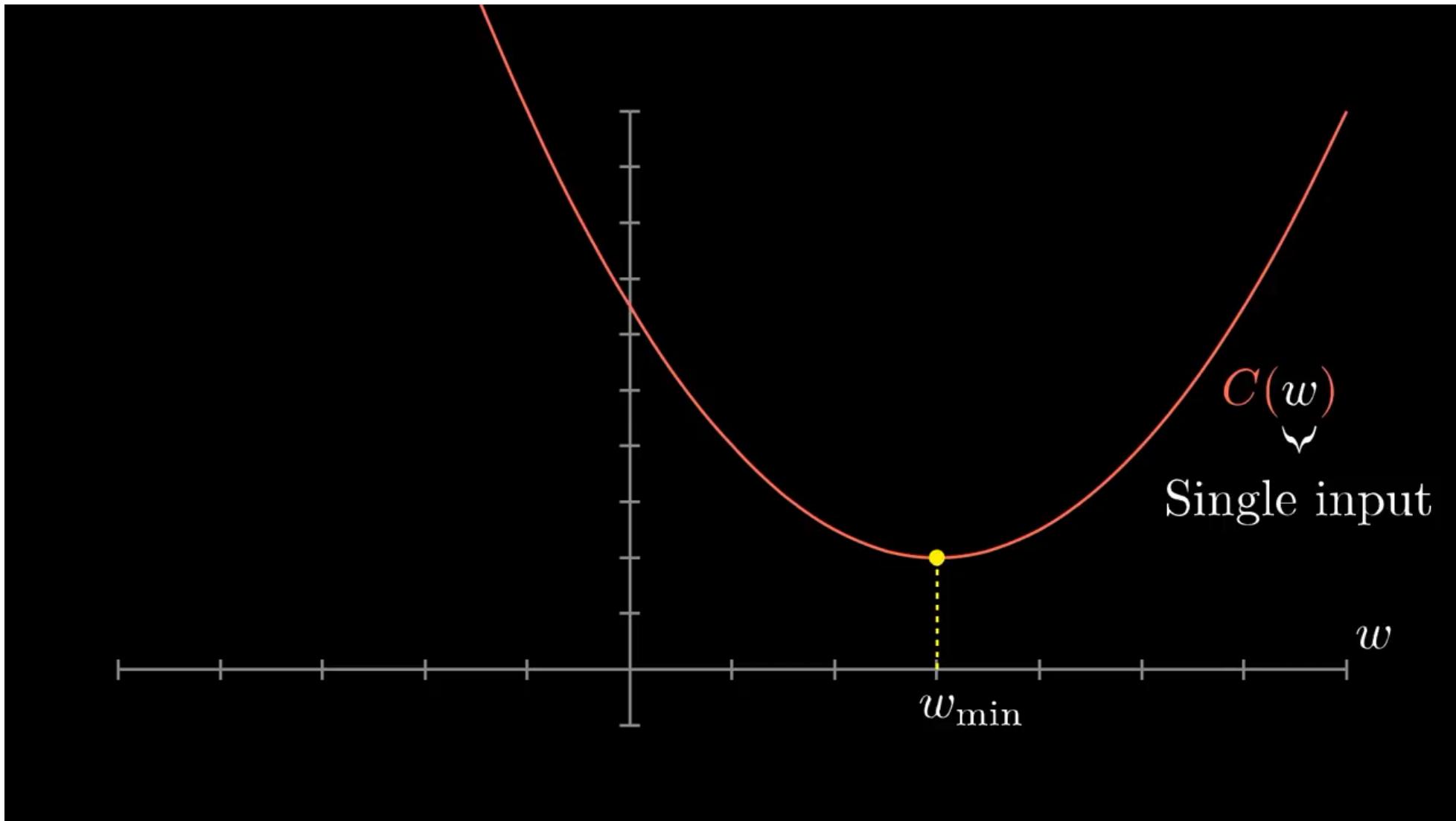


Cost Function

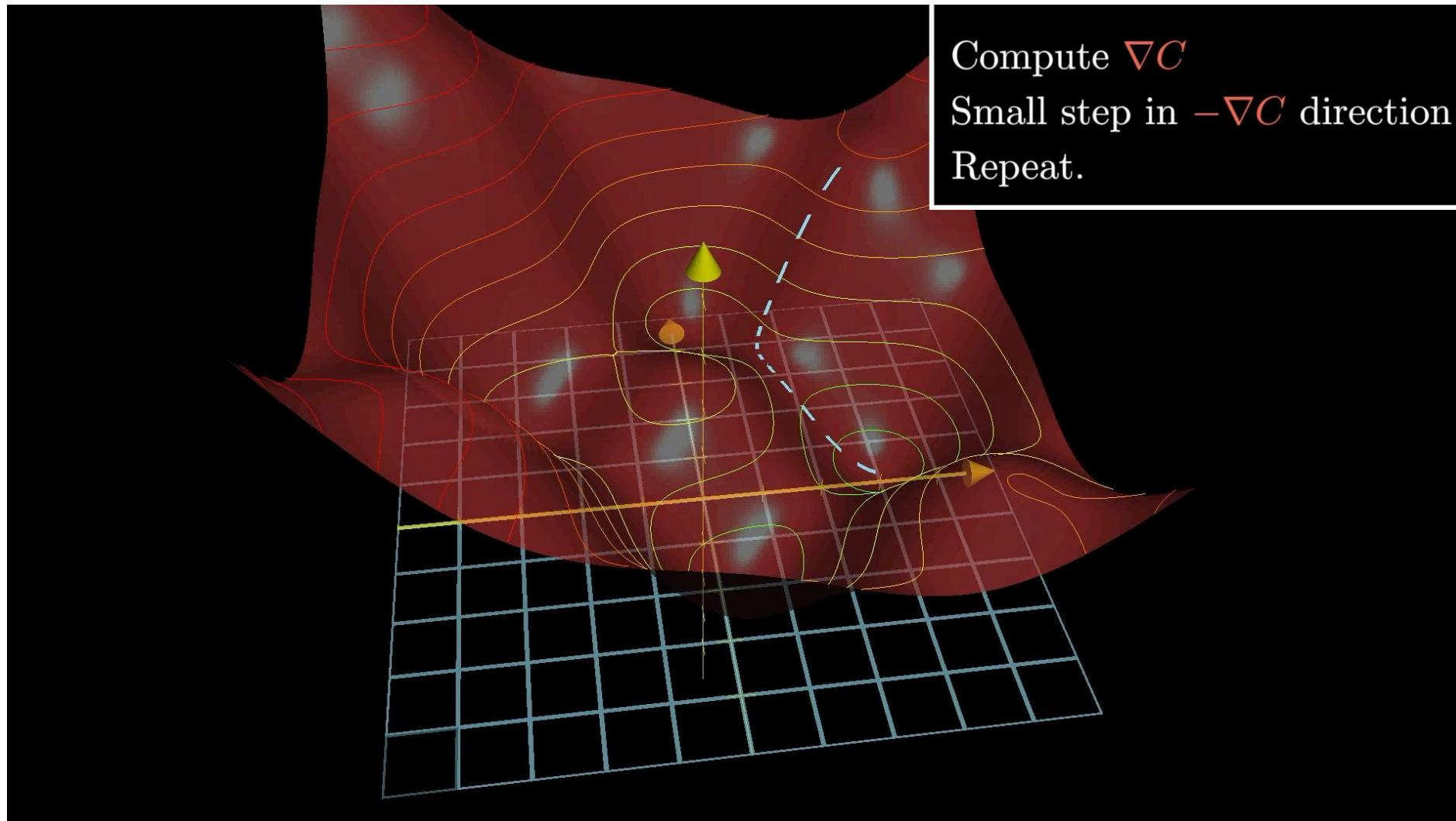


Optimization

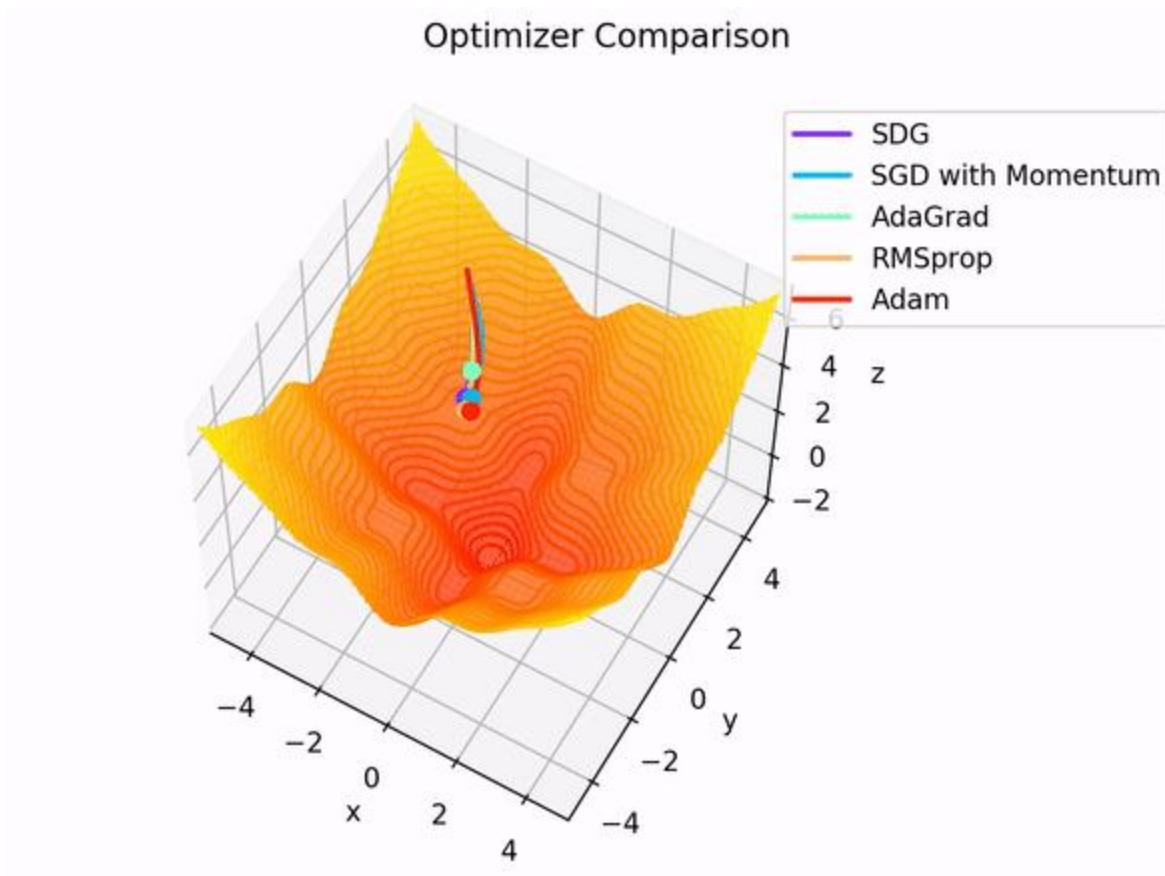
Gradient Descent



Gradient Descent



Gradient Descent



<https://towardsdatascience.com/complete-guide-to-adam-optimization-1e5f29532c3d>

Gradient Descent

13,002 weights and biases

$$\vec{\mathbf{W}} = \begin{bmatrix} 2.25 \\ -1.57 \\ 1.98 \\ \vdots \\ -1.16 \\ 3.82 \\ 1.21 \end{bmatrix}$$

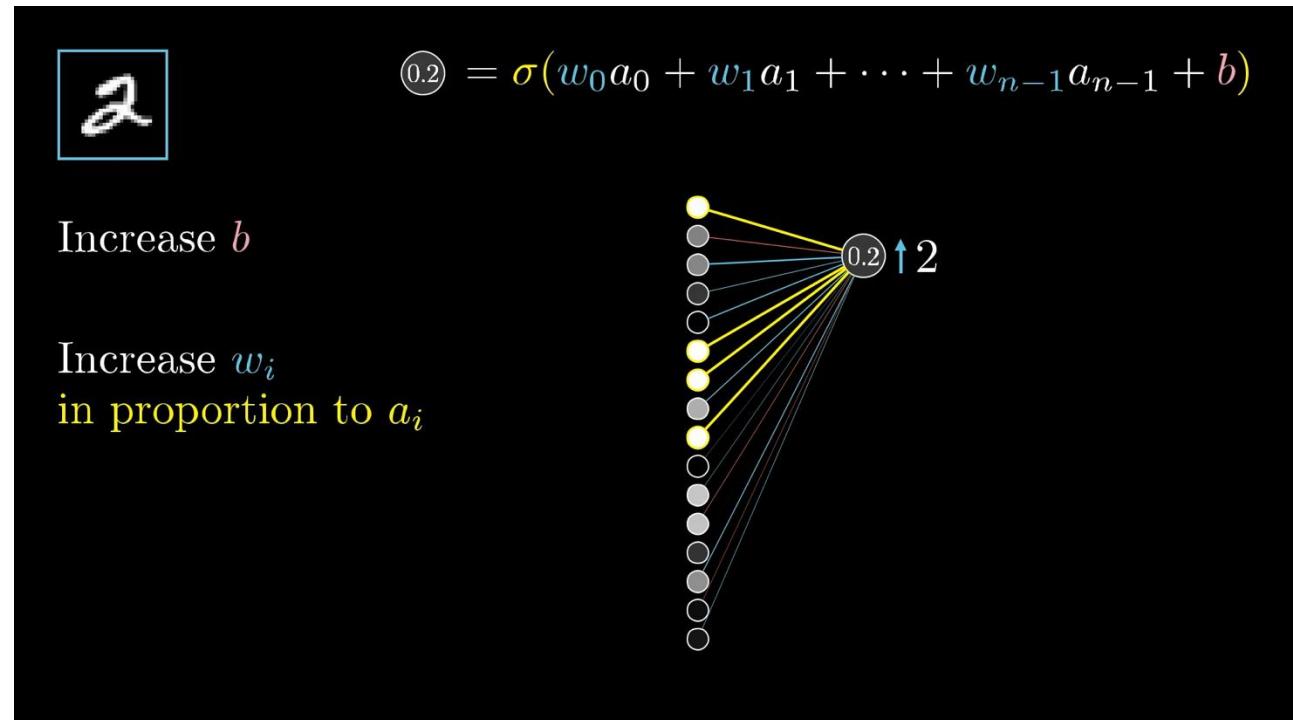
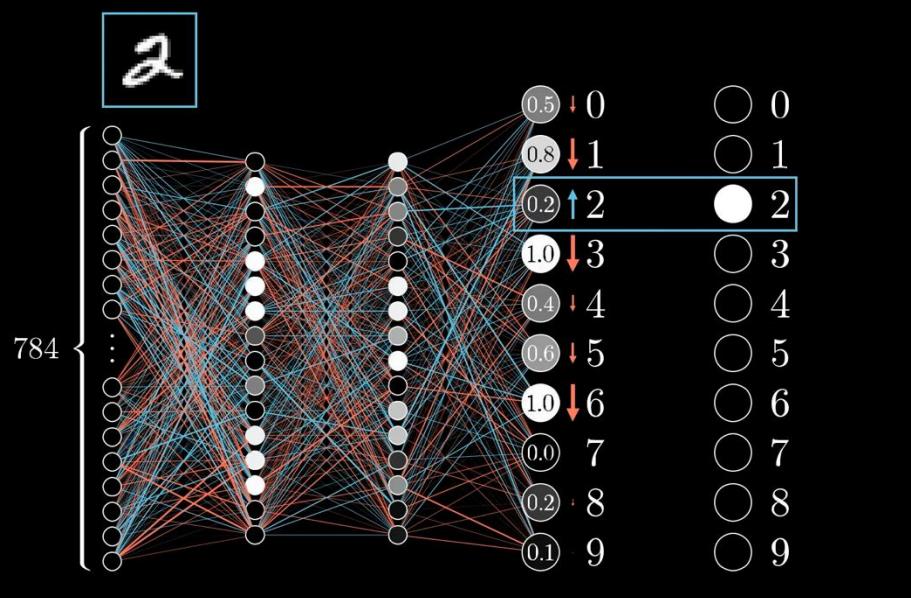
$$-\nabla C(\vec{\mathbf{W}}) = \begin{bmatrix} 0.31 \\ 0.03 \\ -1.25 \\ \vdots \\ 0.78 \\ -0.37 \\ 0.16 \end{bmatrix}$$

w_0 should increase somewhat
 w_1 should increase a little
 w_2 should decrease a lot
 $w_{13,000}$ should increase a lot
 $w_{13,001}$ should decrease somewhat
 $w_{13,002}$ should increase a little

Gradient Descent

$$\vec{\mathbf{W}} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{13,000} \\ w_{13,001} \\ w_{13,002} \end{bmatrix}$$
$$-\nabla C(\vec{\mathbf{W}}) = \left. \begin{bmatrix} 0.31 \\ 0.03 \\ -1.25 \\ \vdots \\ 0.78 \\ -0.37 \\ 0.16 \end{bmatrix} \right\} \text{Example gradient}$$

Backpropagation



Backpropagation

Training in progress. . .



Training NNs

- Training = estimating all the weights $w_{ij}^{(l)}$ for all layers (l) with the aim of minimizing the loss or cost function
- Loss/cost function: how well your classifier/regressor do ?
→ define a function quantifying how far you are from the truth (e.g., Mean Squared Error)

$$loss = \frac{1}{n} \sum_{examplei=1}^n (y_i^{predicted} - y_i^{truth})^2$$

- Training algorithm :

- Step 1 Initialize randomly the weights

For each epoch :

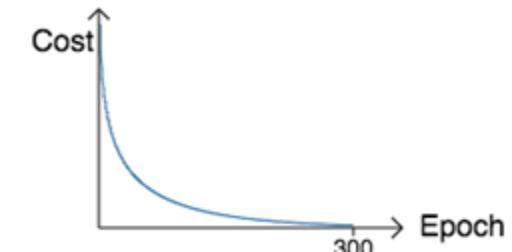
Step 2 Forward pass your examples (eg labeled images of cats and birds) through nnet to compute predicted values (as previously explained)

Step 3 Compute loss

Step 4 Backward propagation

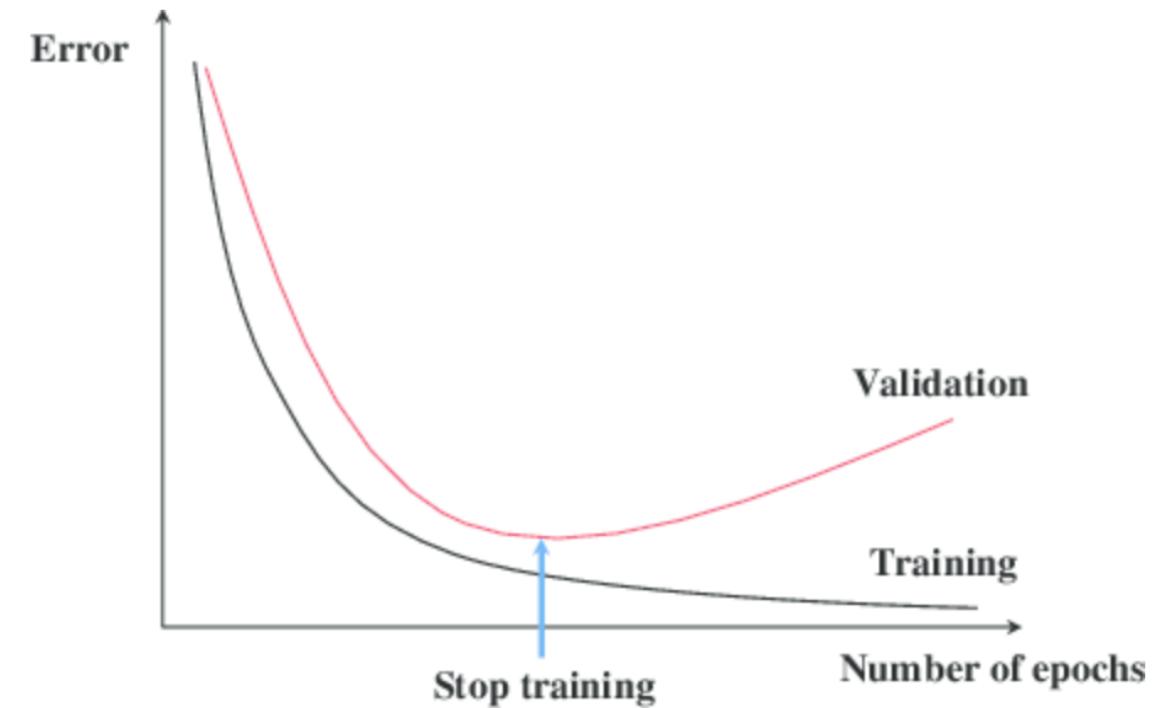
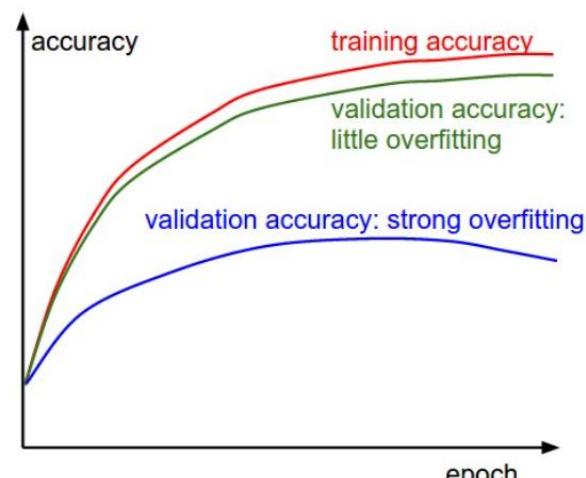
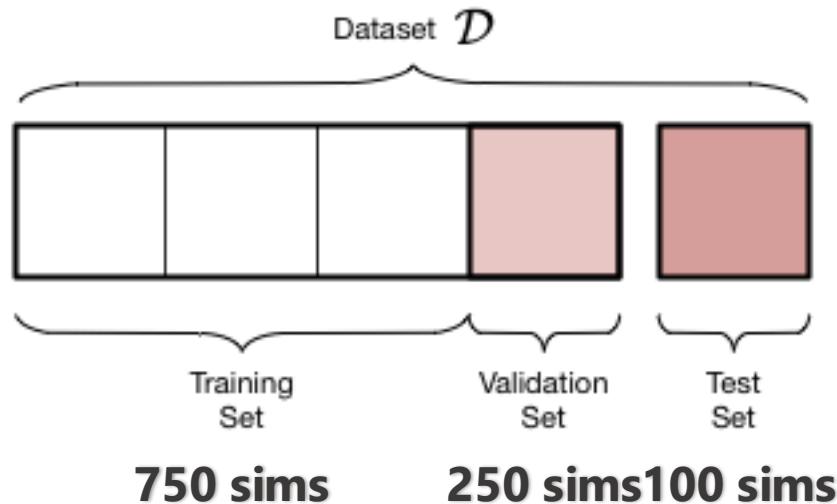
Step 5 Update weights (gradient descent algorithm)

Repeat from step 2 until a plateau is reached



Early Stop to avoid overfitting

<https://se-education.org/learningresources/contents/ai/ml.html>

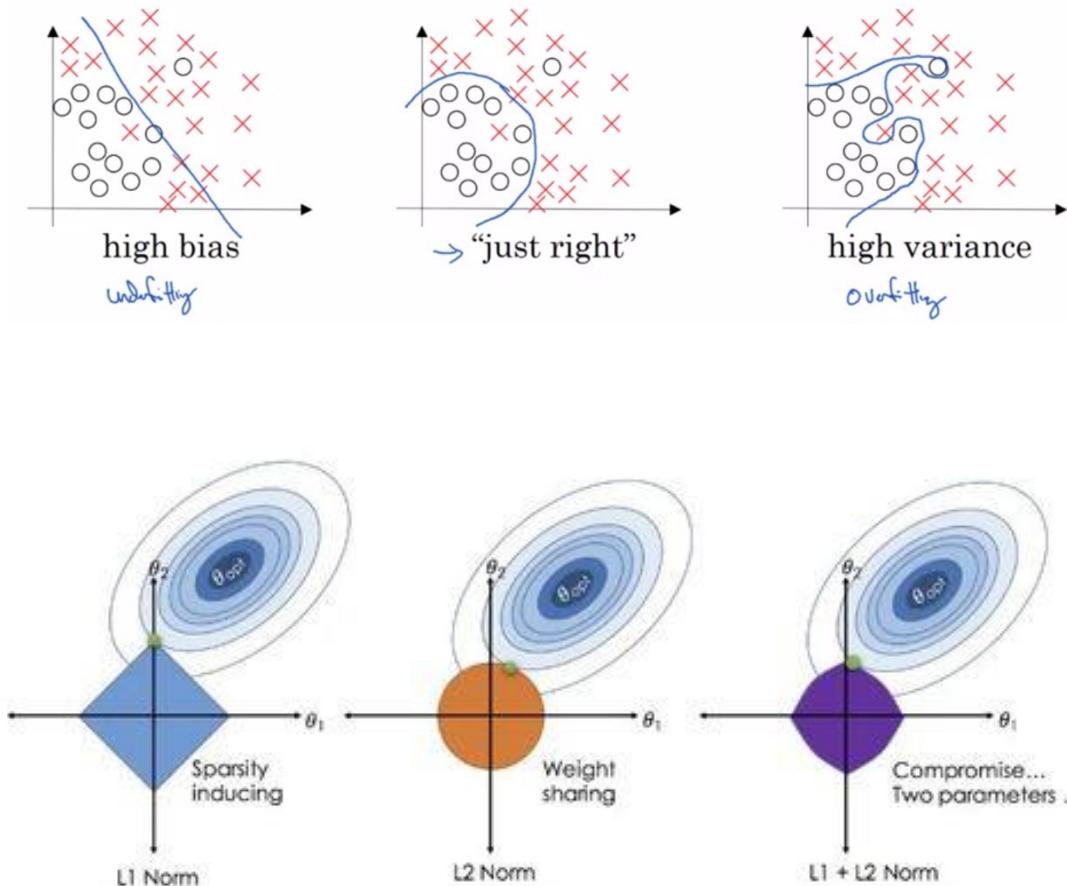


<https://towardsdatascience.com/a-practical-introduction-to-early-stopping-in-machine-learning-550ac88bc8fd>

- **Practical Exercise 2:**
- Access <https://playground.tensorflow.org>.
- Use the graphical interface to see how the neural network results changes when you modify the number of neurons, the input features, and the number of hidden layers.
- Compare the learning capacity when using a sigmoid and a ReLU activation. We will talk about these functions later.

Regularization to avoid overfitting

Bias and Variance

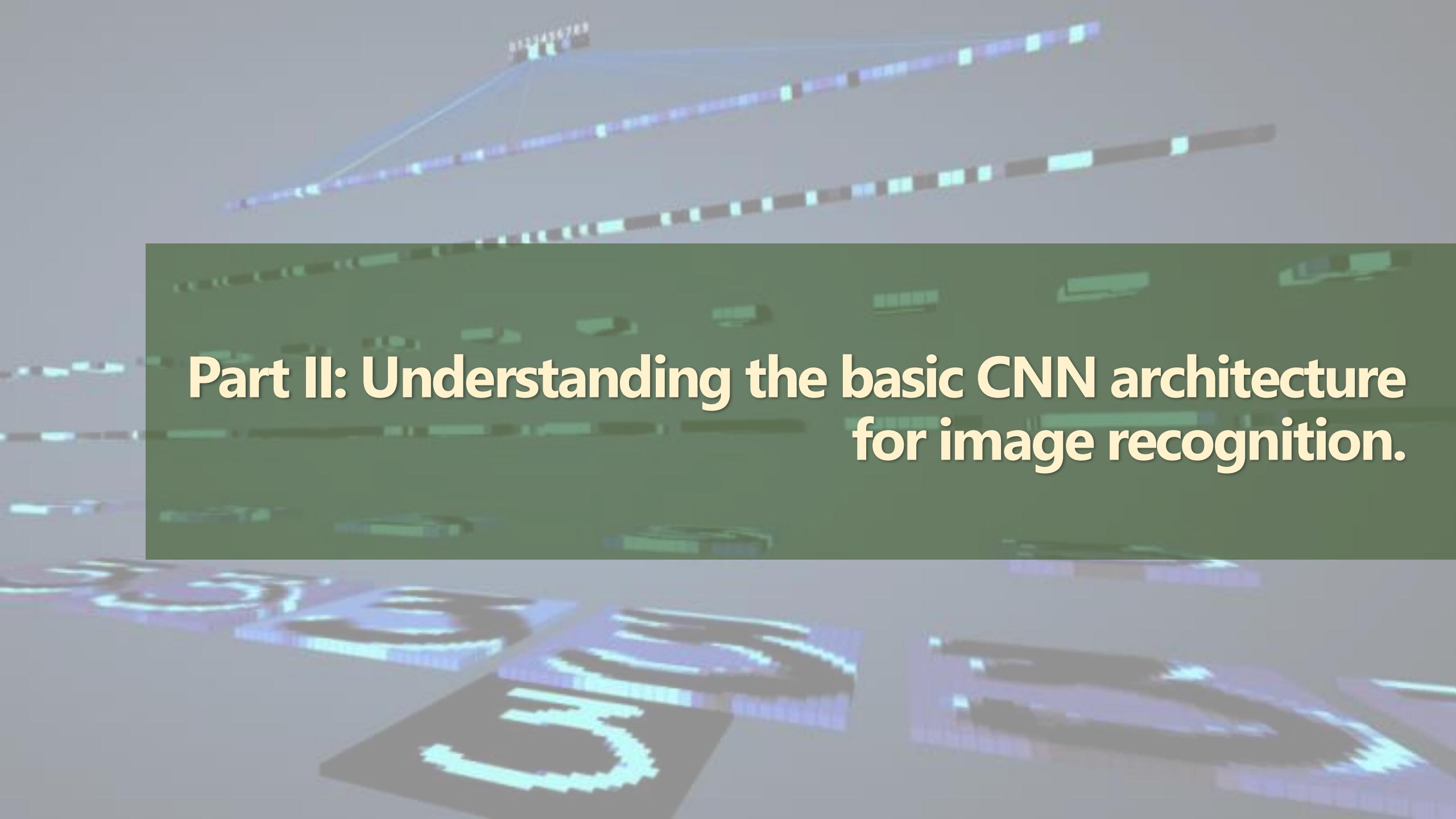


Regularization Type	Advantages	Disadvantages
L1	<ul style="list-style-type: none">1. Performs feature selection by shrinking less important feature weights to zero.2. Can be used for high-dimensional datasets with many irrelevant features.	<ul style="list-style-type: none">1. Not effective for datasets with many important features.2. The solution may not be unique, which can lead to instability in the model.
L2	<ul style="list-style-type: none">1. Provides a smooth solution and improves the generalization performance of the model.2. Can handle datasets with many important features.	<ul style="list-style-type: none">1. May not perform well for datasets with many irrelevant features.2. Does not perform feature selection, and all features are included in the model with non-zero weights.

- **Practical Exercise 3:**
- Access the notebook at
<https://drive.google.com/file/d/1JjHU4MMbJc5NMLr7IJykW8sNqmABr1M5/view?usp=sharing>. It is also on the Github page.

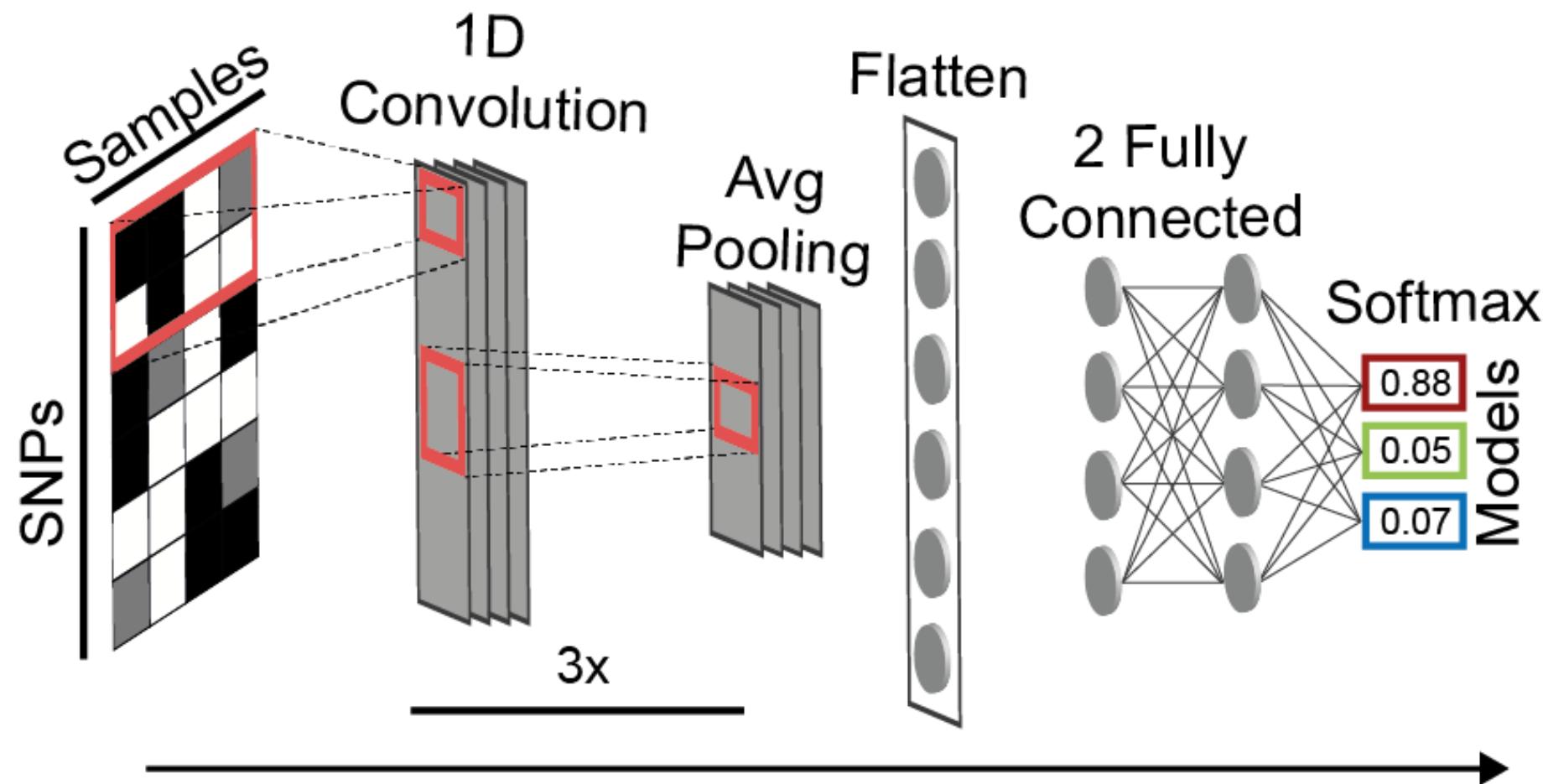
Goals

- Conceive and simulate genetic data under competing demographic scenarios 
- Understand deep learning background and how a CNN works 
- How to use deep learning to compare demographic scenarios

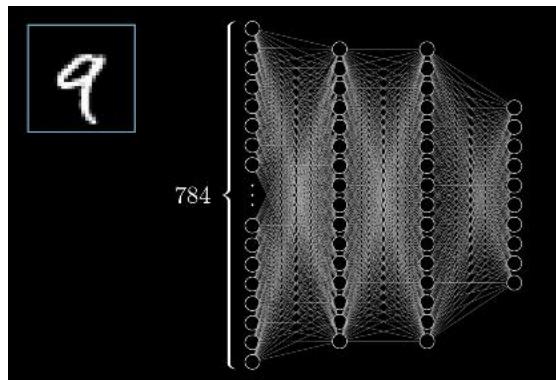


**Part II: Understanding the basic CNN architecture
for image recognition.**

CNN architecture

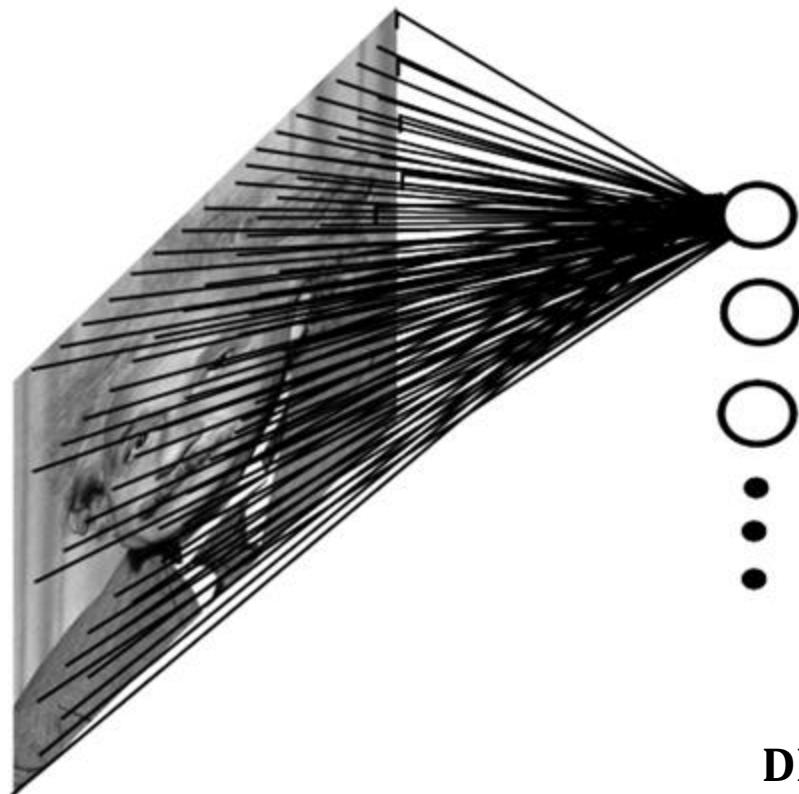


Convolutional Neural Networks (CNNs)

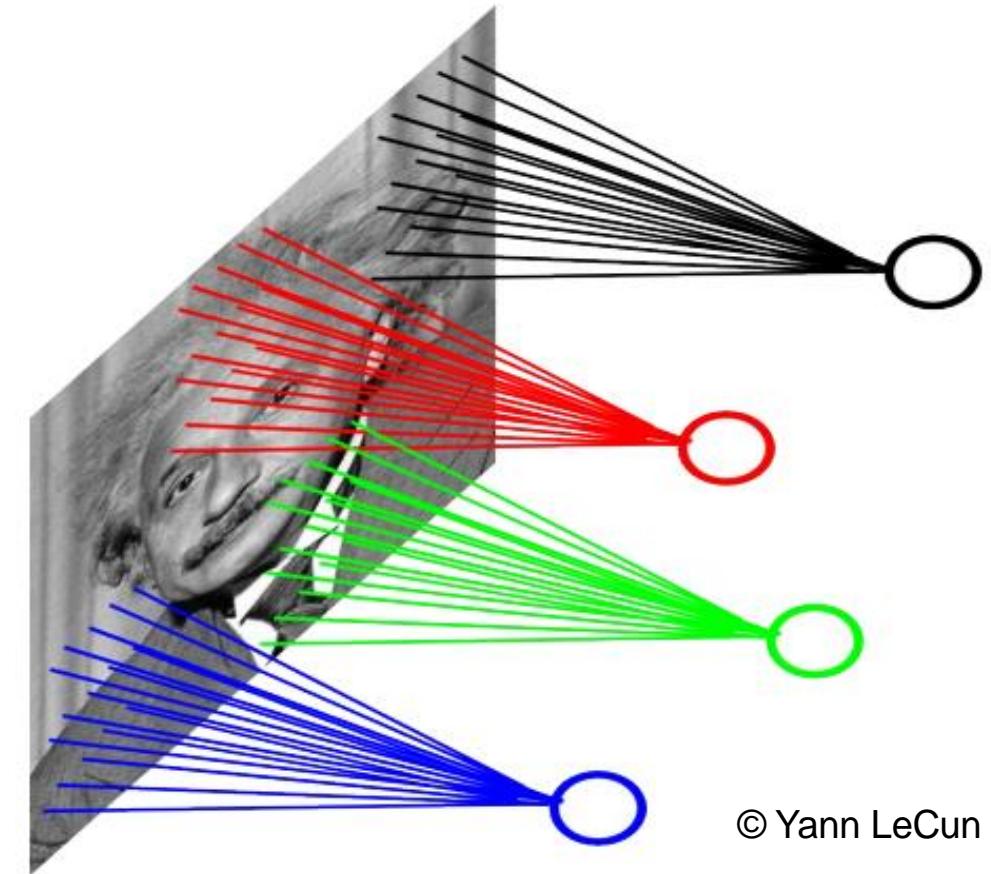


MLP
Fully connected

Convnet
locally connected
→ smaller number of parameters to learn

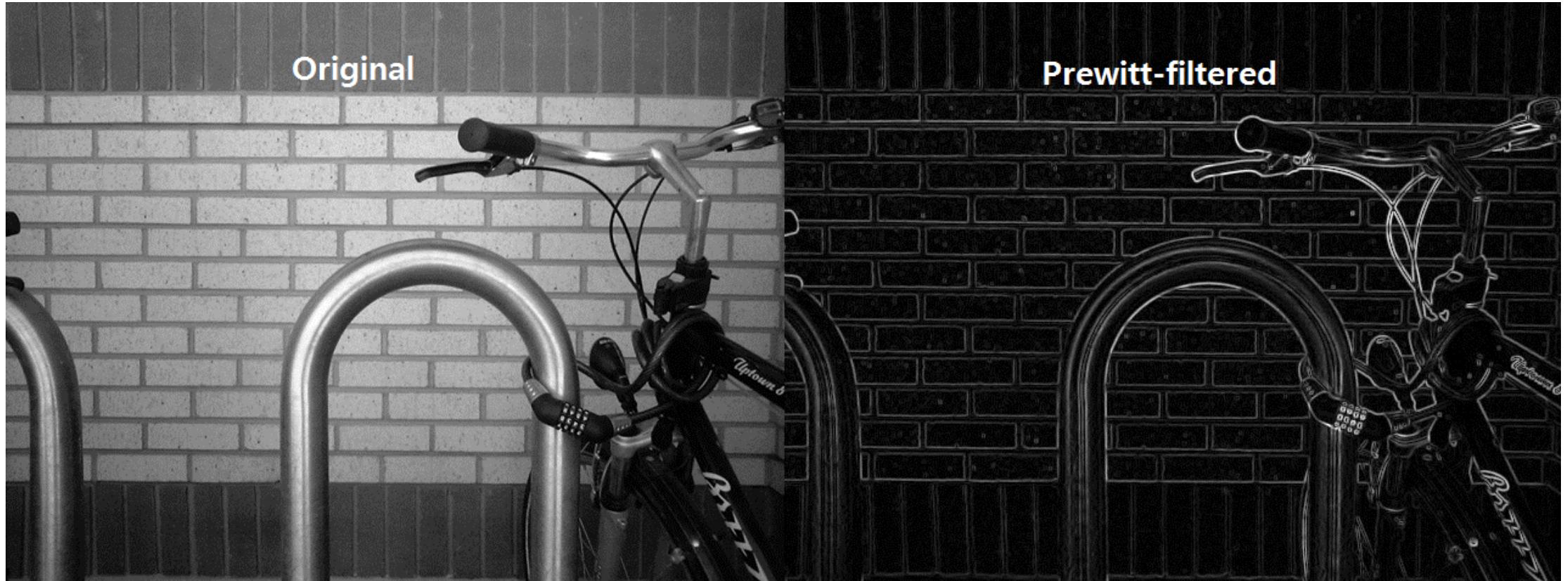


DL witchcraft

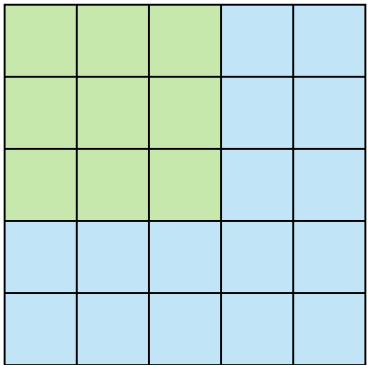


© Yann LeCun

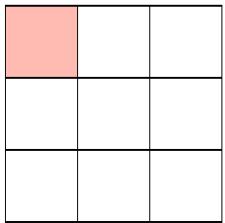
Convolutional Layers



<https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529>

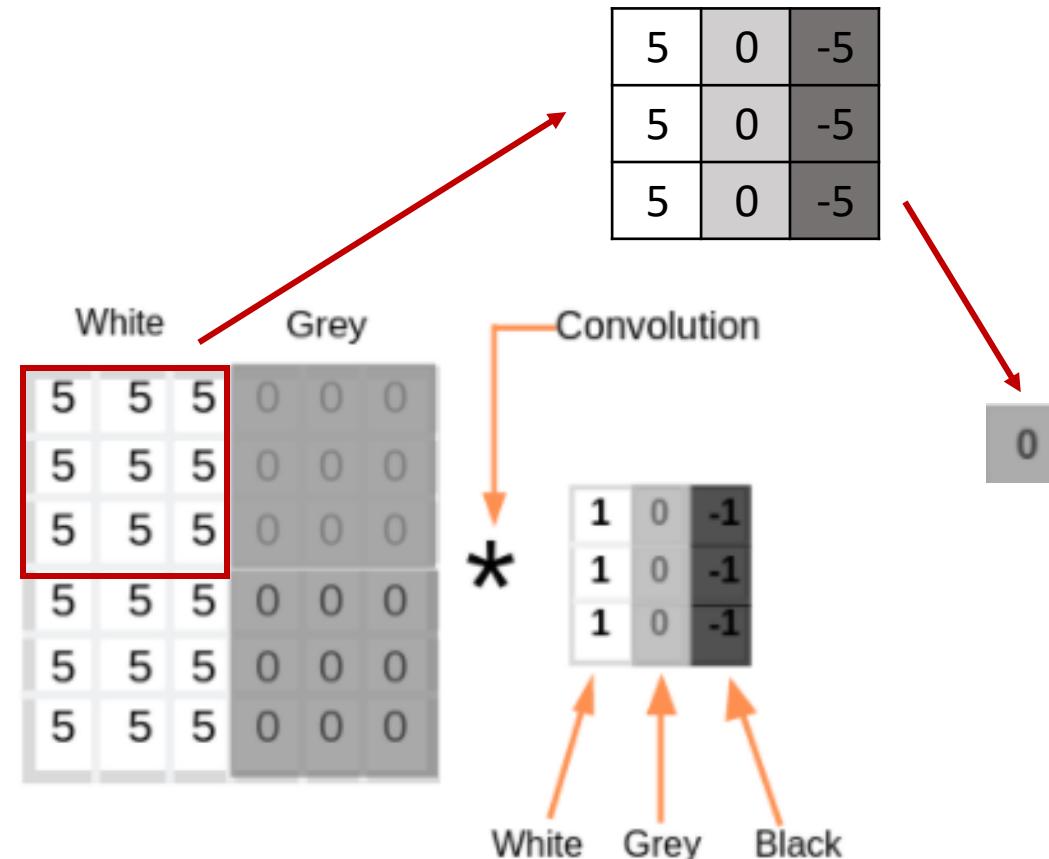
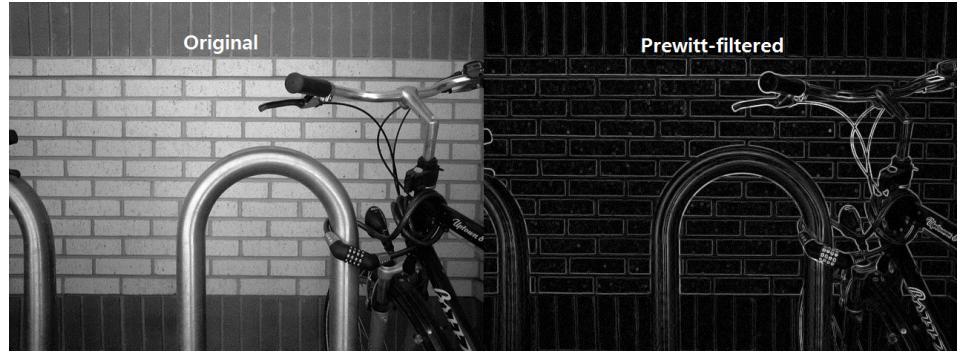


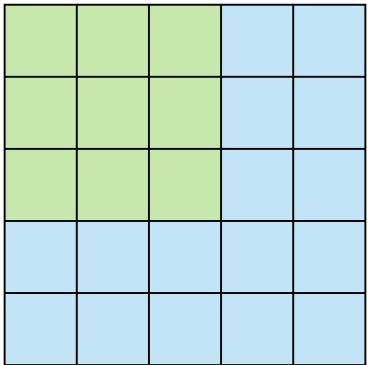
Stride 1



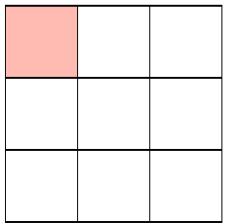
Feature Map

Convolutional Layers

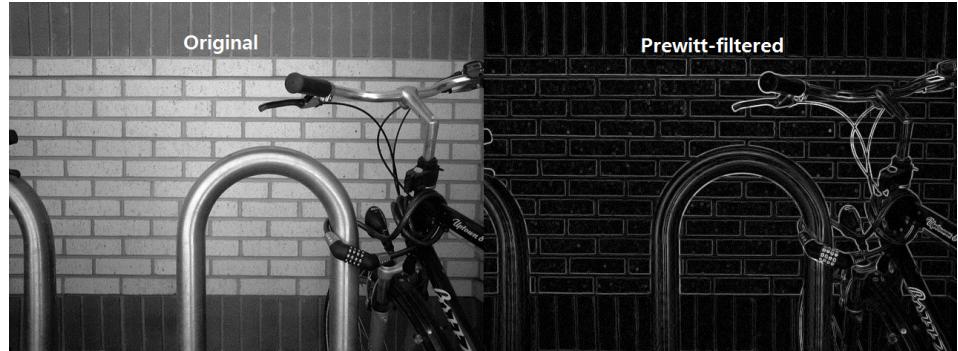




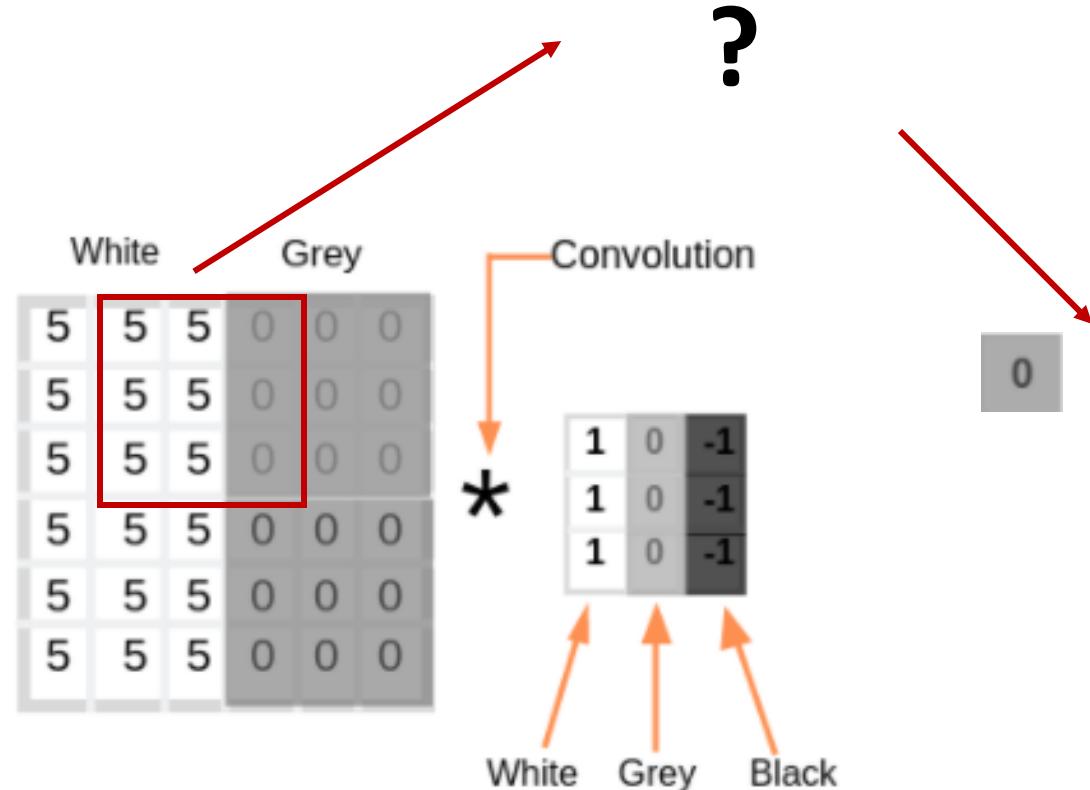
Stride 1

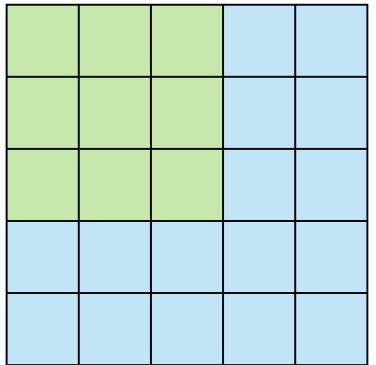


Feature Map

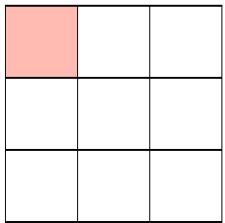


Convolutional Layers





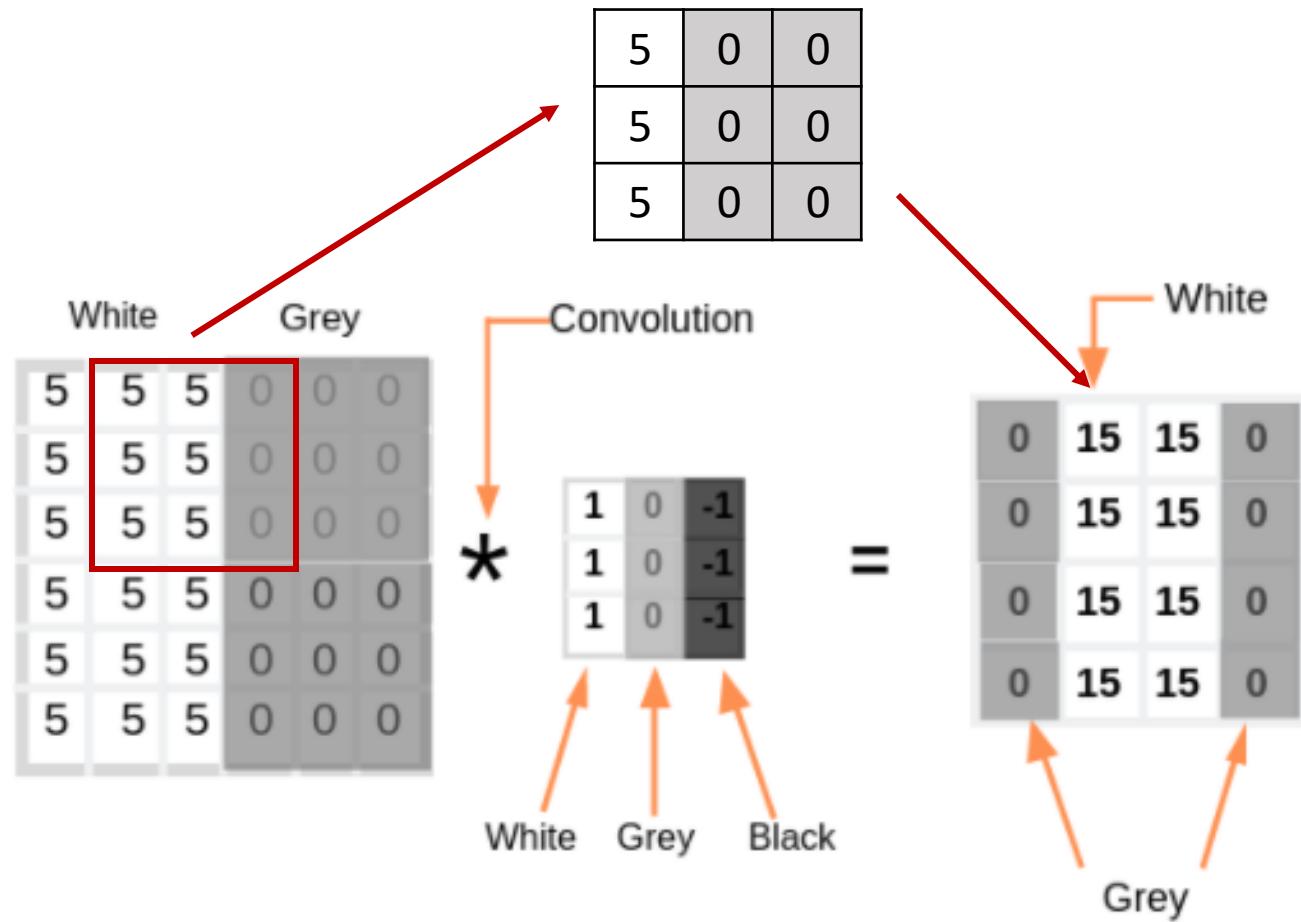
Stride 1



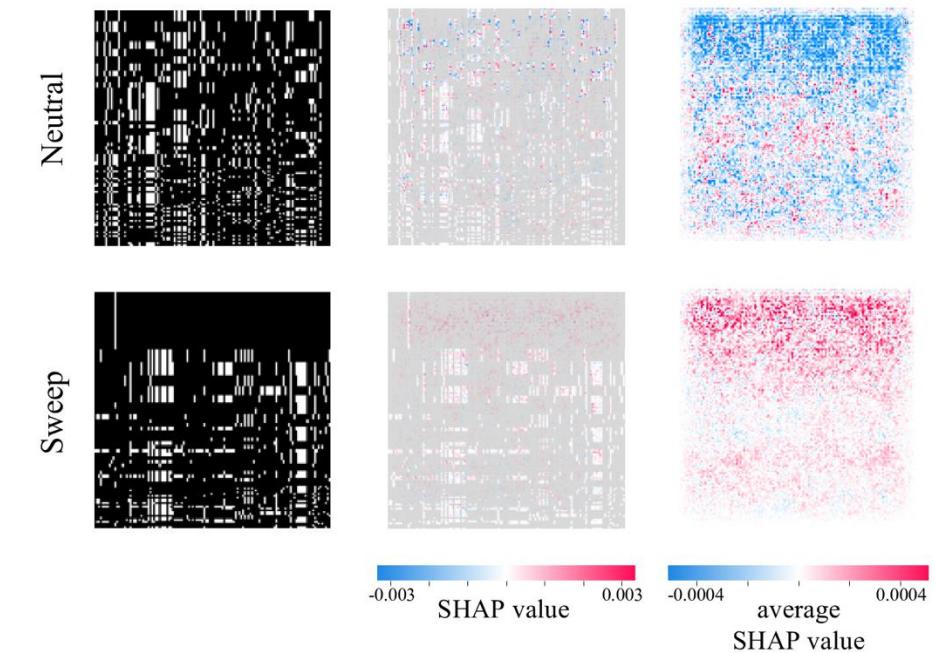
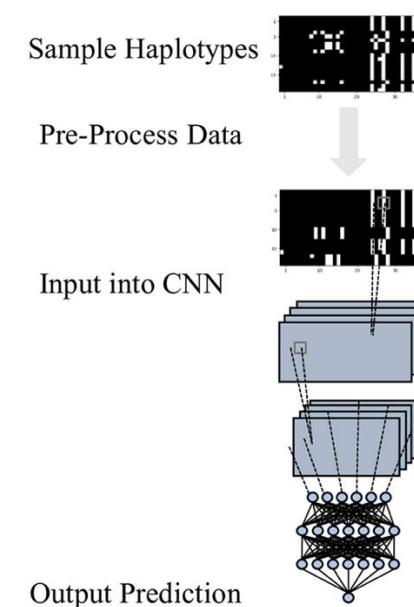
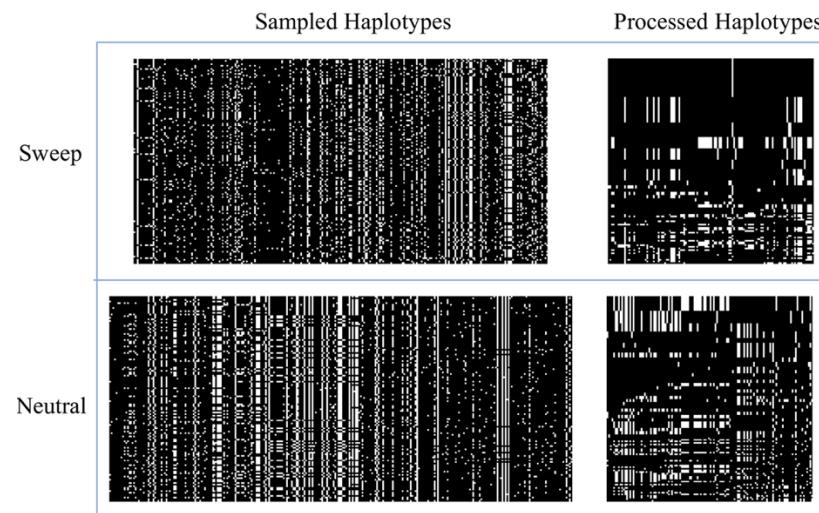
Feature Map



Convolutional Layers



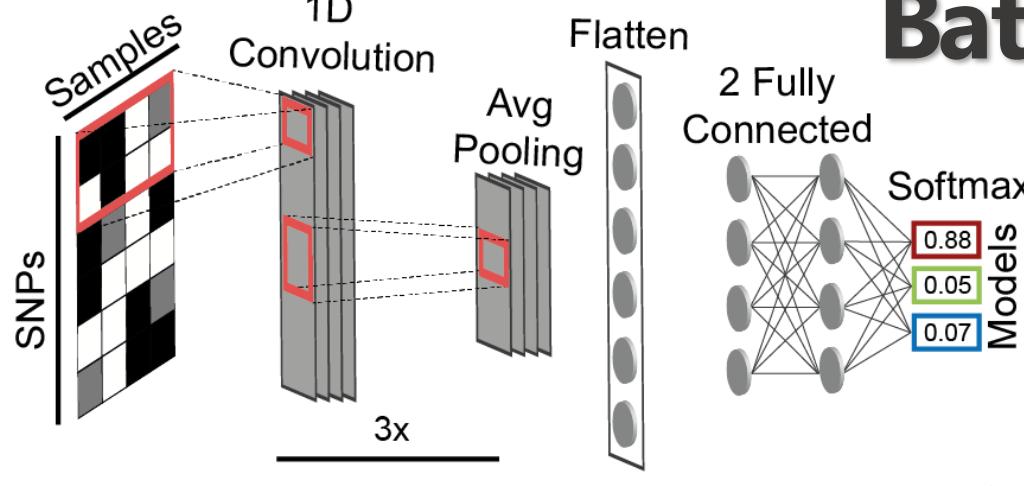
Neural Network for Genomic data



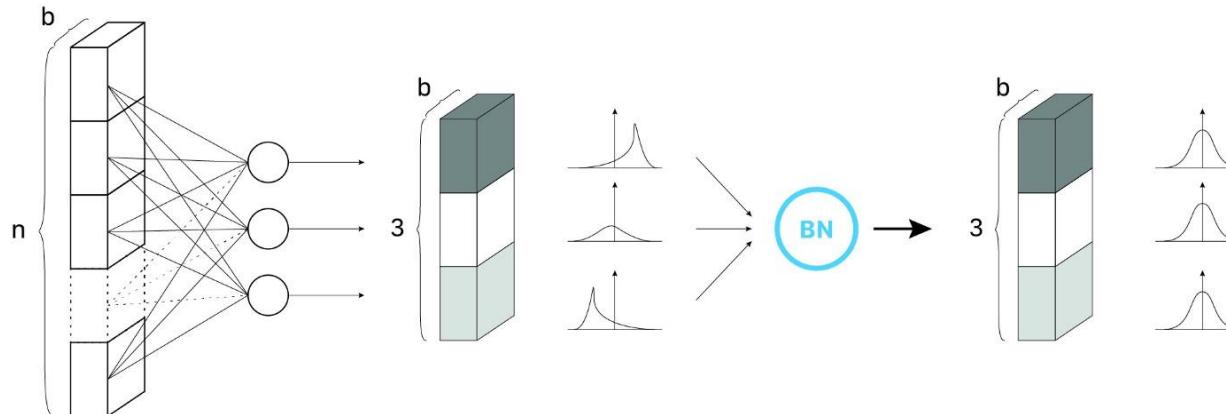
Cecil & Sudgen (2023) *PLoS Comp Biol*



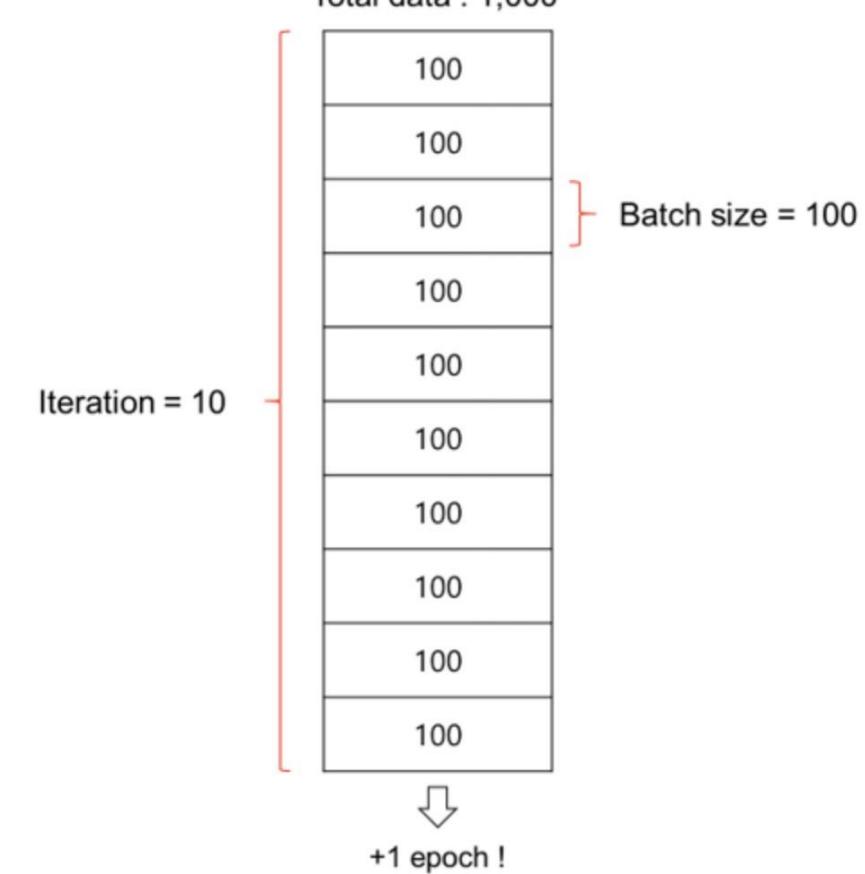
Batches and Batch Normalization



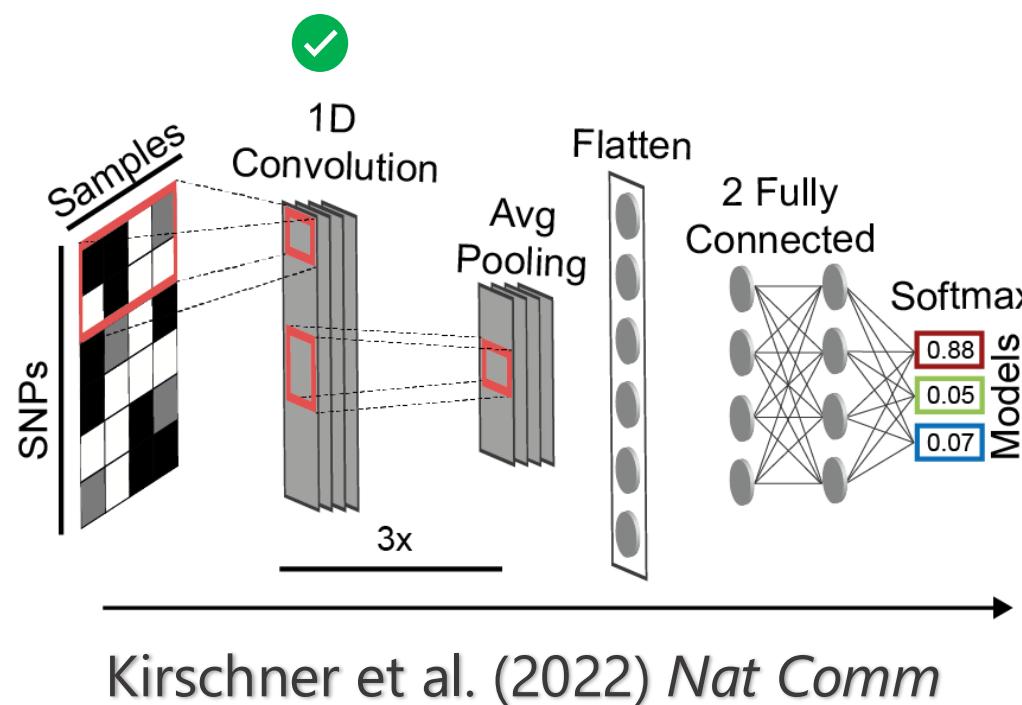
Kirschner et al. (2022) *Nat Comm*



<https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338>

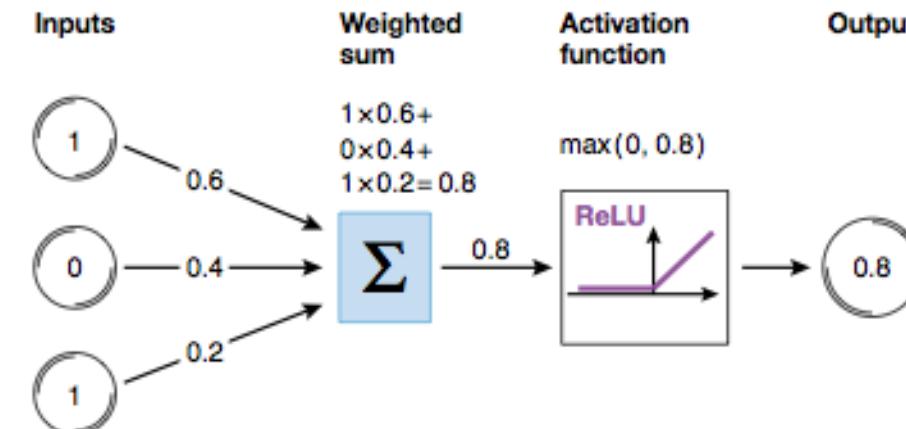


<https://jerryan.medium.com/batch-size-a15958708a6>

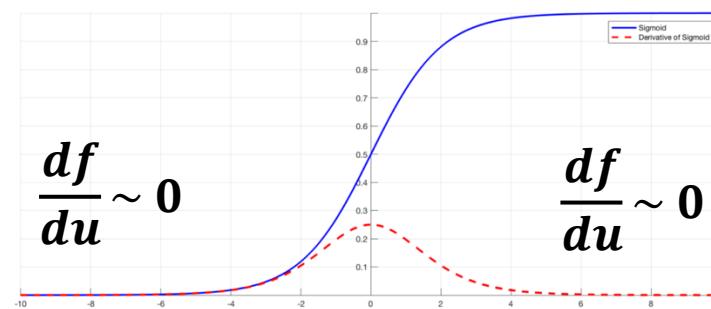


Activation Function

DL witchcraft



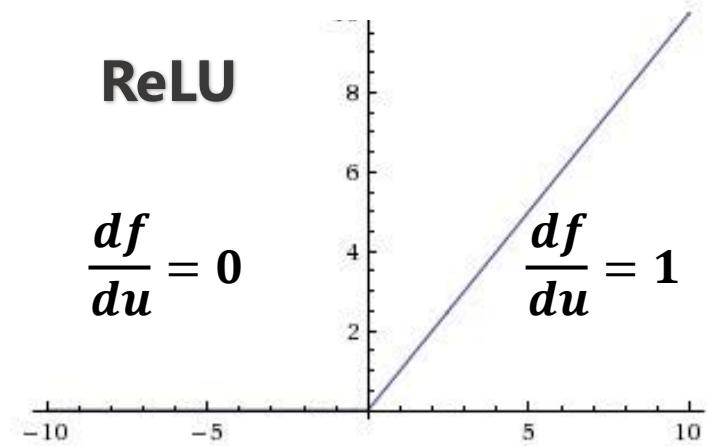
Sigmoid – Vanishing gradients



<https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484>

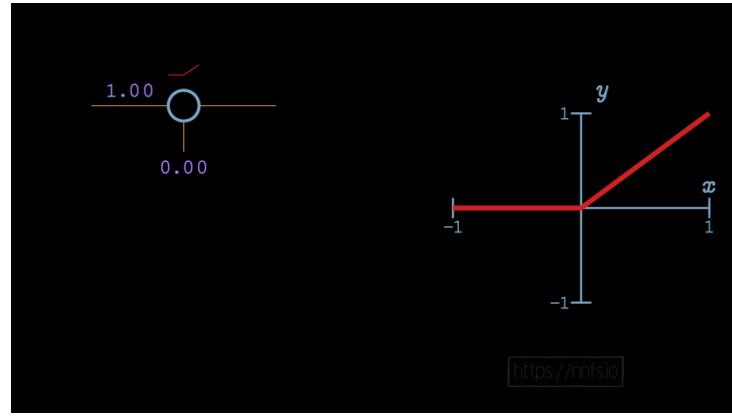
ReLU

$$\frac{df}{du} = 0$$

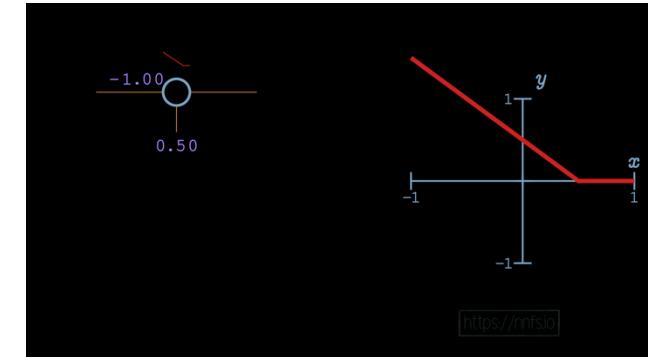


<https://www.kaggle.com/code/dansbecker/rectified-linear-units-relu-in-deep-learning/notebook>

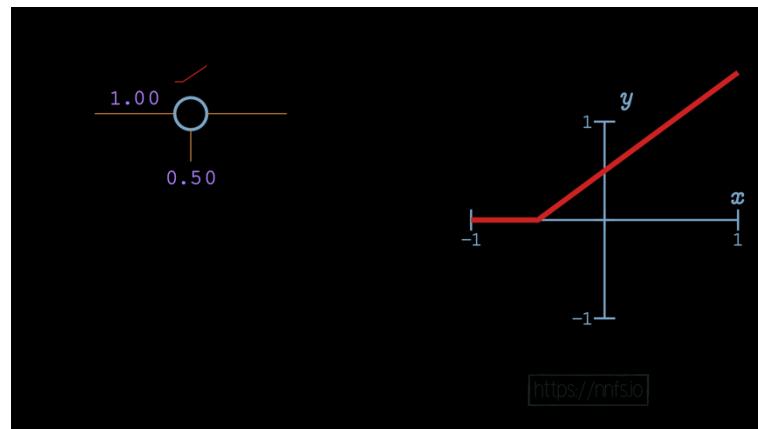
CNN Script – ReLU



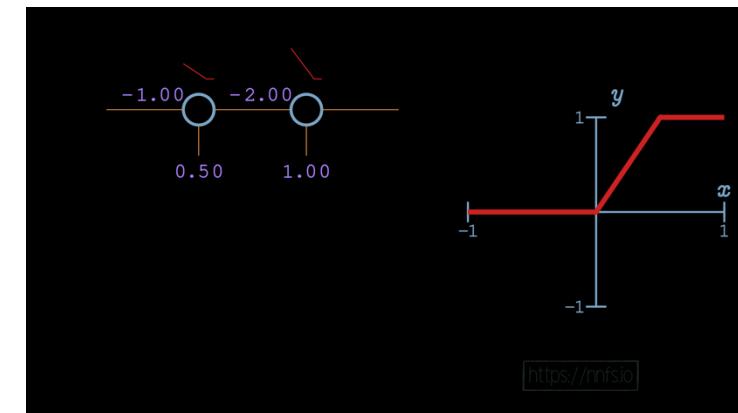
**Weight = 1
Bias = 0**



**Weight = -1
Bias = 0.5**



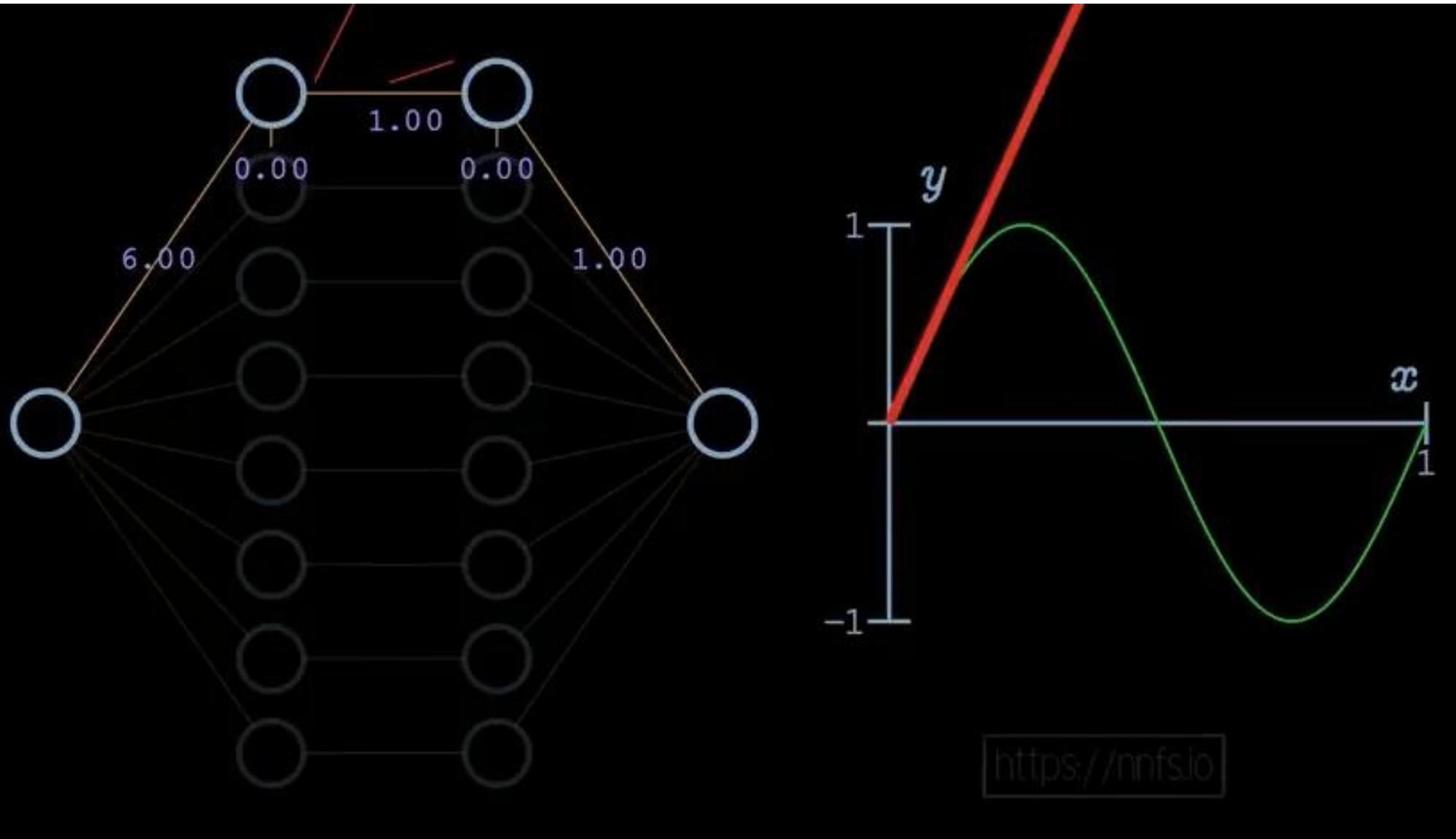
**Weight = 1
Bias = 0.5**



2 neurons

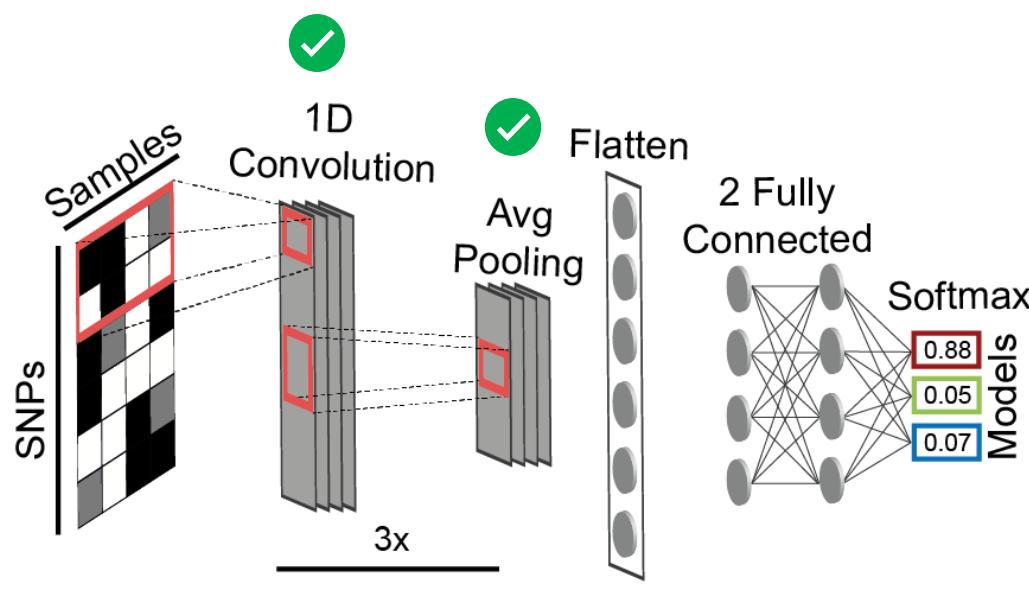
<https://nnfs.io/mvp/>

CNN Script – ReLU

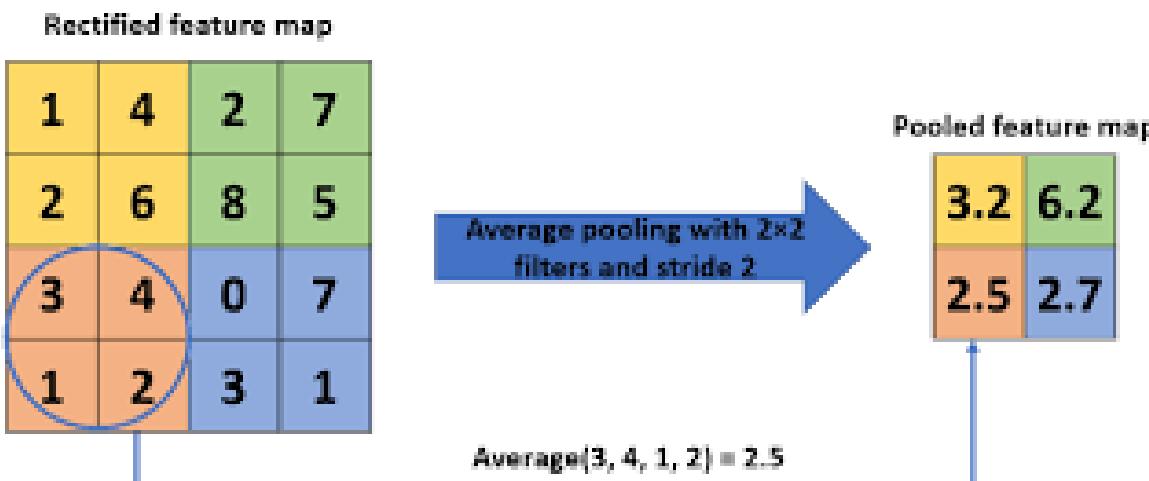


<https://nnfs.io/mvp/>

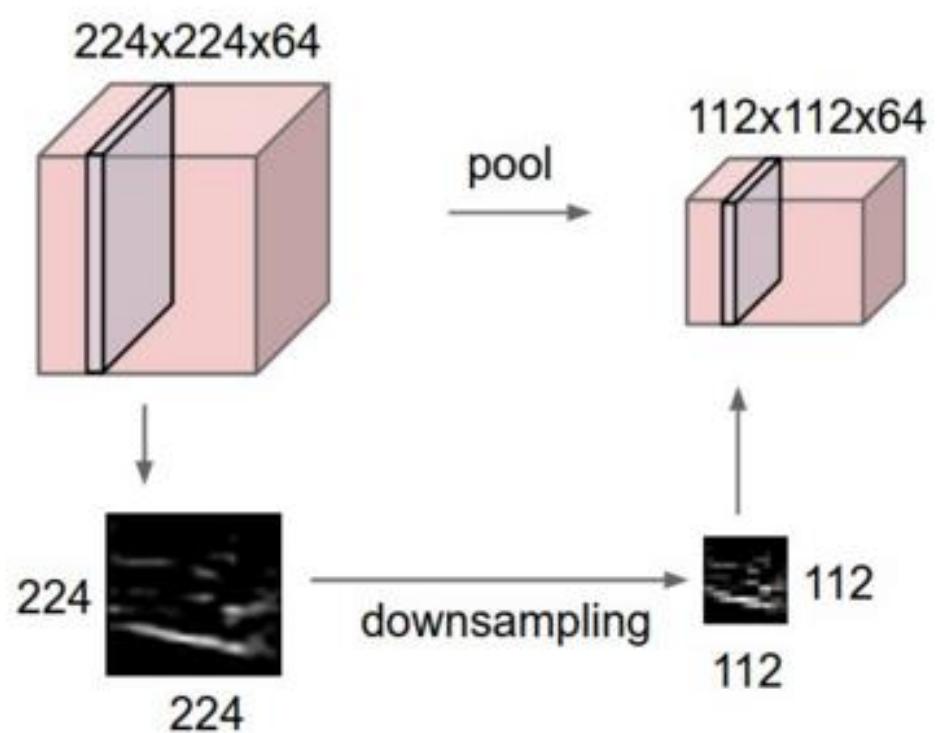
Pooling



Kirschner et al. (2022) *Nat Comm*

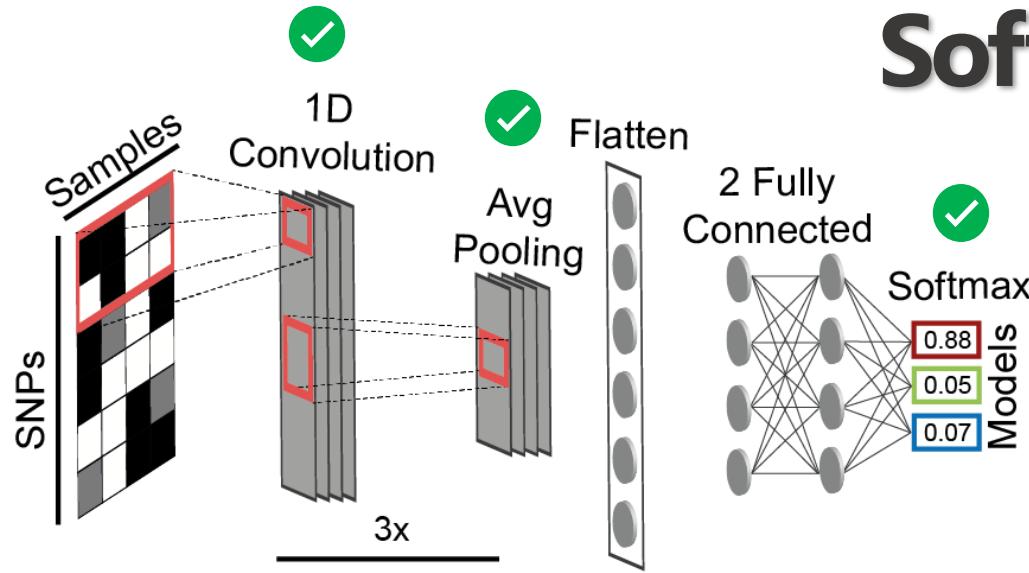


Gholamalinezhad & Khosravi
(2020) *arXiv*

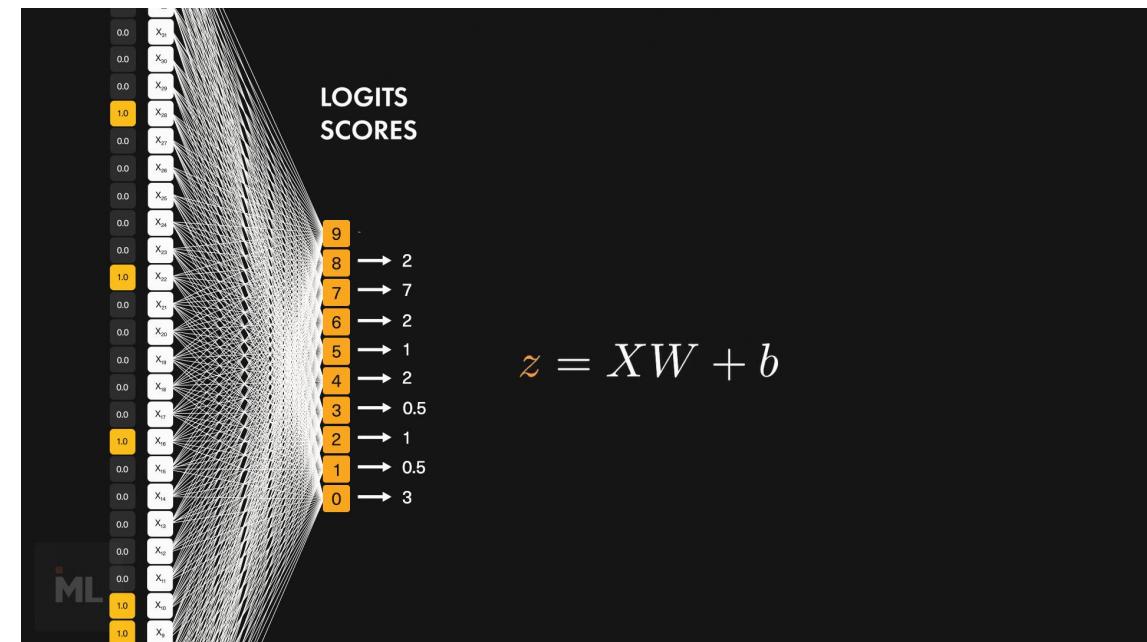


https://leonardoaraujosantos.gitbook.io/artificial-intelligence/machine_learning/deep_learning/pooling_layer

Softmax layer for Classification



Kirschner et al. (2022) *Nat Comm*



Libraries for deep Learning:
Keras, Tensorflow, pyTorch



Table 2. Central parameters of a neural network and recommended settings.

Name	Range	Default value
Learning rate	0.1, 0.01, 0.001, 0.0001	0.01
Batch size	64, 128, 256	128
Momentum rate	0.8, 0.9, 0.95	0.9
Weight initialization	Normal, Uniform, Glorot uniform	Glorot uniform
Per-parameter adaptive learning rate methods	RMSprop, Adagrad, Adadelta, Adam	Adam
Batch normalization	Yes, no	Yes
Learning rate decay	None, linear, exponential	Linear (rate 0.5)
Activation function	Sigmoid, Tanh, ReLU, Softmax	ReLU
Dropout rate	0.1, 0.25, 0.5, 0.75	0.5
L1, L2 regularization	0, 0.01, 0.001	

Hyperparameters

Some of the best hyperparameter optimization libraries are:

1. Scikit-learn
2. Scikit-Optimize
3. Optuna
4. Hyperopt
5. Ray.tune
6. Talos
7. BayesianOptimization
8. Metric Optimization Engine (MOE)
9. Spearmint
10. GPyOpt
11. SigOpt
12. Fabolas

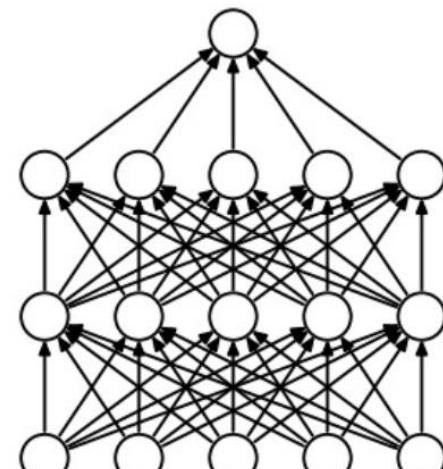
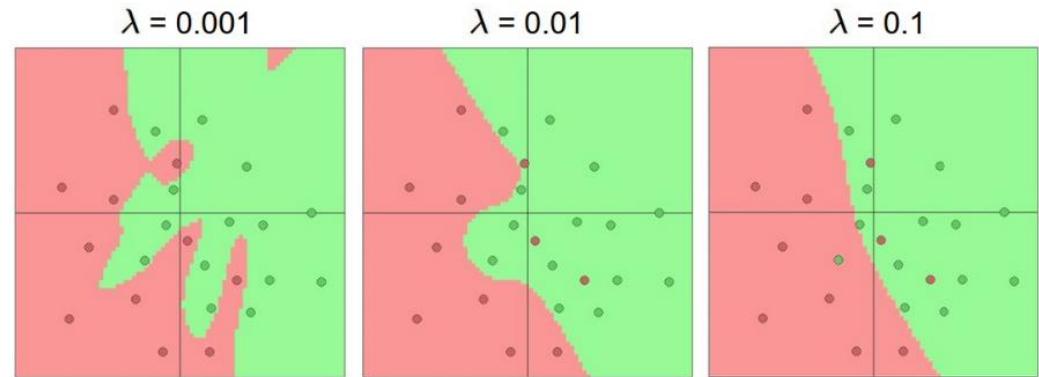
<https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide#hyperopt>

Hyperparameters

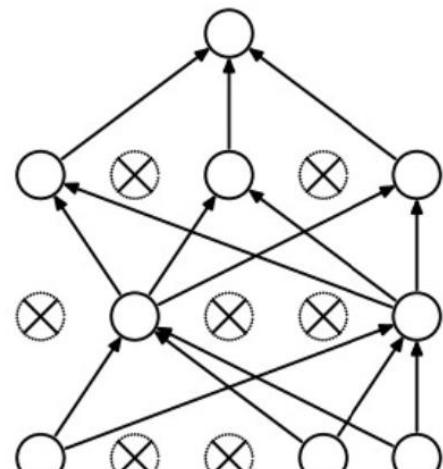
Table 2. Central parameters of a neural network and recommended settings.

Name	Range	Default value
Learning rate	0.1, 0.01, 0.001, 0.0001	0.01
Batch size	64, 128, 256	128
Momentum rate	0.8, 0.9, 0.95	0.9
Weight initialization	Normal, Uniform, Glorot uniform	Glorot uniform
Per-parameter adaptive learning rate methods	RMSprop, Adagrad, Adadelta, Adam	Adam
Batch normalization	Yes, no	Yes
Learning rate decay	None, linear, exponential	Linear (rate 0.5)
Activation function	Sigmoid, Tanh, ReLU, Softmax	ReLU
Dropout rate	0.1, 0.25, 0.5, 0.75	0.5
L1, L2 regularization	0, 0.01, 0.001	

Regularization



(a) Standard Neural Net

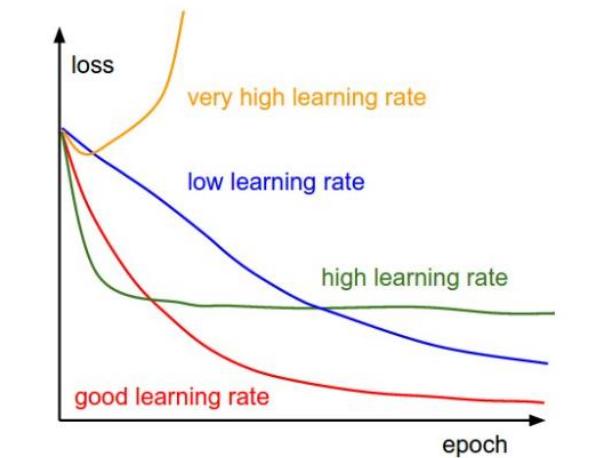


(b) After applying dropout.

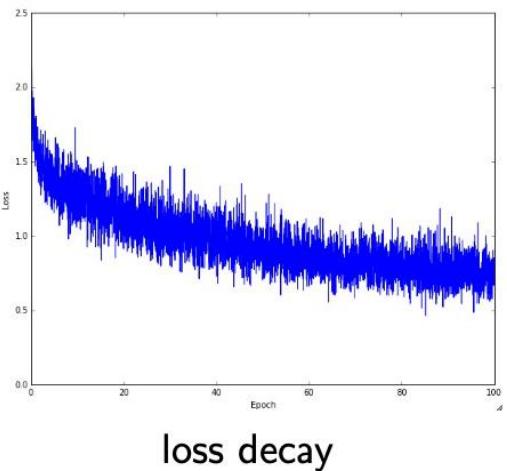
Hyperparameters

Table 2. Central parameters of a neural network and recommended settings.

Name	Range	Default value
Learning rate	0.1, 0.01, 0.001, 0.0001	0.01
Batch size	64, 128, 256	128
Momentum rate	0.8, 0.9, 0.95	0.9
Weight initialization	Normal, Uniform, Glorot uniform	Glorot uniform
Per-parameter adaptive learning rate methods	RMSprop, Adagrad, Adadelta, Adam	Adam
Batch normalization	Yes, no	Yes
Learning rate decay	None, linear, exponential	Linear (rate 0.5)
Activation function	Sigmoid, Tanh, ReLU, Softmax	ReLU
Dropout rate	0.1, 0.25, 0.5, 0.75	0.5
L1, L2 regularization	0, 0.01, 0.001	



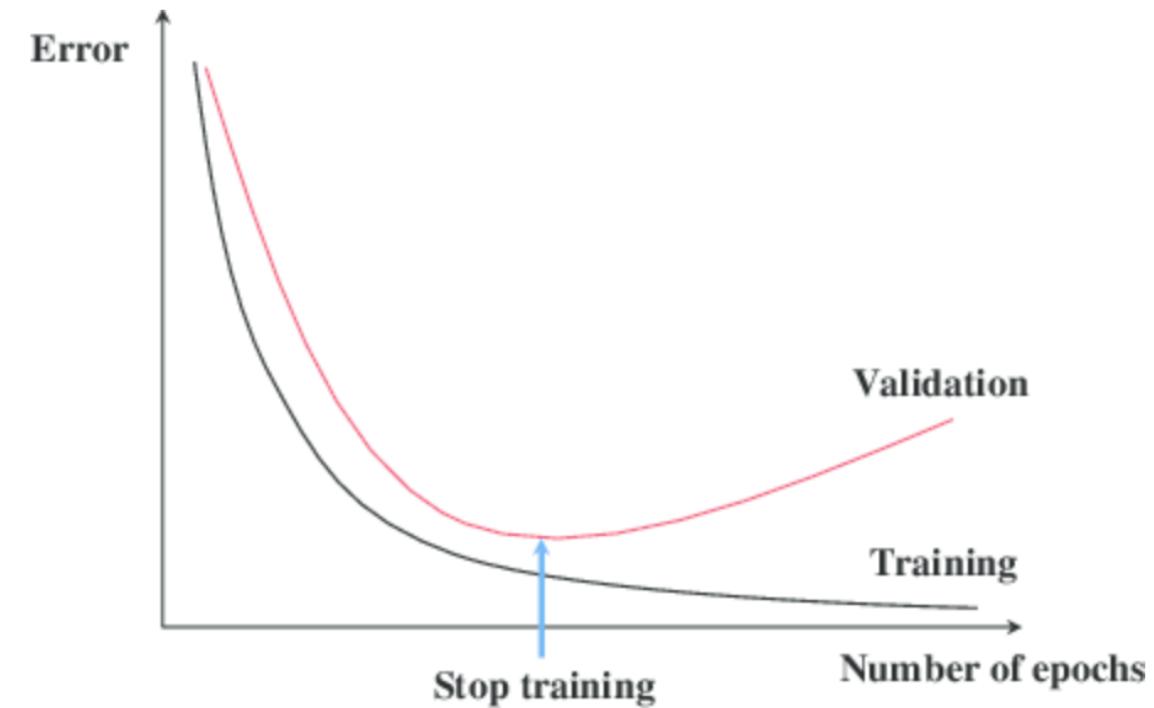
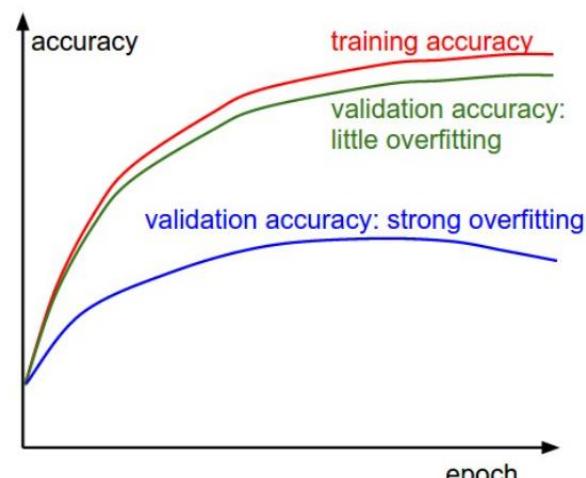
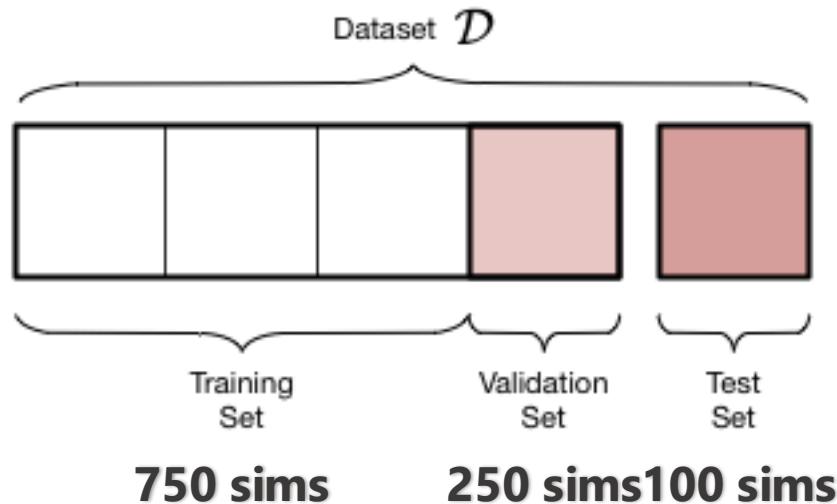
effects of different learning rates



loss decay

Early Stop to avoid overfitting

<https://se-education.org/learningresources/contents/ai/ml.html>



<https://towardsdatascience.com/a-practical-introduction-to-early-stopping-in-machine-learning-550ac88bc8fd>

Part III: Practical on Image classification with CNN.

- **Practical Exercise 4:**
- Access <https://poloclub.github.io/cnn-explainer/>
- Scroll down to the Understanding hyperparameters section. Try to understand what is kernel size, padding, and stride.
- Use the graphical interface to try to better understand how convolutions work and what features are being extracted from the images.
- You can also upload the alignment images (Model1 to Model3.png) from different scenarios (but notice the network was not trained with that type of image). Does the network give an output prediction for the alignments?

Goals

- Conceive and simulate genetic data under competing demographic scenarios 
- Understand deep learning background and how a CNN works 
- How to use deep learning to compare demographic scenarios



DL witchcraft

