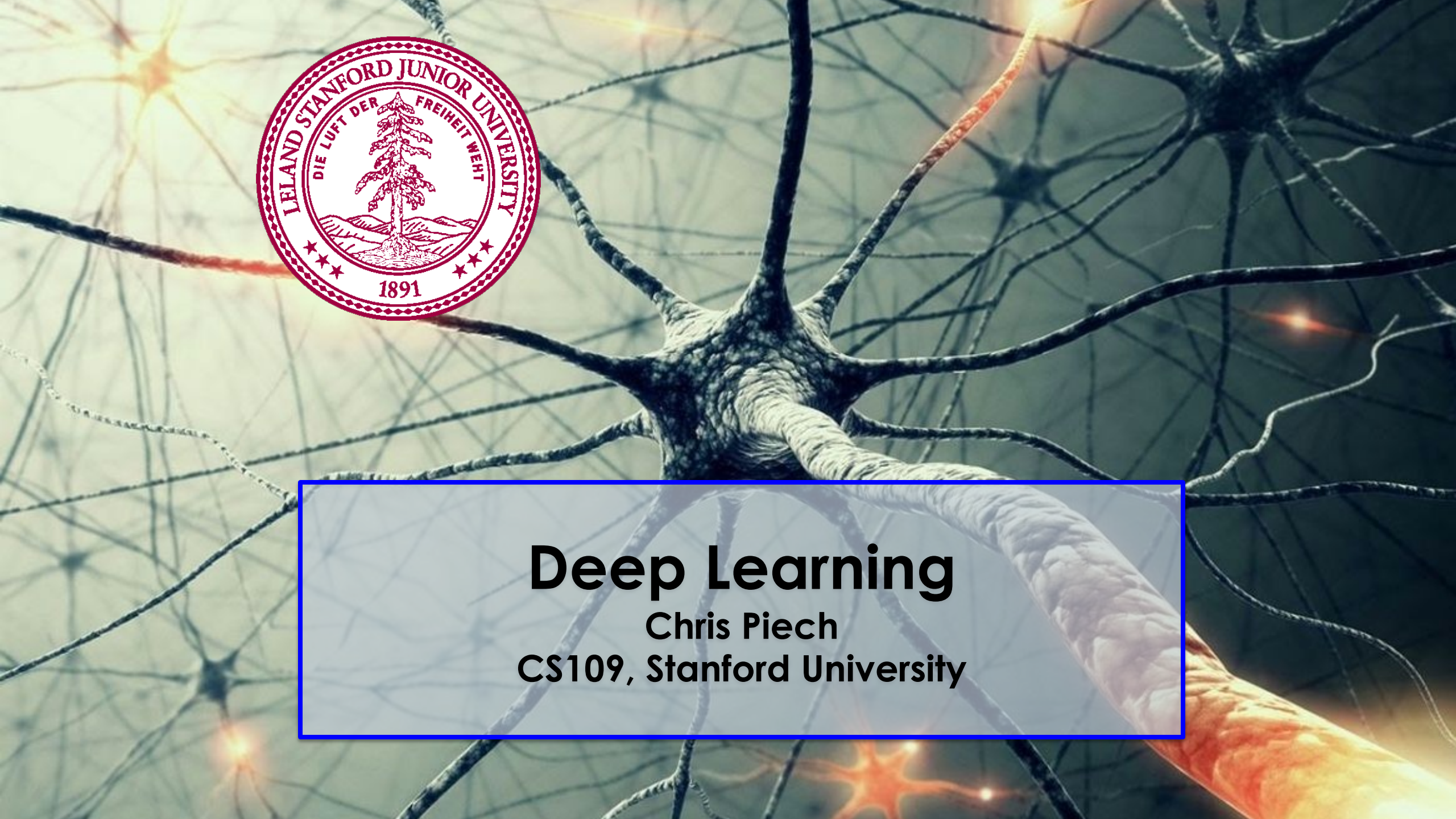
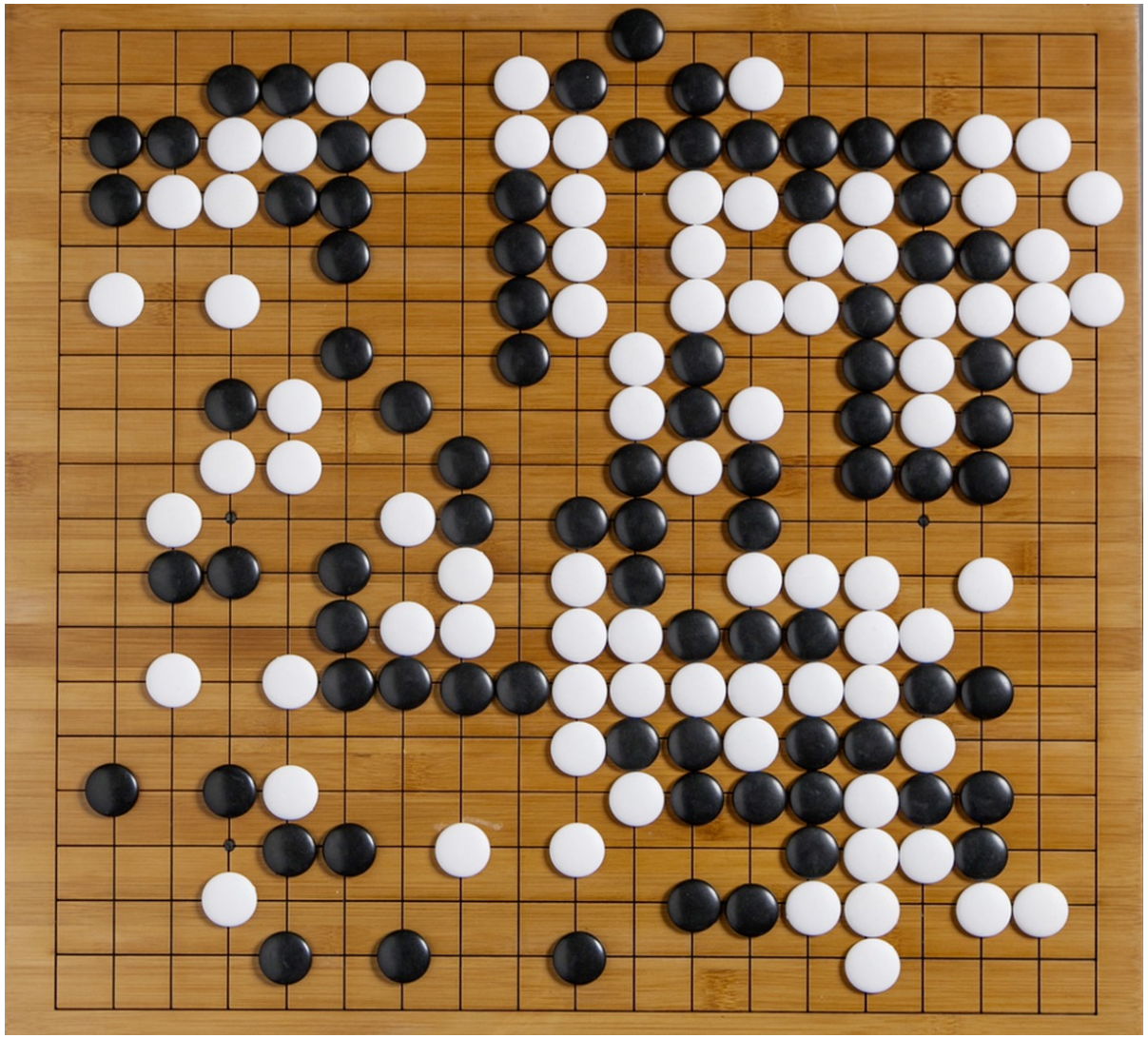


Deep Learning

Chris Piech
CS109, Stanford University

Innovations in deep learning



AlphaGO (2016)

Deep learning (neural networks) is the core idea driving the current revolution in AI.

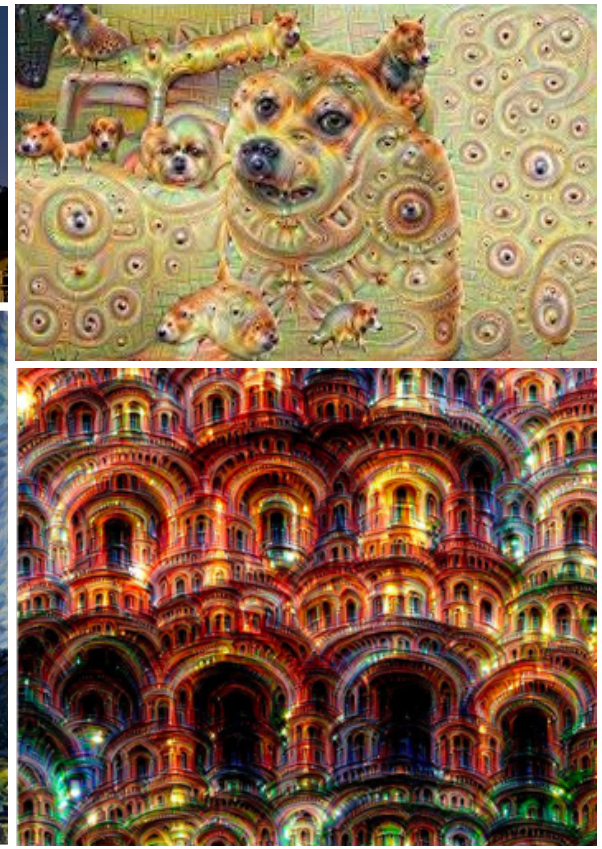
Notes:

- Checkers is the last **solved** game (from game theory, where perfect player outcomes can be fully predicted from any gameboard).
  https://en.wikipedia.org/wiki/Solved_game
- The first machine learning algorithm defeated a world champion in Chess in 1996.
  https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer)

Computers making art

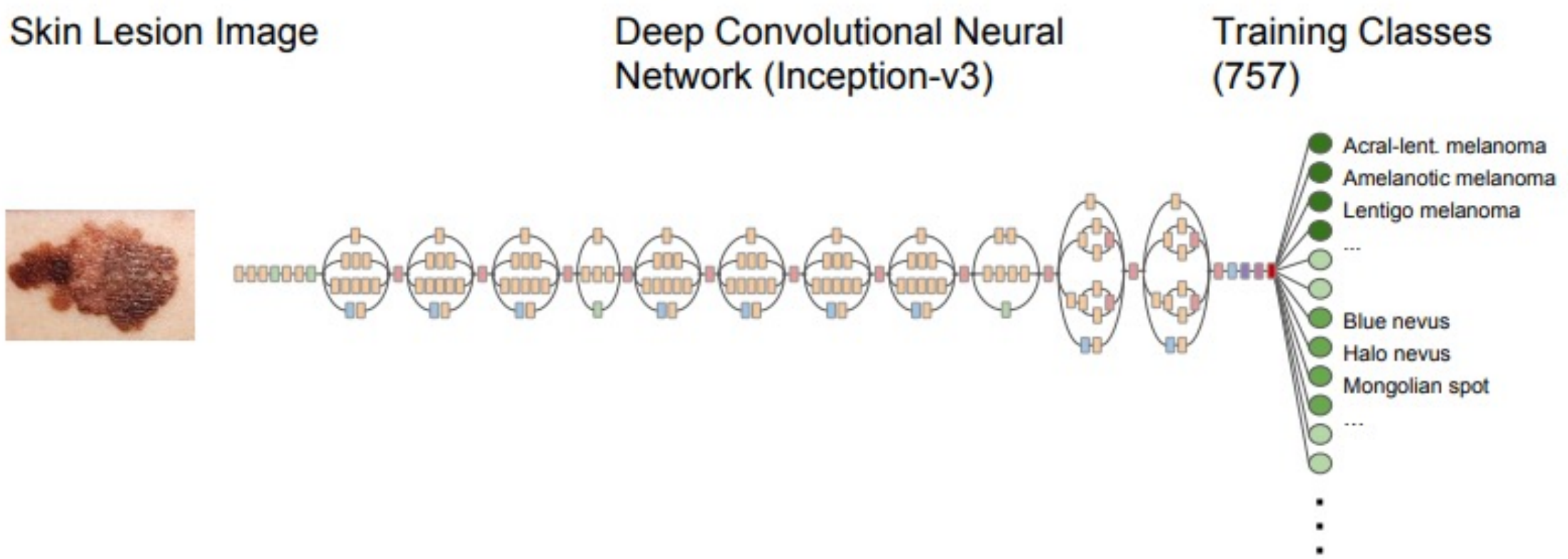
The Next Rembrandt
https://medium.com/@DutchDigital/the-next-rembrandt-bringing-the-old-master-back-to-life-35dfb1653597

A Neural Algorithm of Artistic Style
https://arxiv.org/abs/1508.06576
https://github.com/jcjohnson/neural-style



Google Deep Dream
https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Detecting skin cancer



Esteva, Andre, et al. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* 542.7639 (2017): 115-118.

Announcements

1. Final PSet deadline: we **will** allow for late submissions. Due Wed Dec 1st. Grace period until Friday Dec 3rd.

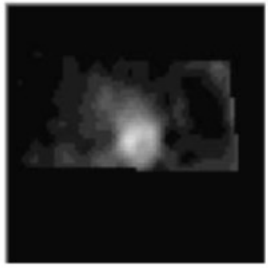
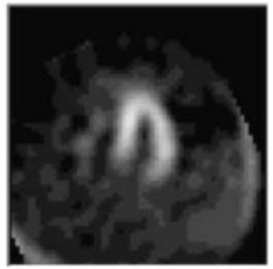
2. Thanksgiving break is coming up. Enjoy!

3. Final period of class will be fast after thanksgiving.

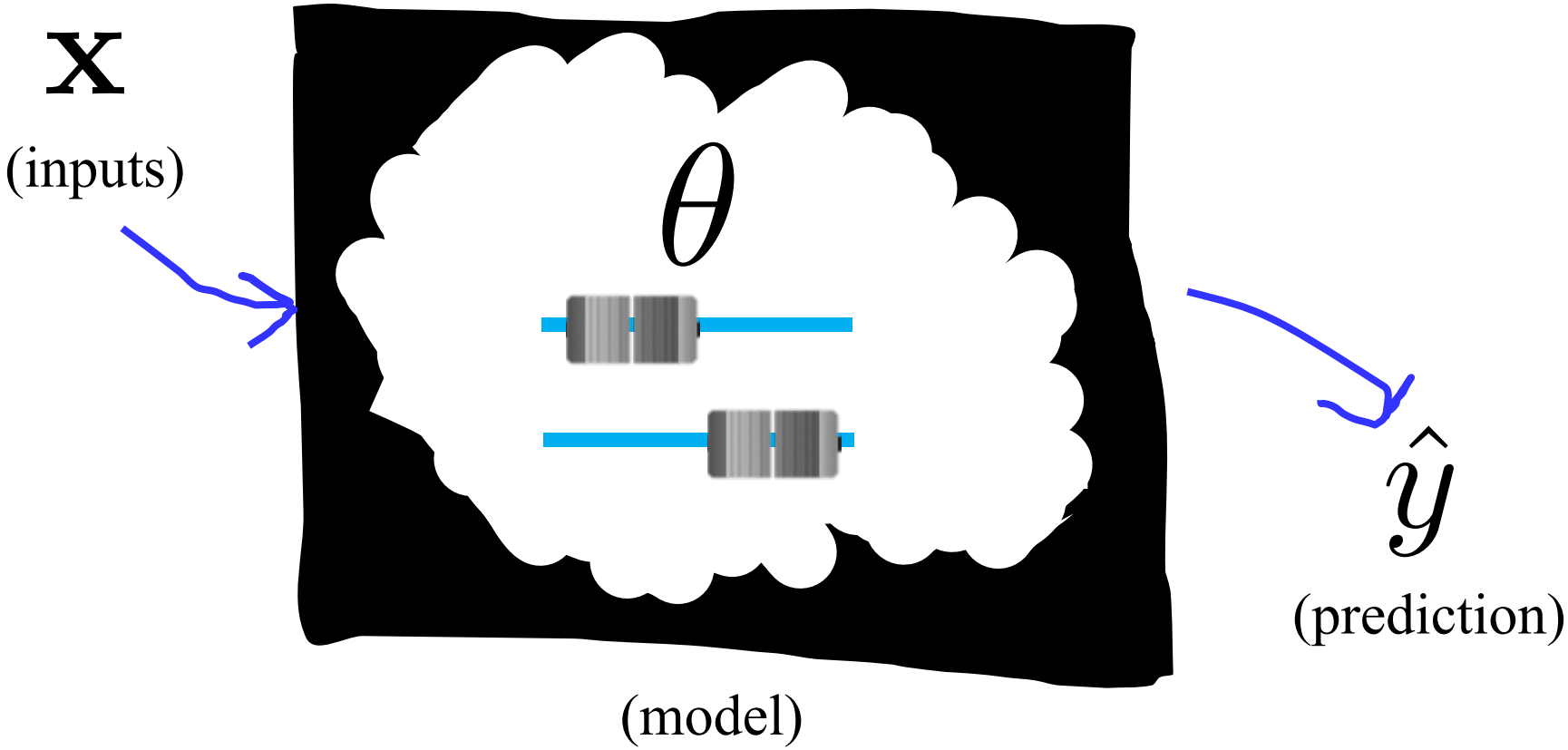
4. No more concept checks. You did it!

Review

Classification Task

	ROI 1	ROI 2	...	ROI *m*	Output
					
Heart 1	0	1		1	0
Heart 2	1	1		1	0
			⋮		⋮
Heart *n*	0	0		0	1

Machine Learning



The Training / Testing Paradigm

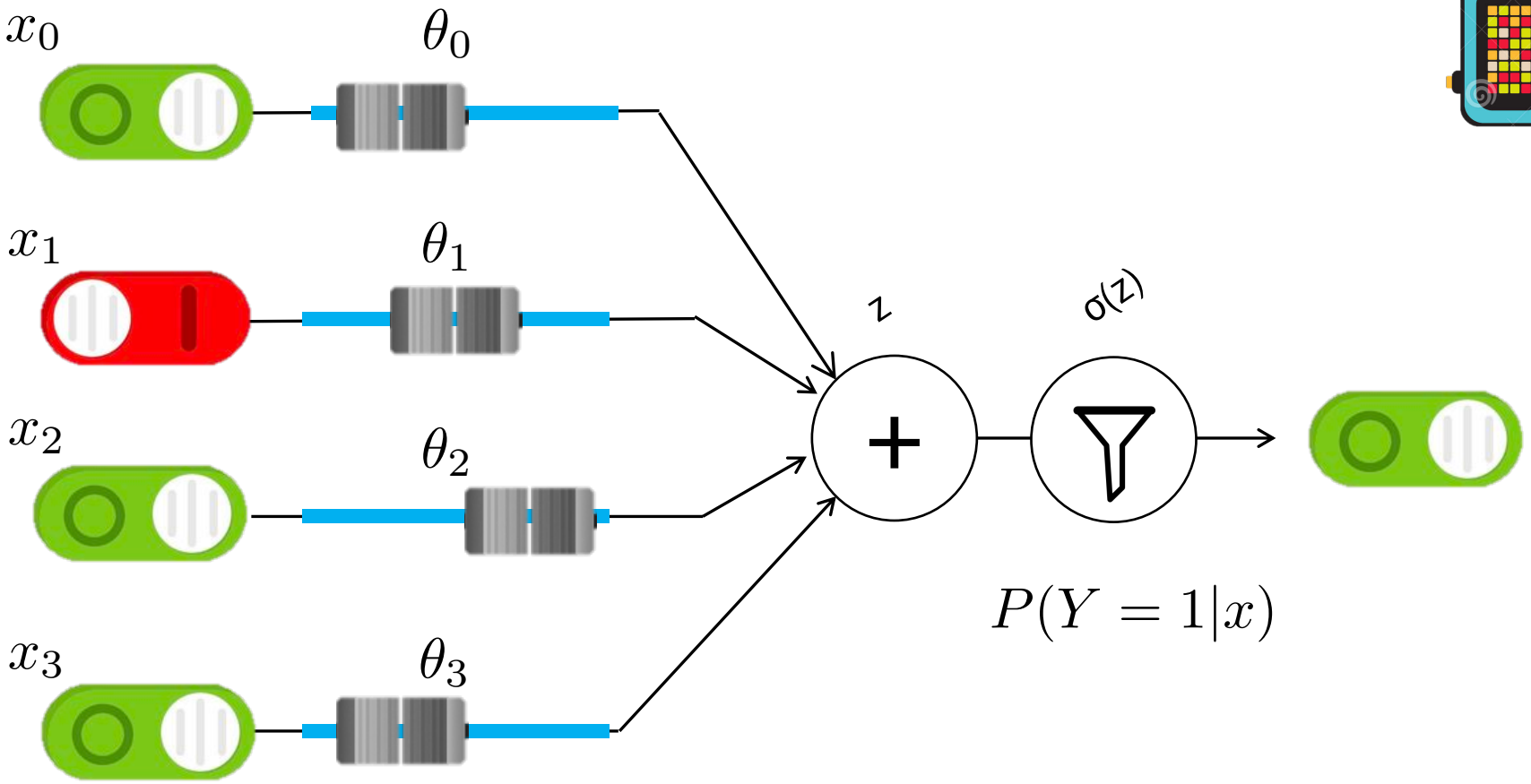
Learn your parameters

Make sure that they work

If your model passes testing…

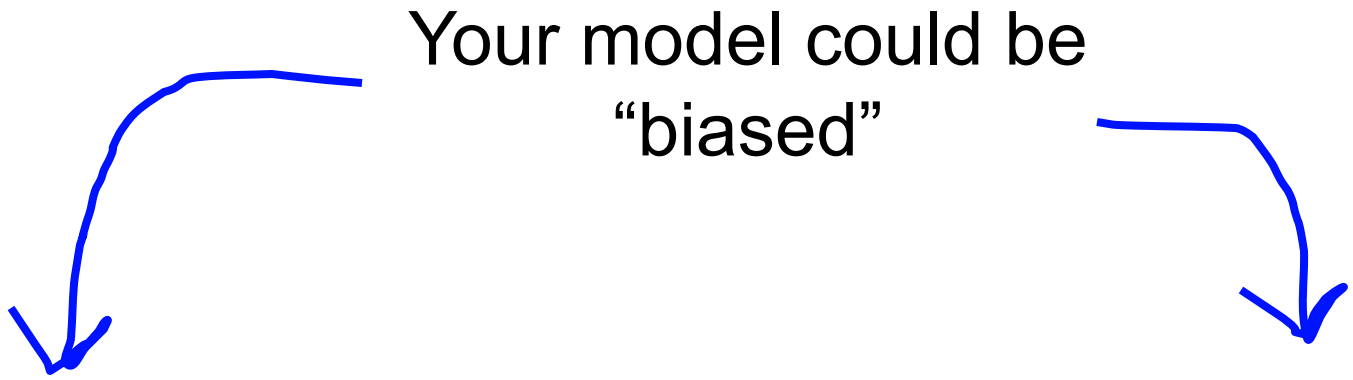
Deployment

Logistic Regression



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

But Watch Out!



Procedural Un-Fairness:

Causes harm by using protected demographics in the decision-making or classification ***process***.

Distributive Un-Fairness:

Causes harm by unfair ***outcomes*** with respect to a protected demographic.

Procedural Fairness is Hard

Sapiezynski et. al 2019,

https://sapiezynski.com/papers/sapiezynski2019algorithms.pdf

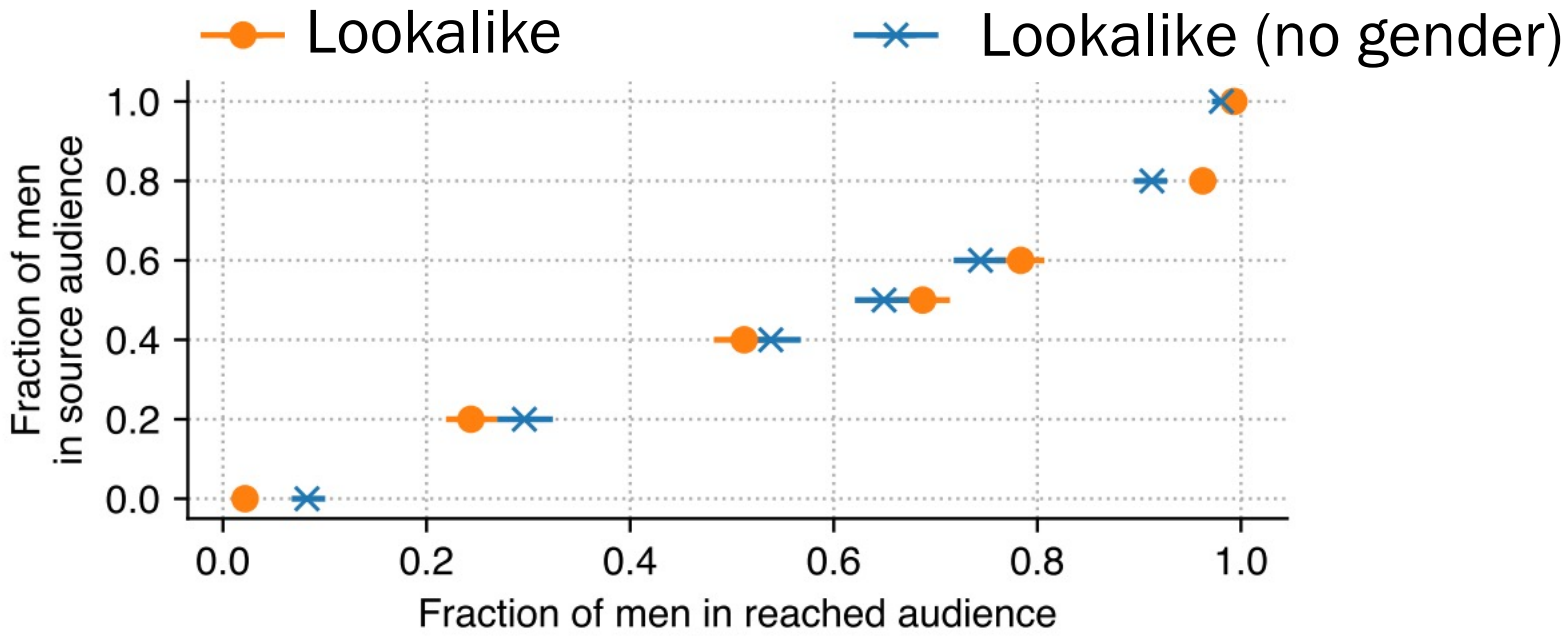


Figure 2: Gender breakdown of ad delivery to Lookalike and Special Ad audiences created from the same source audience with varying fraction of male users, using the same ad creative. We can observe that both Lookalike and Special Ad audiences reflect the gender distribution of the source audience, despite the lack of gender being provided as an input to Special Ad Audiences.

Yo, Piotr, you got your axis backwards ☺

Distributive Fairness #1: Parity

Fairness definition #1: Parity

An algorithm satisfies "parity" if the probability that the algorithm makes a positive prediction ($G = 1$) is the same regardless of begin conditioned on demographic variable.

D: protected demographic
G: guess of your model (aka y hat)
T: the true value (aka y)

$$P(G=1|D=1) = P(G = 1 \mid D = 0)$$

Distributive Fairness #2: Calibration

Fairness definition #2: Calibration

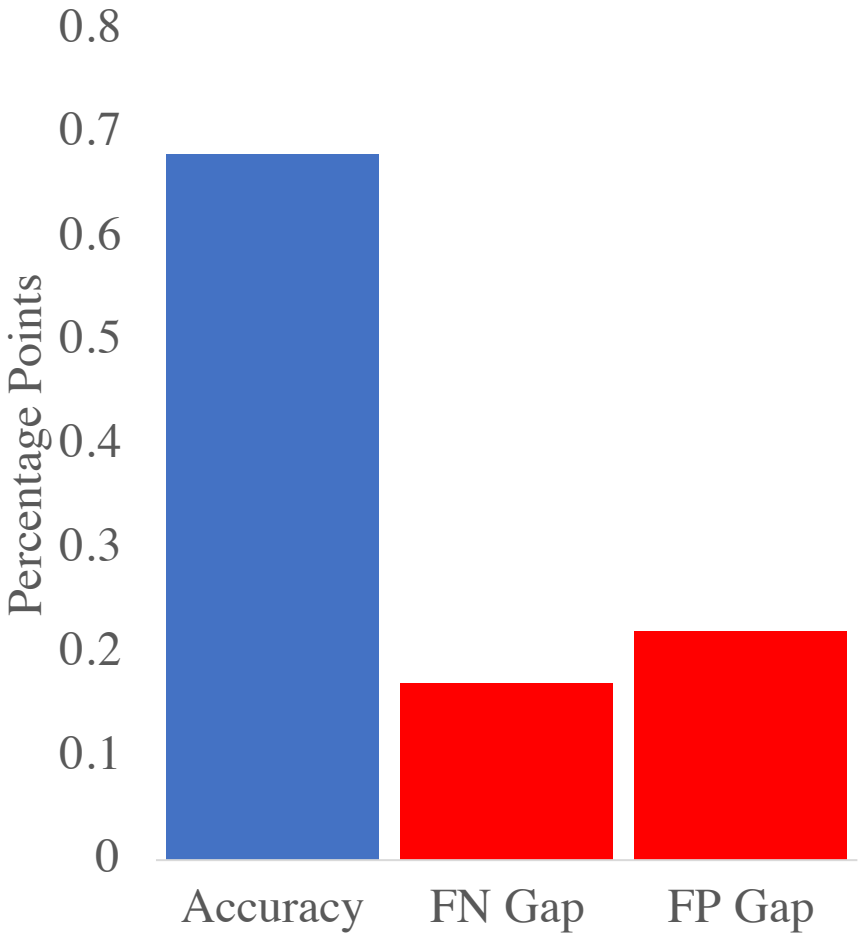
An algorithm satisfies "calibration" if the probability that the algorithm is correct ($G = T$) is the same regardless of demographics.

D: protected demographic
G: guess of your model (aka y hat)
T: the true value (aka y)

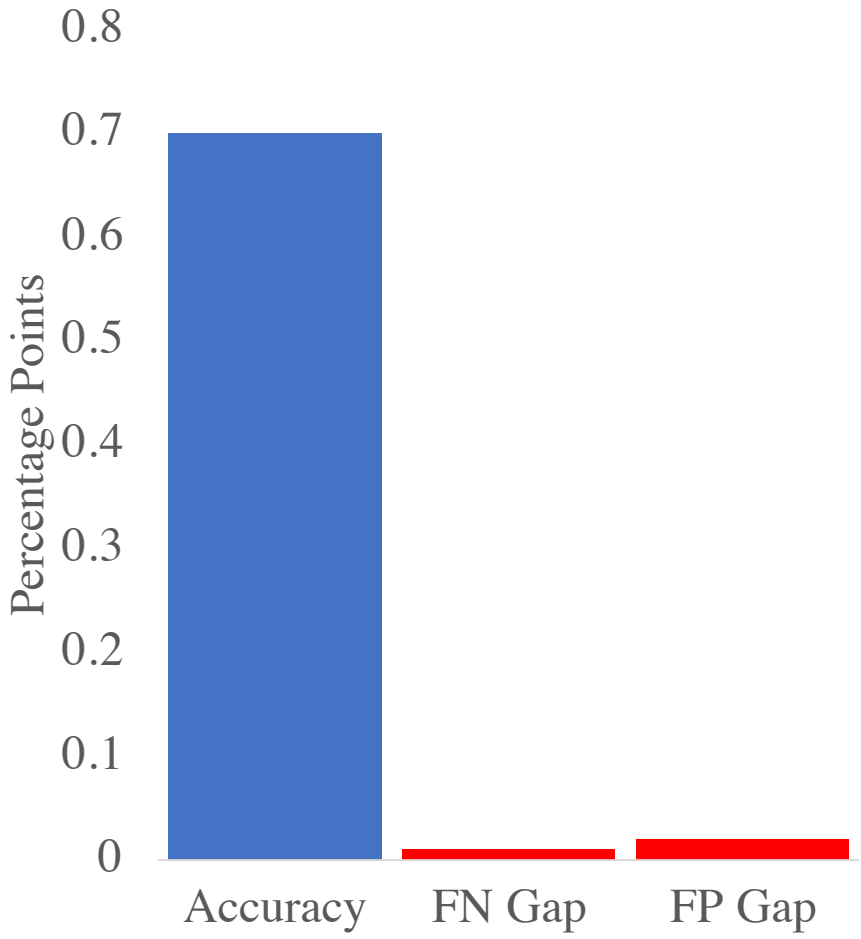
$$P(G = T|D = 0) = P(G = T|D = 1)$$

Can We Train Out Distributive Unfairness?

Before: Compas is Biased

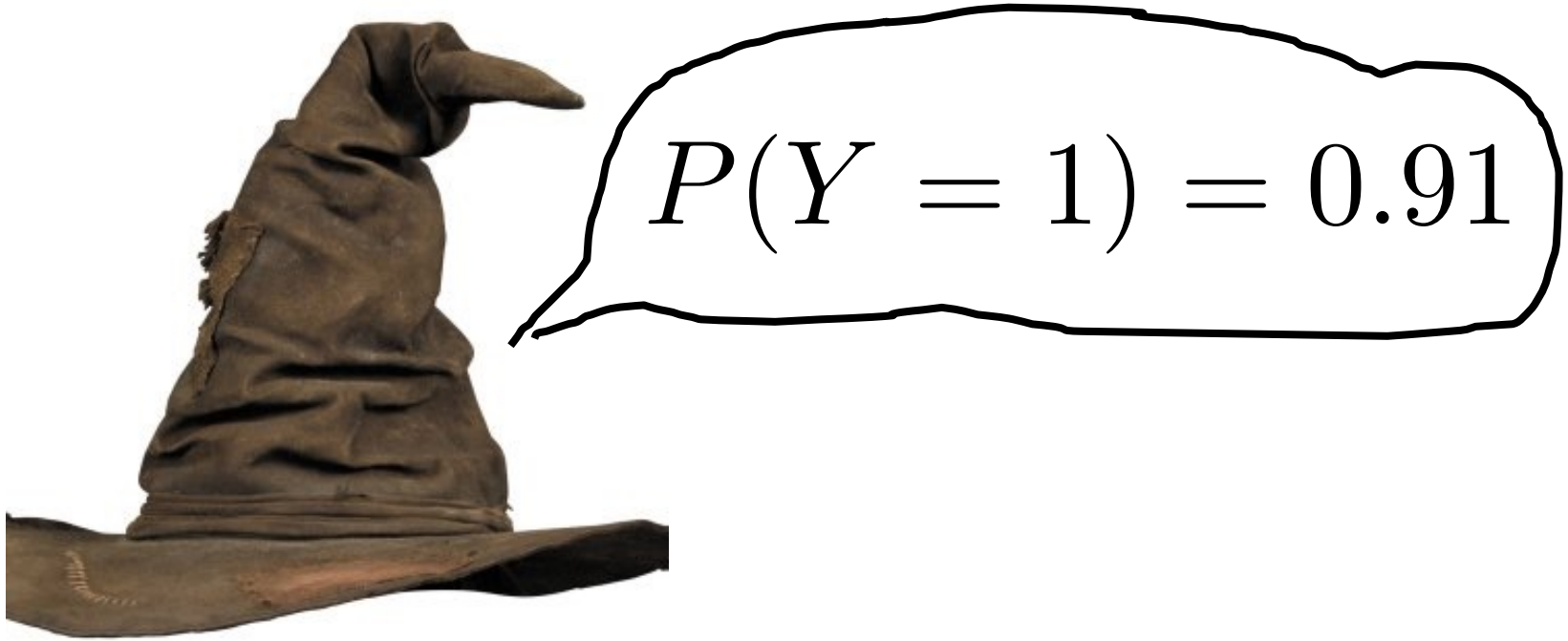


After: FN Gap is reduced



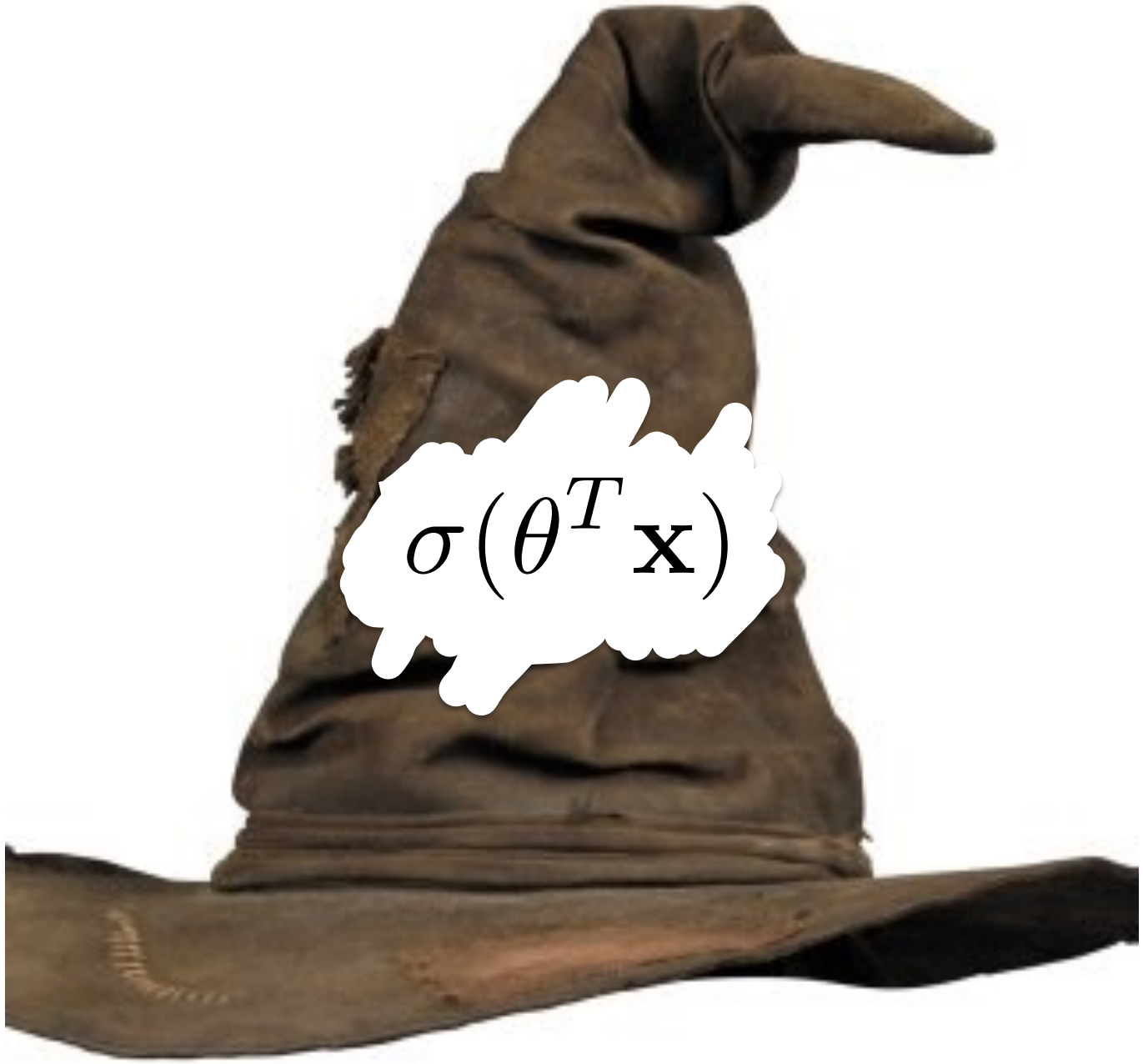
A Journey From Pure Math to Skin Cancer Detection

Logistic Regression is like the Harry Pottery Sorting Hat

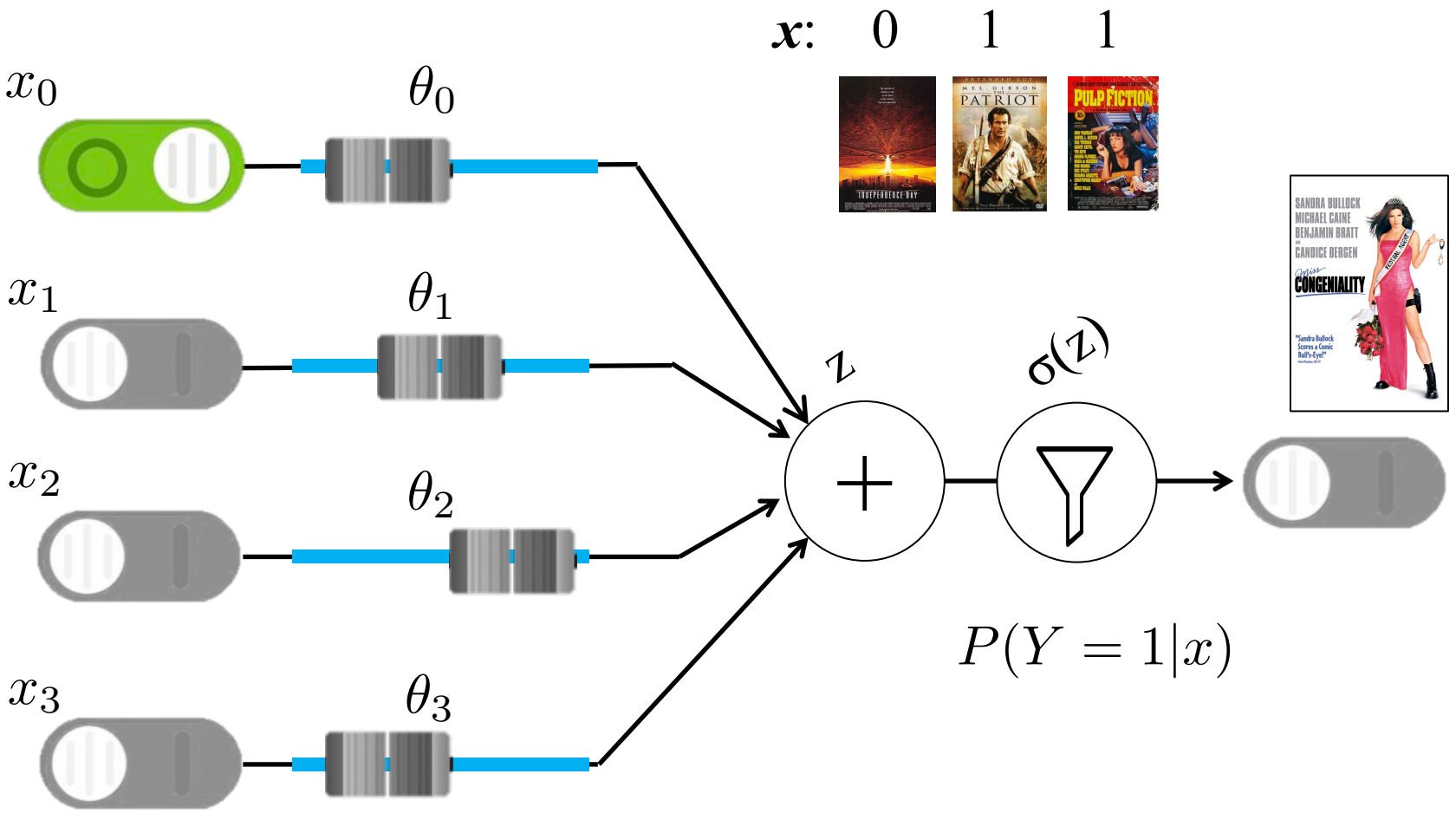


$\mathbf{X}$

Logistic Regression is like the Harry Pottery Sorting Hat

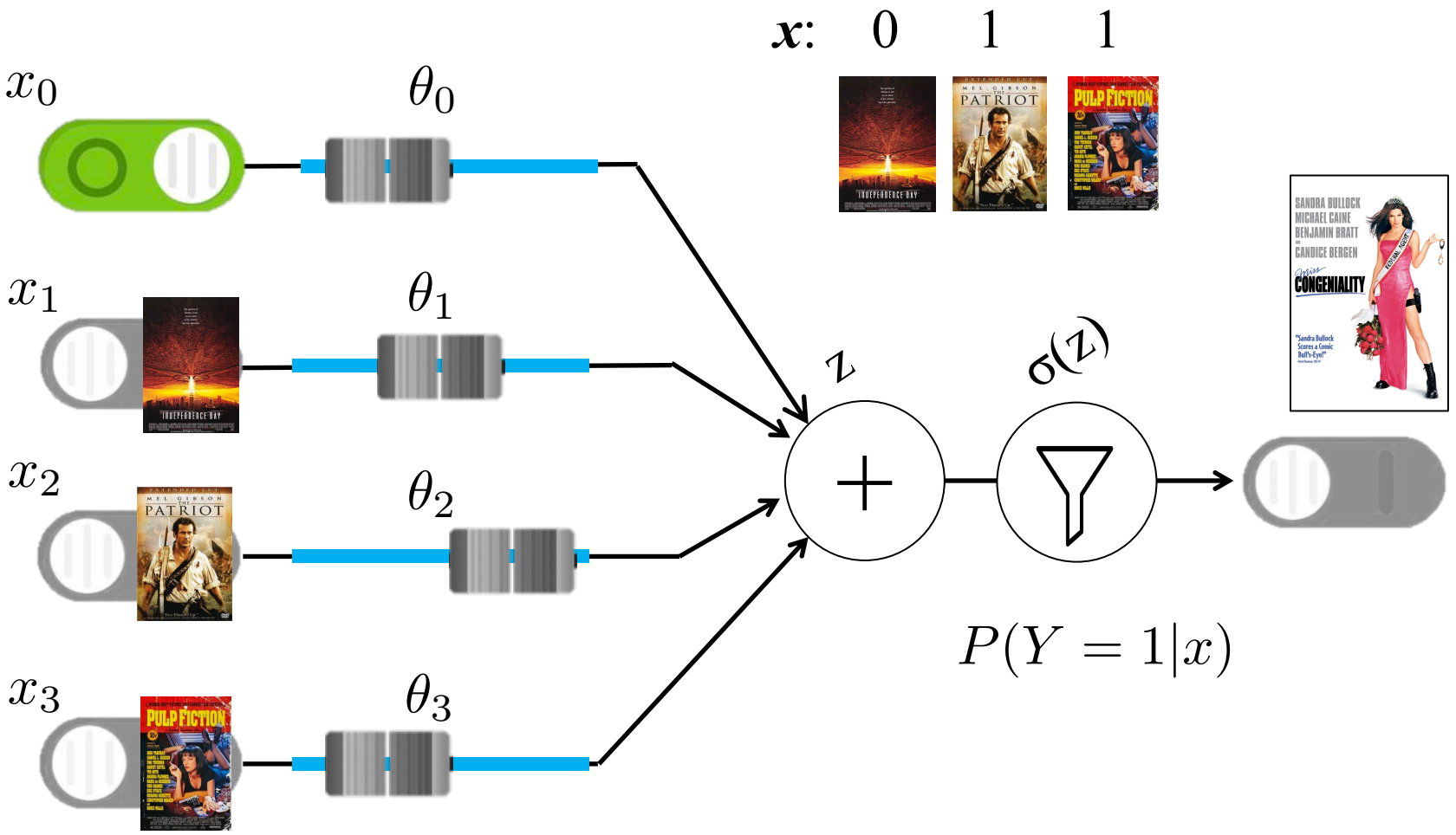


Logistic Regression



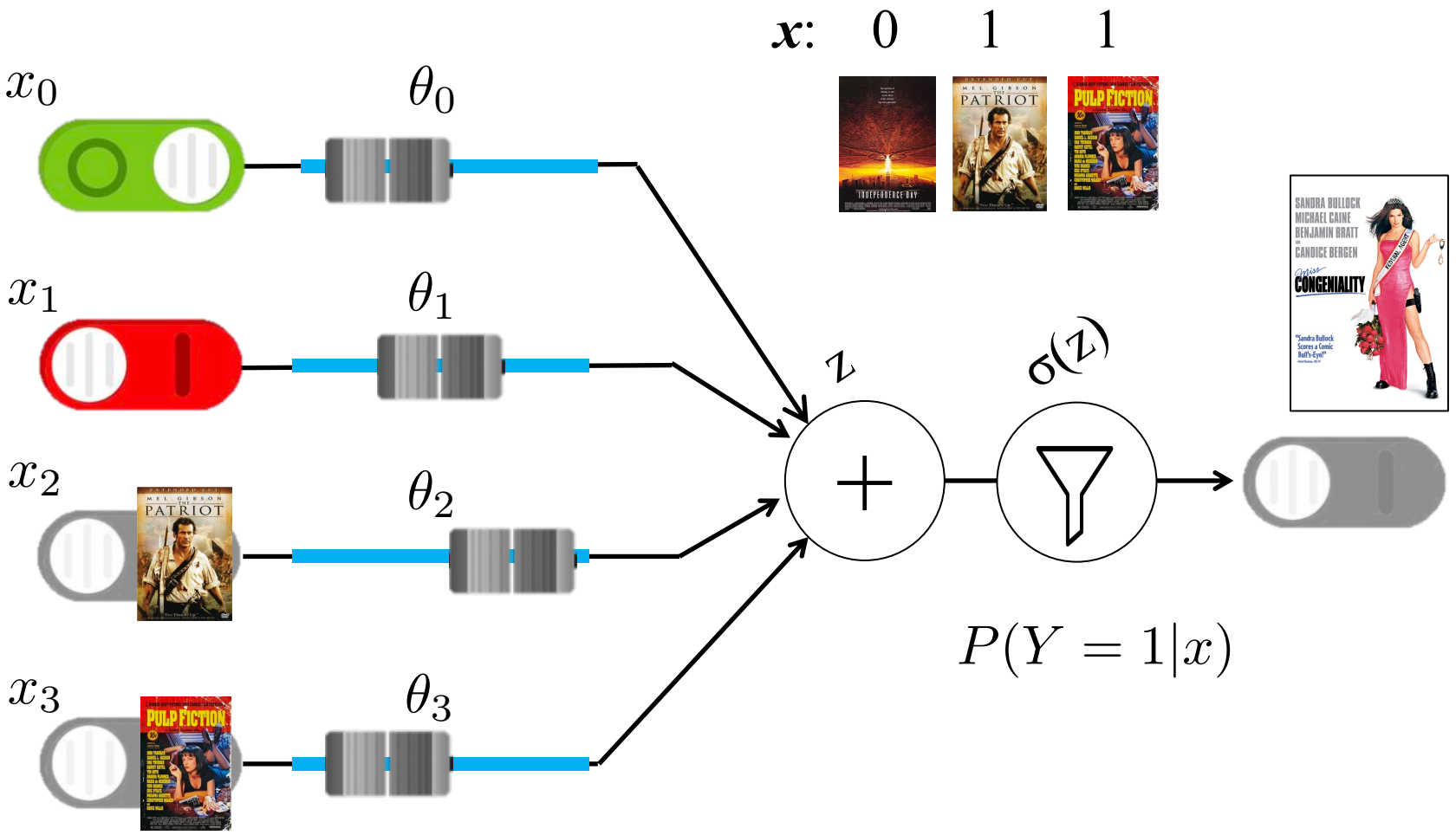
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



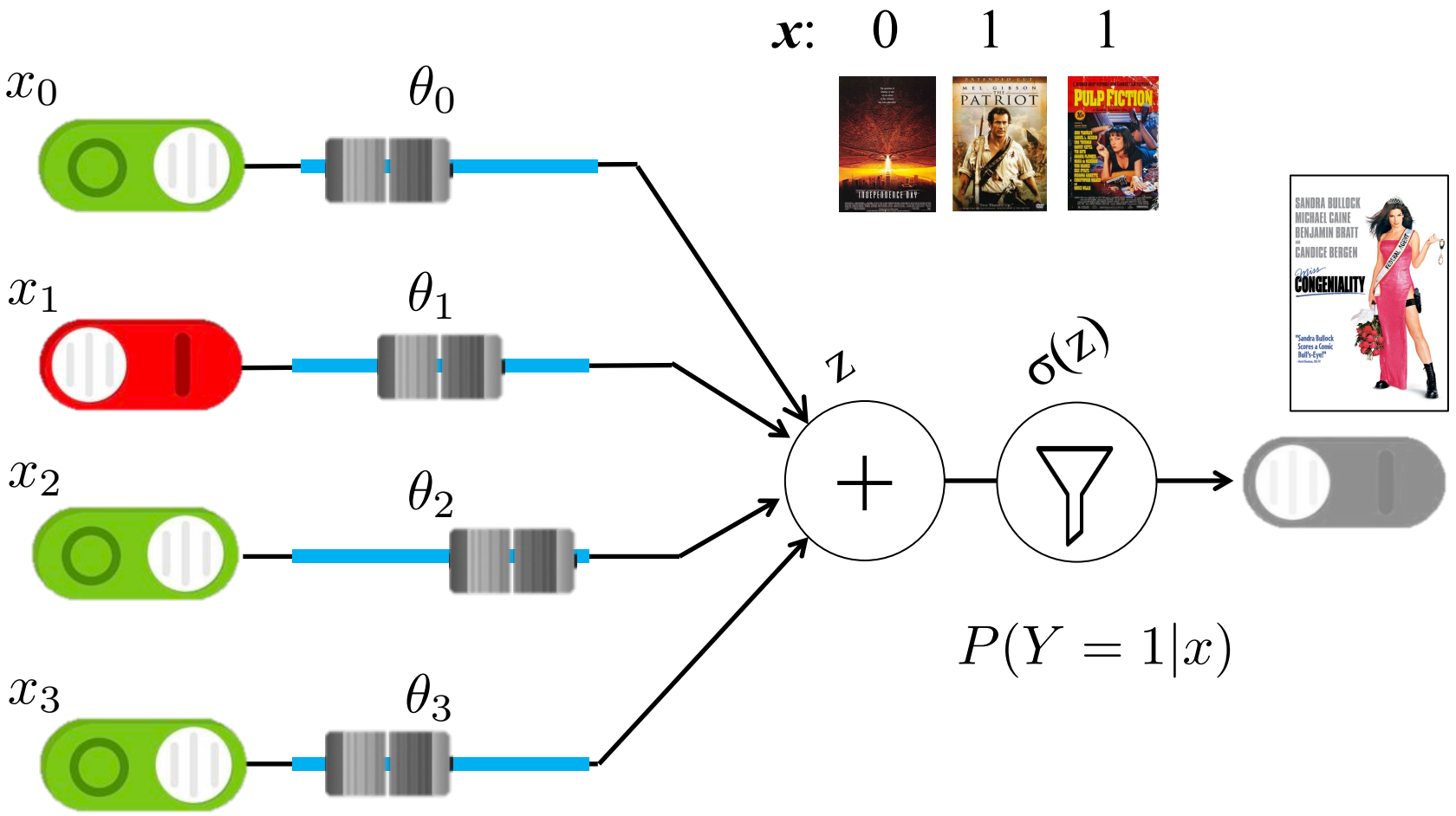
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



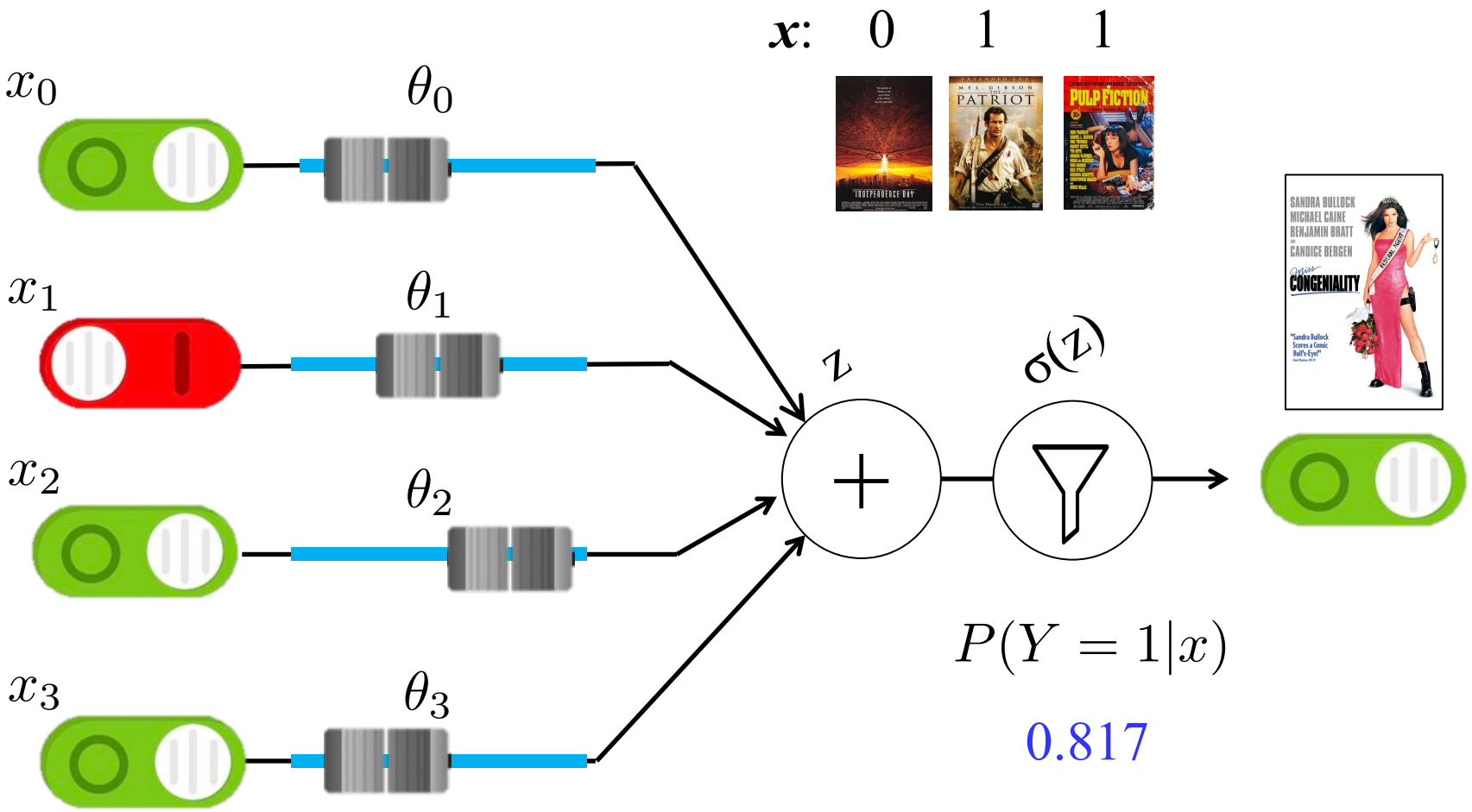
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



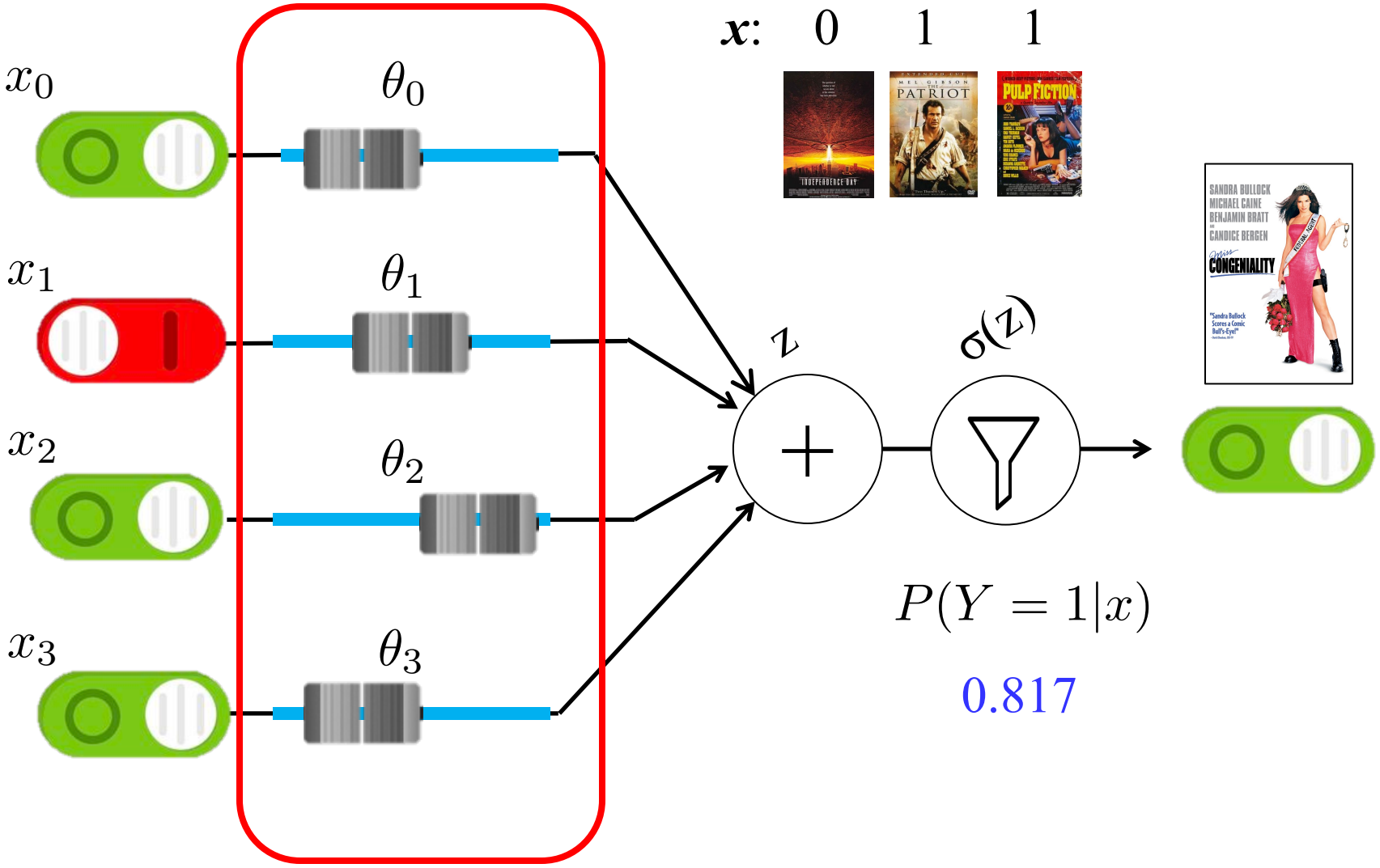
$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Logistic Regression



$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

Math for Logistic Regression

1 Make logistic regression assumption

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$
$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

Often call this $\hat{y}$

2 Calculate the log likelihood for all data

$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

3 Get derivative of log likelihood with respect to thetas

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

Logistic Regression Training

Initialize: $\theta_j = 0$ for all $0 \leq j \leq m$

Repeat many times:

gradient[j] = 0 for all $0 \leq j \leq m$

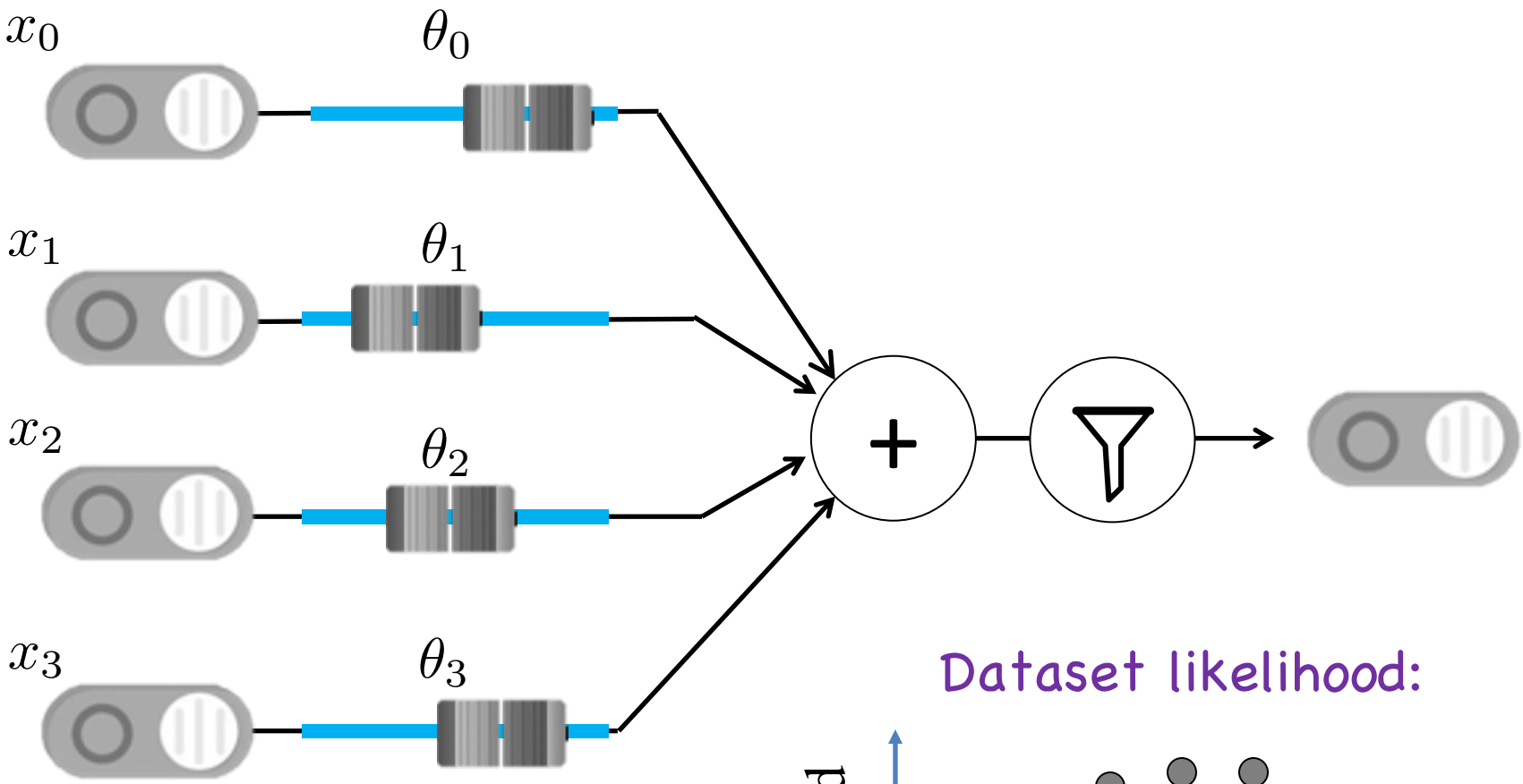
For each training example (x, y):

For each parameter j:

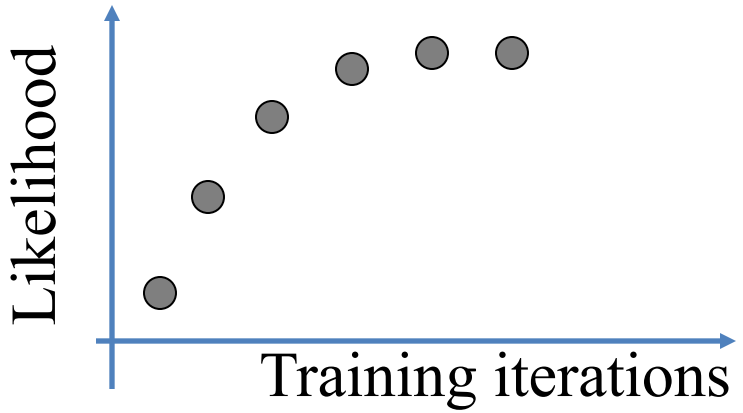
gradient[j] $+= x_j \left(y - \frac{1}{1 + e^{-\theta^T \mathbf{x}}} \right)$

θ_j += η * gradient[j] for all $0 \leq j \leq m$

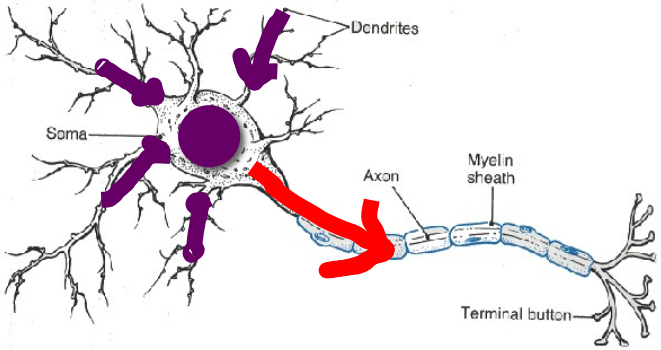
Training

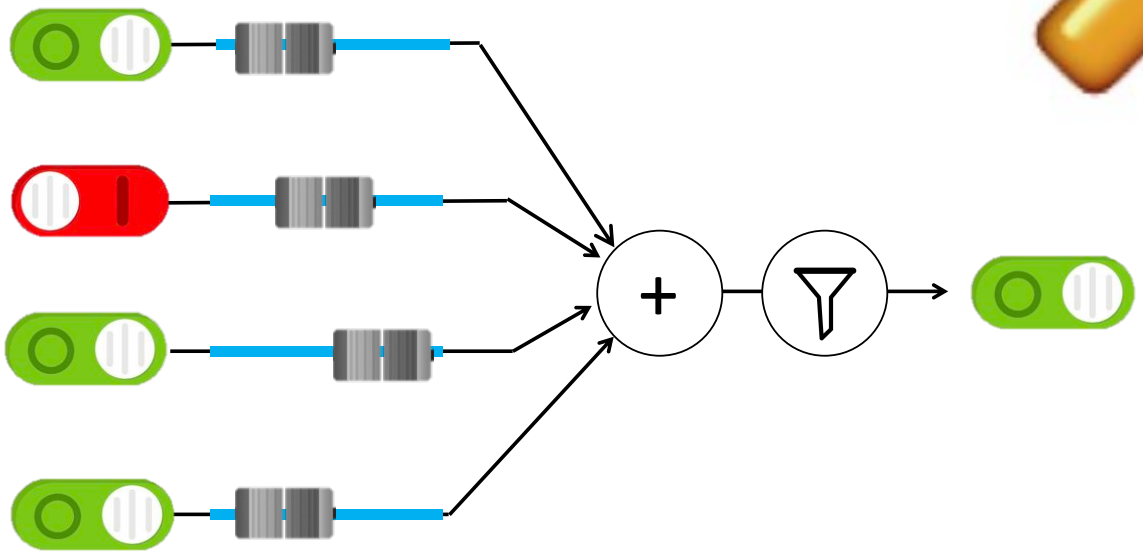


Dataset likelihood:



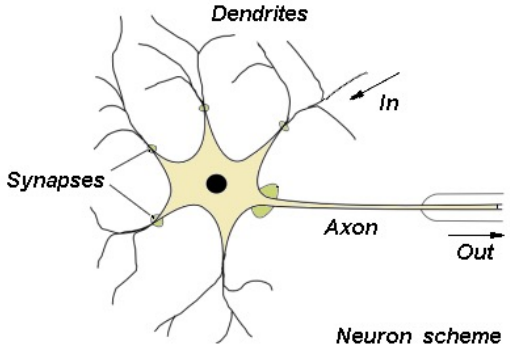
Artificial Neurons

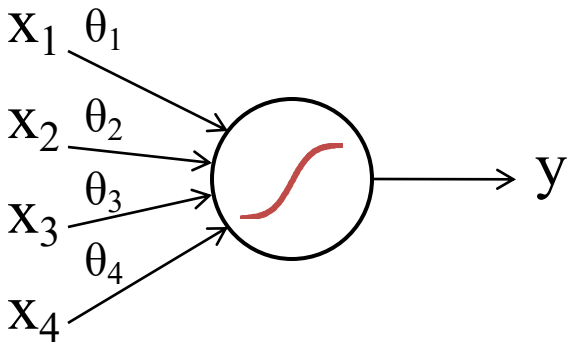




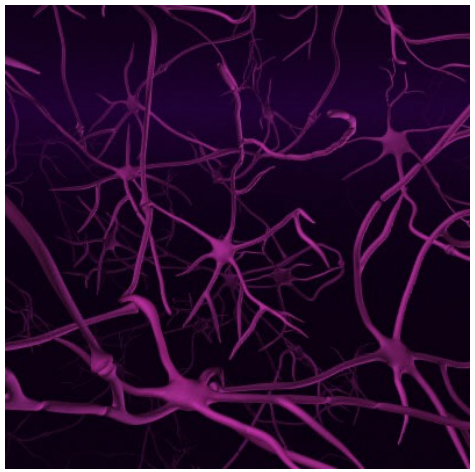
Biological Basis for Neural Networks

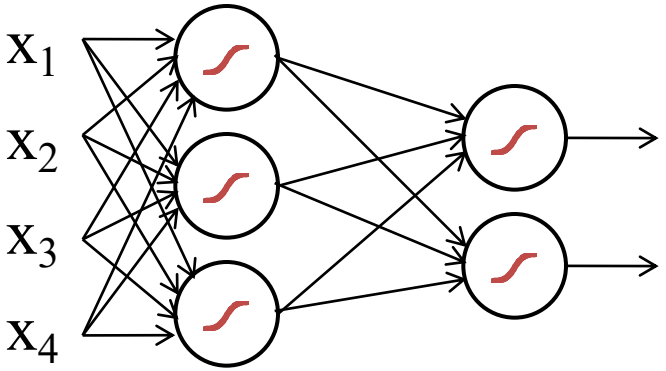
A neuron





Your brain





Actually, it's probably someone else's brain

Speeding up gradient descent

minimizes loss (a function of prediction error)

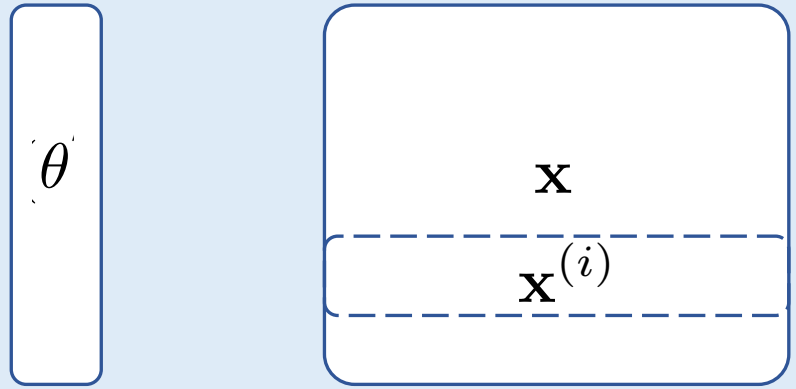
```
initialize θ_j = 0 for 0 ≤ j ≤ m
repeat many times:
    gradient[j] = 0 for 0 ≤ j ≤ m
    for each training example (x,y):
        for each 0 ≤ j ≤ m:
            compute gradient
    θ_j -= η  * gradient[j] for all 0 ≤ j ≤ m
```

1. What if we have 1,200,000 datapoints in our training set?
2. How can we speed up the update?

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

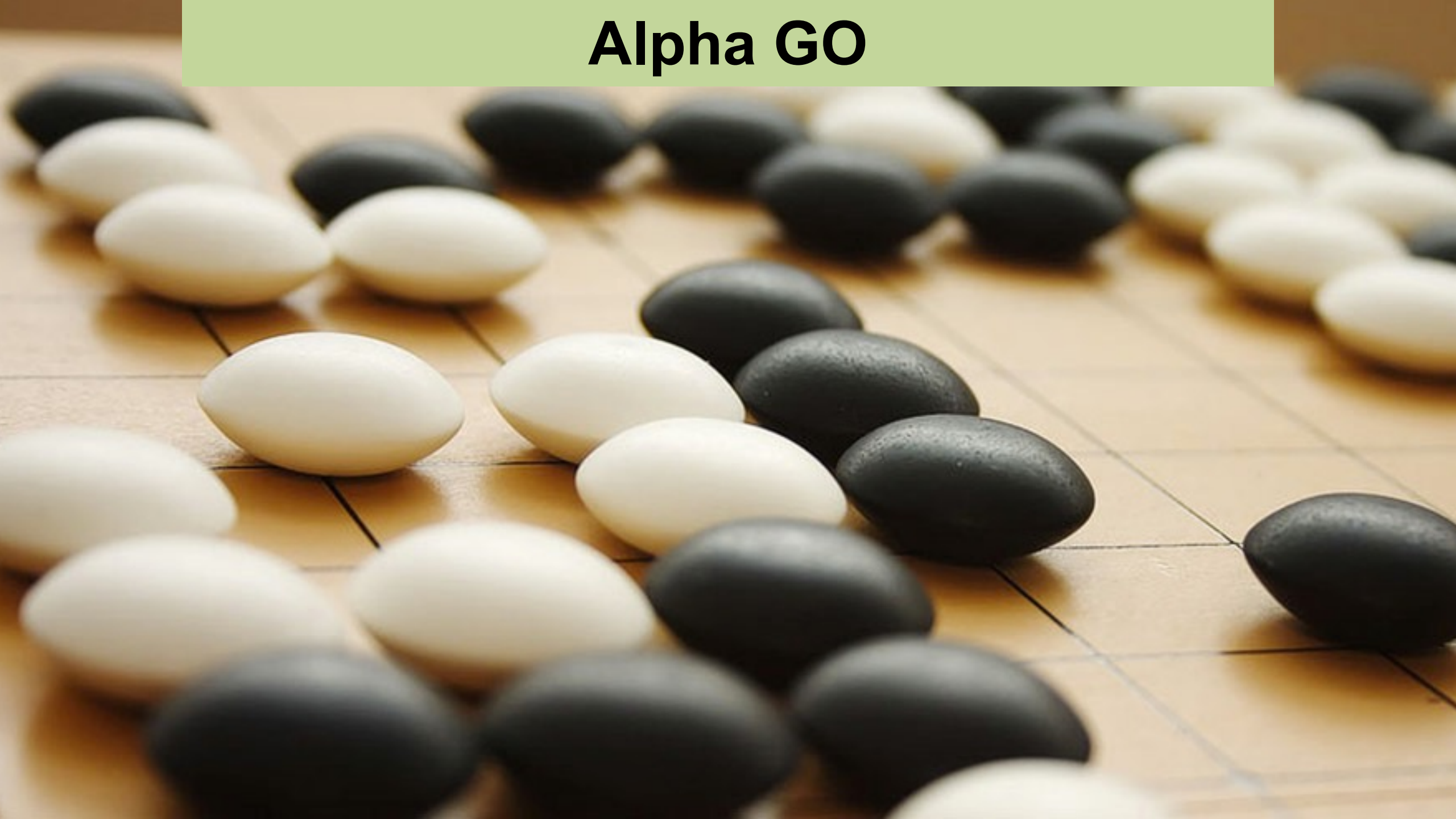
Start with this:

$$\theta^T \mathbf{x}^{(i)}$$



Core idea behind the revolution in AI

Alpha GO



Self Driving Cars

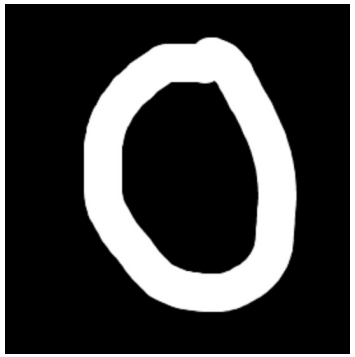
Computers Making Art

(aka Neural Networks)

Deep learning is (at its core) many logistic regression pieces stacked on top of each other.

Digit Recognition Example

Let's make feature vectors from pictures of numbers

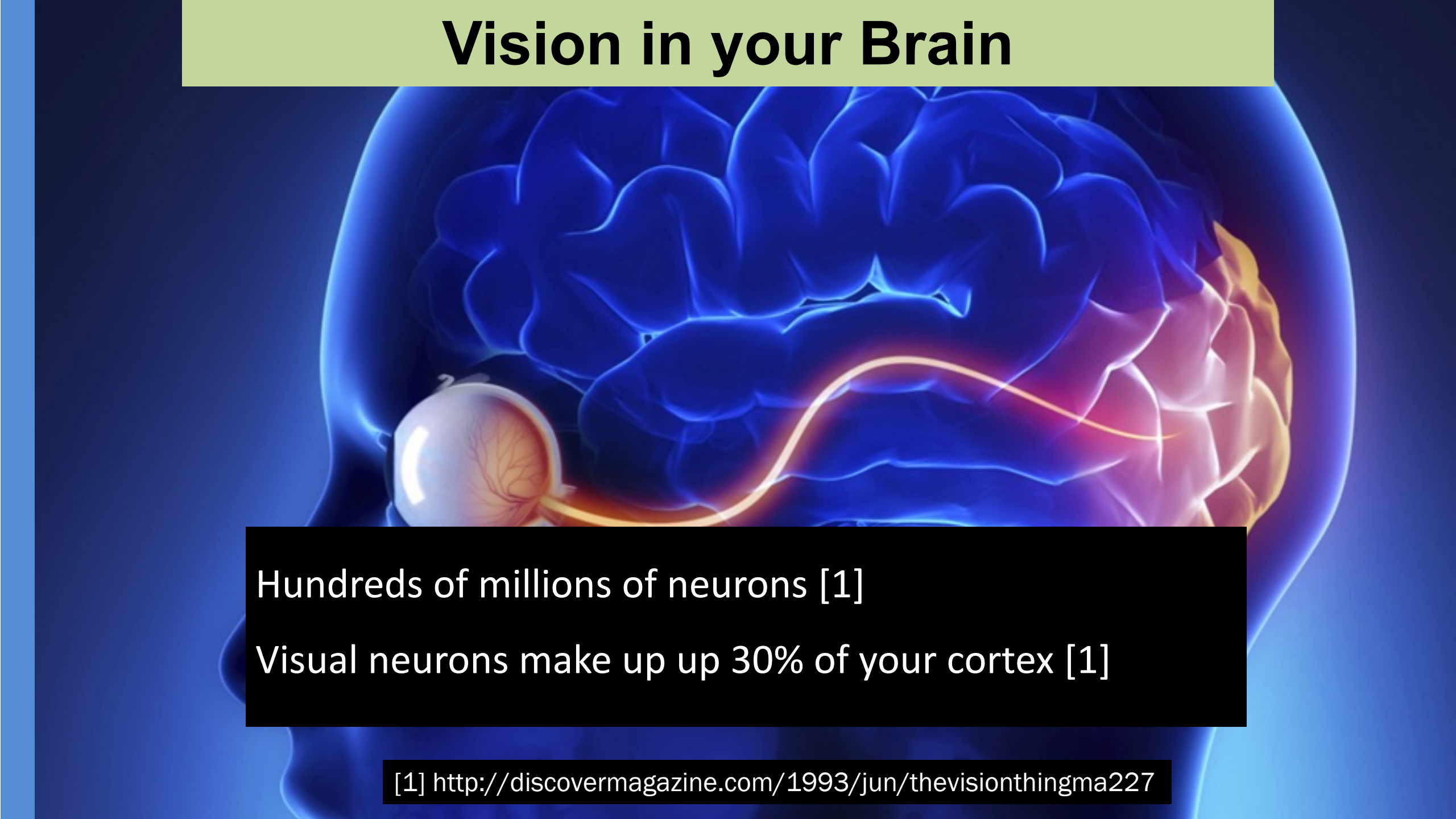


$$\mathbf{x}^{(i)} = [0, 0, 0, 0, \ldots, 1, 0, 0, 1, \ldots 0, 0, 1, 0]$$
$$y^{(i)} = 0$$

$$\mathbf{x}^{(i)} = [0, 0, 1, 1, \ldots, 0, 1, 1, 0, \ldots 0, 1, 0, 0]$$
$$y^{(i)} = 1$$

Computer Vision

Vision in your Brain

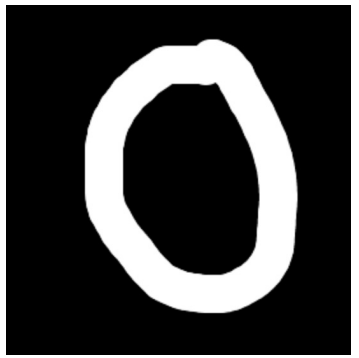


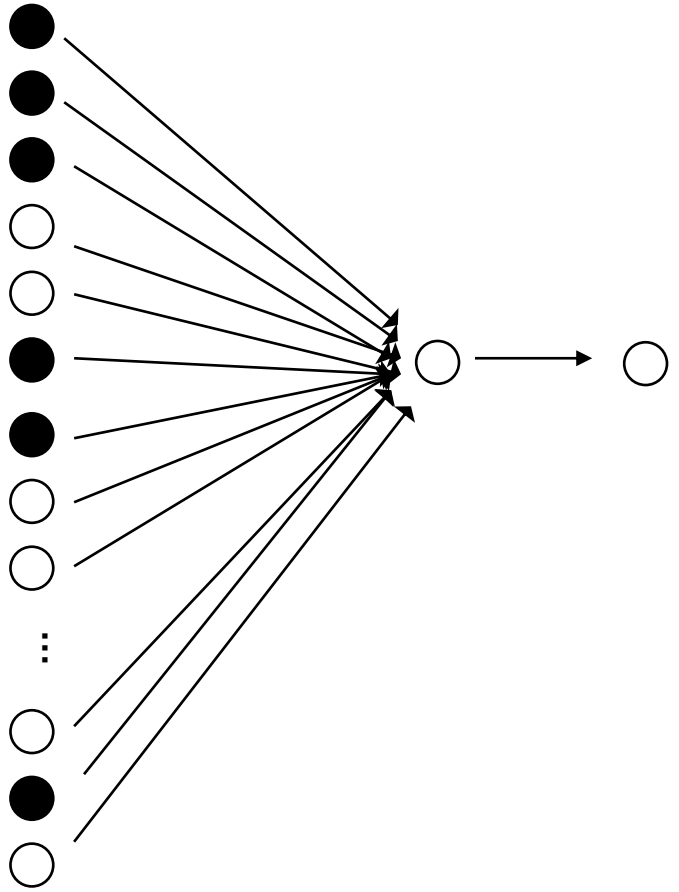
Hundreds of millions of neurons [1]

Visual neurons make up up 30% of your cortex [1]

[1] http://discovermagazine.com/1993/jun/thevisionthingma227

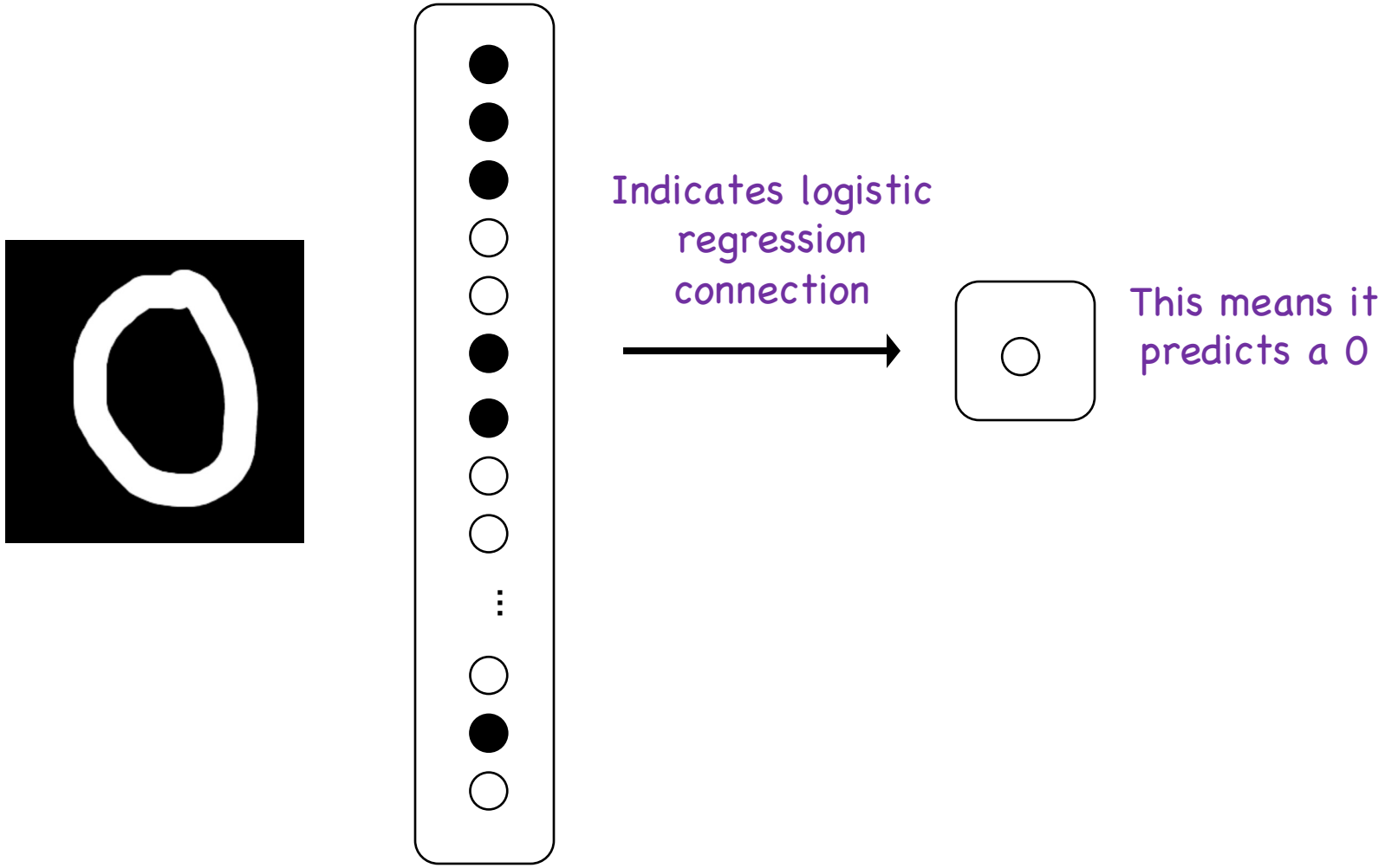
Logistic Regression



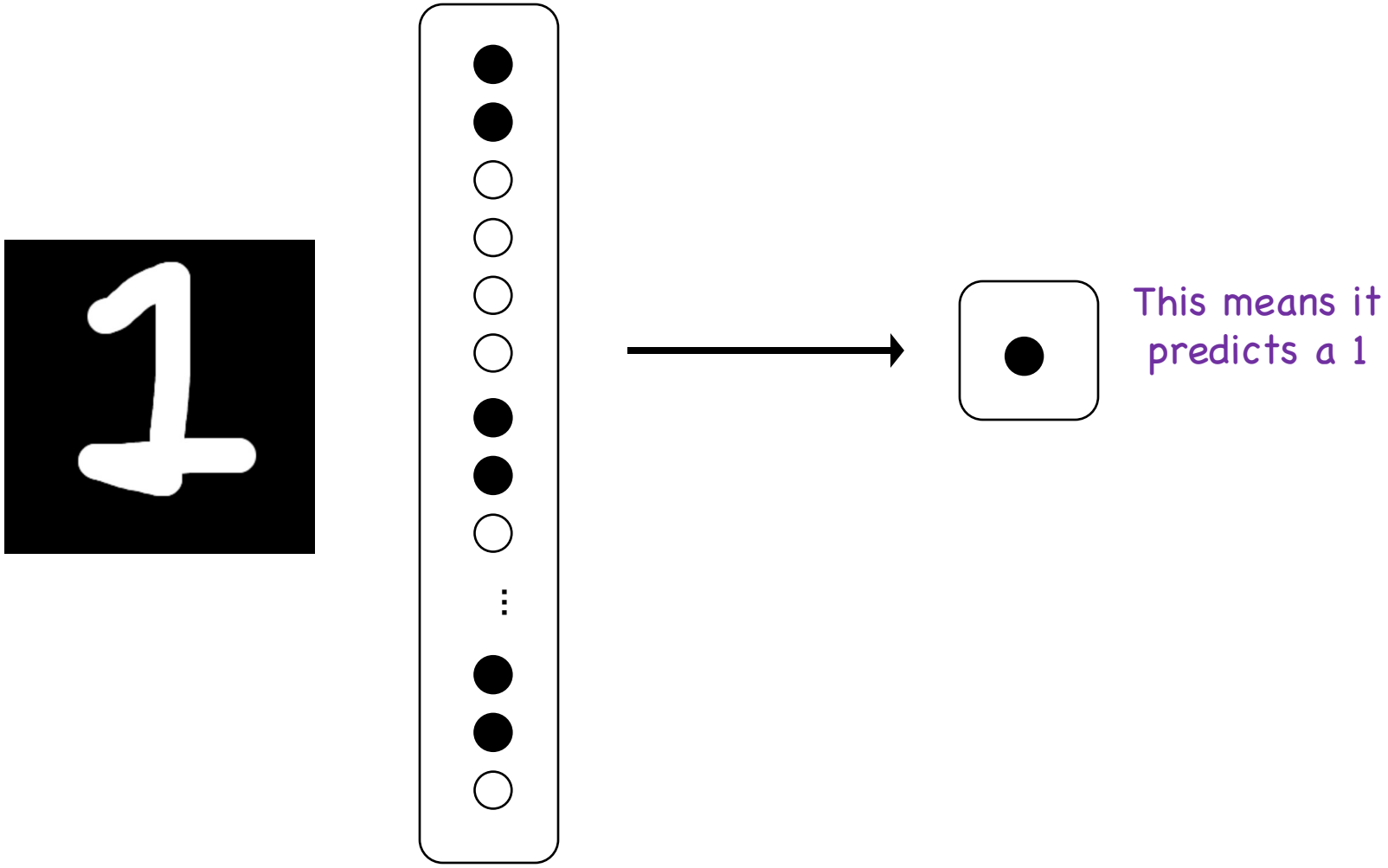


This means it predicts a 0

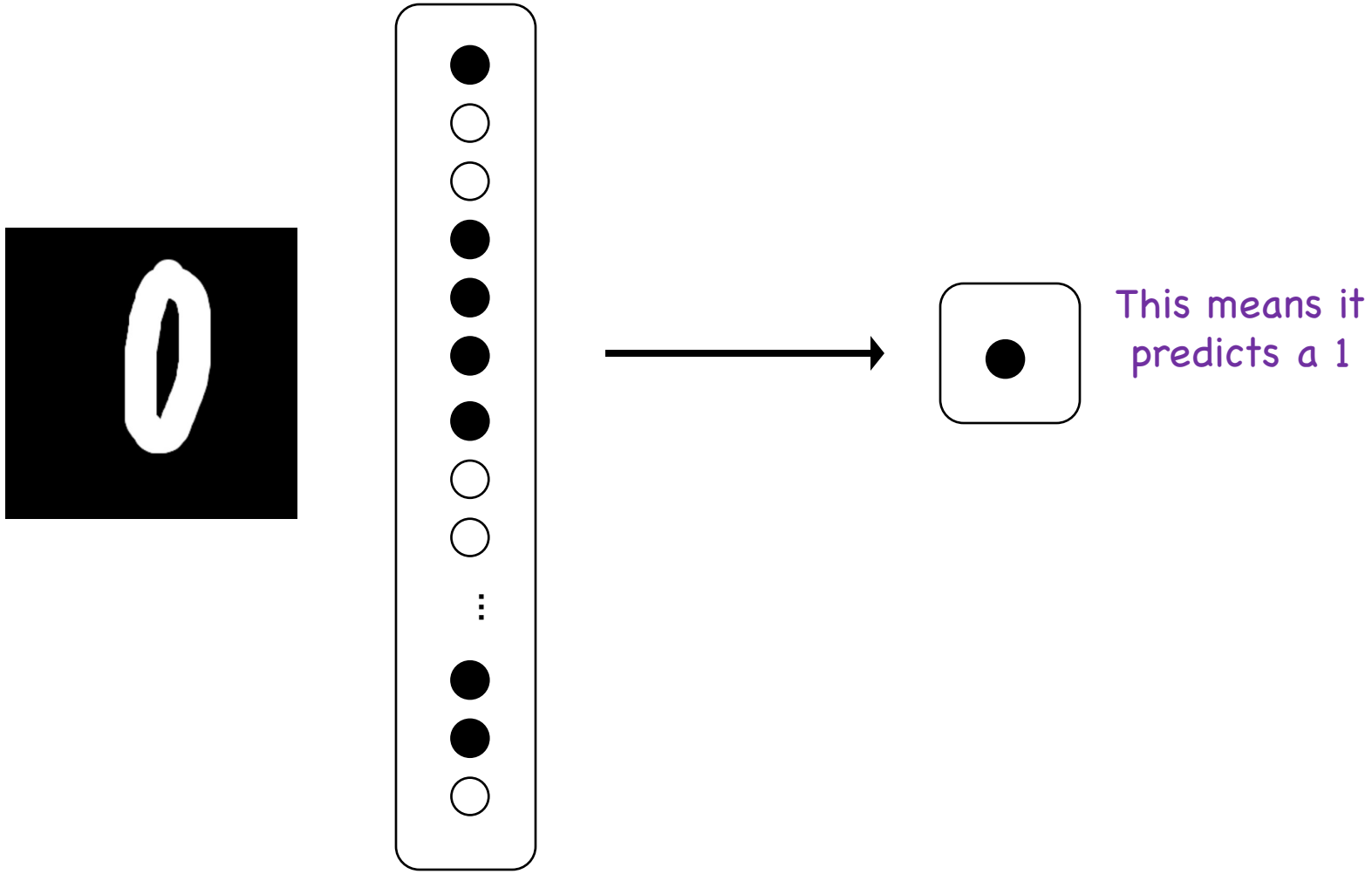
Logistic Regression



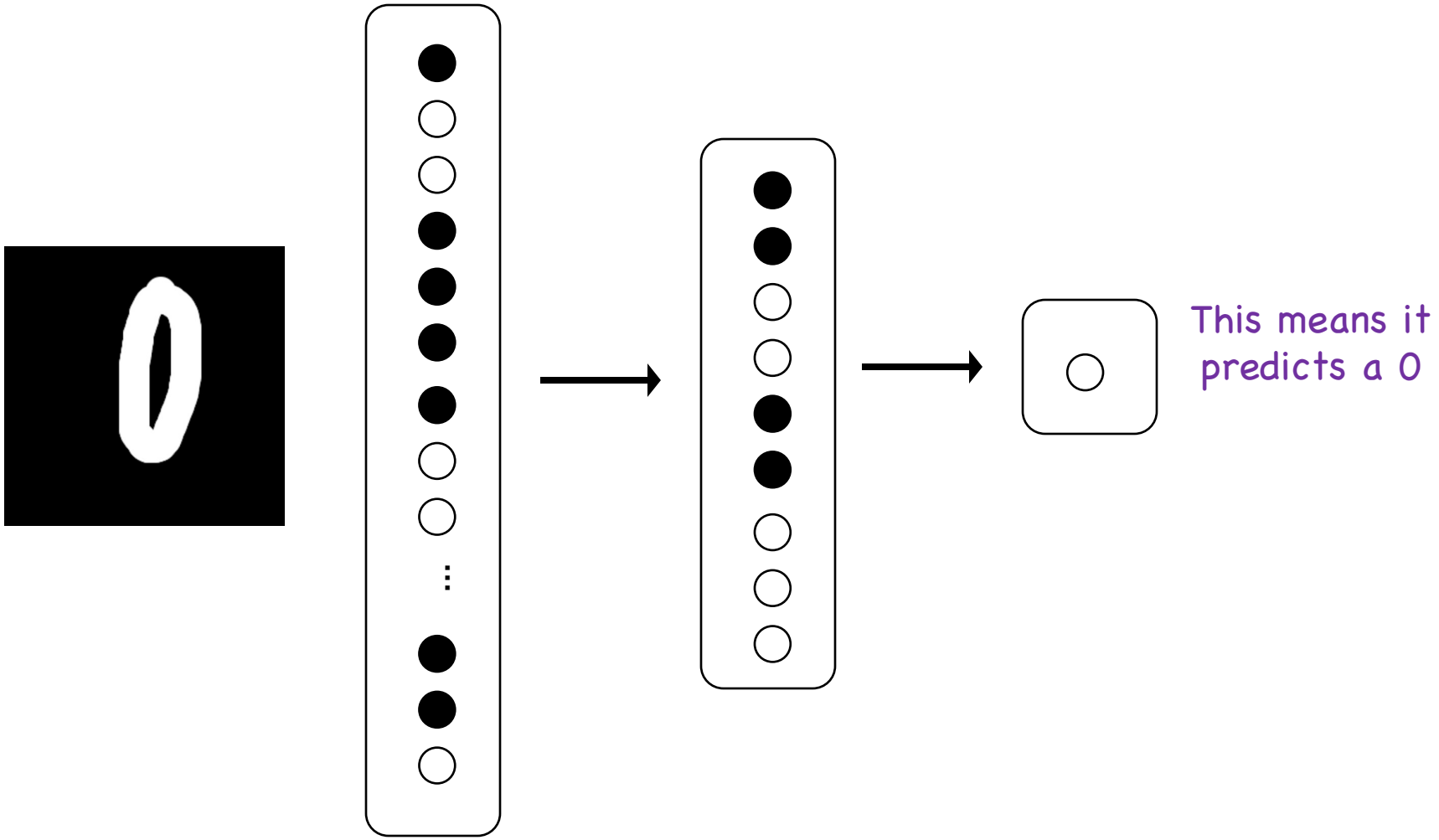
Logistic Regression



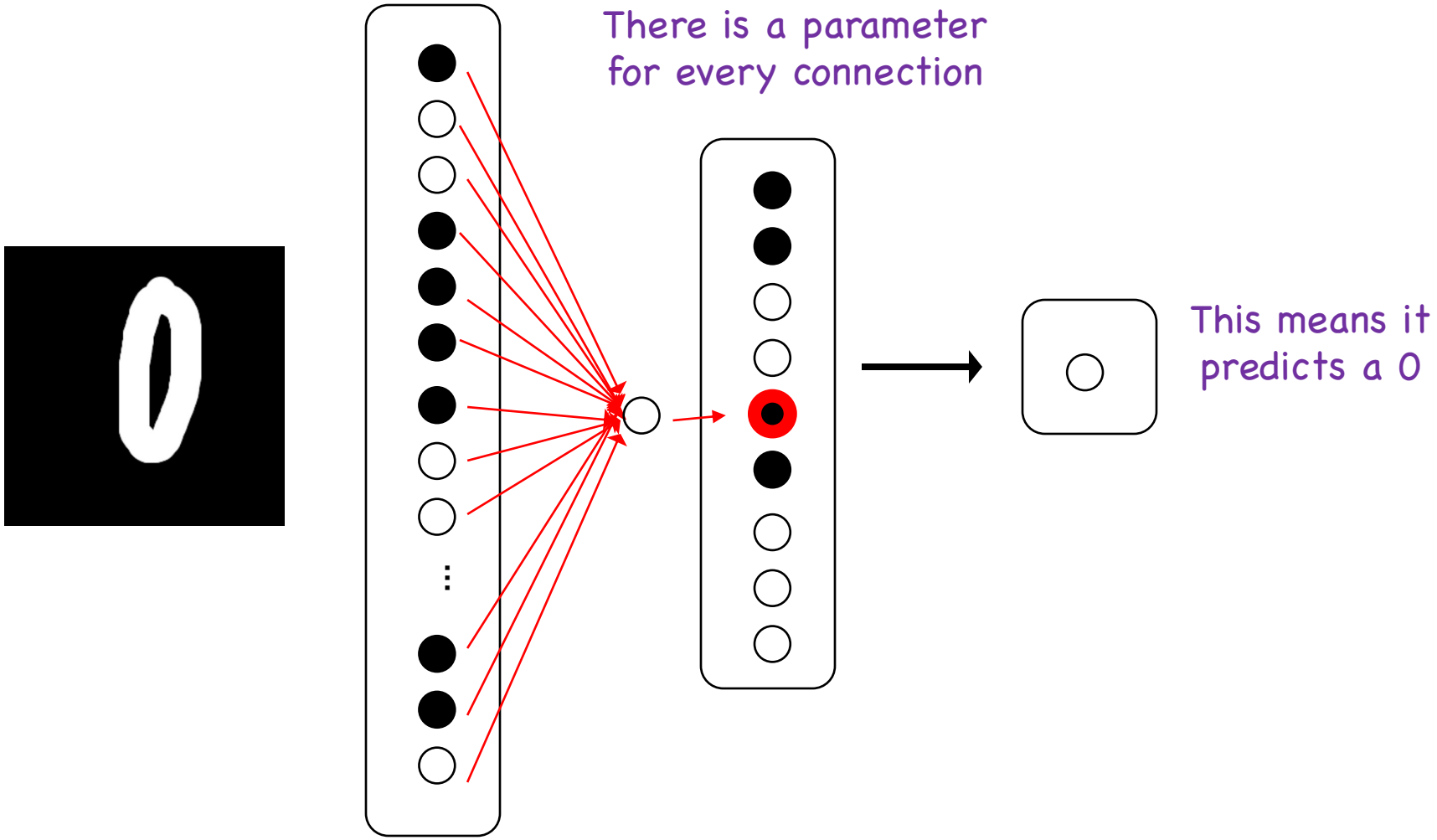
Not So Good



We Can Put Neurons Together

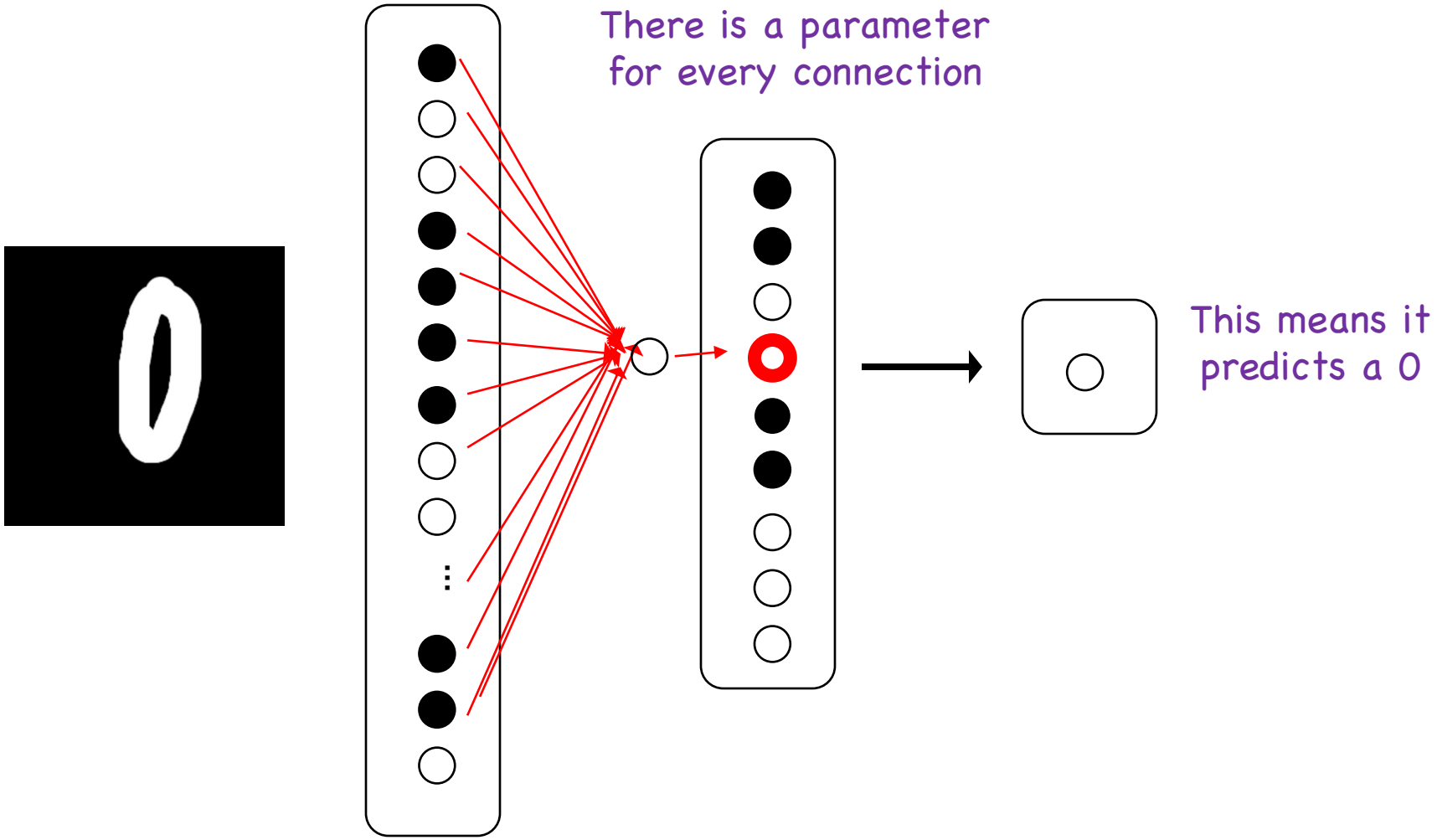


We Can Put Neurons Together



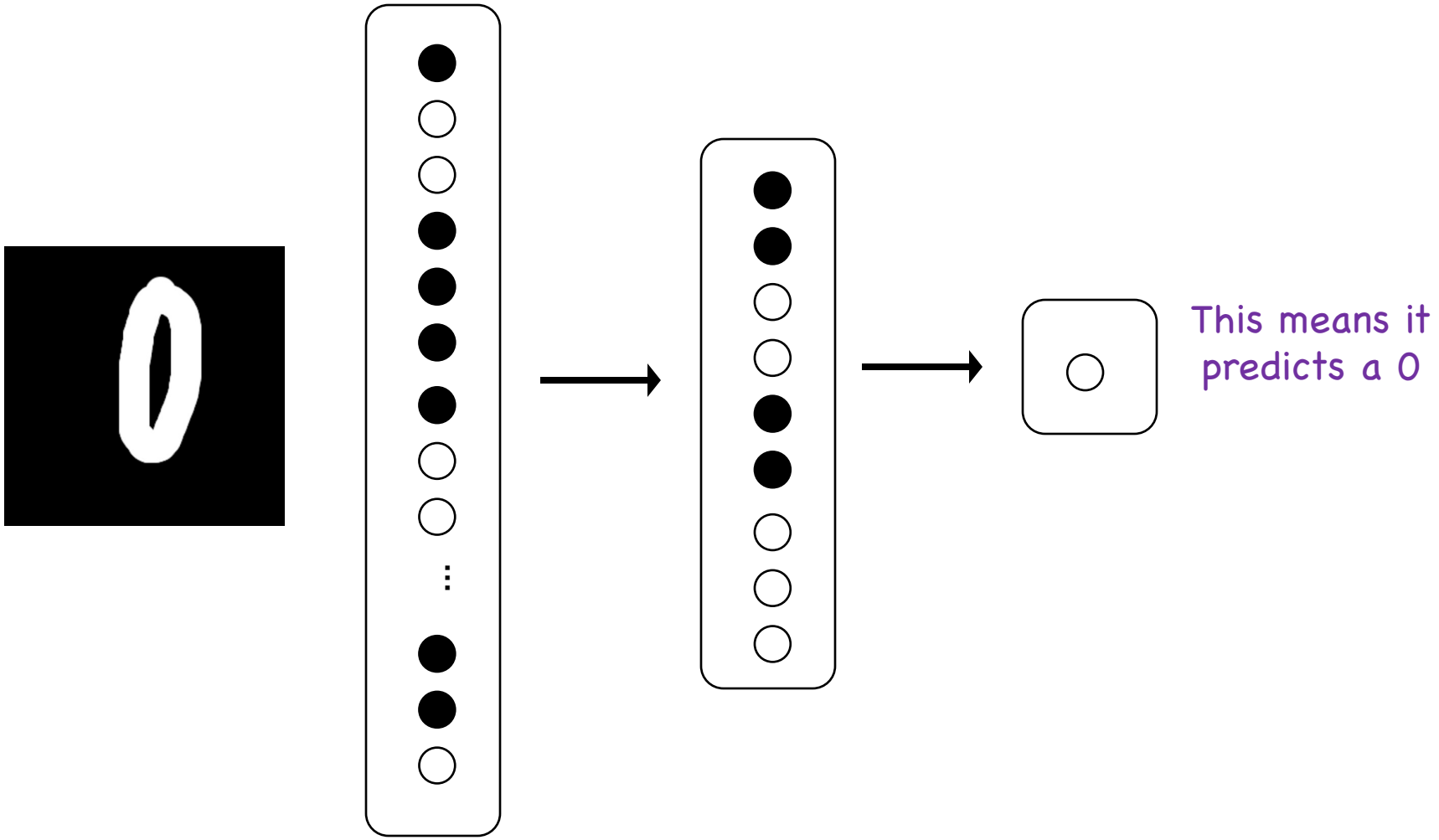
Look at a single "hidden" neuron

We Can Put Neurons Together

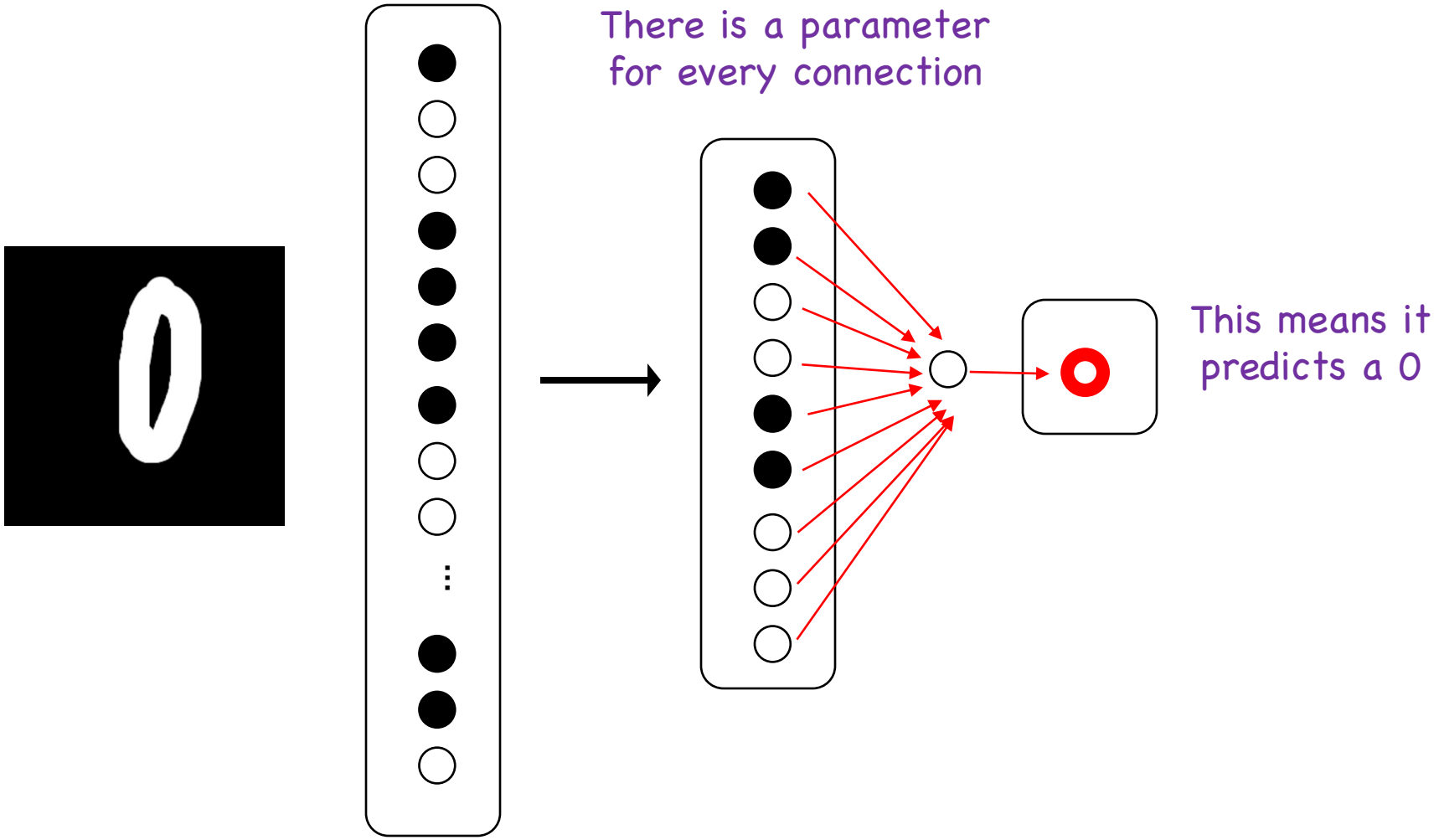


Look at another "hidden" neuron

We Can Put Neurons Together

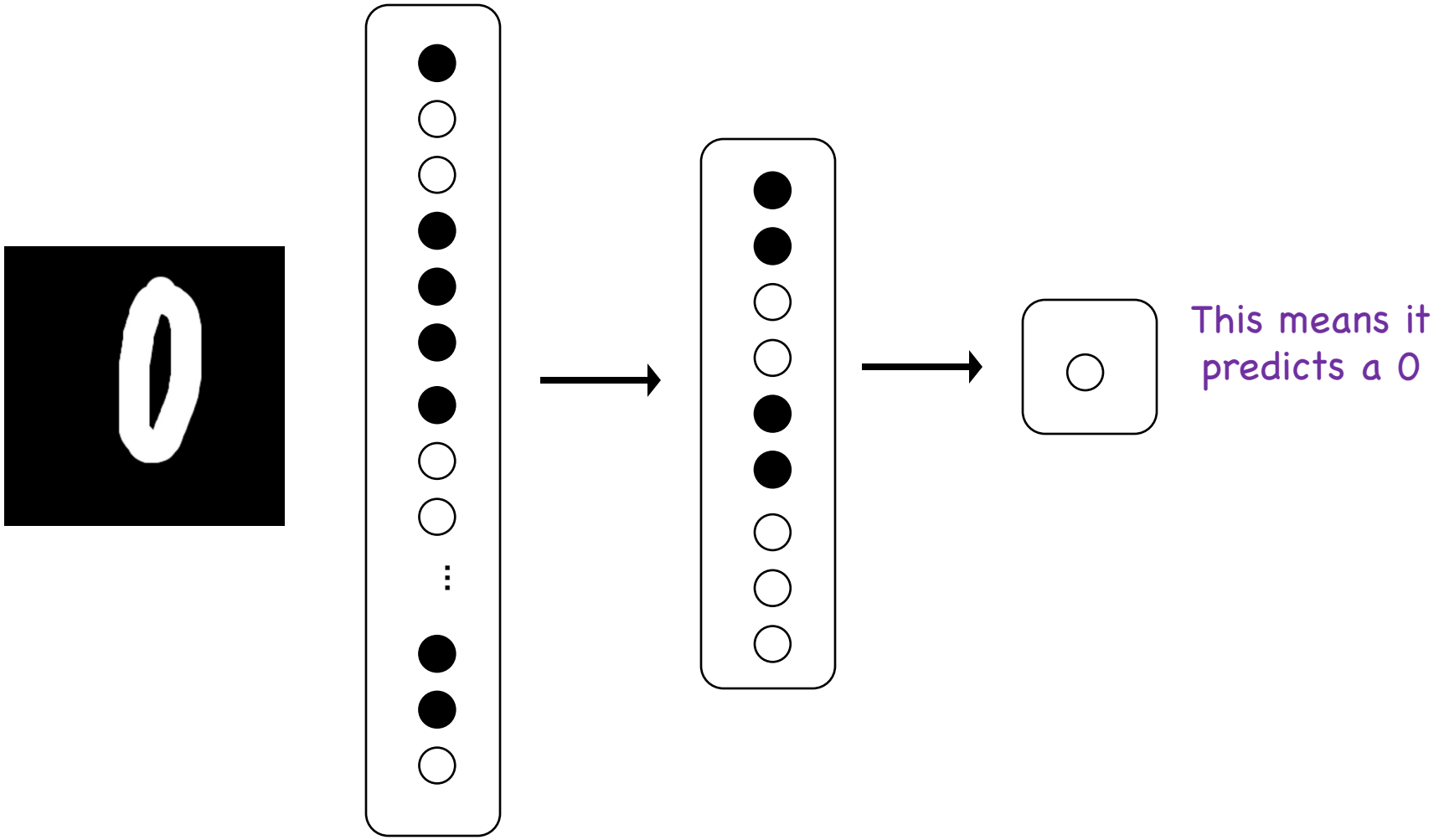


We Can Put Neurons Together

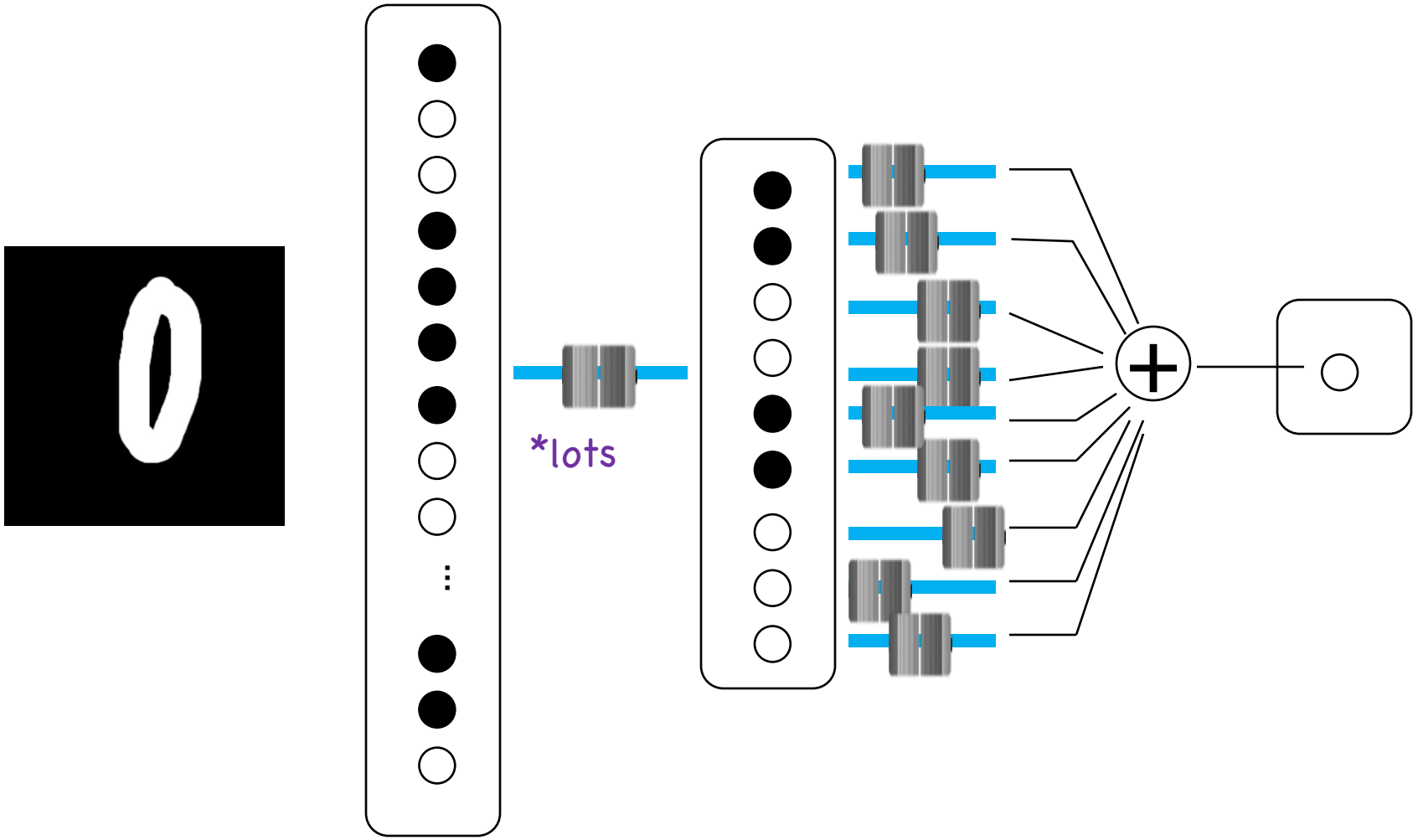


Look at another neuron

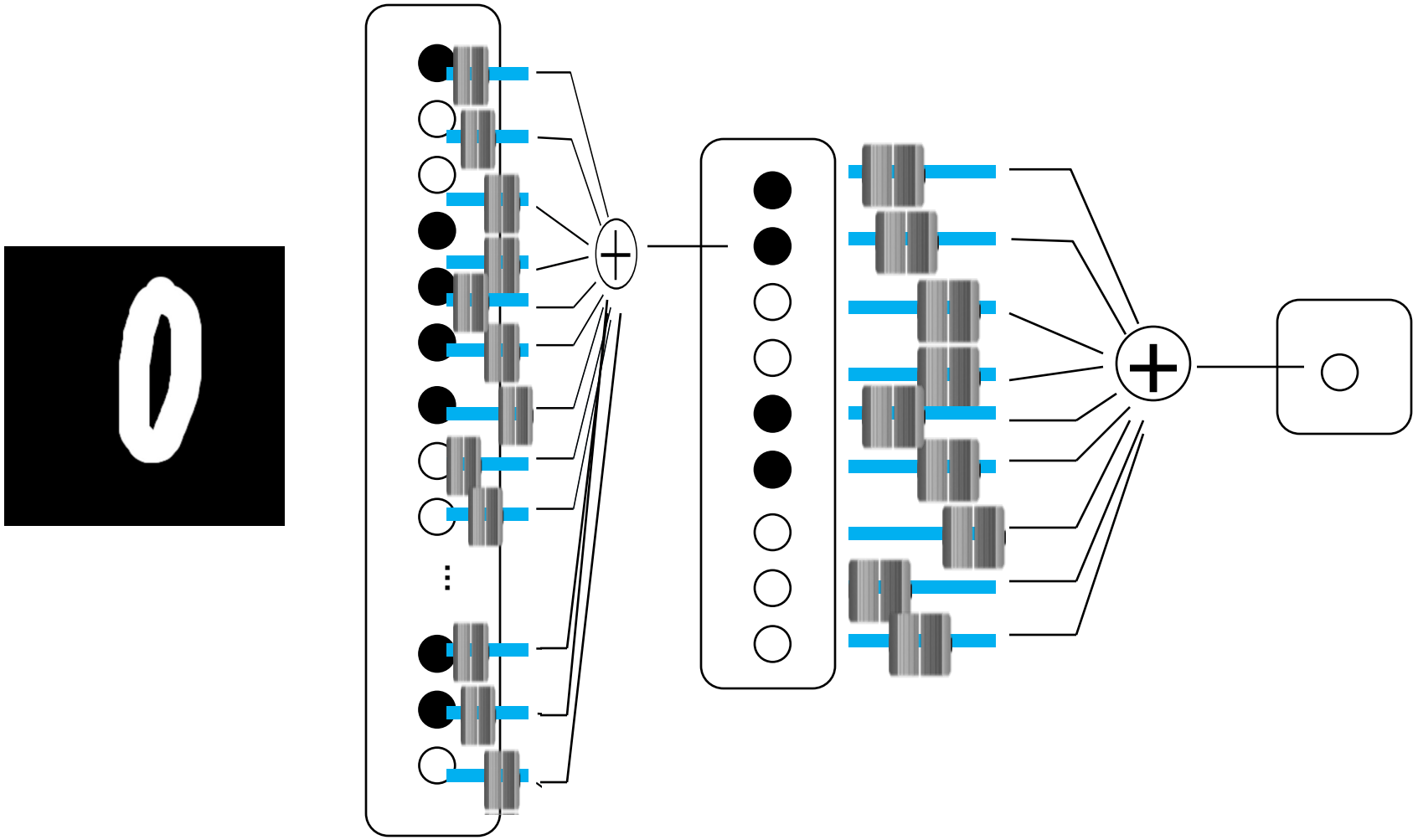
We Can Put Neurons Together



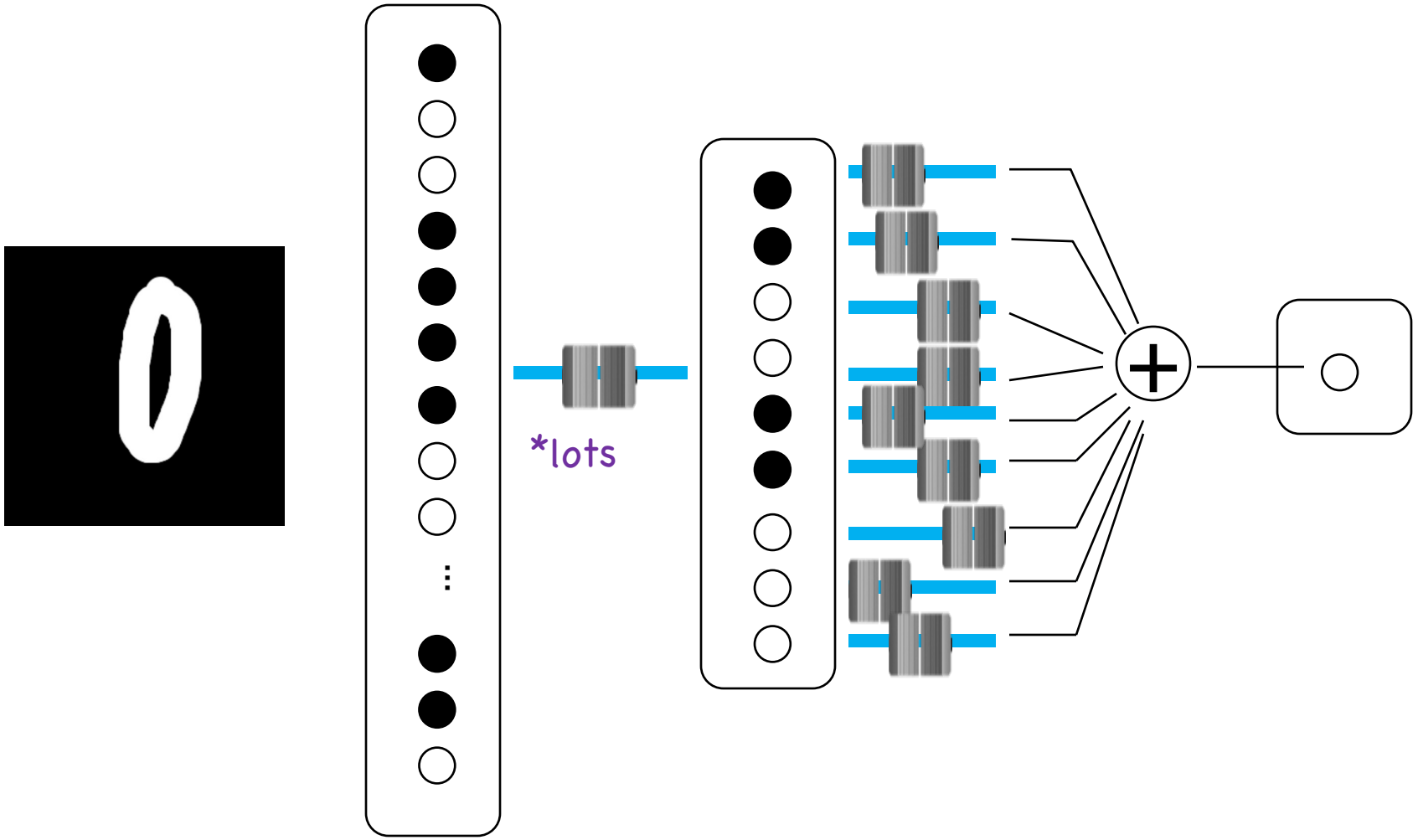
We Can Put Neurons Together



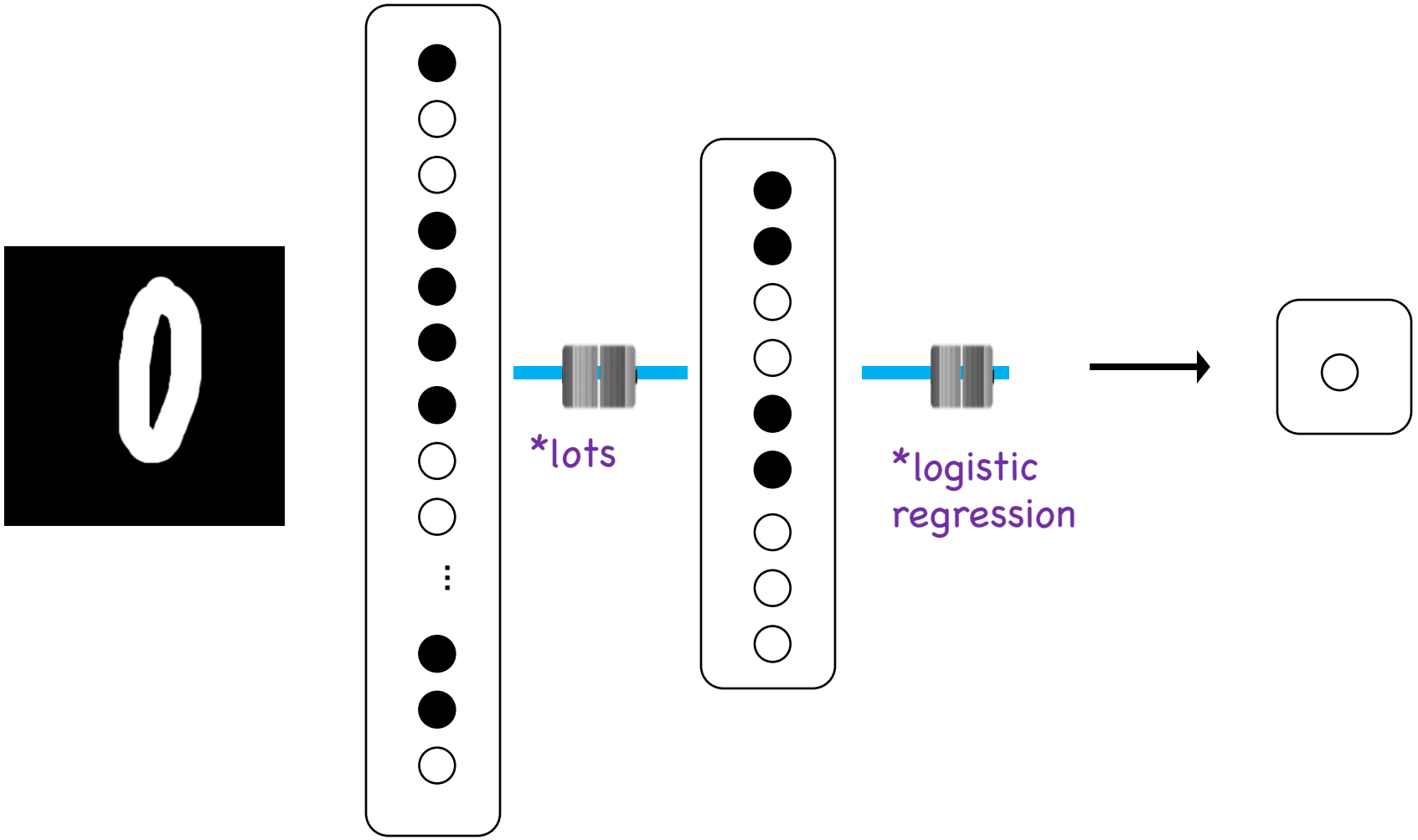
We Can Put Neurons Together



We Can Put Neurons Together



We Can Put Neurons Together



Deep learning

def Deep learning is
maximum likelihood estimation with neural networks.

def A neural network is
(at its core) many logistic regression pieces stacked on top of each other.

$[1,0,\ldots,1]$
$\boldsymbol{x}$, input

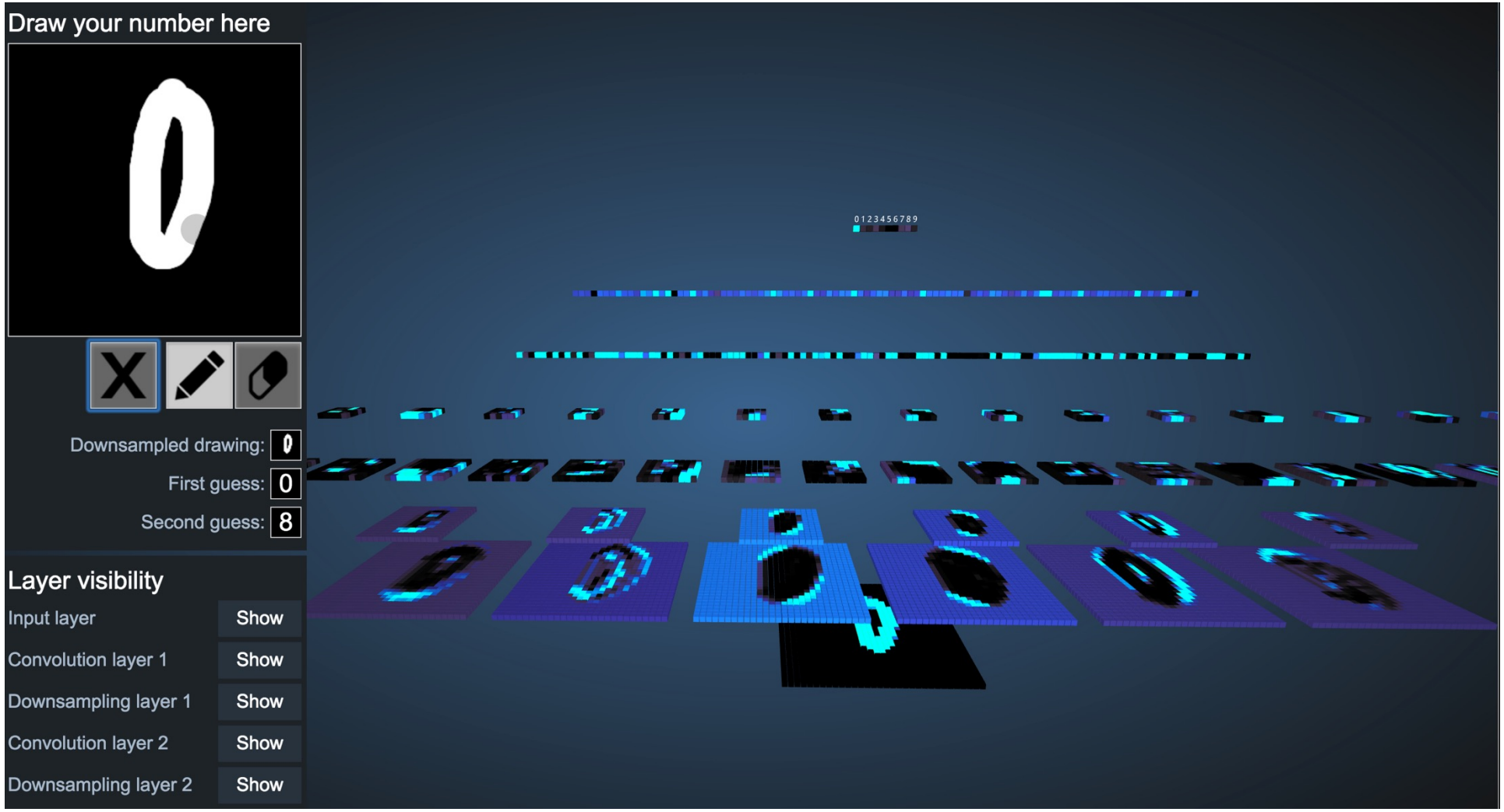
Lots of Logistic (regressions)

$\hat{y}$, output
$P(Y|\boldsymbol{X} = \boldsymbol{x})$

> 0.5?

Yes.
Predict 1

Demonstration



http://scs.ryerson.ca/~aharley/vis/conv/

Deep learning gets its ***intelligence*** from its thetas (aka its parameters)

How do we train?

MLE of Thetas!

First: Learning Goals…

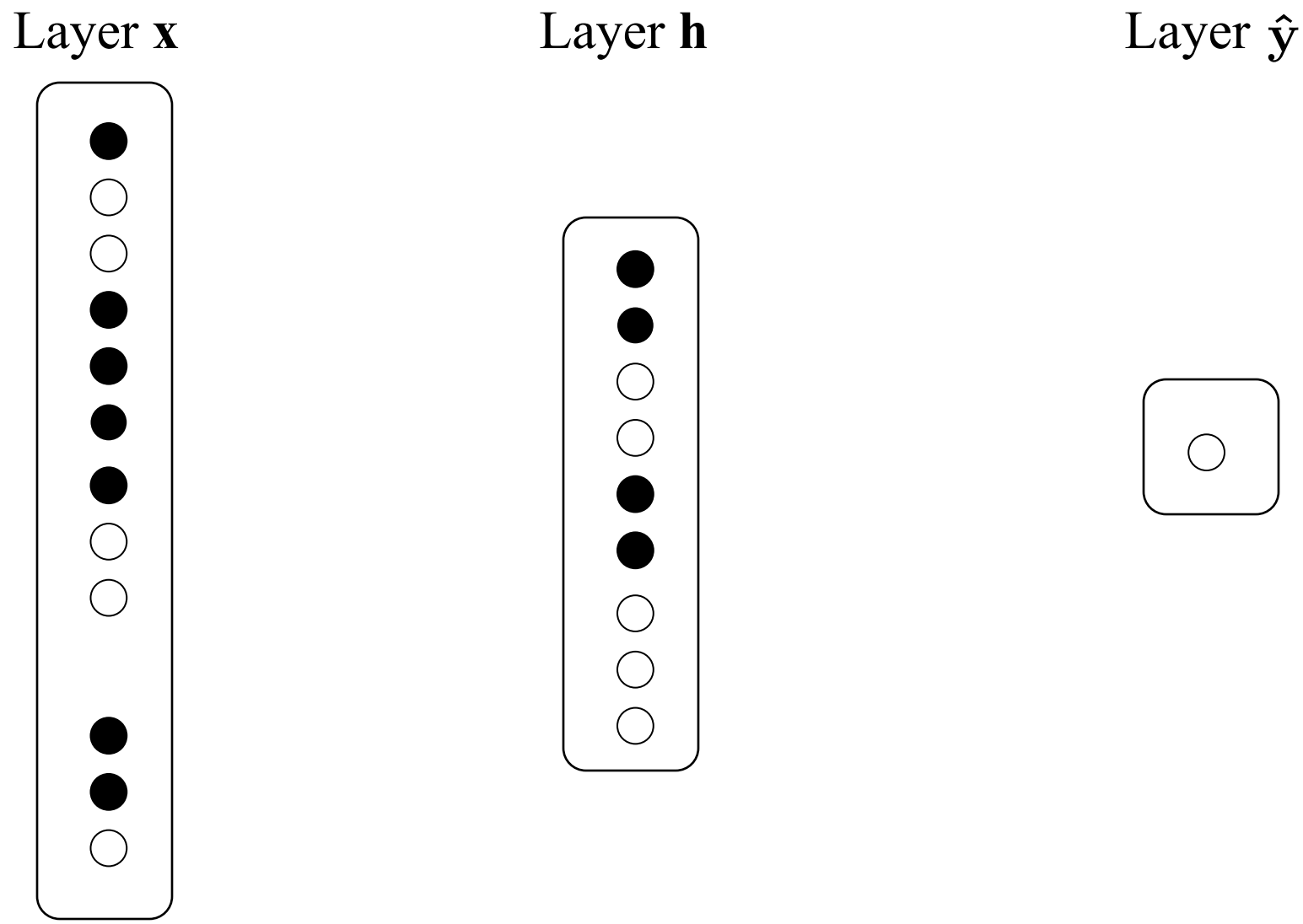
1. Understand Chain Rule as ♡ of Deep Learning

2. Demystify:
Deep Learning is MLE

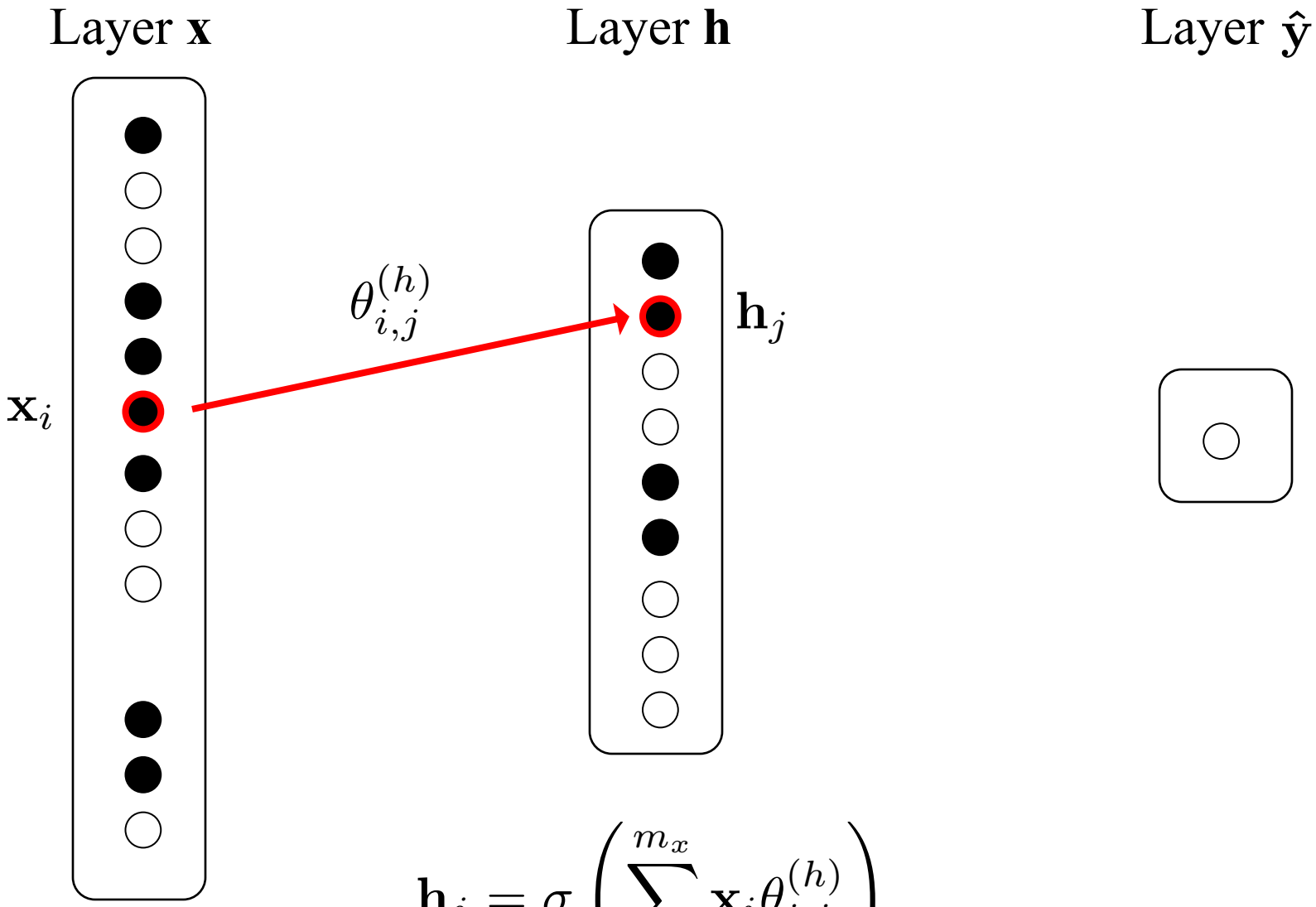
3. Become experts of logistic regression

Math worth knowing:

New Notation

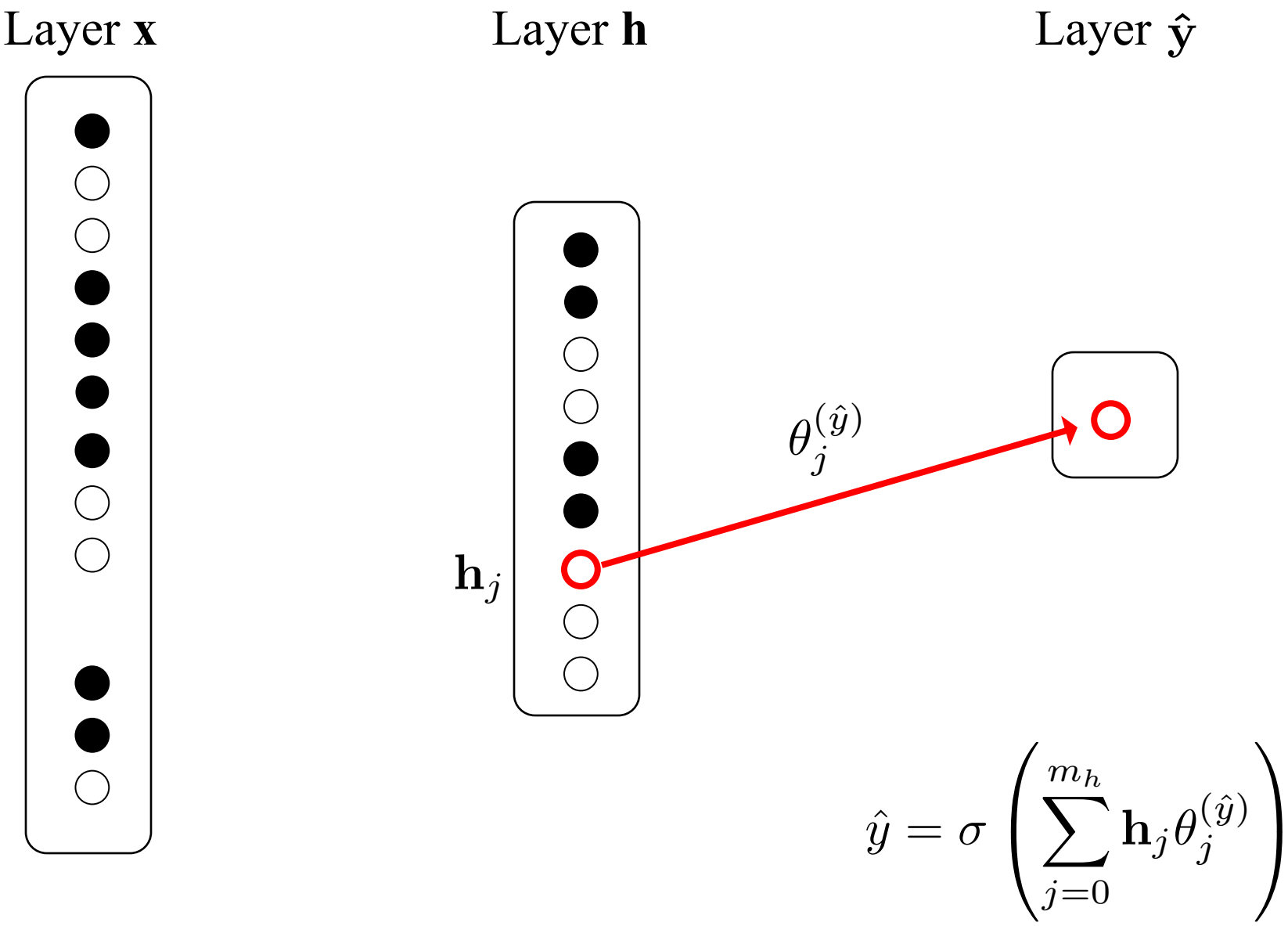


New Notation



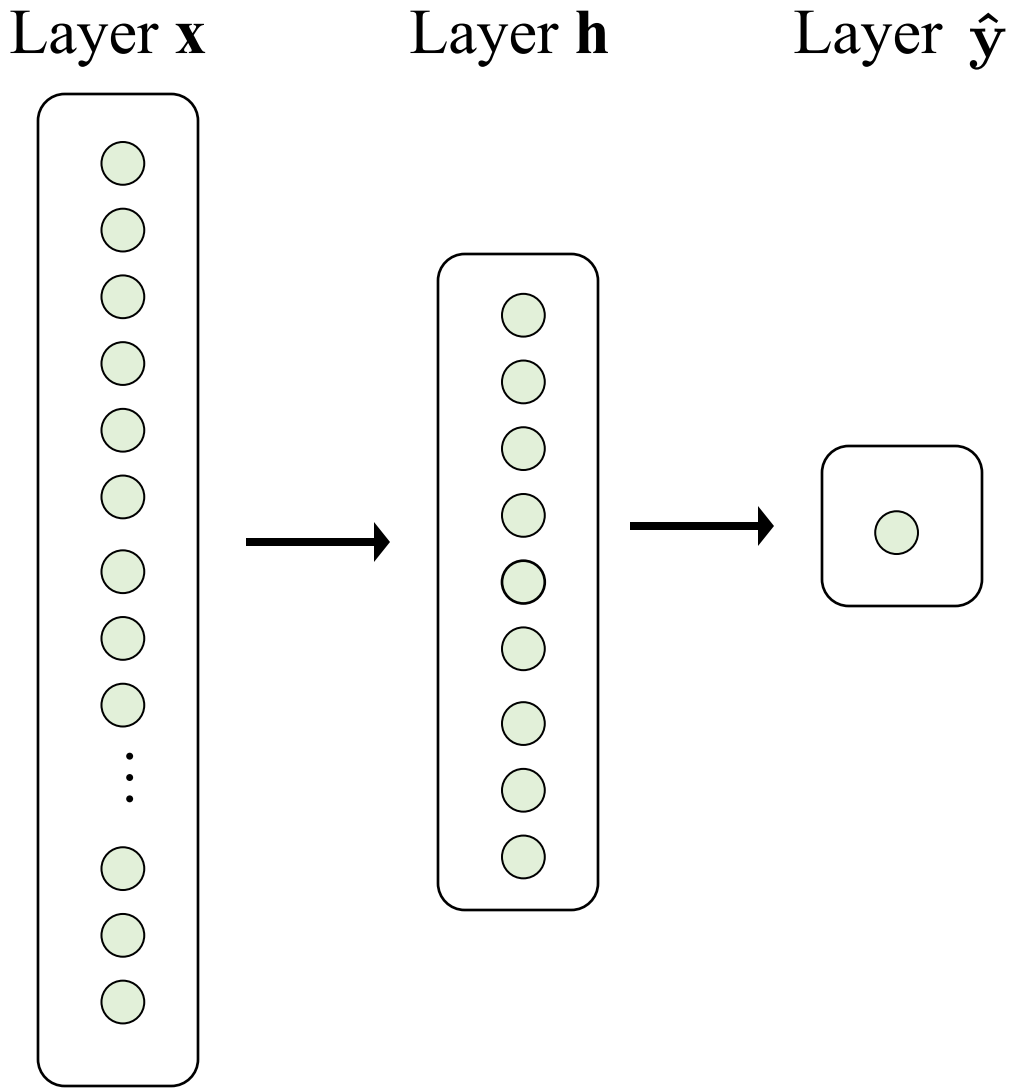
$$\mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

New Notation

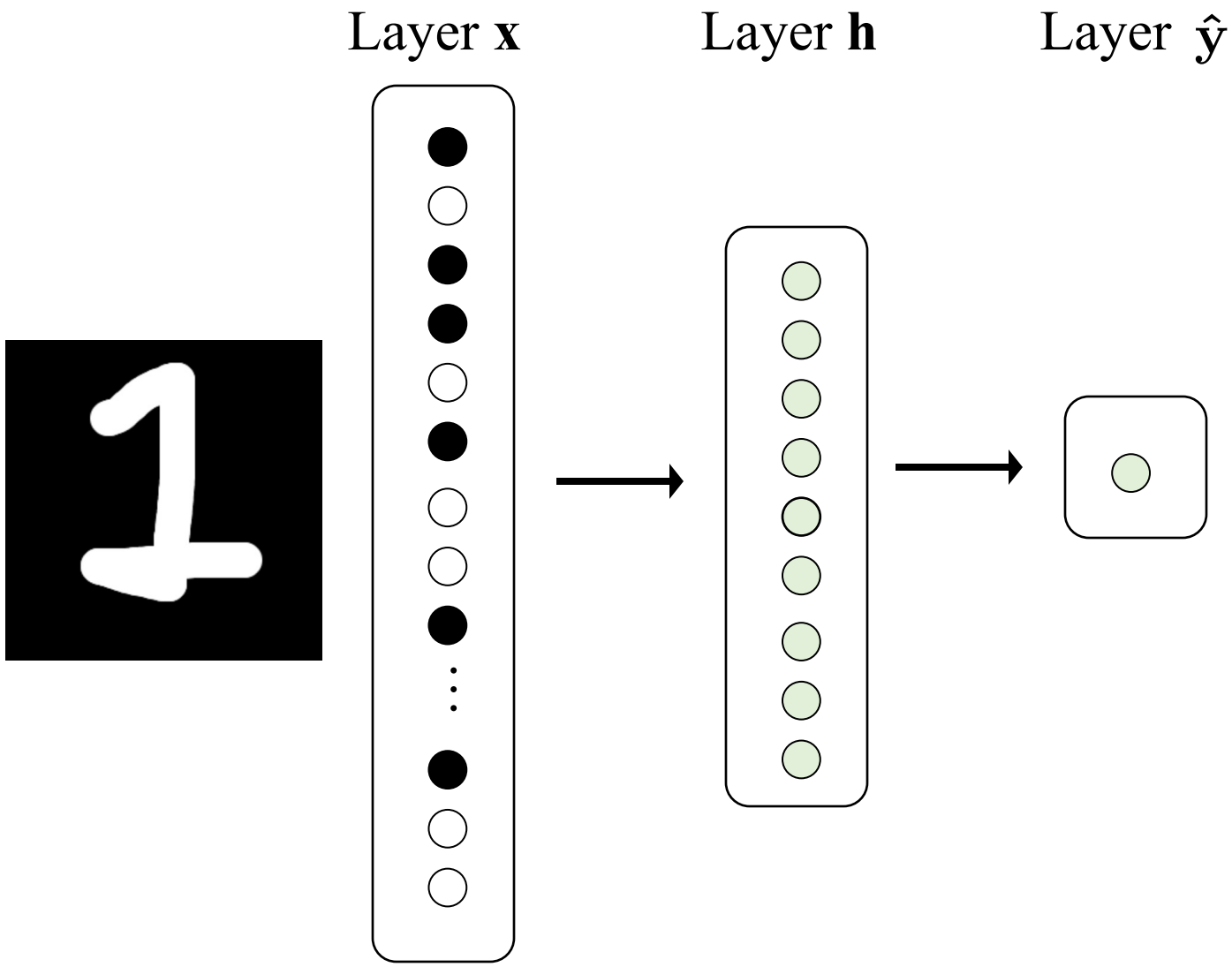


$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

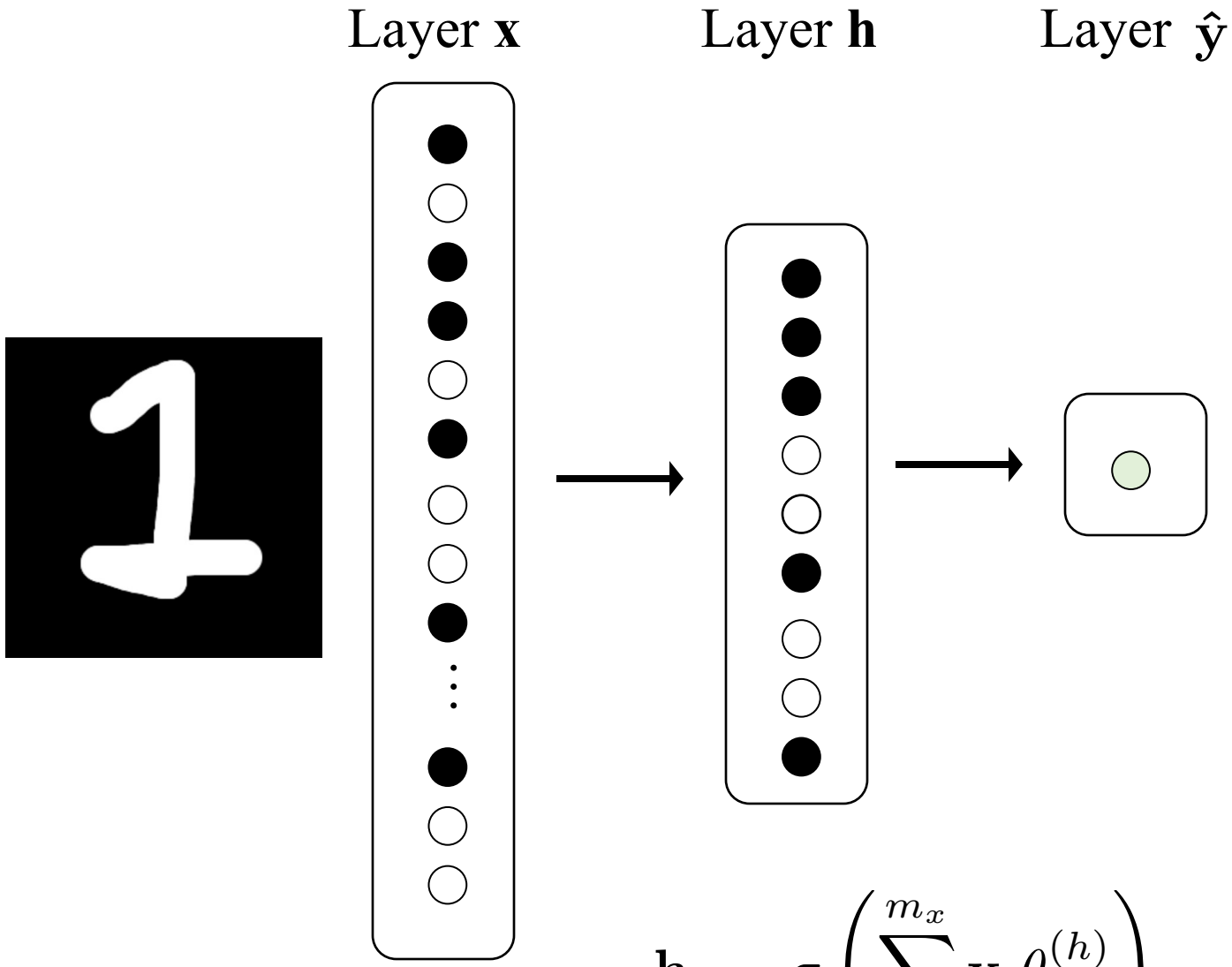
Forward Pass



Forward Pass

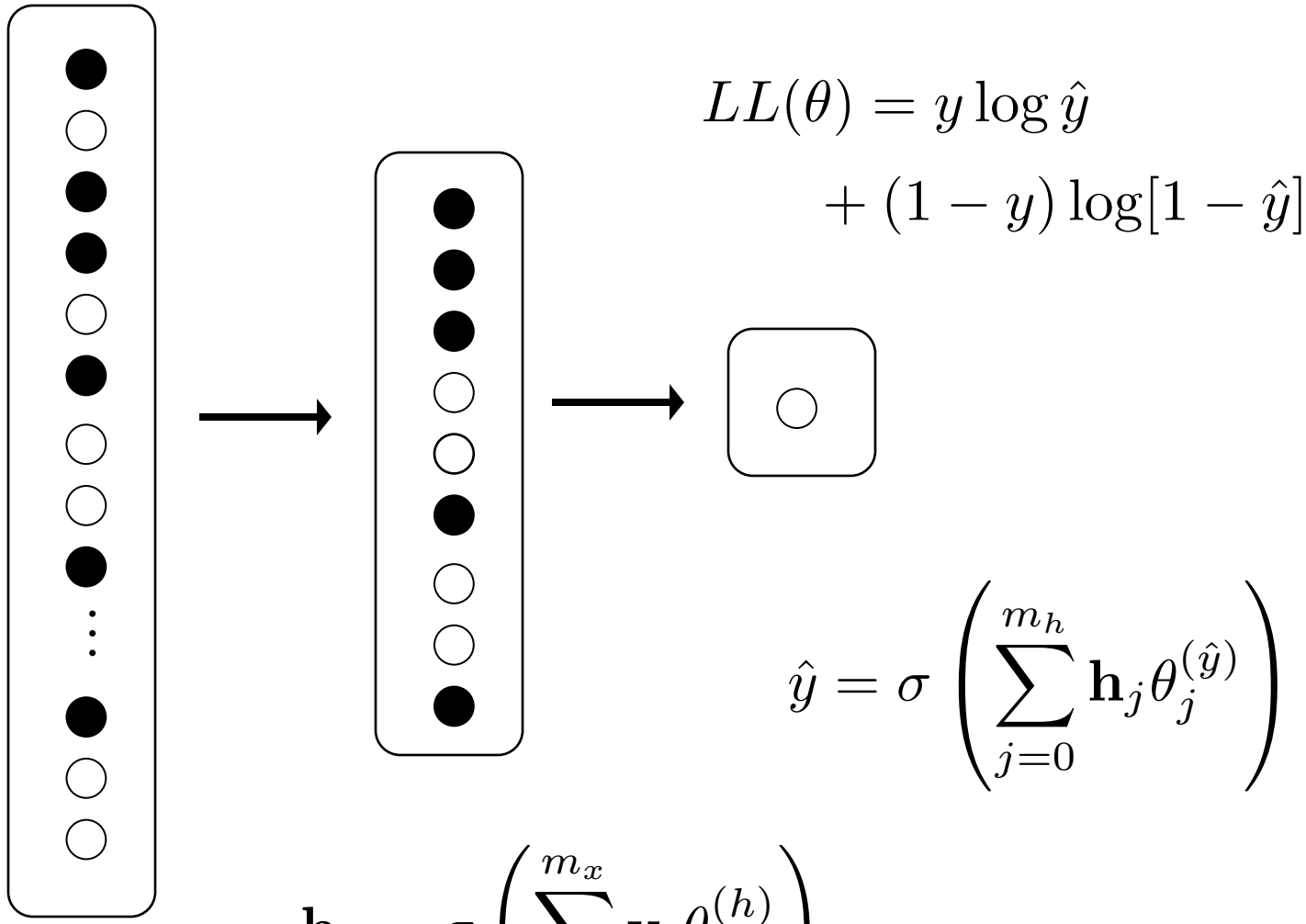


Forward Pass



$$\mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

Forward Pass

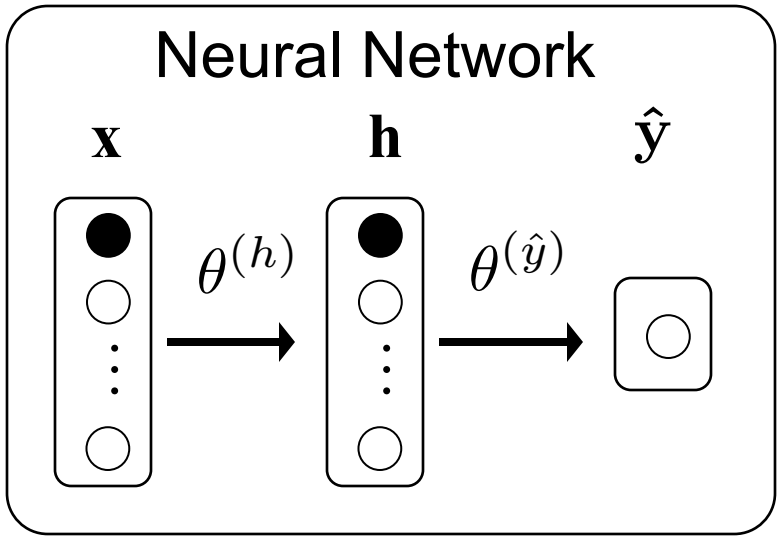


$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

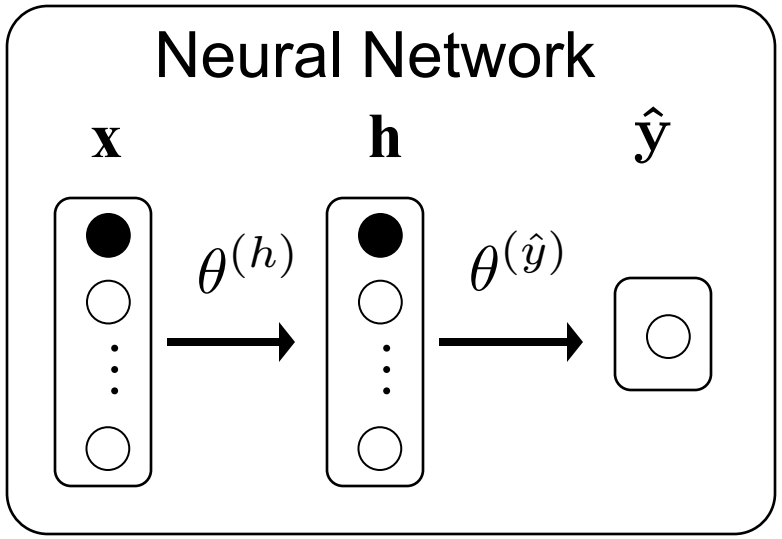
$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

$$\mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

All Together



Smoke Check 1



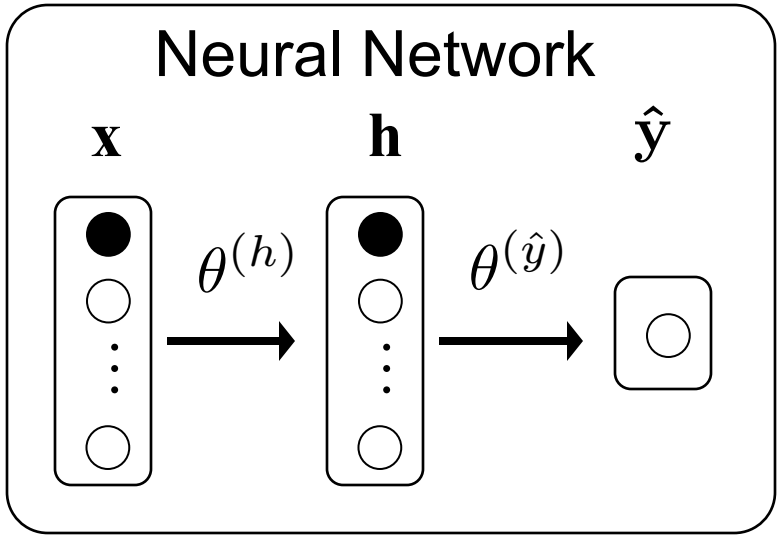
$$|\mathbf{x}| = 40$$

$$|\mathbf{h}| = 20$$

How many parameters in $\theta^{(\hat{y})}$?

a) 2 b) 20 c) 40 d) 800

Smoke Check 2



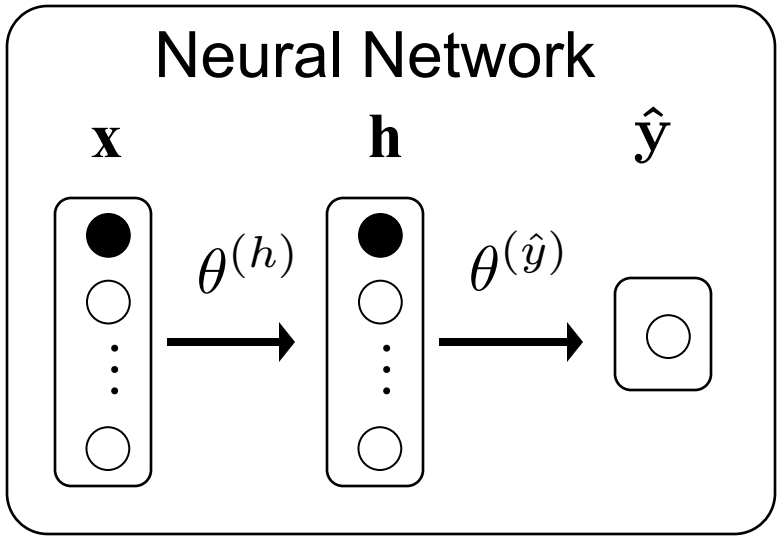
$$|\mathbf{x}| = 40$$

$$|\mathbf{h}| = 20$$

How many parameters in $\theta^{(h)}$?

a) 2 b) 20 c) 40 d) 800

Smoke Check 3



$$|\mathbf{x}| = 40$$

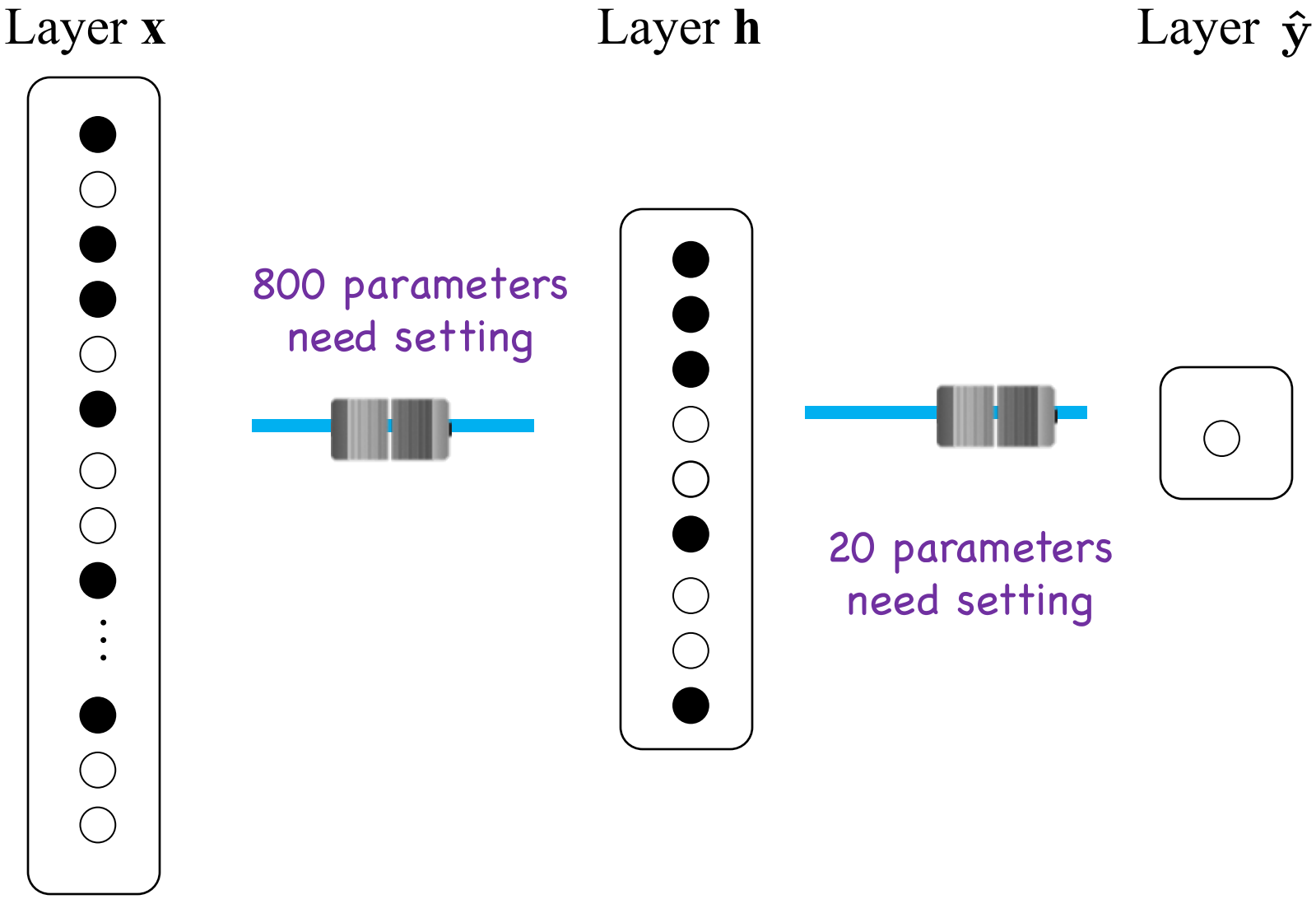
$$|\mathbf{h}| = 20$$

How many parameters in total?

a) 800 b) 20 c) 820 d) 16000

Today: Do Something Brave

Forward Pass



Only Have to Do Three Things

1. Make deep learning assumption

2. Calculate the log probability for all data

3. Get partial derivative of log likelihood with respect to each theta

Smoke Check

3 Get partial derivative of log likelihood with respect to each theta

Why?

Why We Calculate Partial Derivatives

A deep learning model gets its **intelligence** by having **useful thetas**.

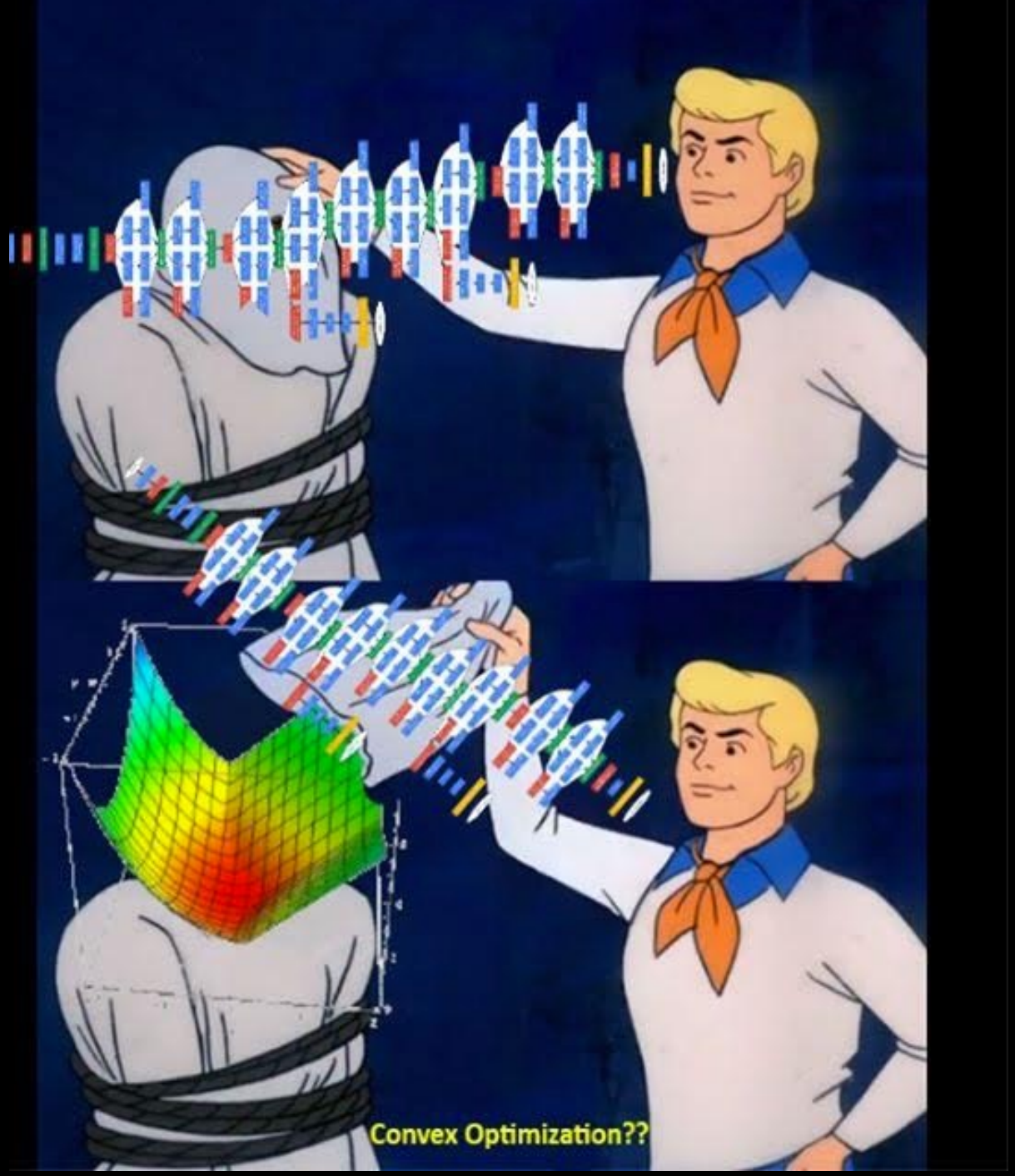
We can find **useful thetas**, by searching for ones that **maximize likelihood** of our training data

We can **maximize likelihood** using **optimization techniques** (such as gradient ascent).

In order to use **optimization techniques**, we need to calculate the **partial derivative** of likelihood with respect to thetas.

Basically MLE is hard because it has so many details

Thanks to Keith Eicher


Convex Optimization??

Only Have to Do Three Things

1. Make deep learning assumption

$$P(Y = 1|X = \mathbf{x}) = \hat{y}$$
$$P(Y = 0|X = \mathbf{x}) = 1 - \hat{y}$$

2. Calculate the log probability for all data

Same Assumption, Same LL

$$P(Y = 1|X = \mathbf{x}) = \hat{y} \qquad \hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right) \qquad \mathbf{h}_j = \sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(h)}\right)$$

For one datum

$$P(Y = y|\mathbf{X} = \mathbf{x}) = (\hat{y})^y (1 - \hat{y})^{1-y}$$

Feel the Bern!

$$Y \sim \text{Bern}(\hat{y})$$

For IID data

$$L(\theta) = \prod_{i=1}^{n} P(Y = y^{(i)}|X = \mathbf{x}^{(i)})$$

$$= \prod_{i=1}^{n} (\hat{y}^{(i)})^{y^{(i)}} \cdot \left[1 - (\hat{y}^{(i)})\right]^{(1-y^{(i)})}$$

Take the log

$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}]$$

Only Have to Do Three Things

1. Make deep learning assumption

$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right)$$

$$P(Y = 1 | X = \mathbf{x}) = \hat{y}$$
$$P(Y = 0 | X = \mathbf{x}) = 1 - \hat{y}$$

2. Calculate the log probability for all data

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}]$$

3. Get partial derivative of log likelihood with respect to each theta

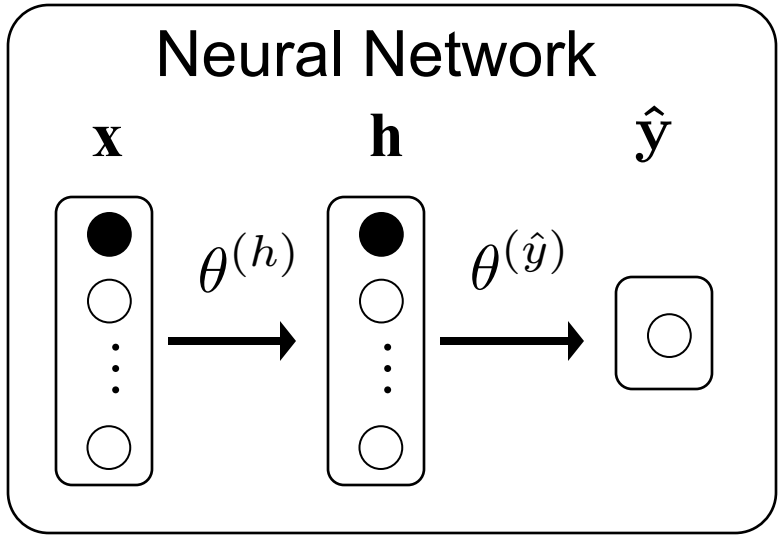
Derivative Goals

Loss with respect to output layer params

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to hidden layer params

$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$

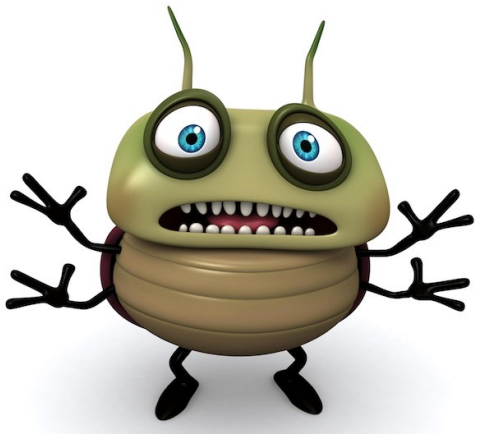


Bad Approach

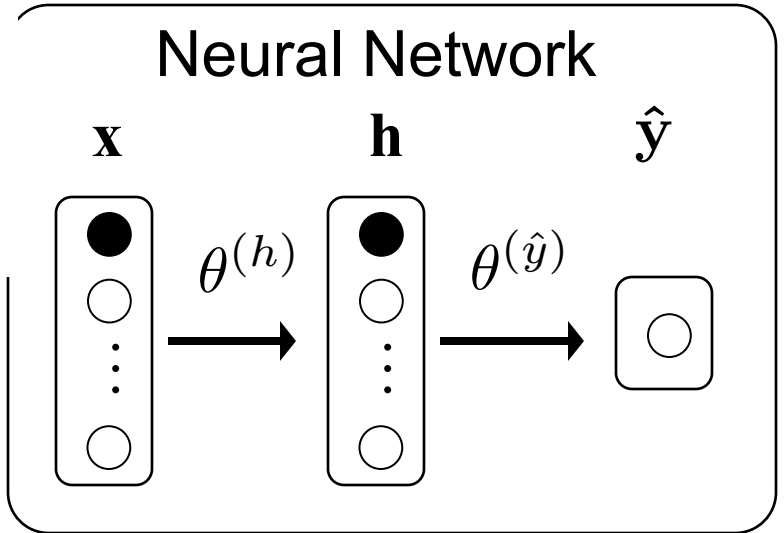
$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

$$\hat{y} = \sigma\left(\sum_{i=0}^{m_h} \mathbf{h}_i \theta_i^{(\hat{\mathbf{y}})}\right)$$

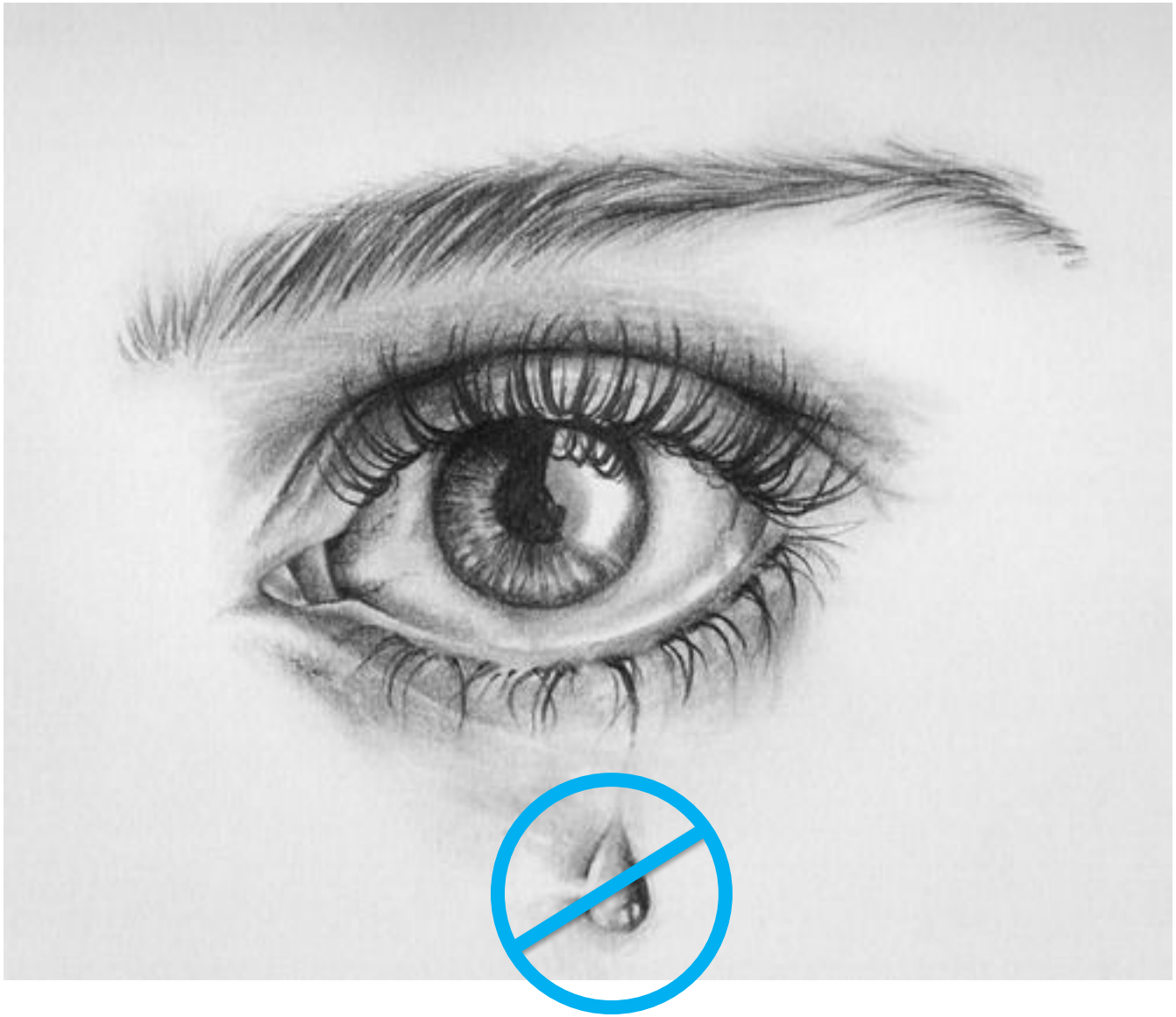
Math bug



$$= \sigma\left(\sum_{i=0}^{m_h}\left[\sigma\left(\sum_{i=0}^{m_x} \mathbf{x}_i \theta_{i,j}^{(\mathbf{h})}\right)\right]\theta_i^{(\hat{\mathbf{y}})}\right)$$



Derivatives Without Tears



Big Idea #1: Chain Rule

Woah Mr Blanton, you were right.
Chain rule is useful!

$$\frac{\partial f(z)}{\partial x} = \frac{\partial f(z)}{\partial z} \cdot \frac{\partial z}{\partial x}$$

First use:

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

Big Idea #2: Sigmoid Derivative

True fact about sigmoid functions

$$\frac{\partial}{\partial z}\sigma(z) = \sigma(z)[1 - \sigma(z)]$$

Big Idea #3: Derivative of Sum

$$LL(\theta) = \sum_{i=0}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}]$$

We only need to calculate the gradient for one training example!

$$\frac{\partial}{\partial x} \sum f(x) = \sum \frac{\partial}{\partial x} f(x)$$

We will pretend we only have one example

$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

We can sum up the gradients of each example to get the correct answer

Recall

Sigmoid has a Beautiful Slope

$$\frac{\partial}{\partial \theta_j}\sigma(\theta^T x)? \qquad \frac{\partial}{\partial z}\sigma(z) = \sigma(z)[1-\sigma(z)]$$

---

where $z = \theta^T x$

$$\frac{\partial}{\partial \theta_j}\sigma(\theta^T x) = \frac{\partial}{\partial z}\sigma(z) \cdot \frac{\partial z}{\partial \theta_j}$$

Chain rule!

$$\frac{\partial}{\partial \theta_j}\sigma(\theta^T x) = \sigma(\theta^T x)[1-\sigma(\theta^T x)]x_j$$

Plug and chug

Sigmoid, you should be a ski hill

This is ~~Sparta~~!!!!!

↑

Stanford

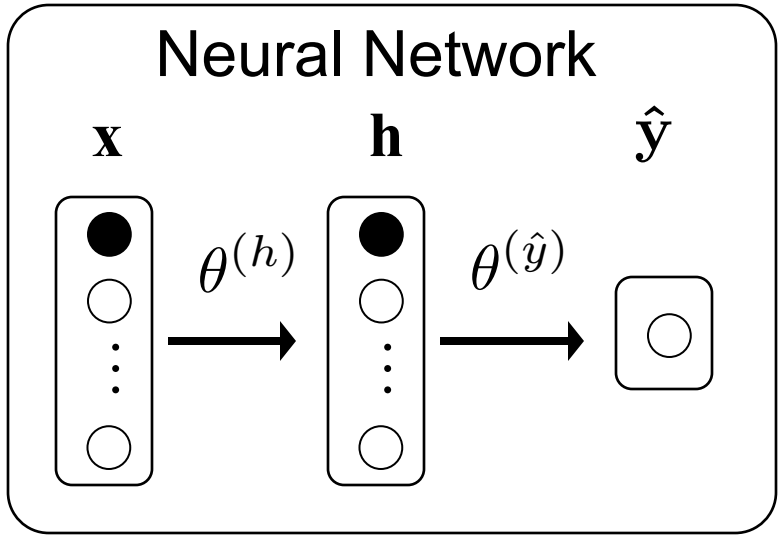
Derivative Goals

Loss with respect to output layer params

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to hidden layer params

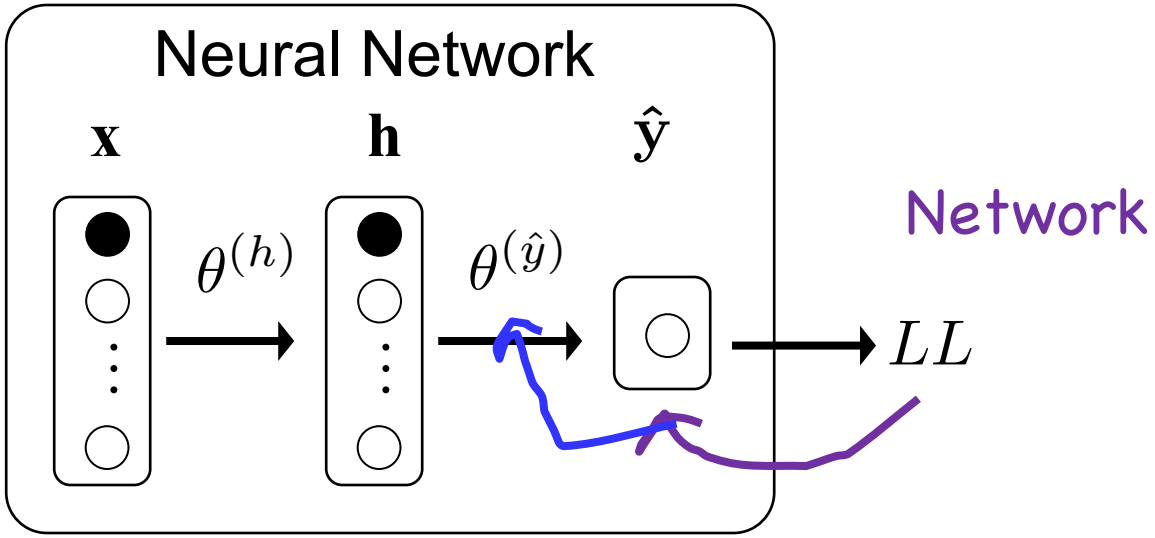
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$



Chain Rule Example 1

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Goal



Network

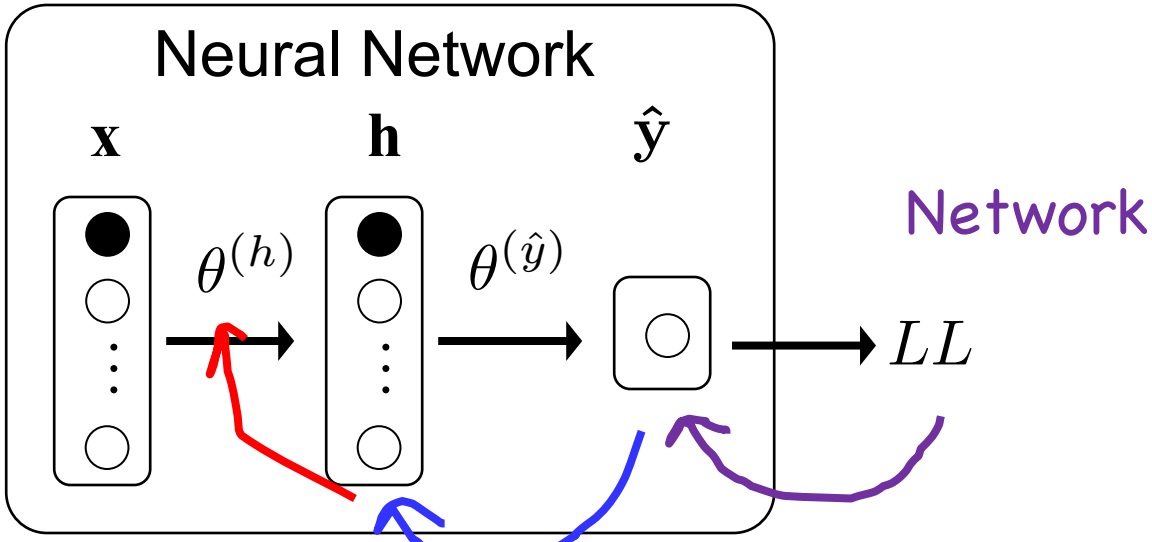
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

Decomposition

Chain Rule Example 2

$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$

Goal



Network

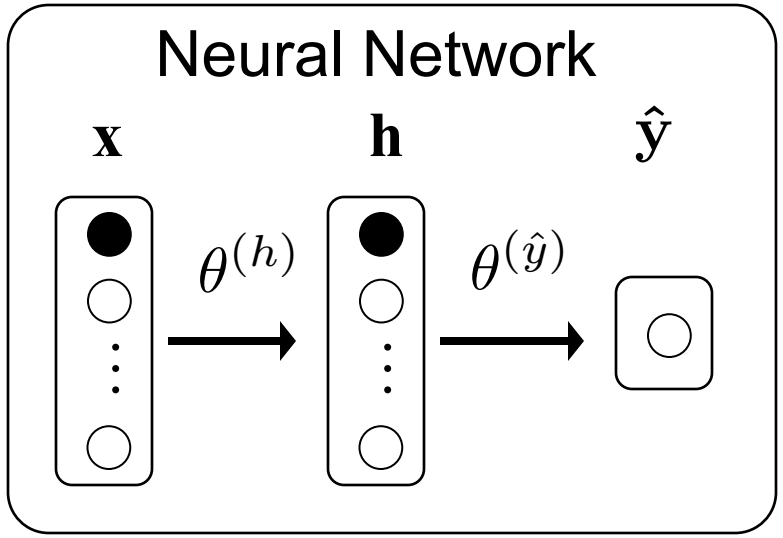
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$

Decomposition

Decomposition

Gradient of output layer params

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$



Gradient of output layer params

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

$$\frac{\partial LL(\theta)}{\partial \hat{y}} = \frac{y}{\hat{y}} + \frac{(1 - y)}{(1 - \hat{y})} \cdot \frac{\partial (1 - \hat{y})}{\partial \hat{y}}$$

$$\frac{\partial LL(\theta)}{\partial \hat{y}} = \frac{y}{\hat{y}} - \frac{(1 - y)}{(1 - \hat{y})}$$

Gradient of output layer params

$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}}$$

$$\hat{y} = \sigma\left(\sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}\right) = \sigma(z) \qquad \text{where} \qquad z = \sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}$$

$$\frac{\partial \hat{y}}{\partial \theta_i^{(\hat{y})}} = \hat{y}[1-\hat{y}] \cdot \frac{\partial}{\partial \theta_i^{(\hat{y})}} \sum_{j=0}^{m_h} \mathbf{h}_j \theta_j^{(\hat{y})}$$

$$= \hat{y}[1-\hat{y}] \cdot h_i$$

What! That's not scary!

Make it Simple

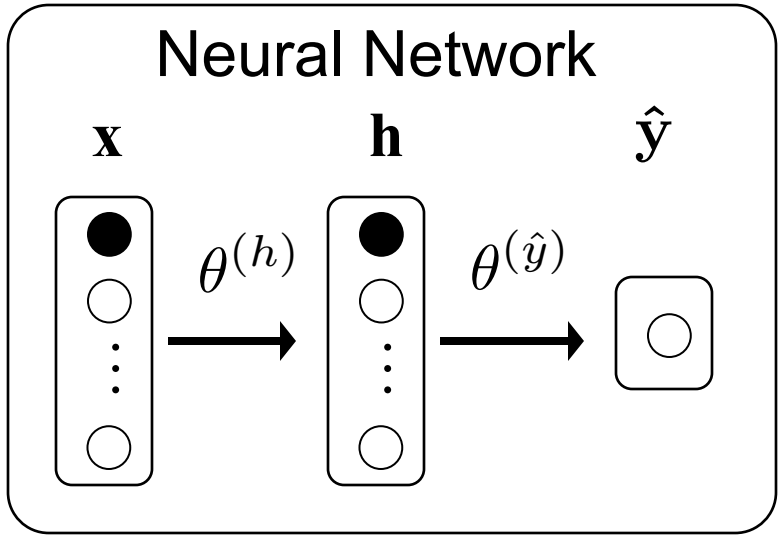
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}} = \text{  } \cdot \text{  }$$

$$\text{  } = \frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}$$

$$\text{  } = \hat{y}[1-\hat{y}] \cdot h_i$$

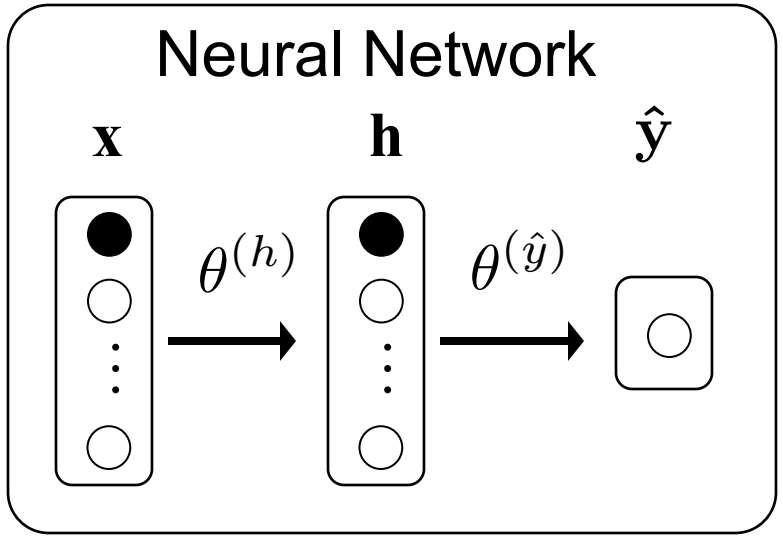
Boom!

$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$



Gradient of hidden layer params

$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$



Gradient of hidden layer params

$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$

$$\hat{y} = \sigma\left(\sum_{i=0}^{m_h} \mathbf{h}_i \theta_i^{(\hat{y})}\right)$$

$$\frac{\partial \hat{y}}{\partial \mathbf{h}_j} = \hat{y}[1-\hat{y}]\theta_j^{(\hat{y})}$$

Wait is it over?

Gradient of hidden layer params

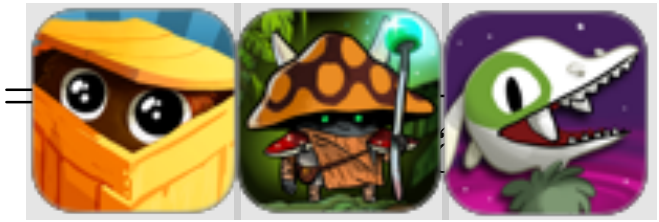
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \frac{\partial LL}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \mathbf{h}_j} \cdot \frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}}$$

$$\mathbf{h}_j = \sigma\left(\sum_{k=0}^{m_x} \mathbf{x}_k \theta_{k,j}\right)$$

$$\frac{\partial \mathbf{h}_j}{\partial \theta_{i,j}^{(h)}} = \mathbf{h}_j[1 - \mathbf{h}_j]\mathbf{x}_i$$

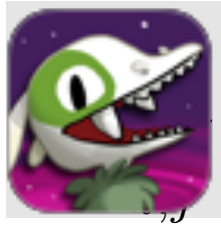
That one too?

Make it Simple

$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}} = \text{  }$$

$$\text{  } = \frac{y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}$$

$$\text{  } = \hat{y}[1-\hat{y}]\theta_j^{(\hat{y})}$$

$$\text{  } = \mathbf{h}_j[1-\mathbf{h}_j]\mathbf{x}_j$$

Congrats. You now know Backpropagation

Moment of silence

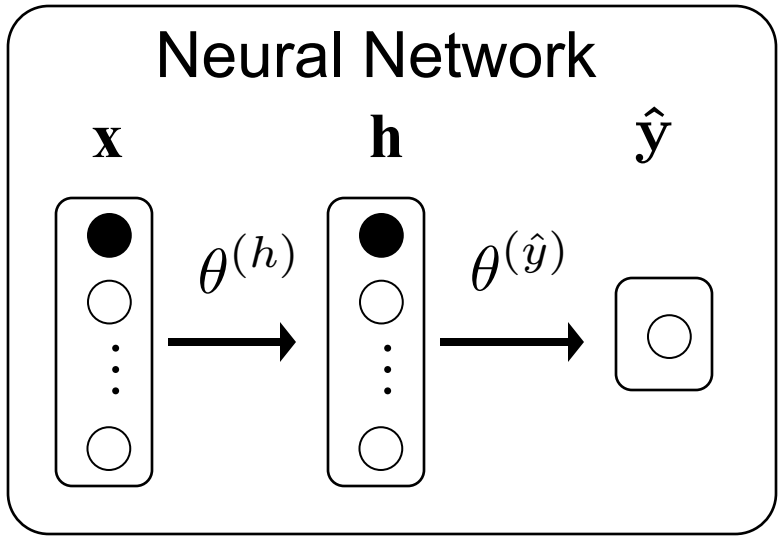
Summary: Simple Calculations For

Loss with respect to output layer params

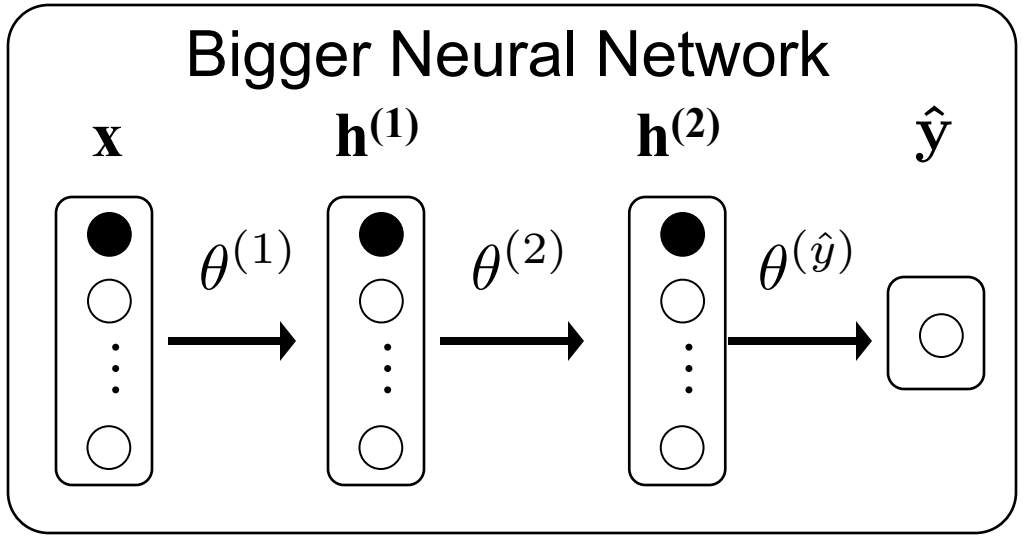
$$\frac{\partial LL(\theta)}{\partial \theta_i^{(\hat{y})}}$$

Loss with respect to hidden layer params

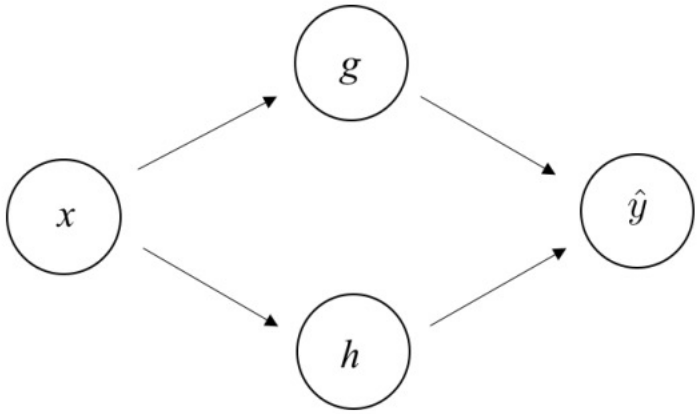
$$\frac{\partial LL(\theta)}{\partial \theta_{i,j}^{(h)}}$$



What Would You Do Here?



What if You Had a Neural Network Like This?



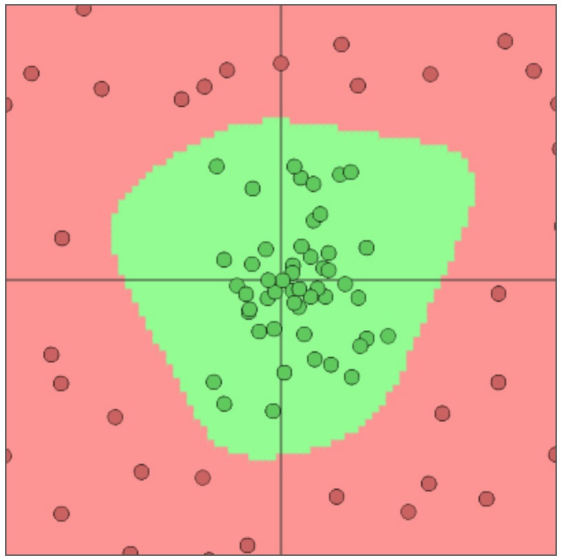
$$g = \text{sigmoid}(\theta_1 \cdot x)$$
$$h = \text{sigmoid}(\theta_2 \cdot x)$$
$$\hat{y} = \text{sigmoid}(\theta_3 \cdot g + \theta_4 \cdot h)$$
$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

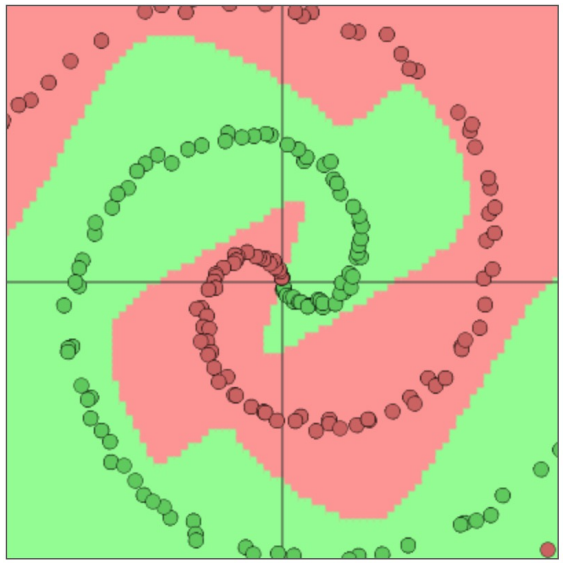
1. Calculate partial derivative for one data instance
2. Use chain rule!
3. Sigmoid derivatives come out simple if you use the right decomp.
4. You don’t need to give the most reduced answer

Chain rule:
Game changer for
artificial intelligence

Neural Networks Can Learn Complex Functions

- Some data sets/functions are not separable



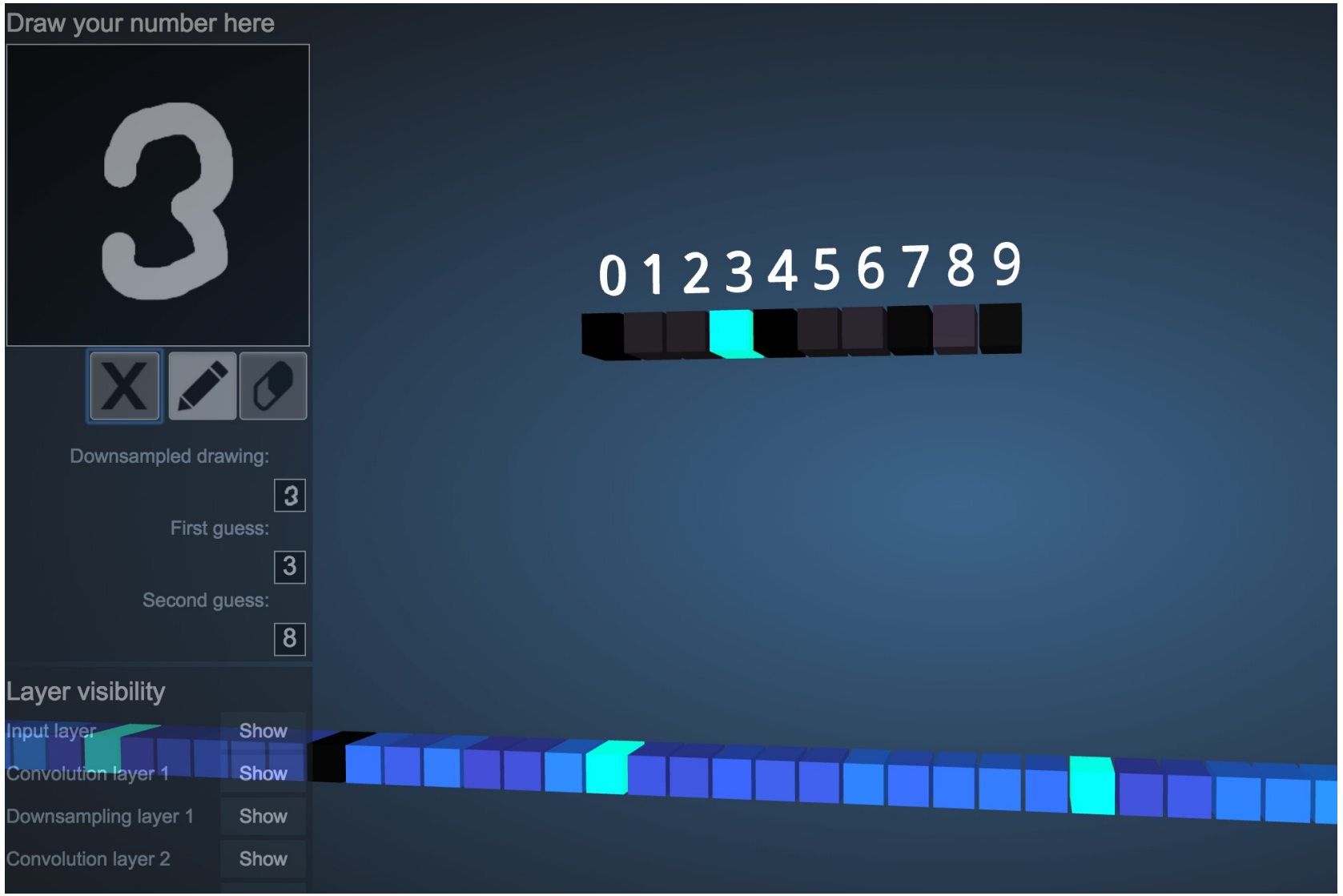


  - These are classifiers learned by neural networks

http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Some Extra Ideas!

Multiple Outputs



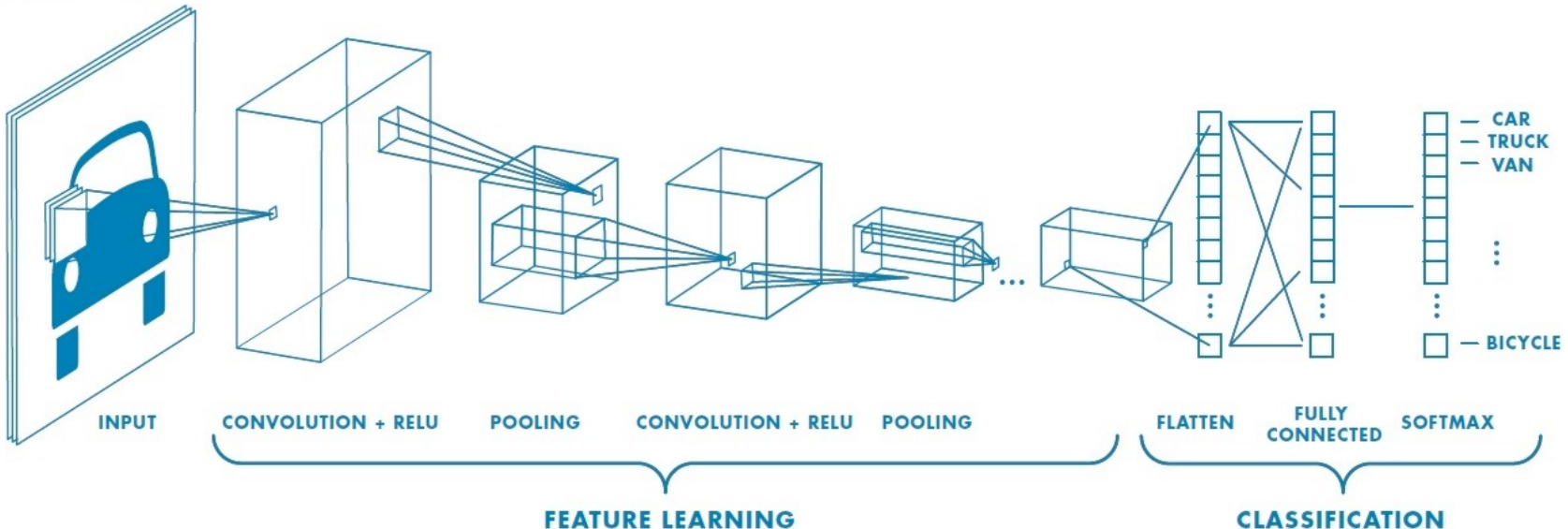
Multiple Output Classification?

Softmax is a generalization of the sigmoid function that squashes a K -dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\text{softmax}(\mathbf{z})$ of real values in the range [0, 1] that add up to 1.

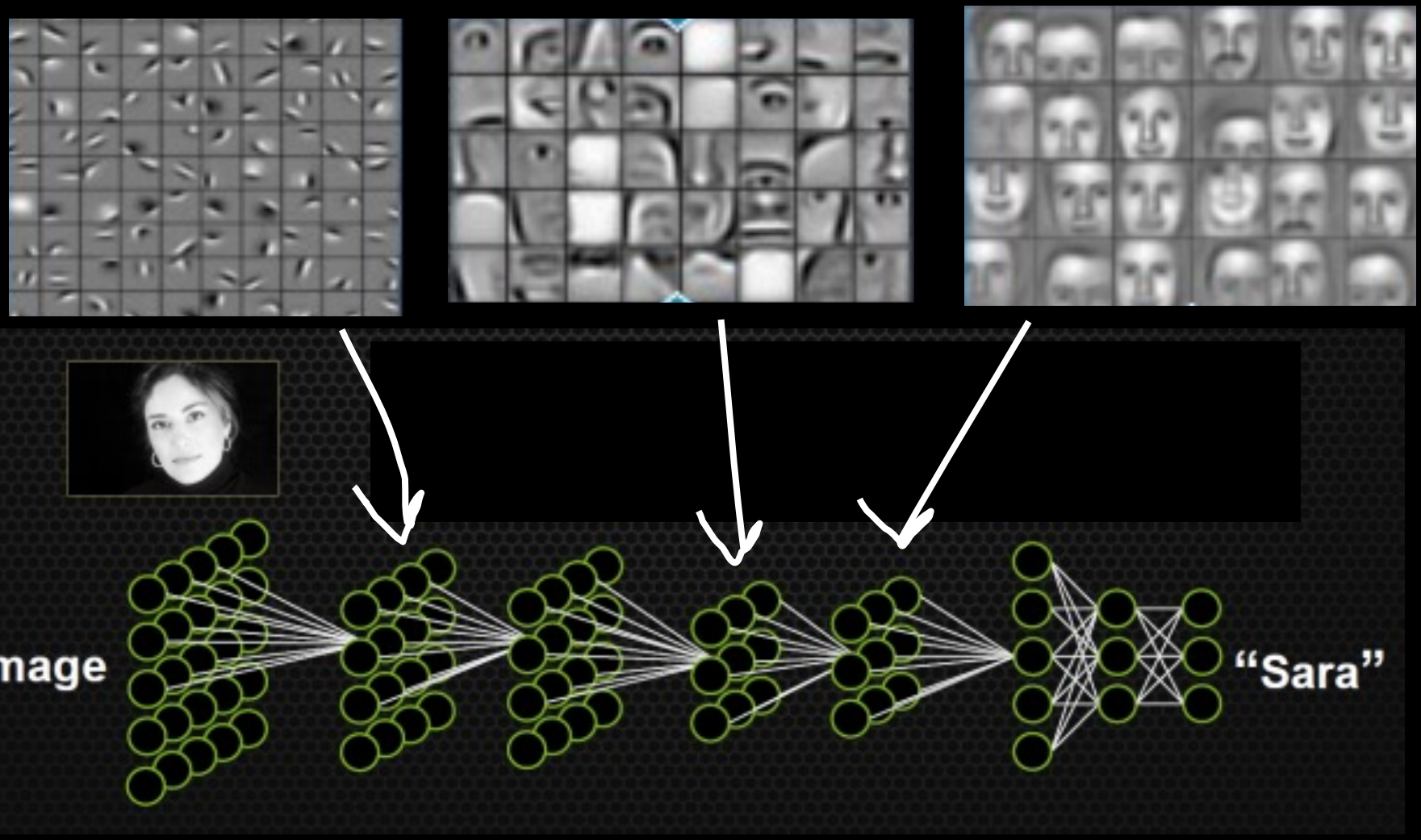
$$P(Y = j|\mathbf{X} = \mathbf{x}) = \text{softmax}(f(\mathbf{x}))_j$$

Shared Weights?

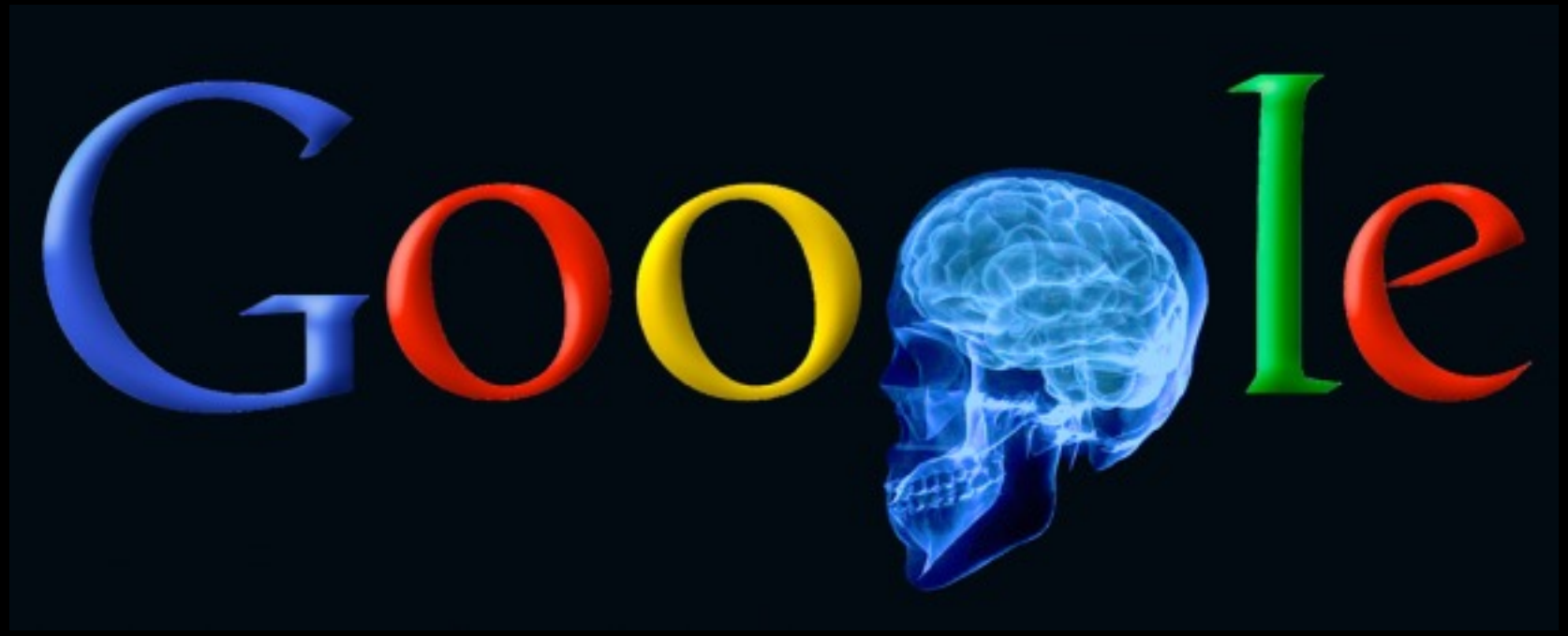
Convolution it turns out if you want to force some of your weights to be shared for different neurons, the math isn't that much harder. This is used a lot for vision (CNN).



Works for any number of layers



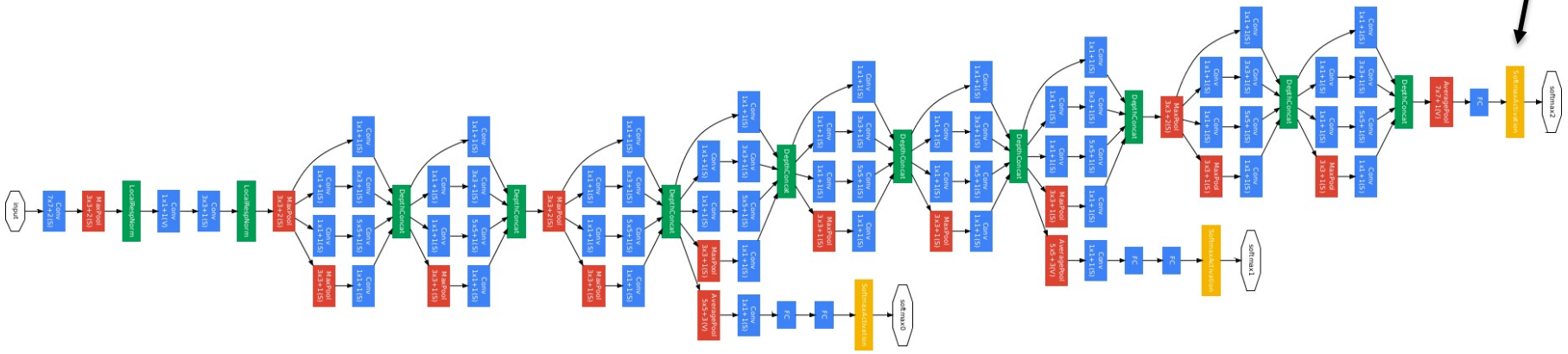
GoogLeNet Brain



1 Trillion Artificial Neurons

GoogLeNet Brain

Multiple,
Multi class output



22 layers deep

The Cat Neuron

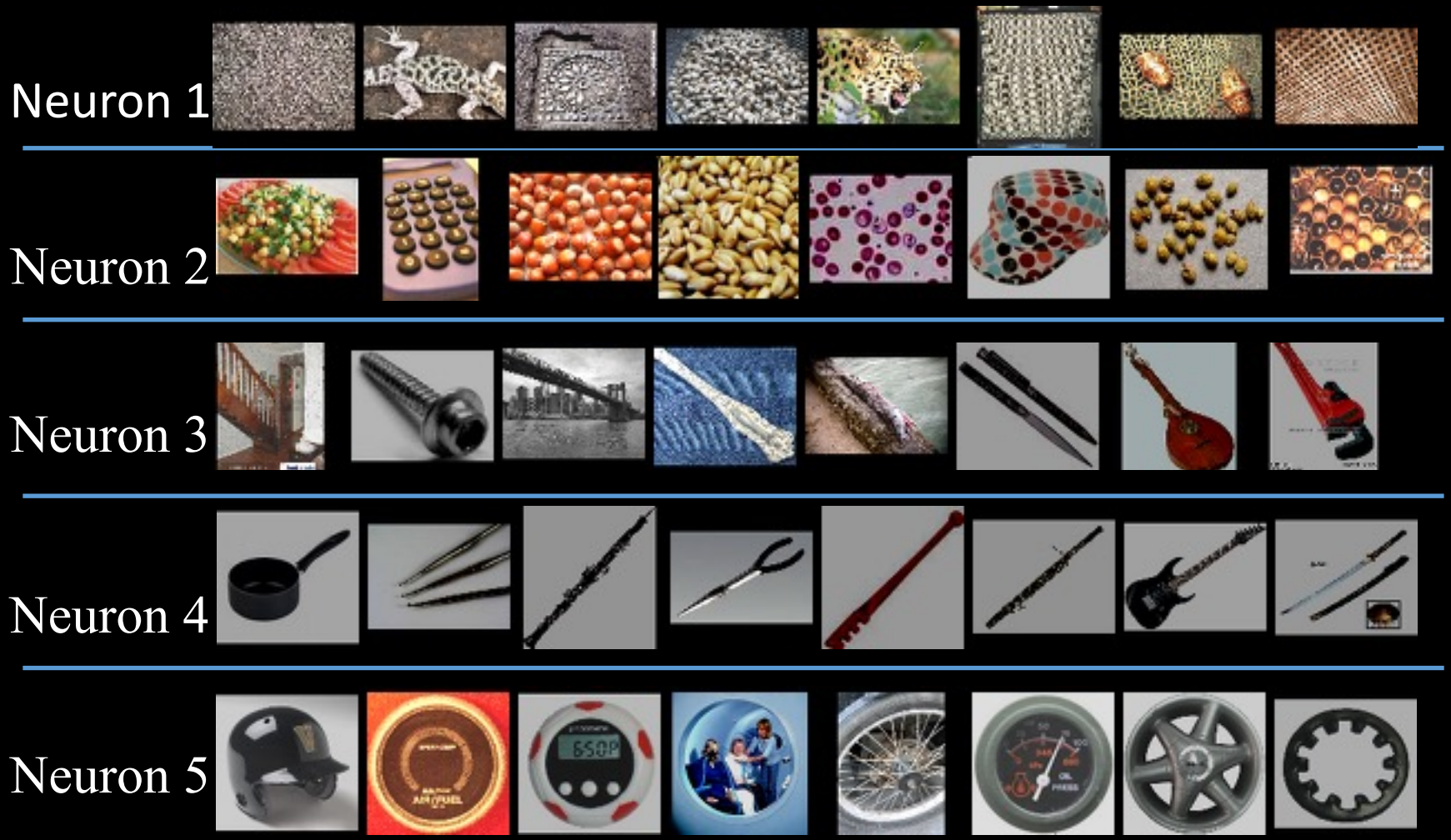
Top stimuli from the test set

Optimal stimulus
by numerical optimization

Le, et al., *Building high-level features using large-scale unsupervised*

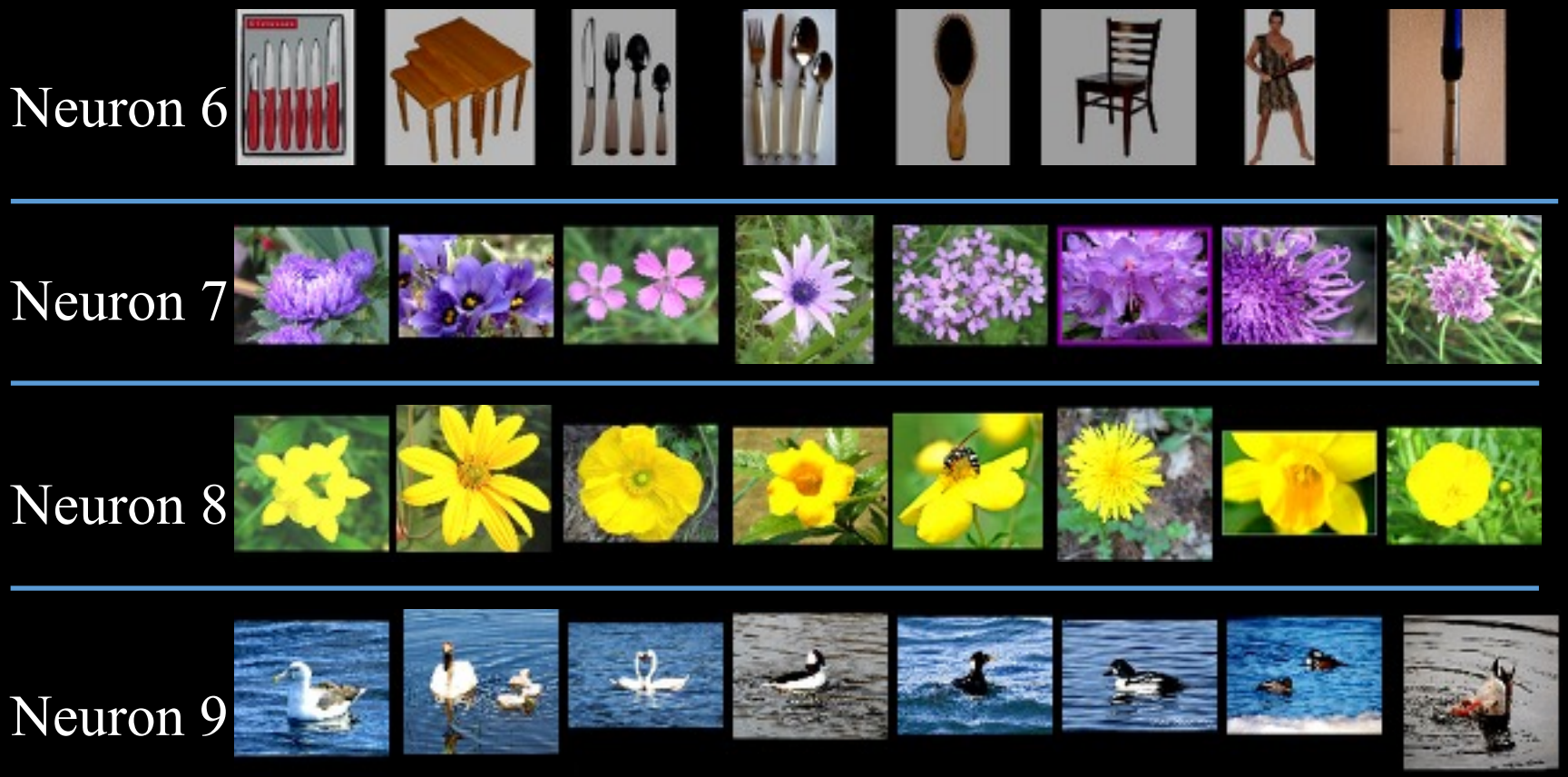

Hire the smartest people in the world
Invent cat detector

Best Neuron Stimuli



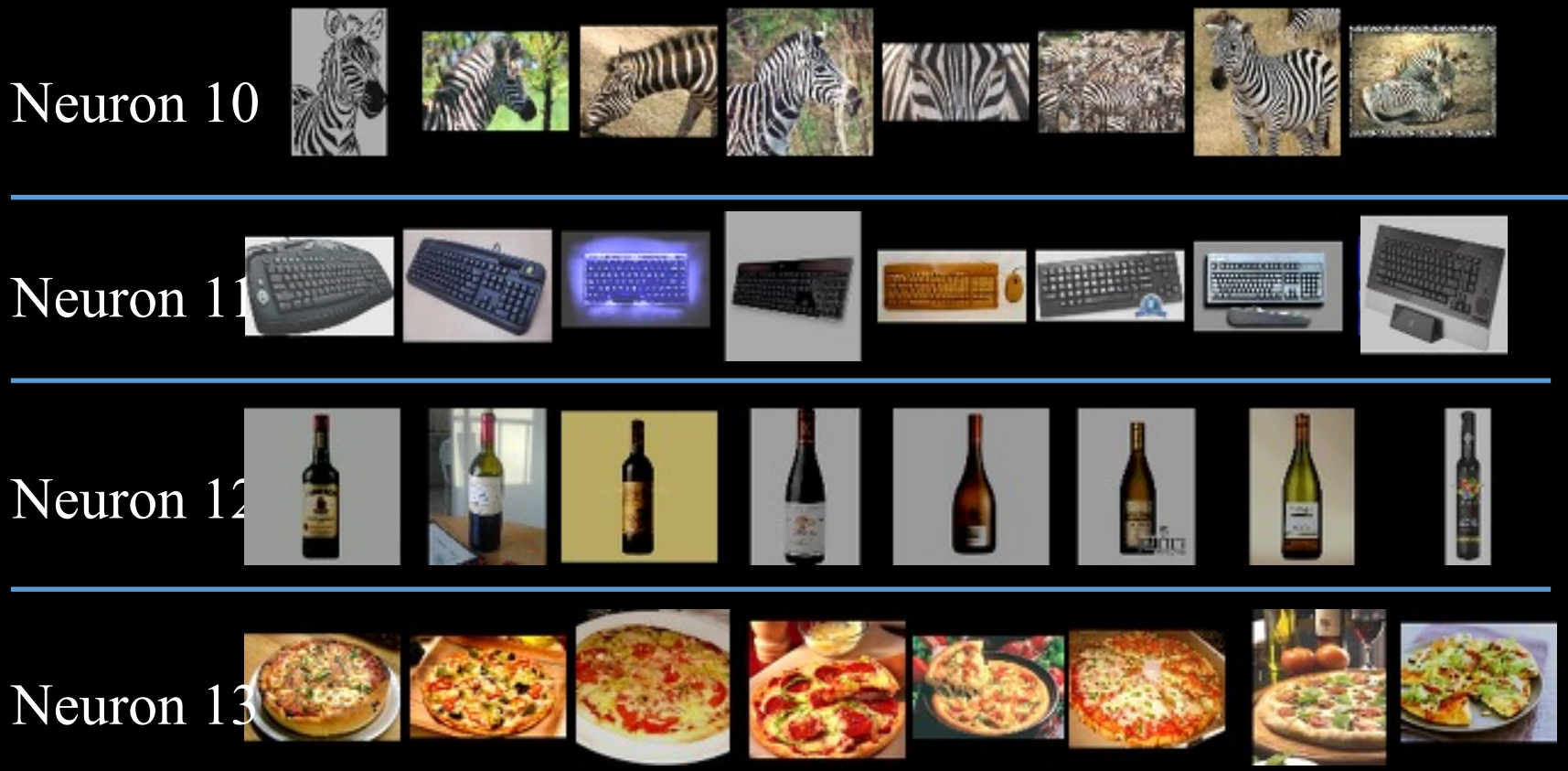
Le, et al., *Building high-level features using large-scale unsupervised*

Best Neuron Stimuli



Le, et al., *Building high-level features using large-scale unsupervised*

Best Neuron Stimuli



Le, et al., *Building high-level features using large-scale unsupervised*

ImageNet Classification

22,000 categories

14,000,000 images

Hand-engineered features (SIFT, HOG, LBP),
Spatial pyramid, SparseCoding/Compression

Le, et al., *Building high-level features using large-scale unsupervised*

22,000 is a lot!

…
smoothhound, smoothhound shark, Mustelus mustelus
American smooth dogfish, Mustelus canis
Florida smoothhound, Mustelus norrisi
whitetip shark, reef whitetip shark, Triaenodon obseus
Atlantic spiny dogfish, Squalus acanthias
Pacific spiny dogfish, Squalus suckleyi
hammerhead, hammerhead shark
smooth hammerhead, Sphyrna zygaena
smalleye hammerhead, Sphyrna tudes
shovelhead, bonnethead, bonnet shark, Sphyrna tiburo
angel shark, angelfish, Squatina squatina, monkfish
electric ray, crampfish, numbfish, torpedo
smalltooth sawfish, Pristis pectinatus
guitarfish
roughtail stingray, Dasyatis centroura
butterfly ray
eagle ray
spotted eagle ray, spotted ray, Aetobatus narinari
cownose ray, cow-nosed ray, Rhinoptera bonasus
manta, manta ray, devilfish
Atlantic manta, Manta birostris
devil ray, Mobula hypostoma
grey skate, gray skate, Raja batis
little skate, Raja erinacea
…

Stingray

Mantaray

0.005%

Random guess

1.5%

Pre Neural Networks

?

GoogLeNet

Le, et al., *Building high-level features using large-scale unsupervised*

0.005%

Random guess

1.5%

Pre Neural Networks

43.9%

GoogLeNet

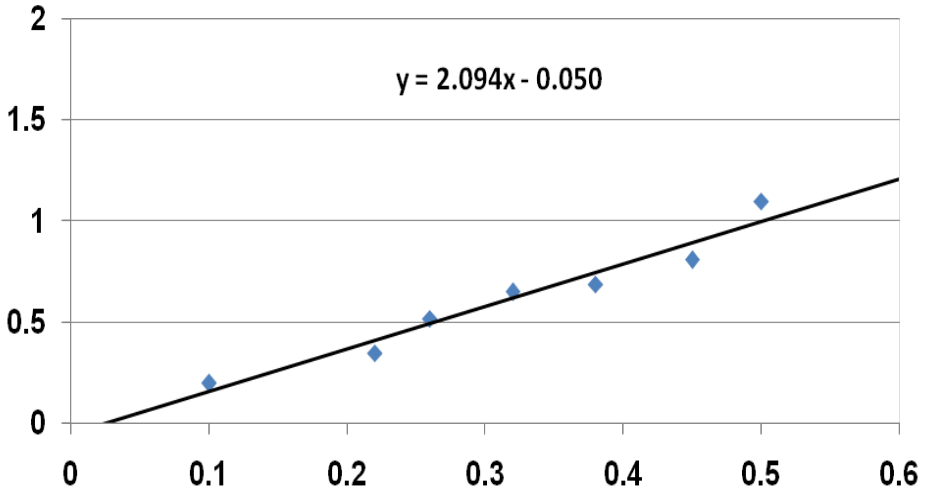
Szegedy et al, Going Deeper With Convolutions, CVPR 2015

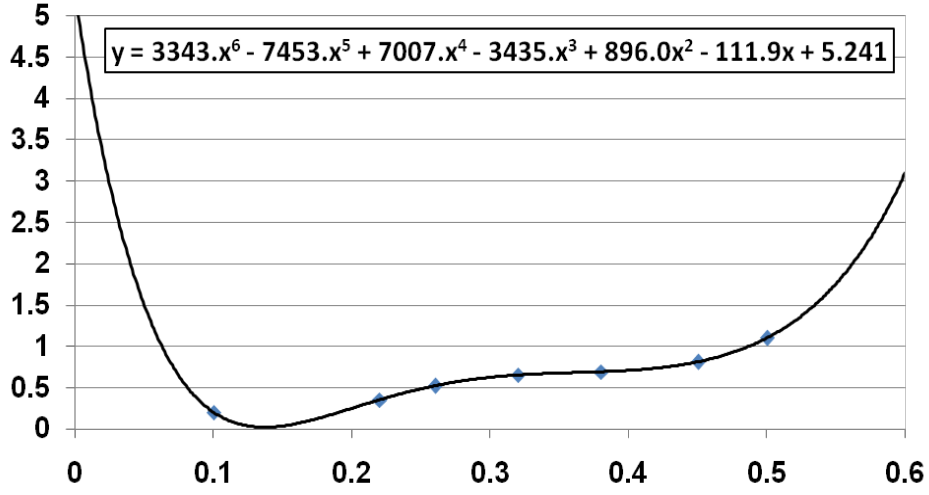
0.005%	1.5%	95.1%
Random guess	Pre Neural Networks	SE-ResNet

How many parameters
is too many?

Good ML = Generalization

- Goal of machine learning: build models that ***generalize*** well to predicting new data
  - "Overfitting": fitting the training data too well, so we lose generality of model

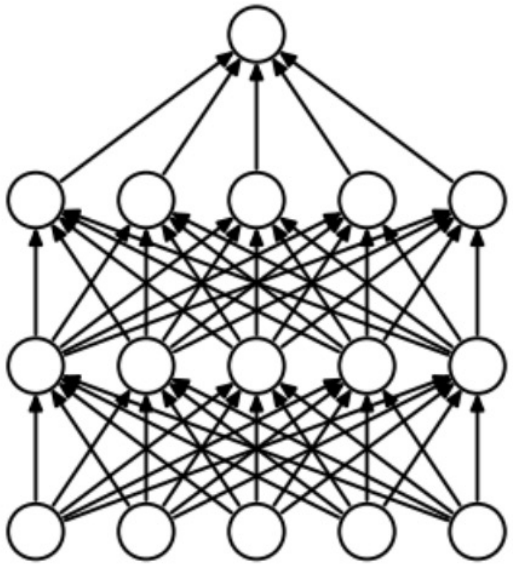




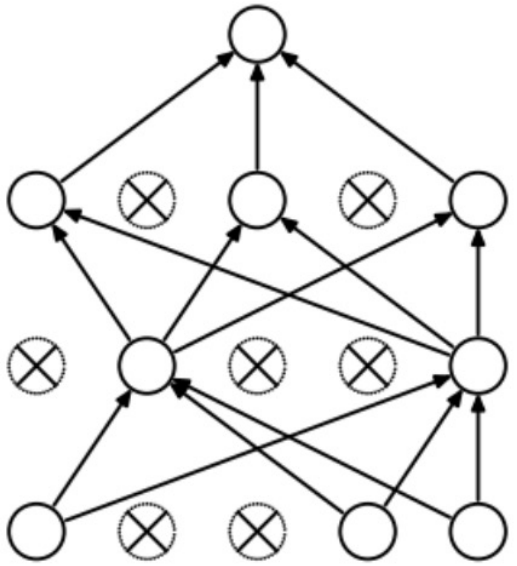
  - Polynomial on the right fits training data perfectly!
  - Which would you rather use to predict a new data point?

Prevent Overfitting?

Dropout when your model is training, randomly turn off your neurons with probability 0.5. It will make your network more robust.



(a) Standard Neural Net



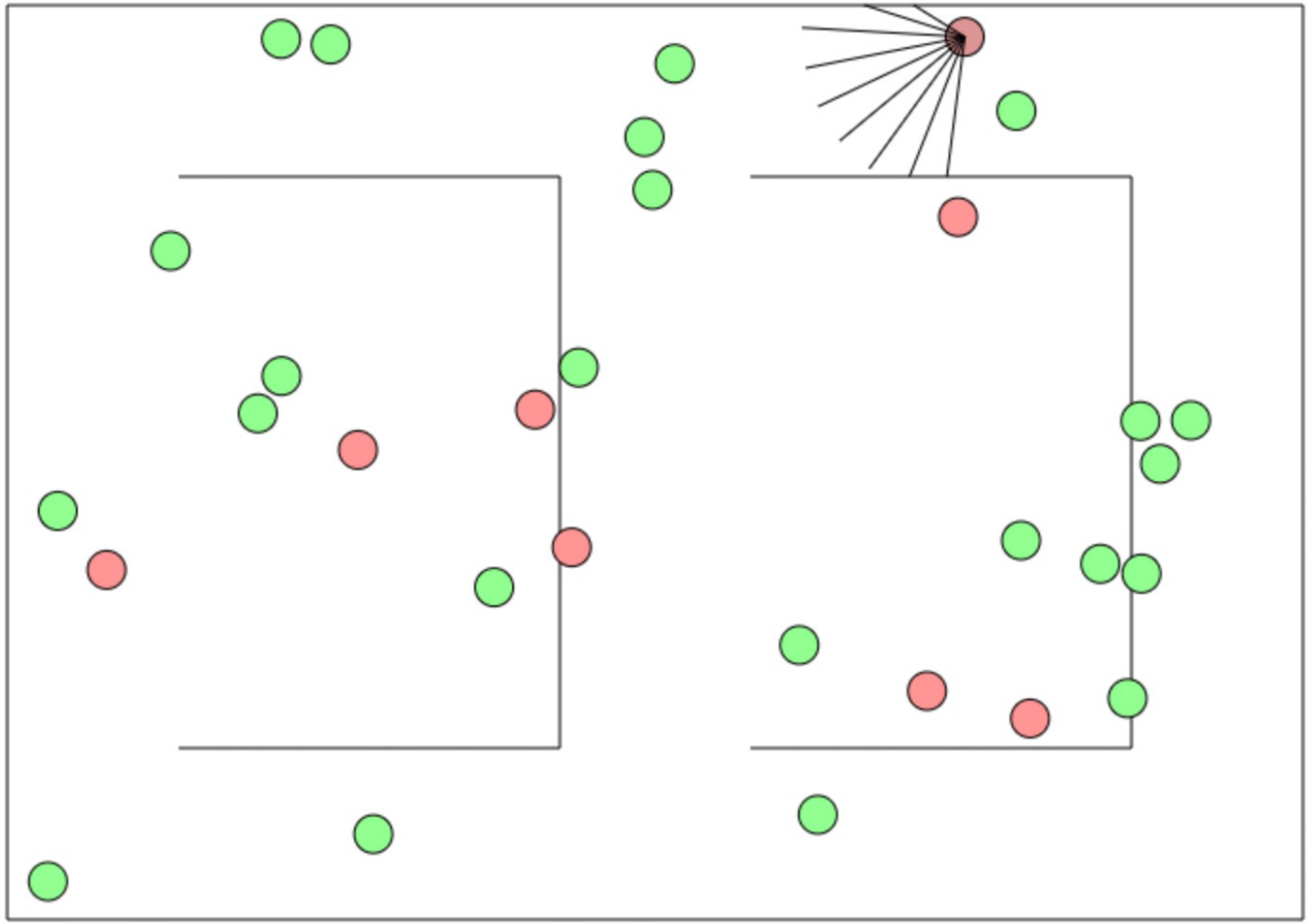
(b) After applying dropout.

Not everything is classification

Making Decisions?

Deep Reinforcement Learning
Instead of having the output of a model be a probability you can make it an expectation.

Deep Reinforcement Learning

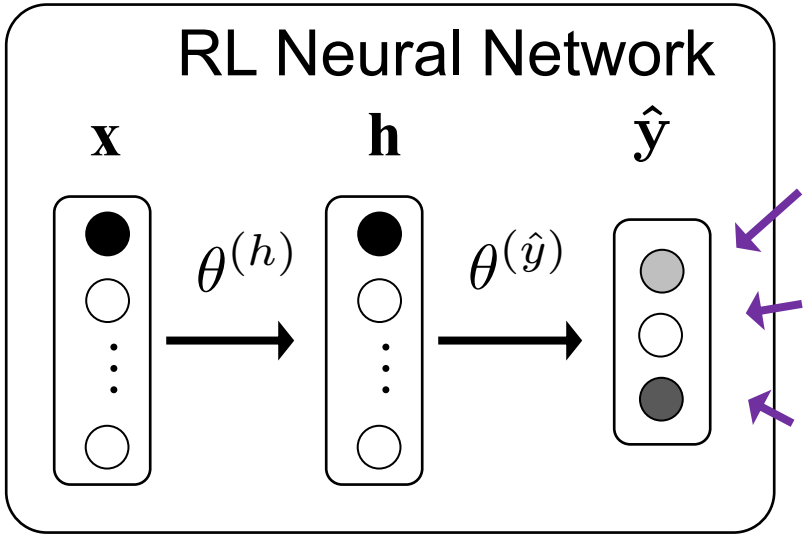


http://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html

Deep Reinforcement Learning

R is a reward and A_i is a legal action

Input is a representation of current state (S)



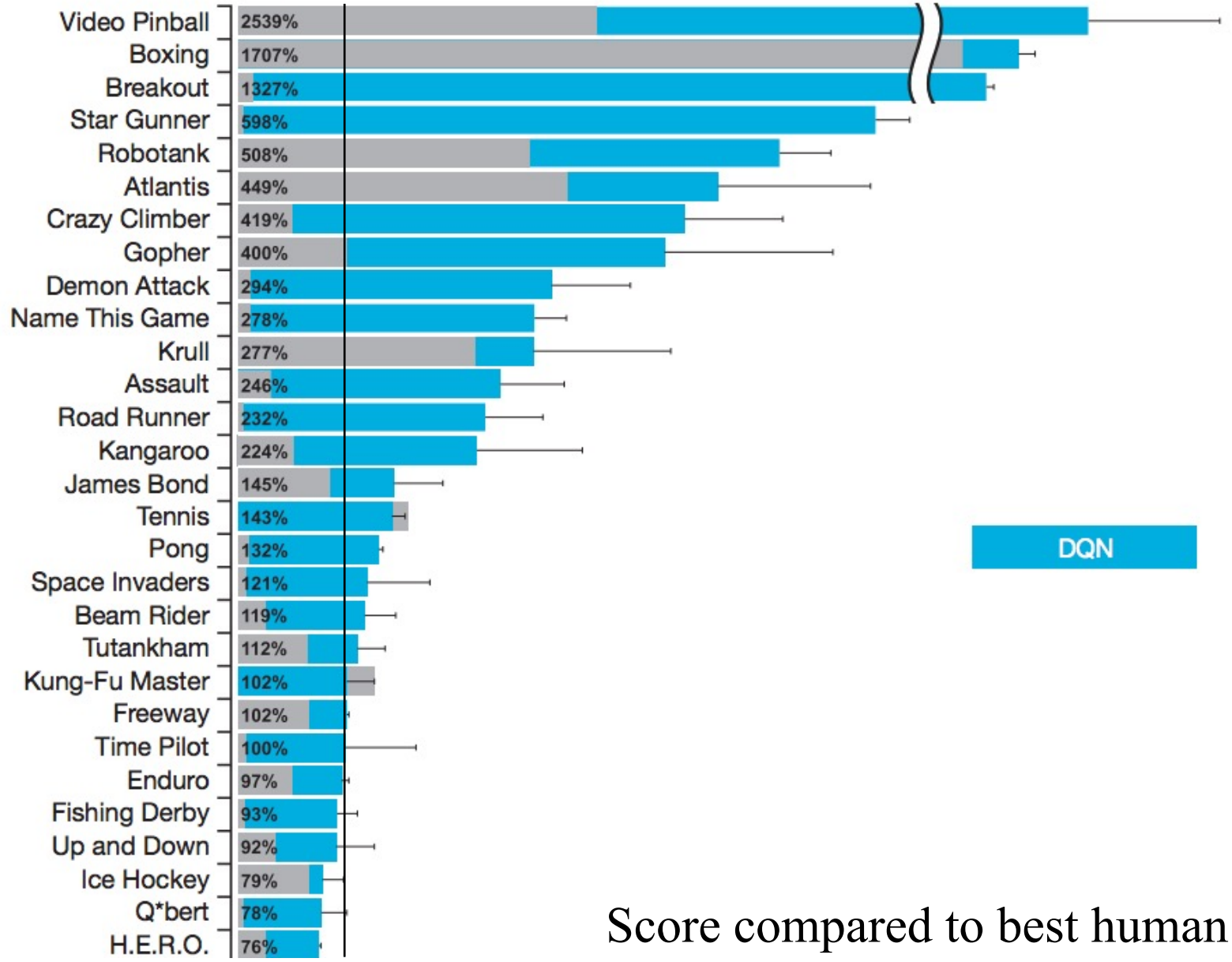
E[R | A_1, S]

E[R | A_2, S]

E[R | A_3, S]

Interpret outputs as expected reward for a given action

Deep Mind Atari Games



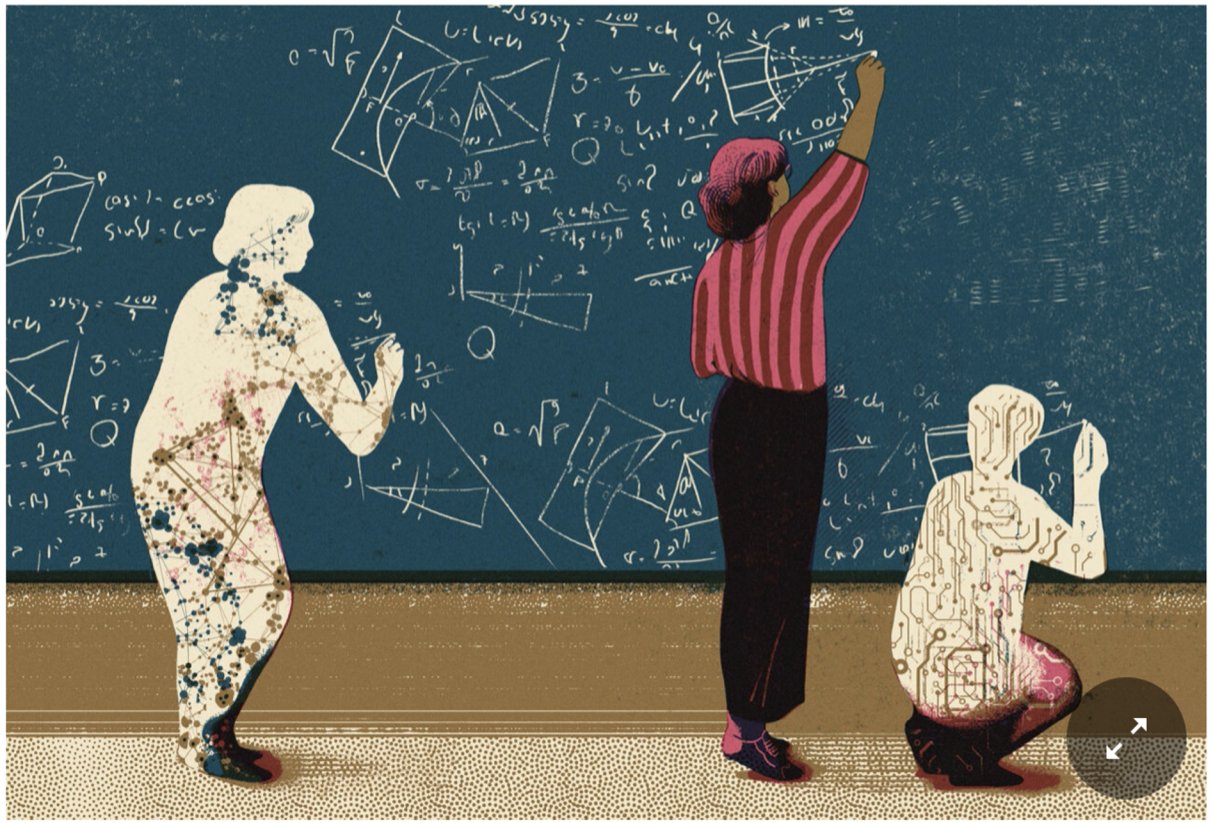
Score compared to best human

The New York Times

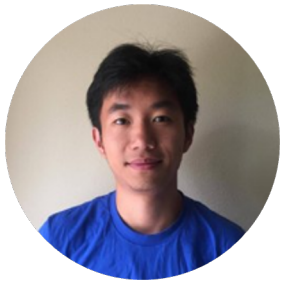
Can A.I. Grade Your Next Test?

Neural networks could give online education a boost by providing automated feedback to students.

Stanford | News



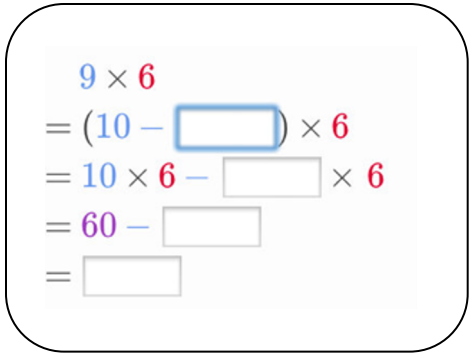
https://www.nytimes.com/2021/07/20/technology/ai-education-neural-networks.html



Give Feedback to Open Ended Work?

Human Level

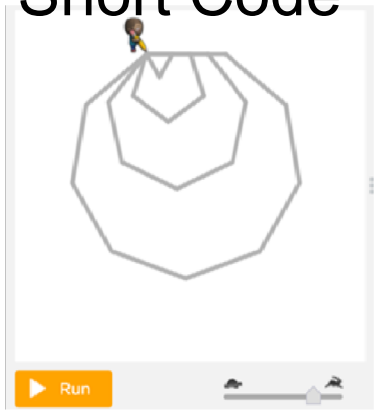
Simple Algebra



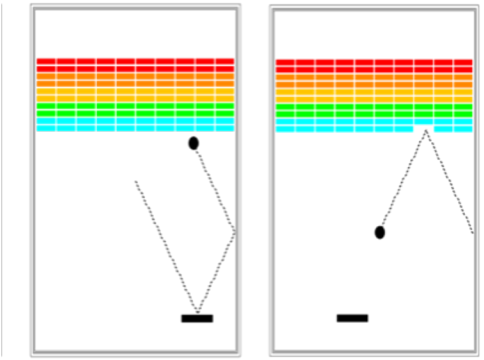
Short Answer

Why did the original colonists come to America?

Short Code



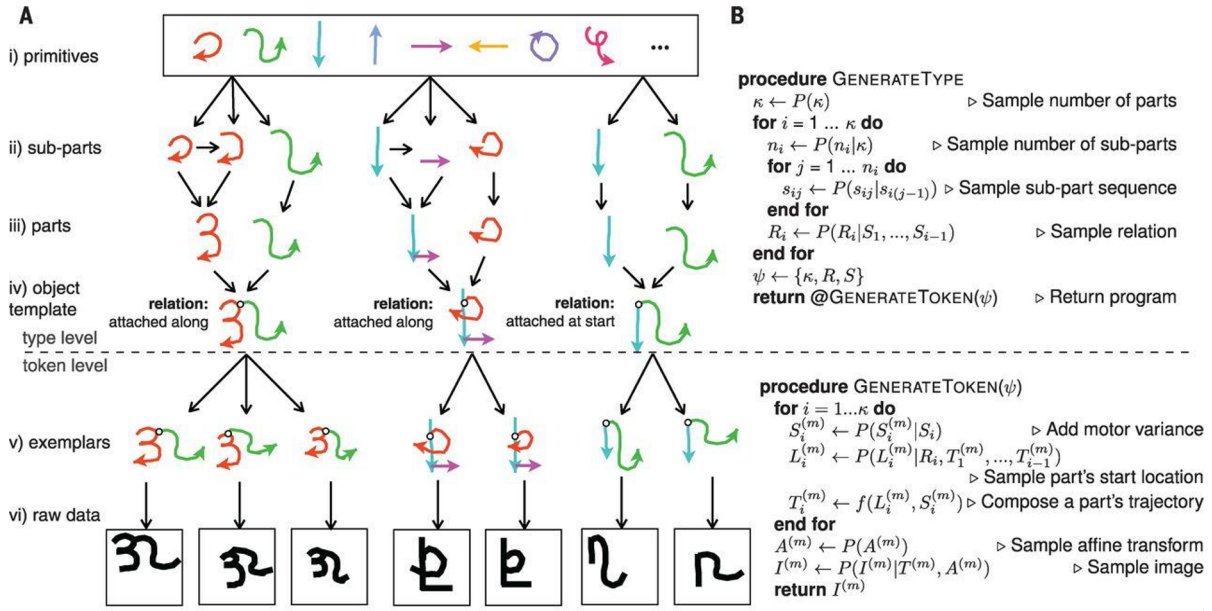
Long Code



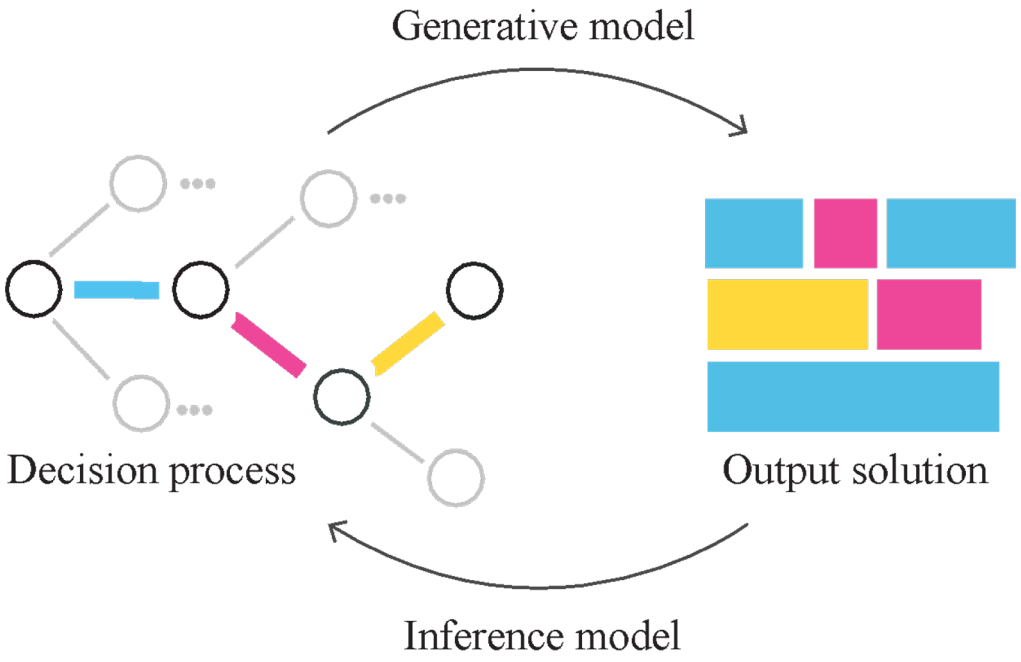
Deep Learning
Circa 2019

Generative Model of Grading

Lake et al, 2015

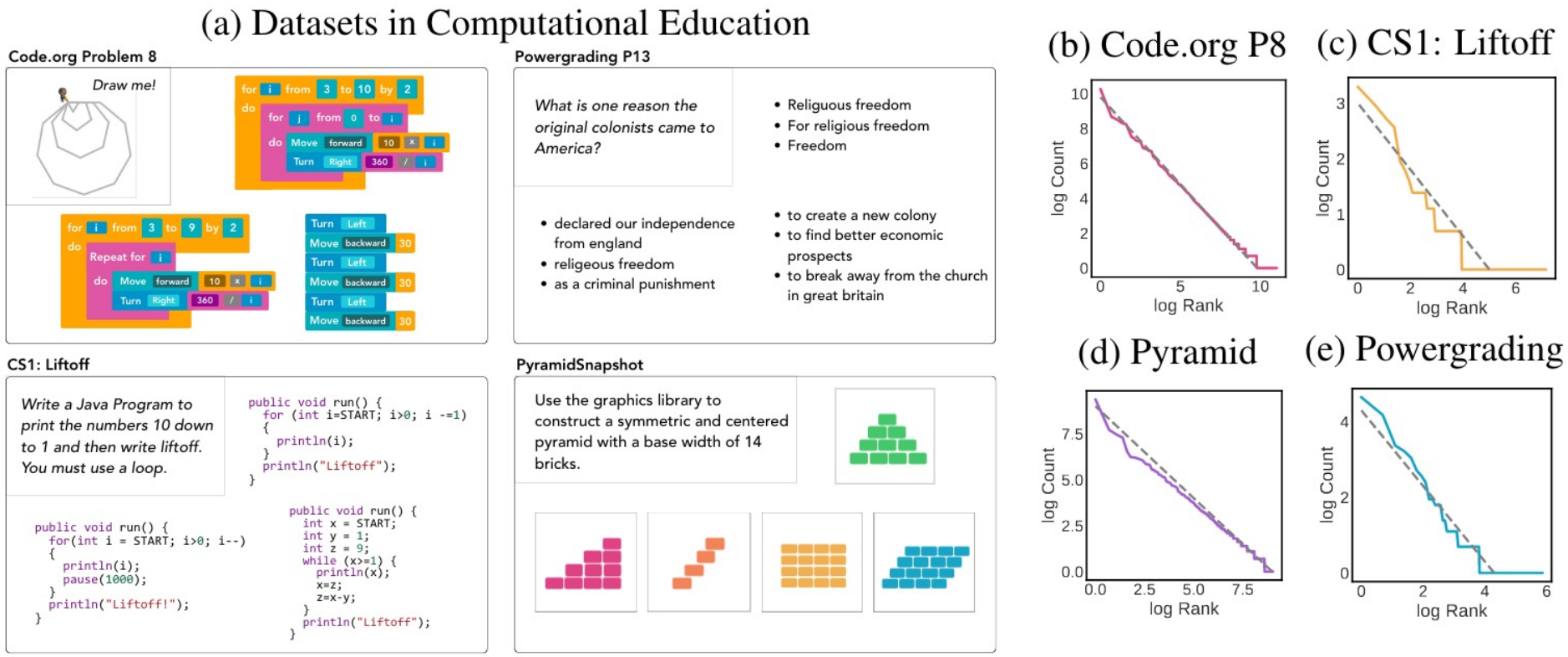


Our Team, EDM 2021

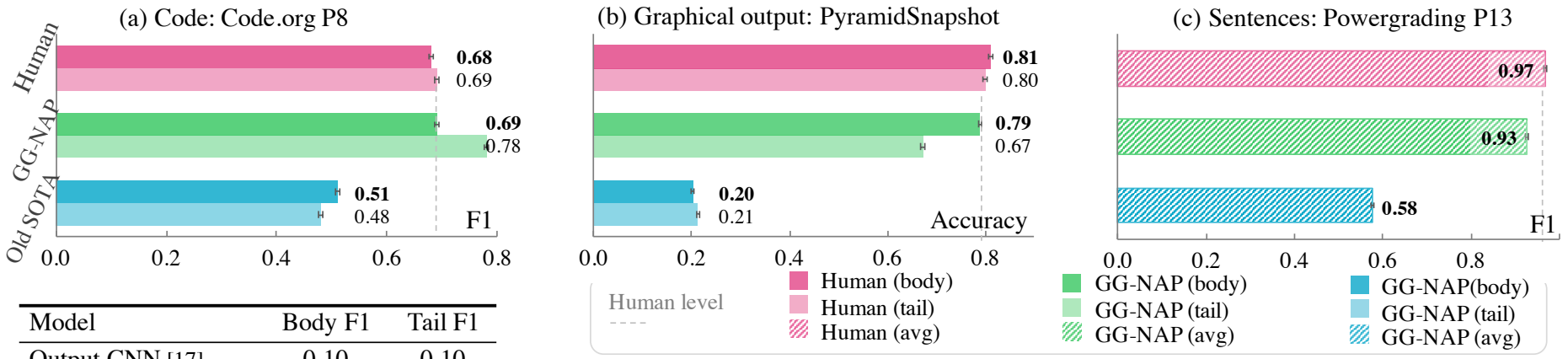


Generative Grading: Near Human-level Accuracy for Automated Feedback on Richly Structured Problems
Ali Malik, Mike Wu, Vrinda Vasavada, Jinpeng Song, Madison Coots, John Mitchell, Noah Goodman, Chris Piech

Generative Grading



Generative Grading



Model	Body F1	Tail F1
Output CNN [17]	0.10	0.10
Program RNN [11]	0.27	0.22
MVAE [16]	0.38	0.26
Rubric Sampling [17]	0.51	0.48
GG-LSH	0.31	0.33
GG-NAP	**0.69**	**0.78**
Human	0.68	0.69

Model	Body F1	Tail F1
kNN [??]	0.20	0.12
NeuralNet [??]	0.20	0.21
GG-kNN	timeout	timeout
GG-NAP	**0.79**	**0.67**
Human	0.81	0.80

Model	Avg F1	Tail Acc
Handcrafted [5]	0.58	-
T&N Best [12]	0.55	-
GG-kNN	0.78	0.63
GG-NAP	**0.93**	**0.76**
Human	0.97	0.90

Give Feedback to Open Ended Work?

Human Level

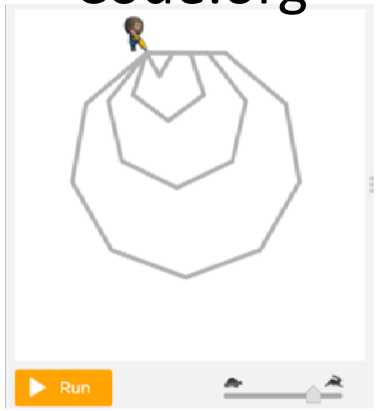
Simple Algebra

$$
\begin{aligned}
&9 \times 6 \\
&= (10 - \square) \times 6 \\
&= 10 \times 6 - \square \times 6 \\
&= 60 - \square \\
&= \square
\end{aligned}
$$

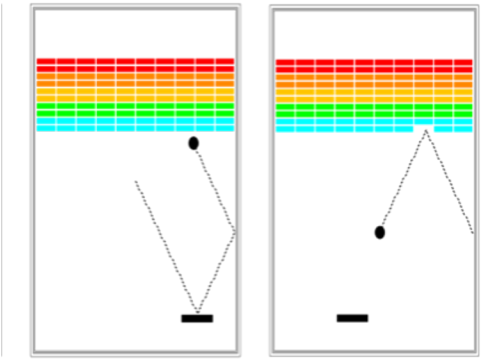
Short Answer

Why did the original colonists come to America?

Code.org



Stanford Midterm



Deep Learning Circa 2019

Generative Grading

Too hard?

Midterm Grading Challenge

Question

Rock Paper Scissors (30 points)

Write a `ConsoleProgram` that has a user play rock paper scissors against a computer until either the user or the computer has **three** "wins". To make the code simpler, use integers to represent the different plays (0 is rock, 1 is paper, 2 is scissors). Example run:

Student Answer

```
1  public class RockPaperScissors extends ConsoleProgram {
2
3      /* constants */
4      private static final int ROCK = 0;
5      private static final int PAPER = 1;
6      private static final int SCISSORS = 2;
7      private static final int N_WINS = 3;
8
9      private RandomGenerator rg = new RandomGenerator();
10
11     public void run() {
12         introMessage();
13         for (int i = 0; i < N_WINS; i++) {
14             inputNumber;
15             roundWinner;
16         }
17         gameWinner;
18     }
19
20     private void introMessage() {
21         println("Rock Paper Scissors!");
22         println("0) Rock");
23         println("1) Paper");
24         println("2) Scissors");
25         println(" ");
26     }
27
28     private void inputNumber() {
29         int a = readInt("Your move: ");
30         int computerMove = rg.nextInt (3);
31         int b = readInt("Computer move: " + computerMove);
32
33     private int computer = 0;
34     private int human = 0;
35
36     private void roundWinner {
37         if (a == 1 && b == 0) {
38             println("Human wins.");
```

Feedback

Grade: 26/30 points ✔ grade submitted!

Rubric: Standard

- For loop instead of a while loop (should go up to 3 wins by computer or user not 3 games)

User Inputting Move (3 points)

- (•) Perfect (0 points)
- ( ) Minor Error (1 points)
- ( ) Major Error (2 points)
- ( ) Totally Wrong (3 points)

Computer Generating Move (4 points)

- (•) Perfect (0 points)
- ( ) Minor Error (1 points)
- ( ) Major Error (2 points)
- ( ) Major Errors/No Attempt (4 points)

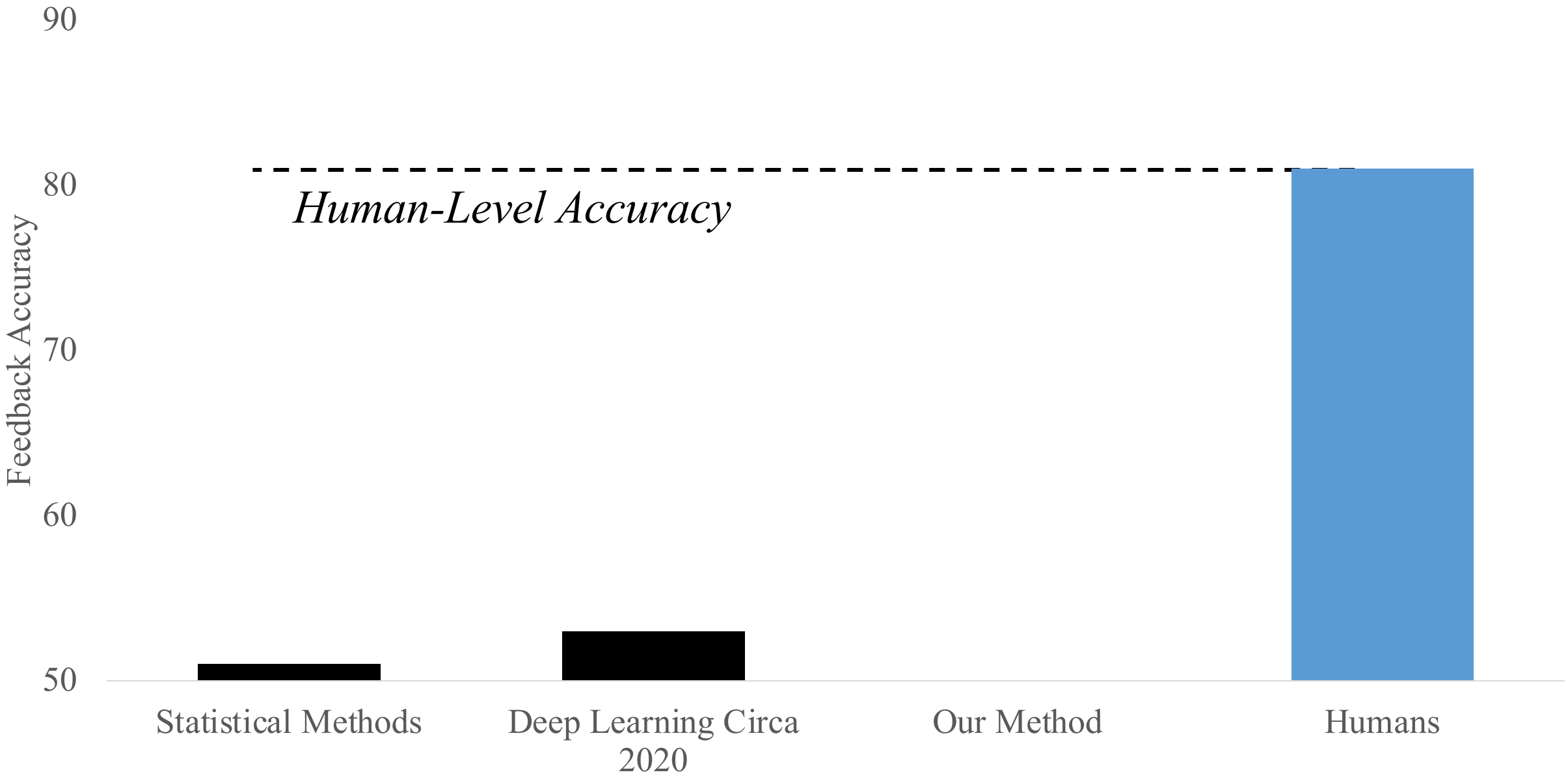
Determining the winner of a round (6 points)

- (•) Perfect (0 points)
- ( ) Minor Error (1 points)
- ( ) Major Error (2 points)

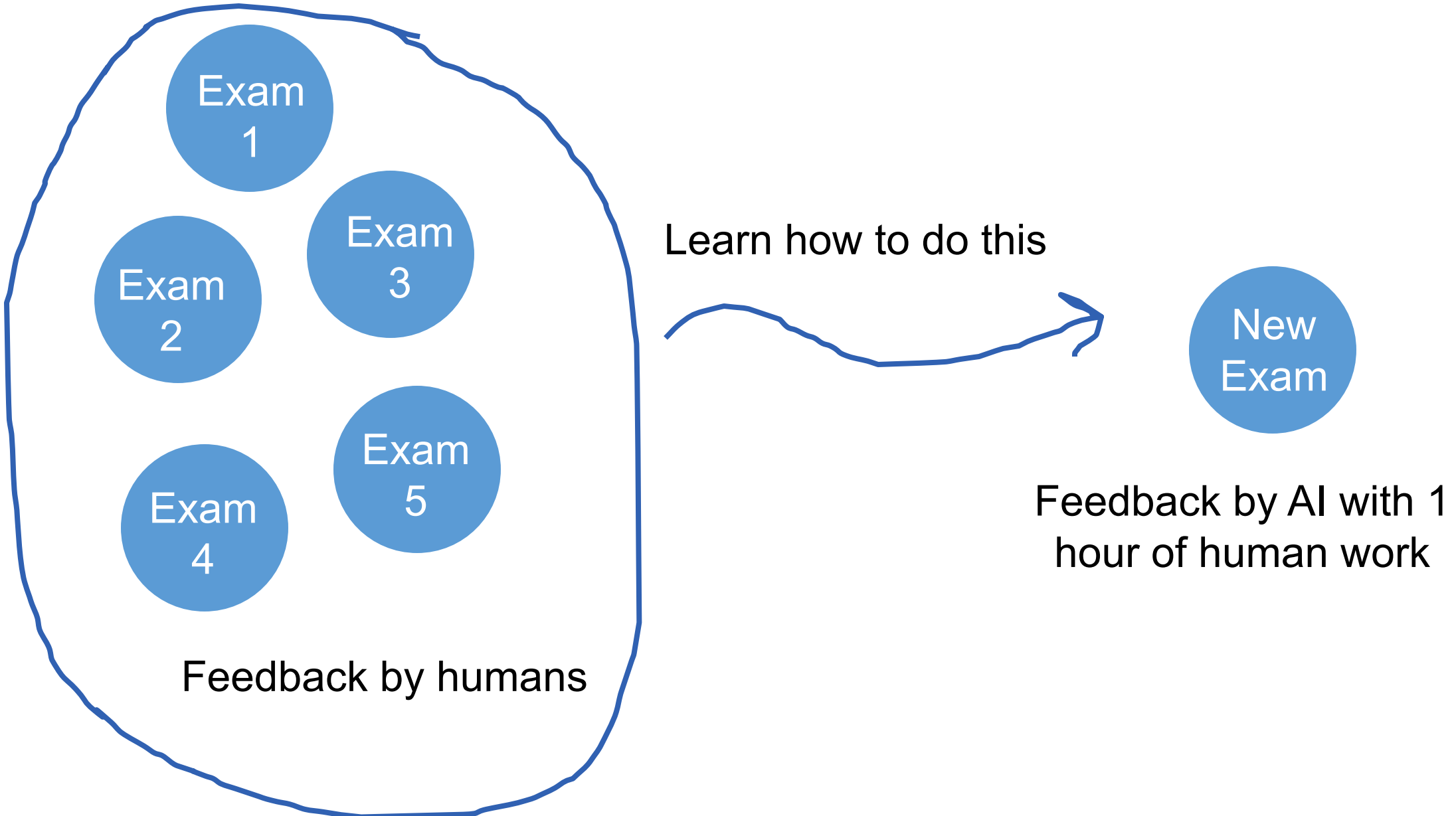
Exam / Question 3 / Student

‹ Previous | Submit Grade | Next ›

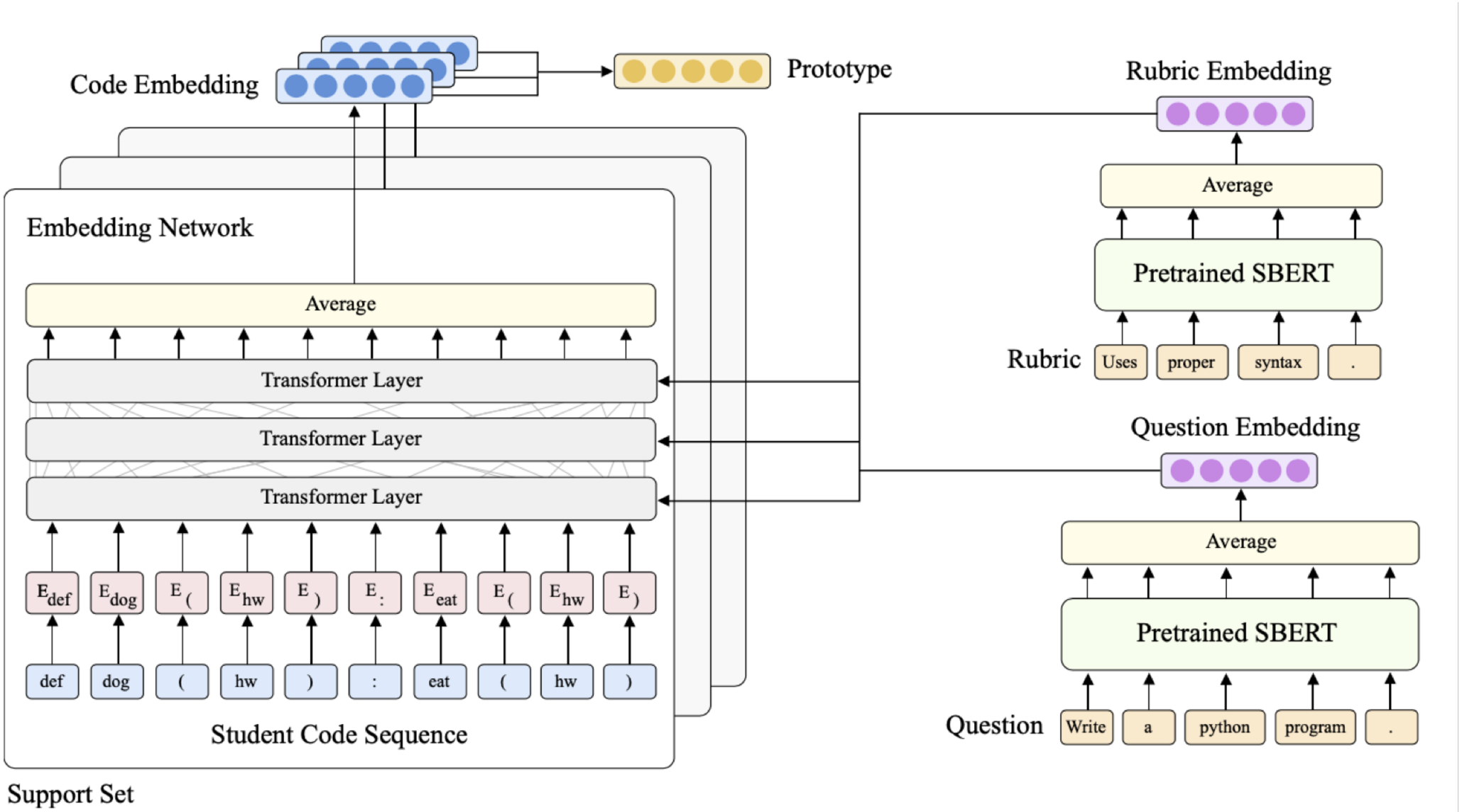
Rubric Level Accuracy on Few-Shot Grading a Novel Question



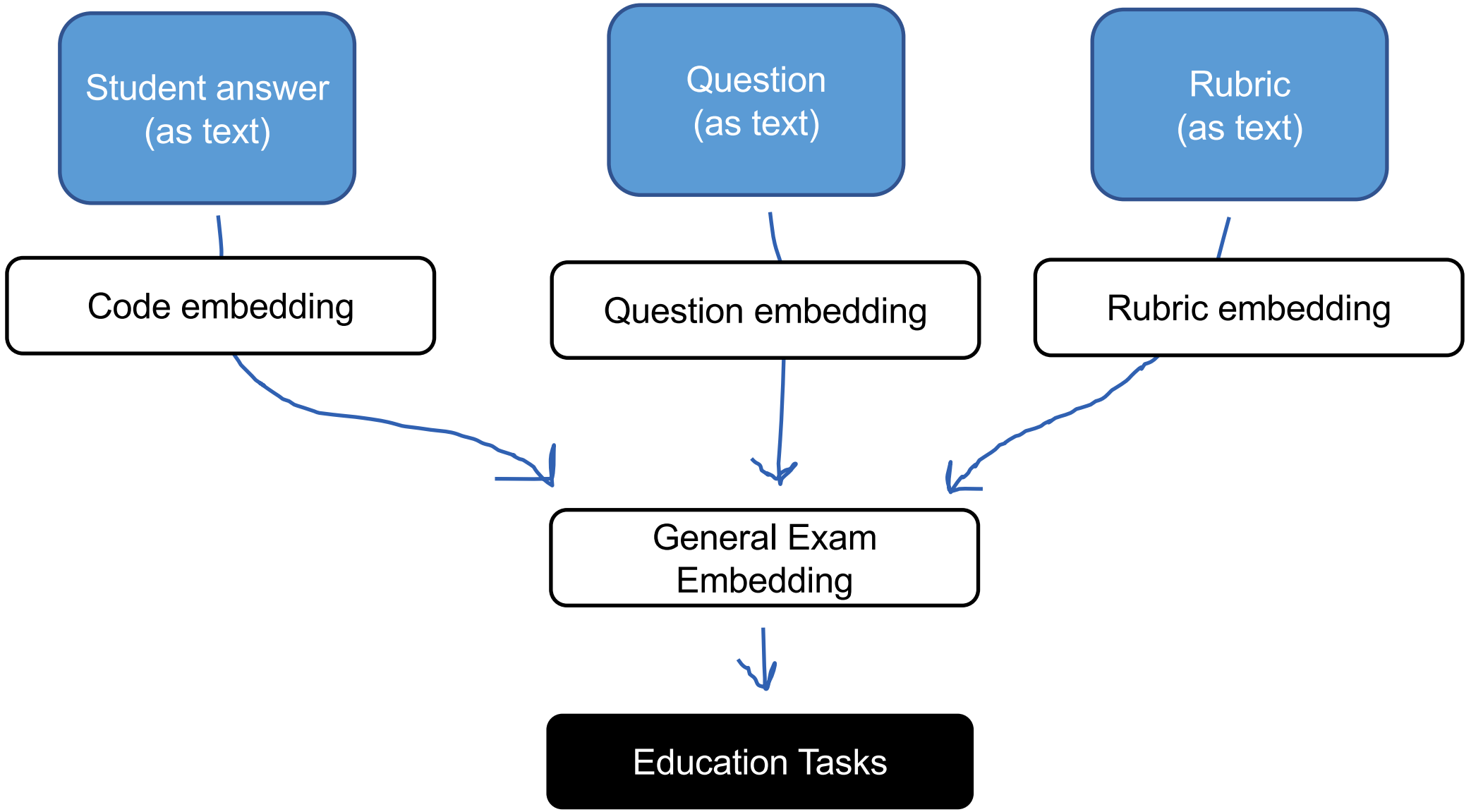
Give Feedback on Fresh Stanford Midterm



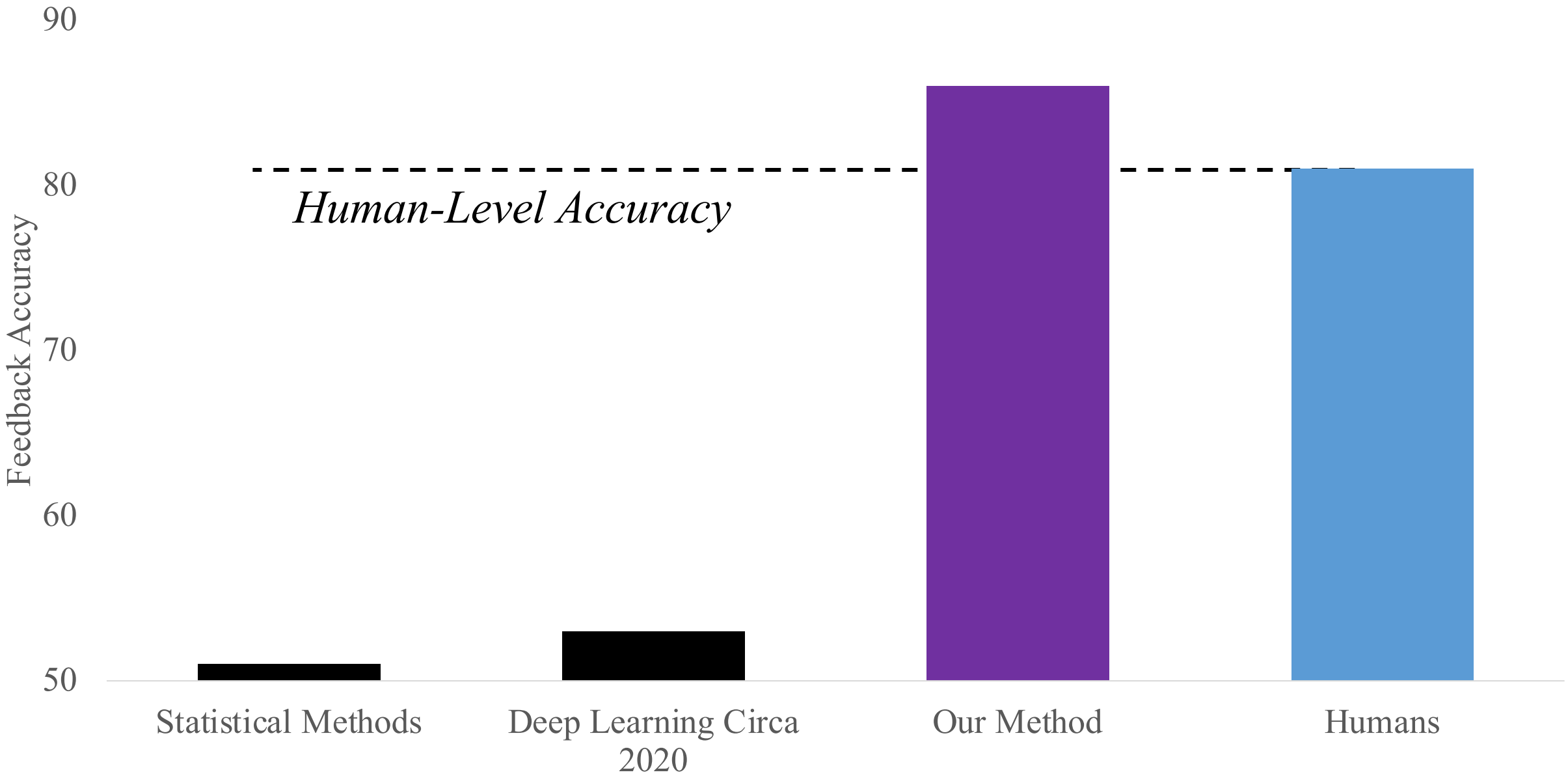
Invented the Proto-Transformer



General Exam Grading Model

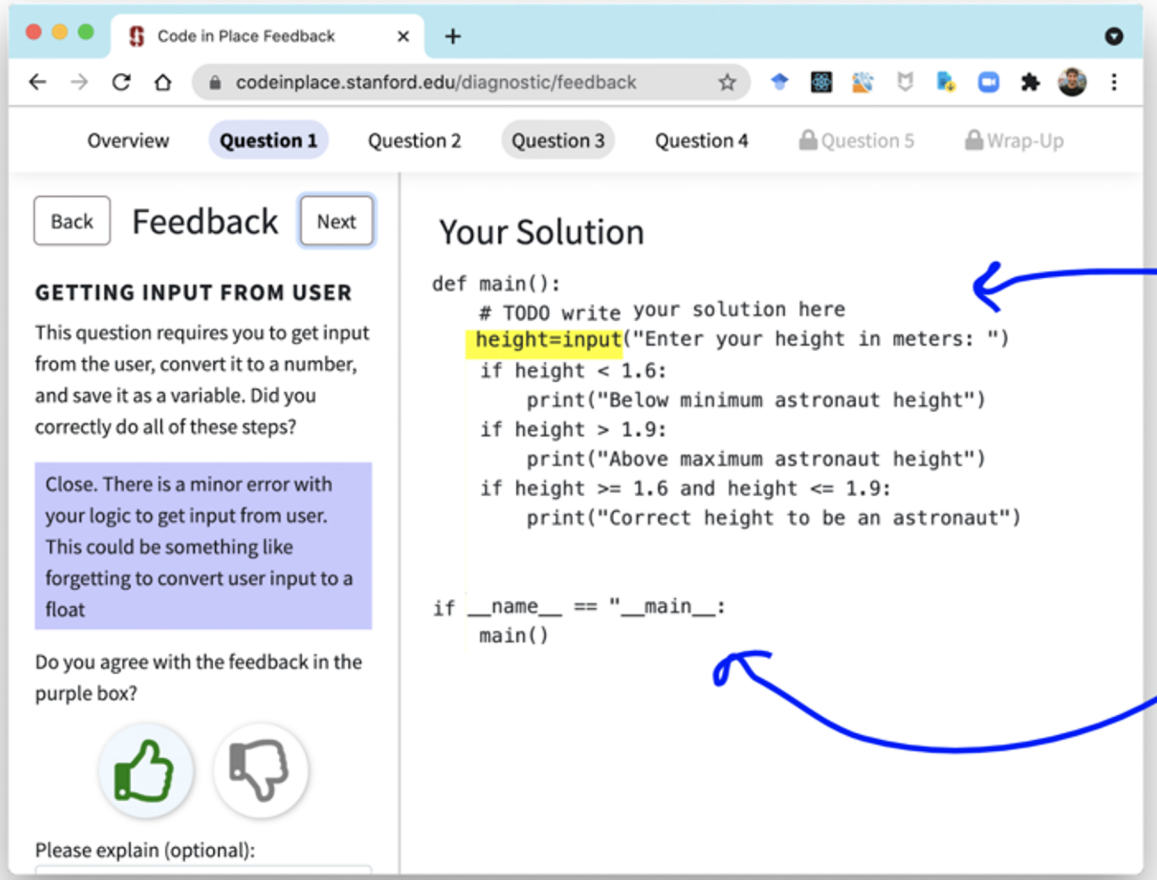


Rubric Level Accuracy on Few-Shot Grading a Novel Question



Gave Feedback to 3,500 Real Students

Do you agree? AI feedback **97.9%**. Human feedback 96.7%

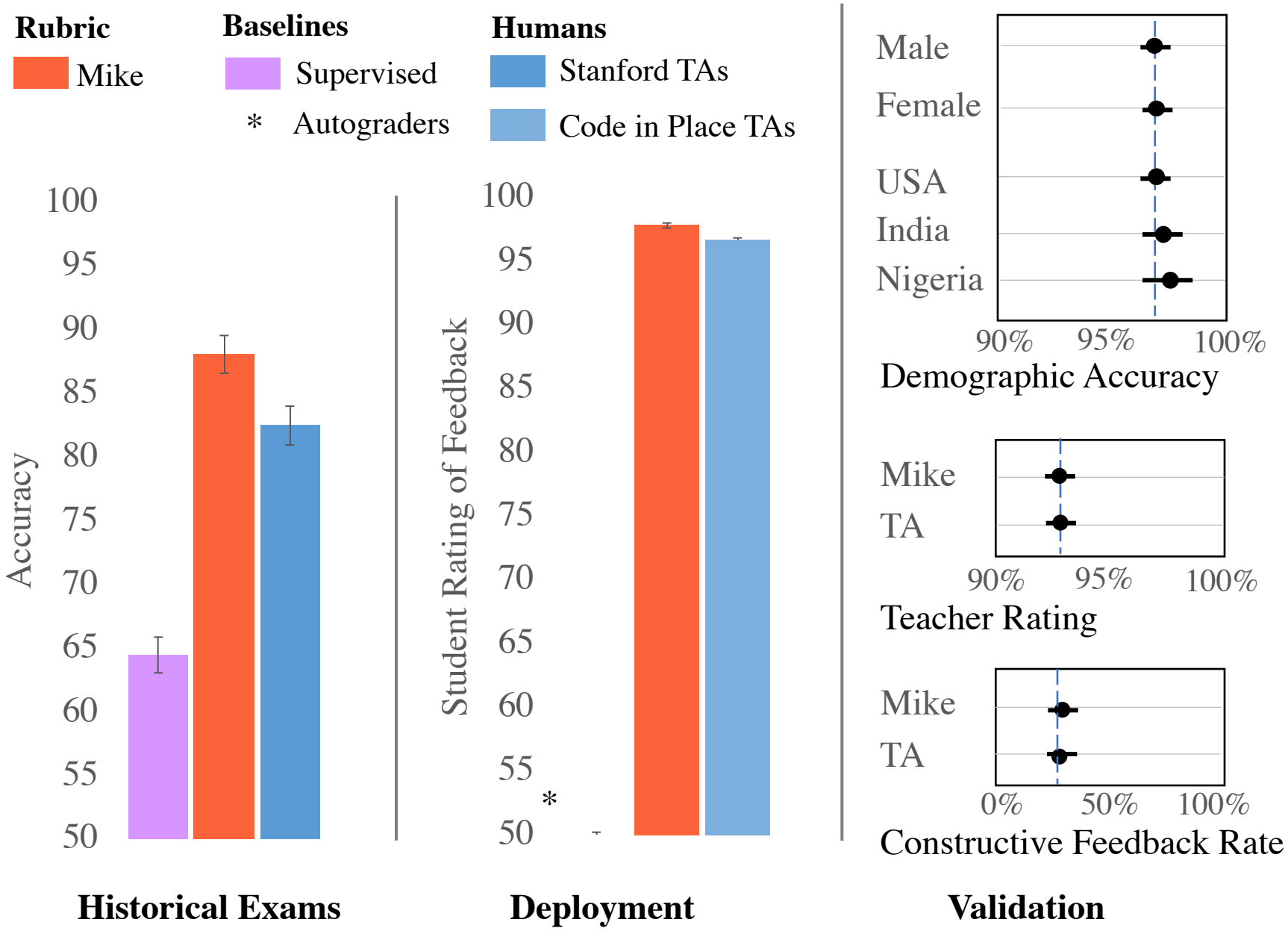


AI generated feedback

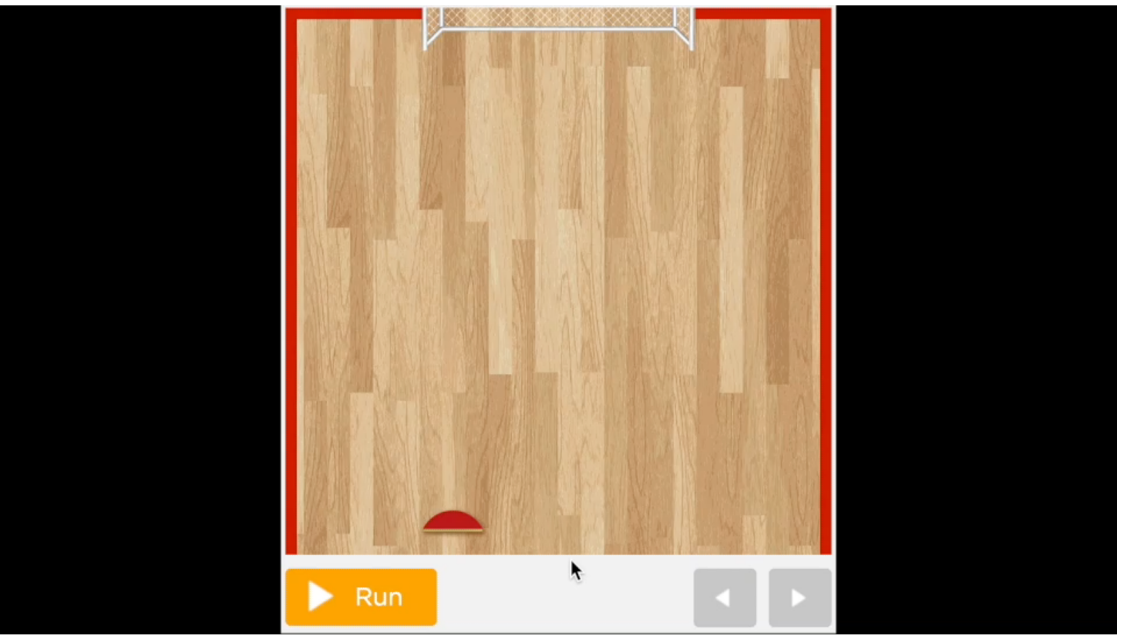
Students evaluate the feedback

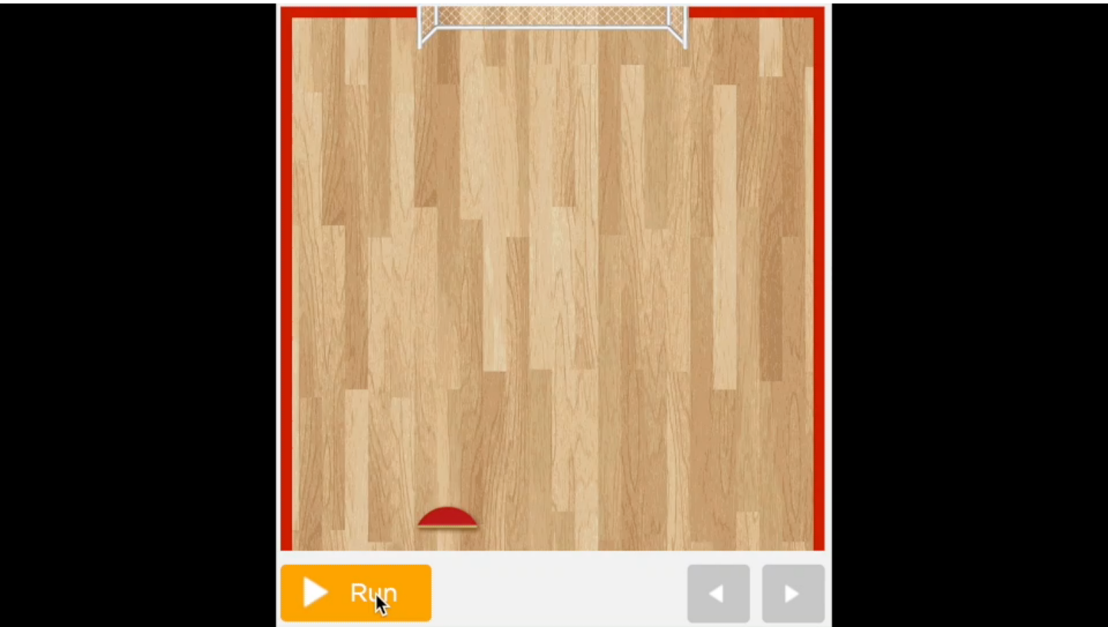
Algorithm uses attention to highlight where in the code the error comes from

Syntax error (missing ") here would prevent auto graders from being useful.


Rubric
Mike
Baselines
Supervised
* Autograders
Humans
Stanford TAs
Code in Place TAs
Accuracy
100
95
90
85
80
75
70
65
60
55
50
Historical Exams
Student Rating of Feedback
100
95
90
85
80
75
70
65
60
55
50
*
Deployment
Male
Female
USA
India
Nigeria
90%
95%
100%
Demographic Accuracy
Mike
TA
90%
95%
100%
Teacher Rating
Mike
TA
0%
50%
100%
Constructive Feedback Rate
Validation

But what about interactive, creative assignments?



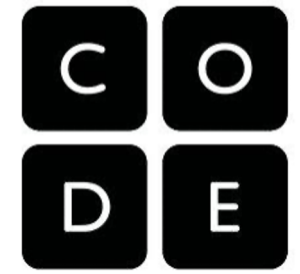


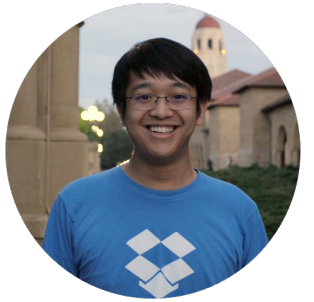
1M ungraded code.org assignments.

The AI is shown a brand new student game. Does it work?

Simultaneously learn to grade and play to grade.

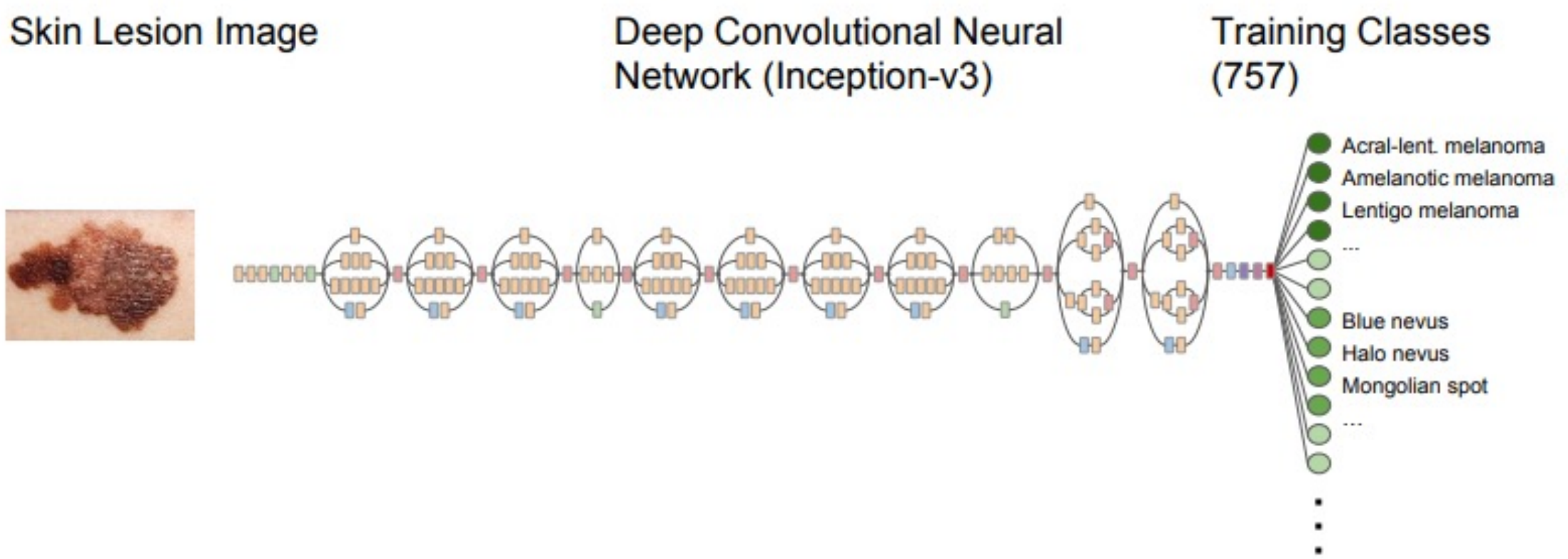
Majority class: 50%
Code-as-text: 67%
Play-to-grade: **94%**





Piech

Detecting skin cancer



Esteva, Andre, et al. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* 542.7639 (2017): 115-118.