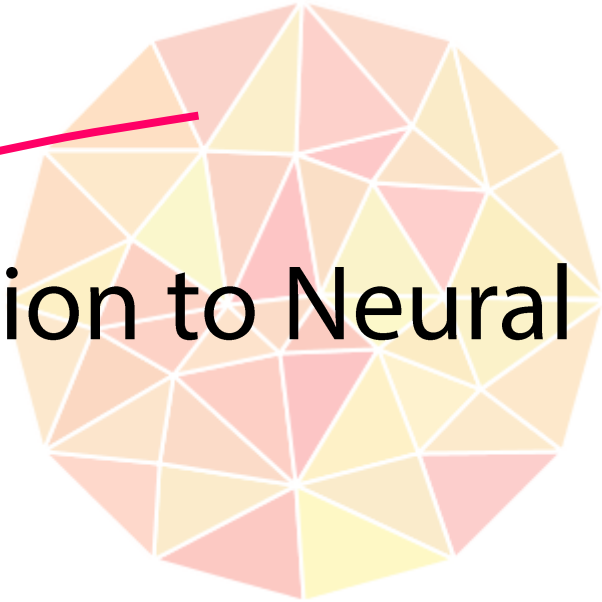
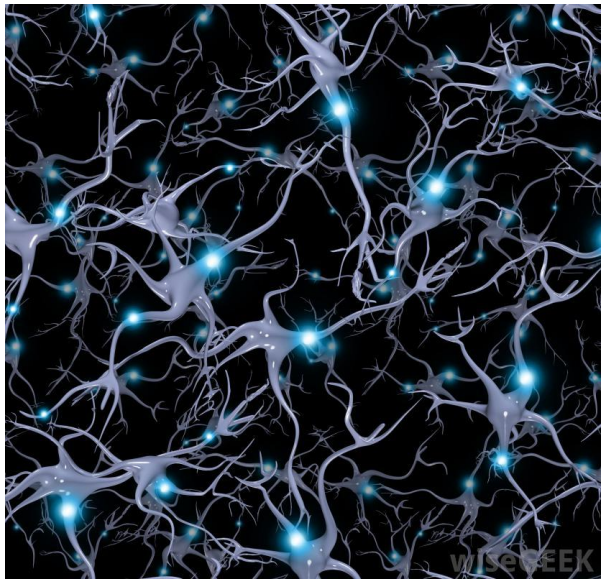
A Quick and Dirty

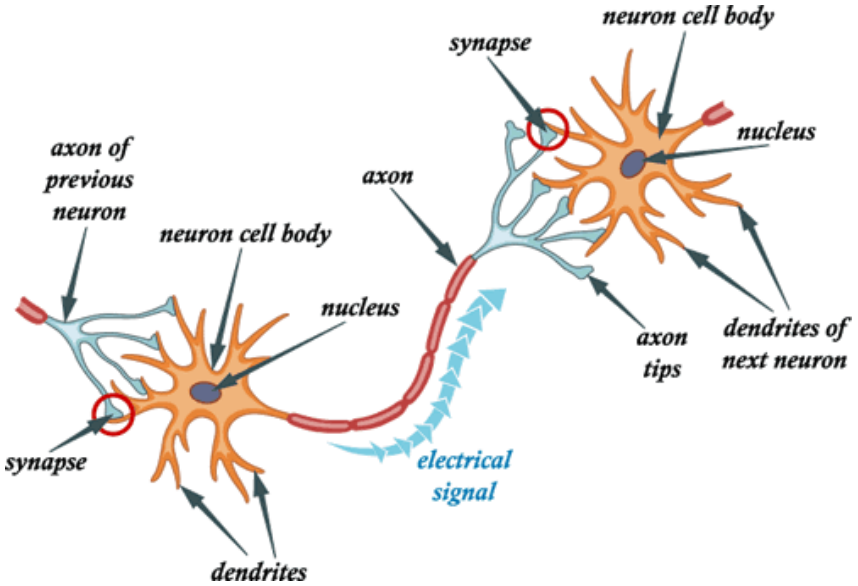


Introduction to Neural Networks

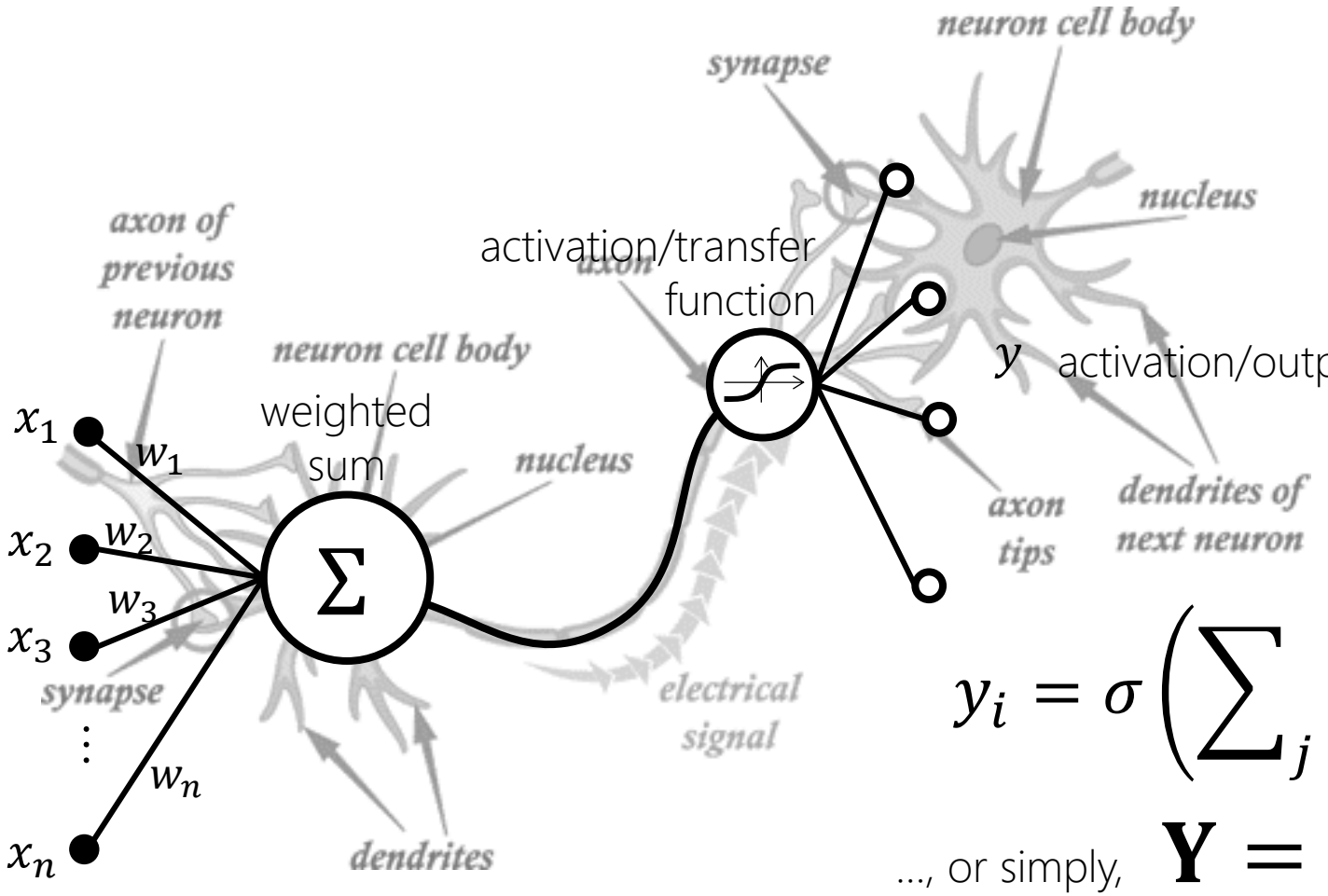
Stephen Baek

Biological Neural Networks





Artificial Neural Networks (ANN)



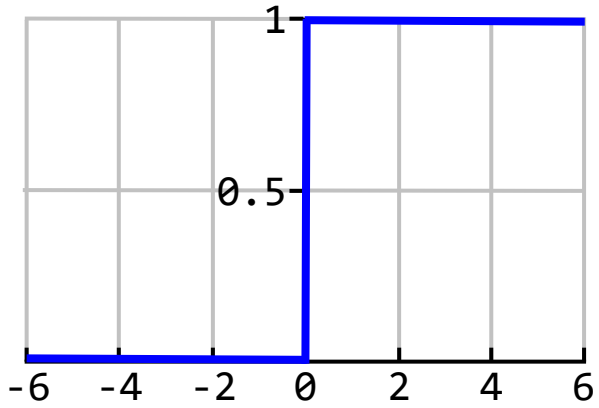
$$y_i = \sigma\left(\sum_j w_{i,j} x_j\right)$$

..., or simply, $\mathbf{Y} = \sigma(\mathbf{XW})$

Activation/Transfer Functions

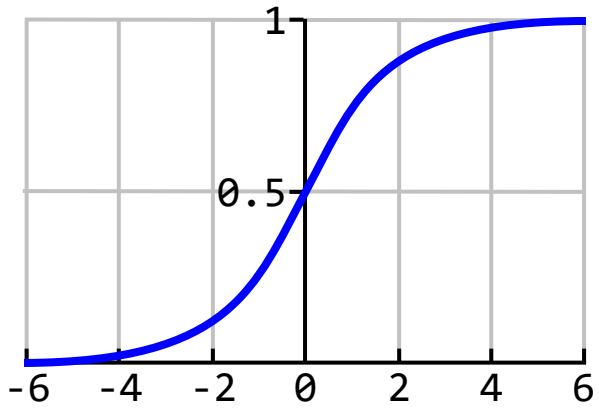
- "Threshold" in biological systems
- Step Function:

$$y = \sigma(z) = \begin{cases} 1 & \text{if } u \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



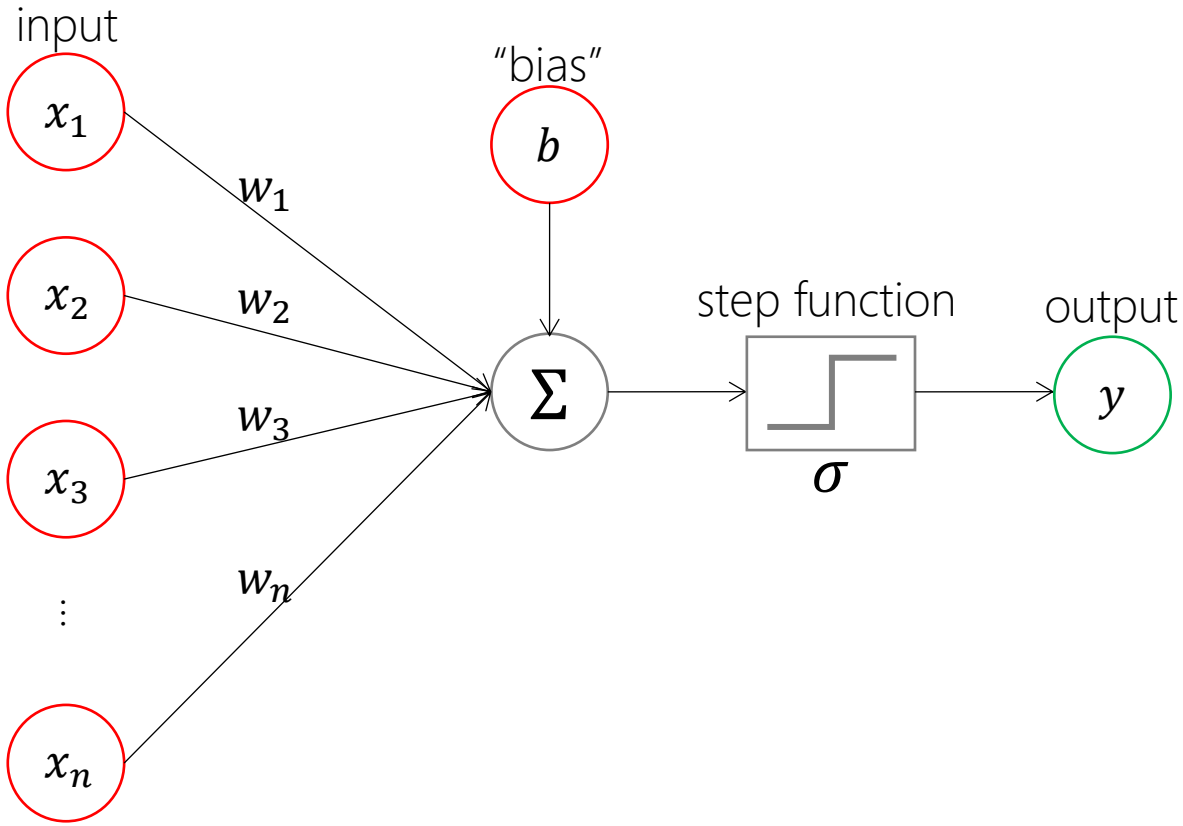
- Sigmoid Function:

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



The simplest model: the "Perceptron"

- Frank Rosenblatt (1957)



$$y = \sigma\left(\sum_j w_j x_j + b\right)$$

For each example $\left(x^{(t)}, \hat{y}^{(t)}\right)$, try to minimize:

$$\mathcal{L} = \frac{1}{2}\left(y^{(t)} - \hat{y}^{(t)}\right)^2$$

$$\frac{\partial \mathcal{L}}{\partial w_j} = \left(y^{(t)} - \hat{y}^{(t)}\right)\frac{\partial y^{(t)}}{\partial w_j}$$

$$= \left(y^{(t)} - \hat{y}^{(t)}\right)x_j^{(t)}$$

Weight update scheme:

$$w(t+1) = w(t) - r\left(y^{(t)} - \hat{y}^{(t)}\right)x_j^{(t)}$$

learning rate

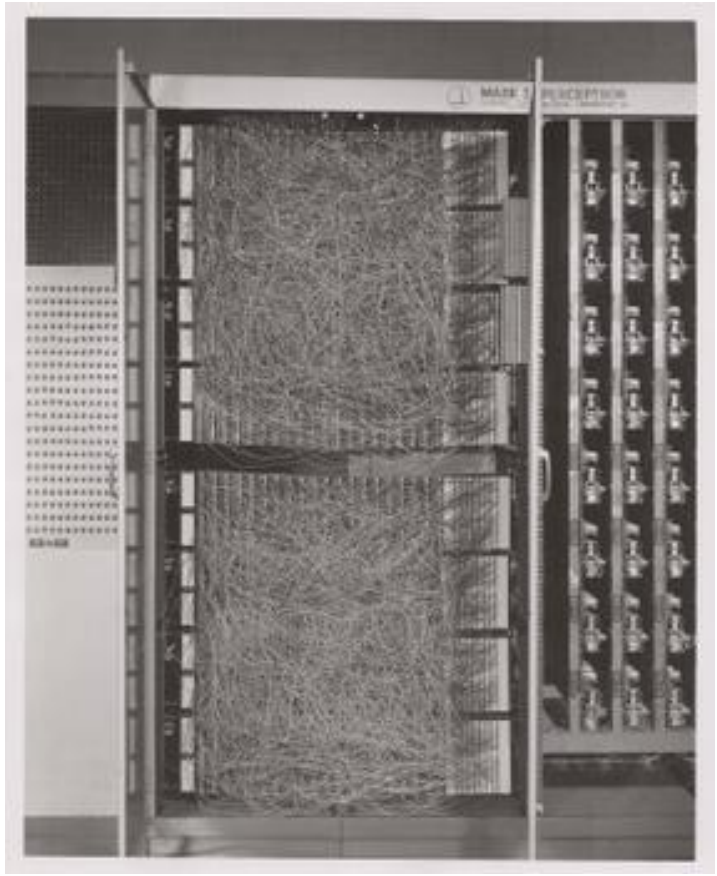
The simplest model: the “Perceptron”

- Frank Rosenblatt (1957)
  - The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.
  - Designed for image recognition.
  - Connected to a camera that used a 20x20 cadmium sulfide photocell array to produce a 400-pixel image.

http://www.rutherfordjournal.org/article040101.html

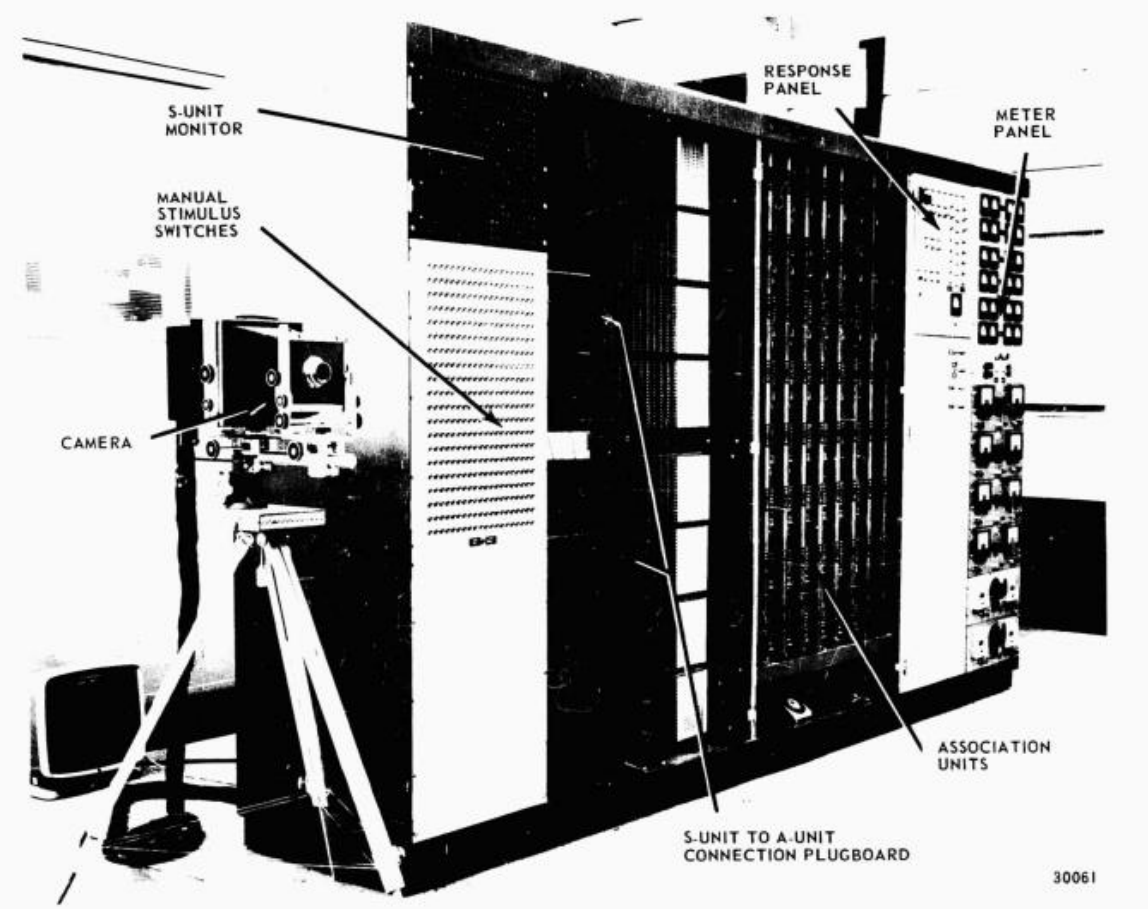
The simplest model: the “Perceptron”

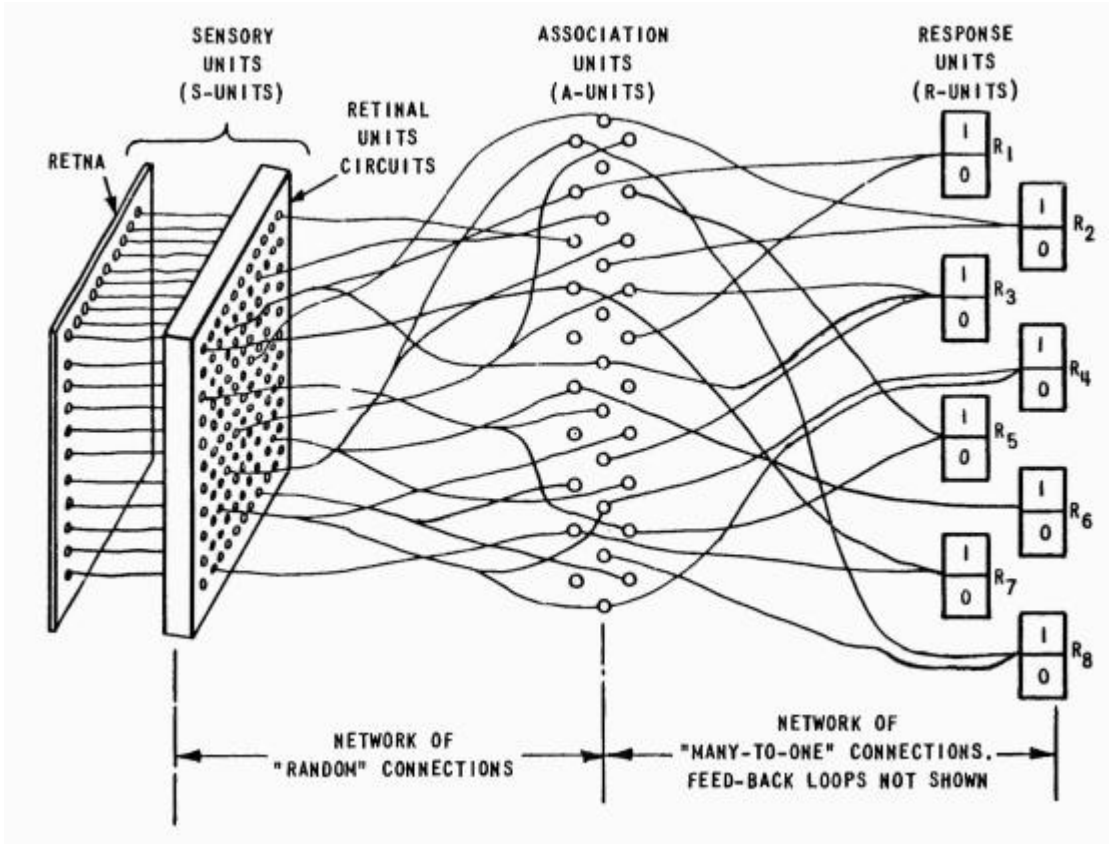
- Frank Rosenblatt (1957)
  - The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.
  - Designed for image recognition.
  - Connected to a camera that used a 20x20 cadmium sulfide photocell array to produce a 400-pixel image.
  - Weights were encoded in potentiometers.
  - Weight updates were performed by electric motors.



Wikipedia – Perceptron

The simplest model: the “Perceptron”





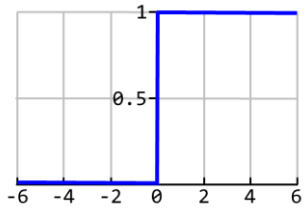
Mark 1 Perceptron Operators’ Manual (http://www.dtic.mil/dtic/tr/fulltext/u2/236965.pdf)

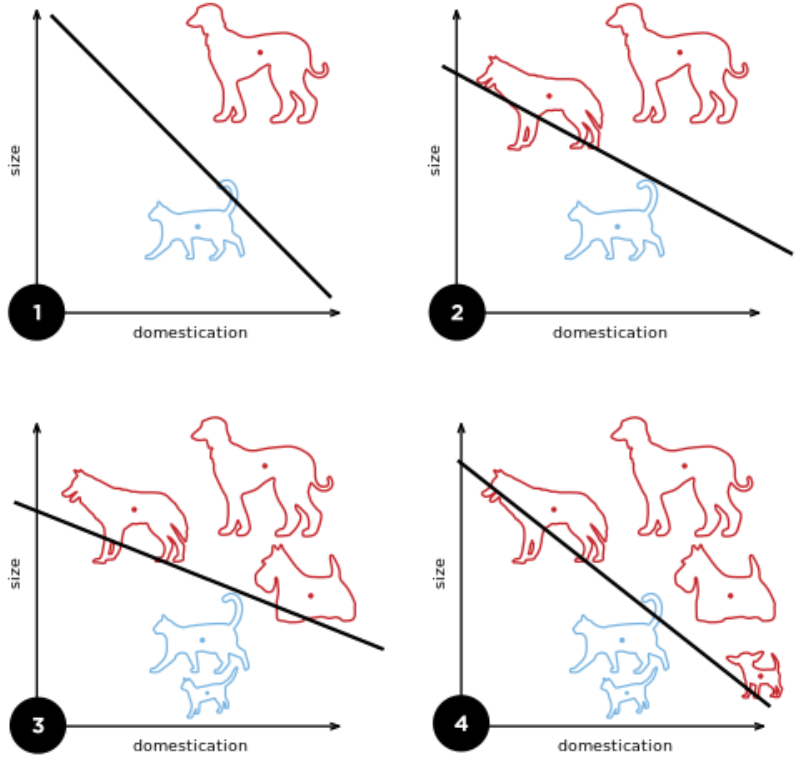
Perceptron as Linear Binary Classification

- Perceptron returns 1 iff. $\sum_j w_j x_j + b \geq 0$ and 0 otherwise.
- Essentially, this is equivalent to linear binary classification problem

$$z = \sum_j w_j x_j + b$$

$$y = \sigma(z) = \begin{cases} 1 & \text{if } u \geq 0 \\ 0 & \text{otherwise} \end{cases}$$





Wikipedia – Perceptron

The XOR Problem

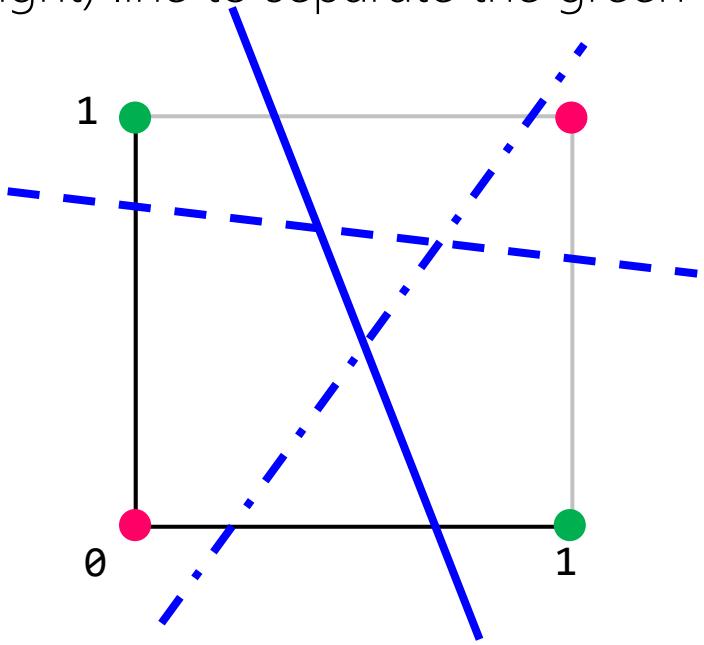
- XOR: Exclusive OR:

x_1	x_2	y
0	0	0
0	1	1
1	1	0
1	0	1

- Q. Can you implement the XOR gate with the perceptron model?

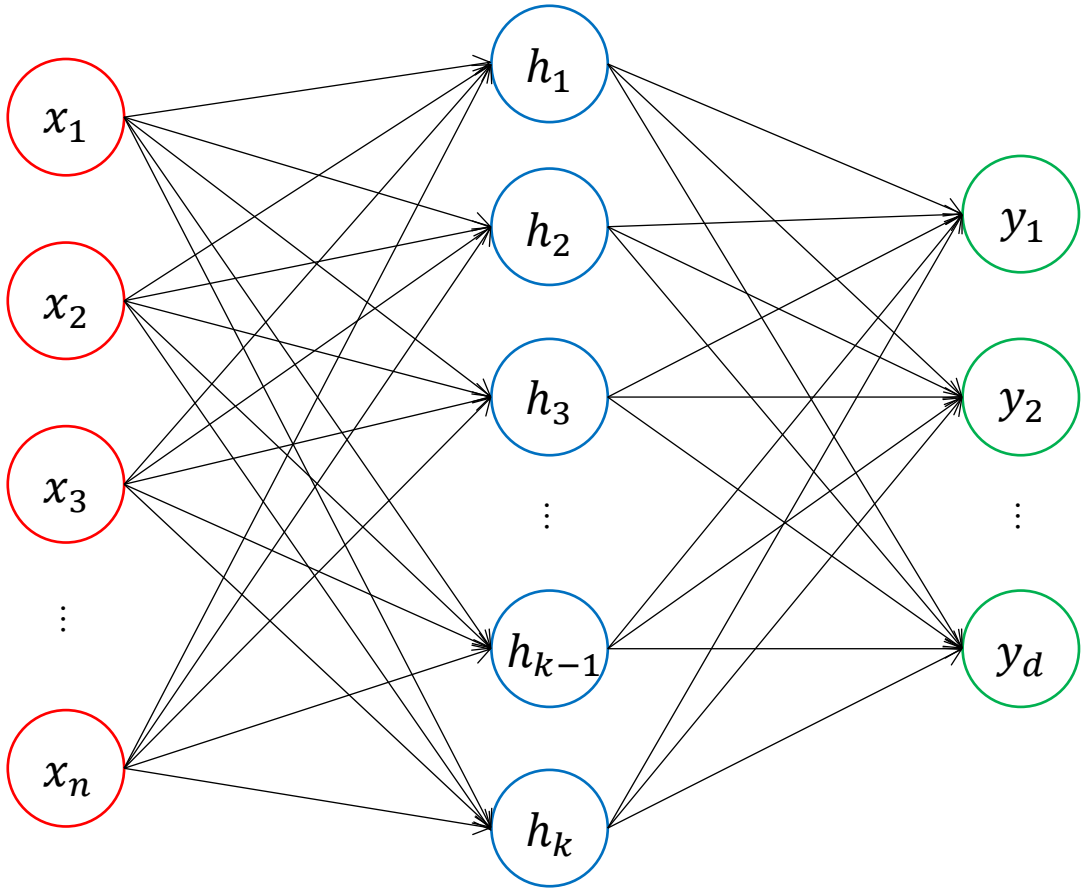
The XOR Problem

- The quick answer is NO.
- Why?
  - a perceptron is a linear classifier.
  - Can you draw a single (straight) line to separate the green and red dots?



Multi-Layer Perceptron (MLP)

- Minsky & Papert. (1969)

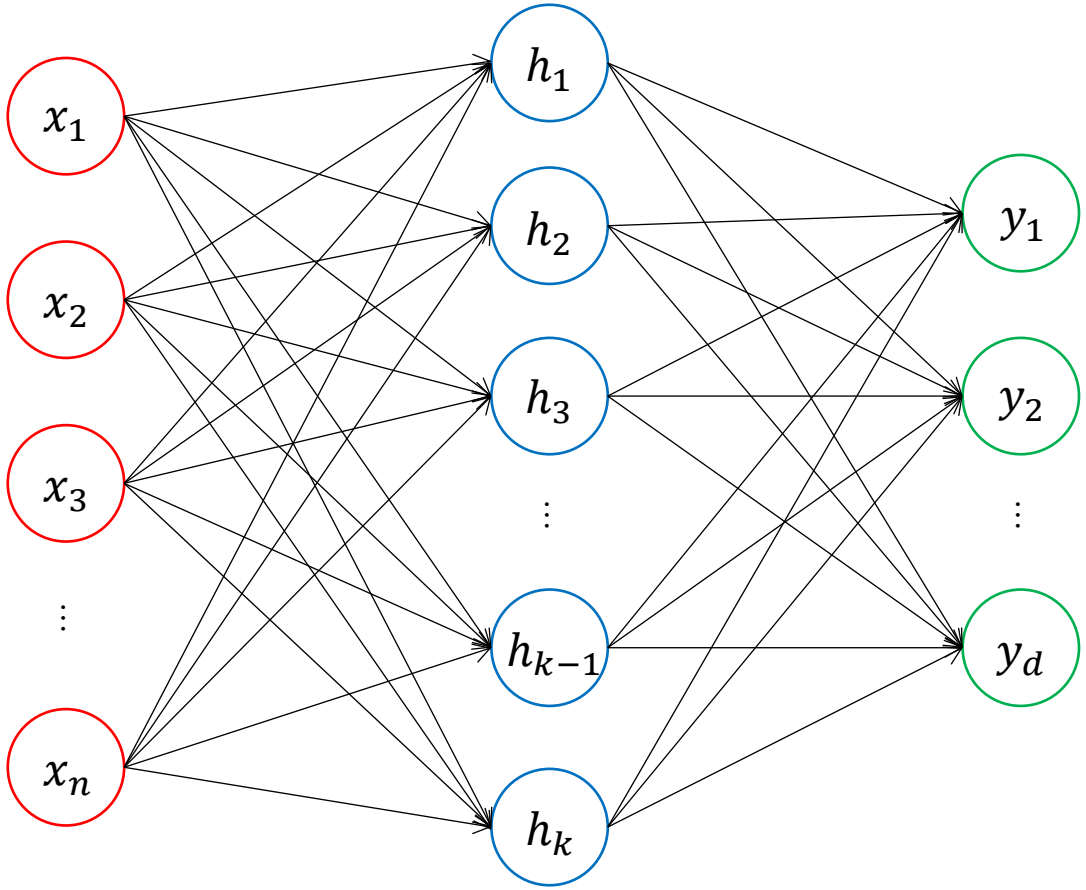


The perceptron has shown itself worthy of study despite (and even because of!) its severe limitations. It has many features to attract attention: its linearity; its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. There is no reason to suppose that any of these virtues carry over to the many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension to multilayer systems is sterile.

Minsky & Papert (1969, pp. 231-232)

Multi-Layer Perceptron (MLP)

- Minsky & Papert. (1969)



$$h_i = \sigma_h\left(\sum_j w_{h,i,j}x_j + b_h\right) \quad y_k = \sigma_o\left(\sum_i w_{o,k,i}h_i + b_o\right)$$

$$\mathcal{L} = \frac{1}{2}\left\|\mathbf{y}^{(t)} - \hat{\mathbf{y}}^{(t)}\right\|^2$$

$$\frac{\partial \mathcal{L}}{\partial w} = ?$$

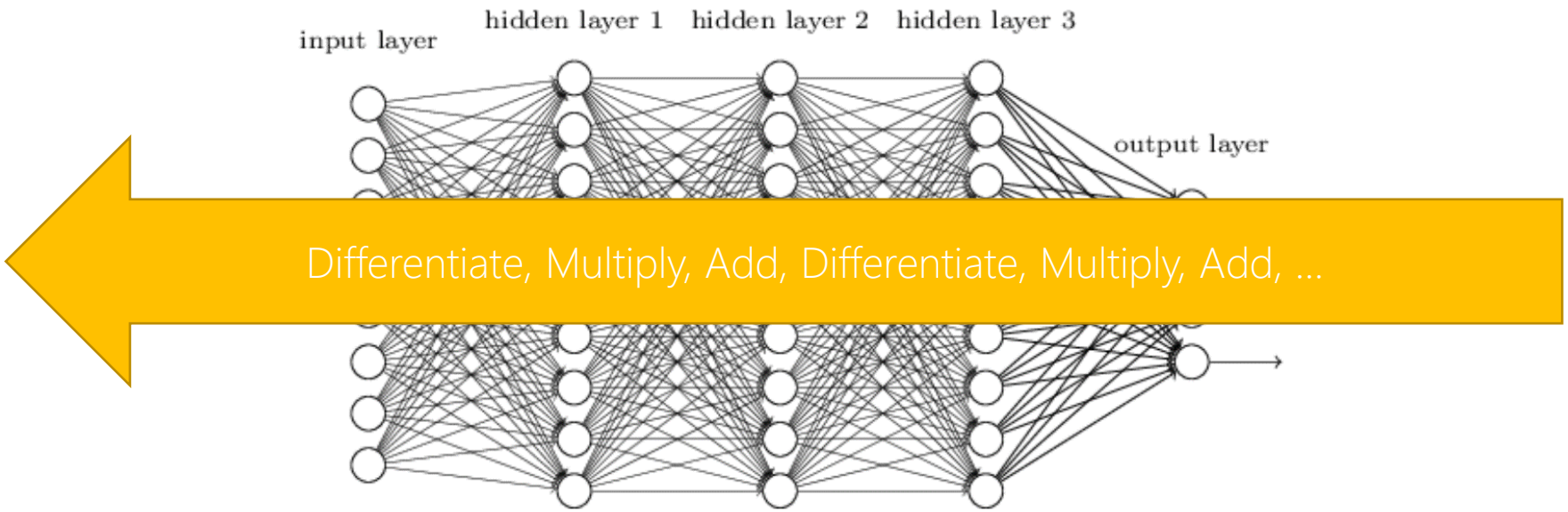
- M&P's criticism on (single-layer) perceptrons
- No practical way to train MLPs

→ The First AI Winter (1970s)


A FEW
MOMENTS LATER

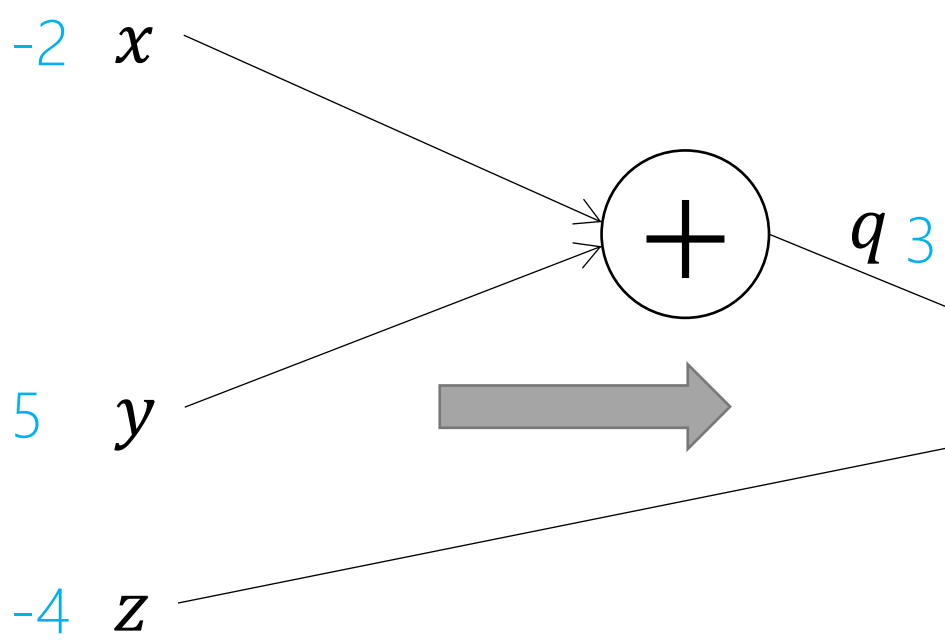
Backpropagation

- Rumelhart, Hinton, and Williams. (1986).
- A popular training method for neural nets
- Propagate what? “the gradient of the current error”



Backpropagation

- A simple example: $f(x, y, z) = (x + y)z$
  - Computational graph:



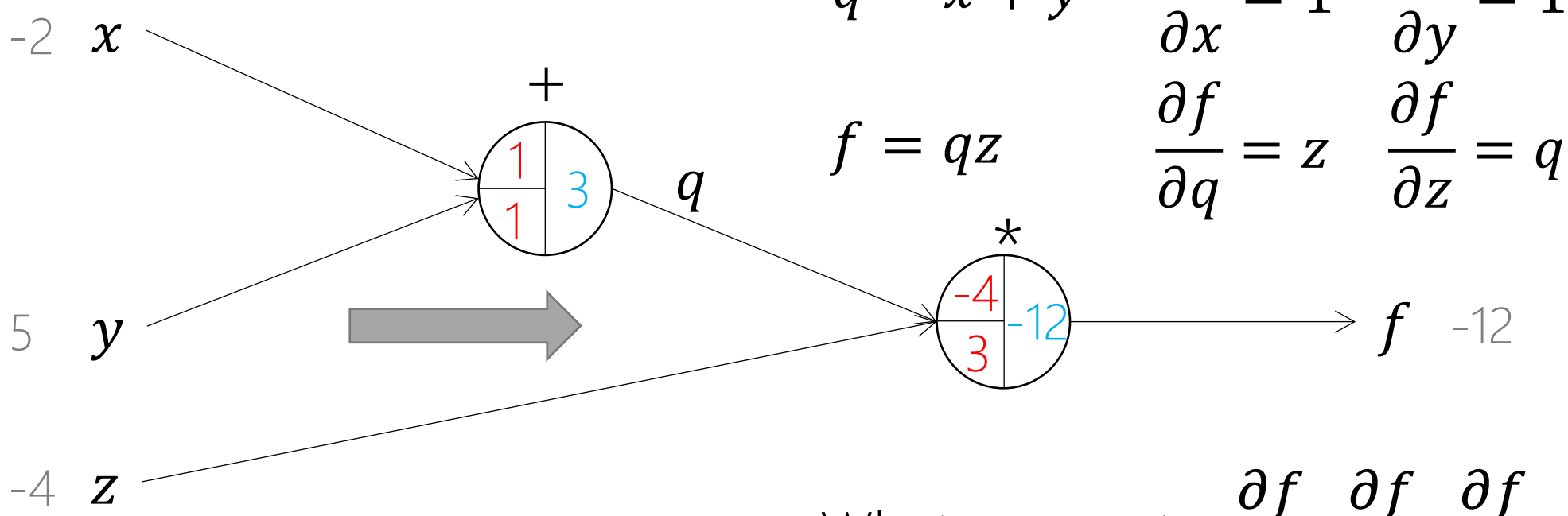
$$q = x + y \quad \frac{\partial q}{\partial x} = 1 \quad \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z \quad \frac{\partial f}{\partial z} = q$$

What we want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation

- A simple example: $f(x, y, z) = (x + y)z$
  - Computational graph:



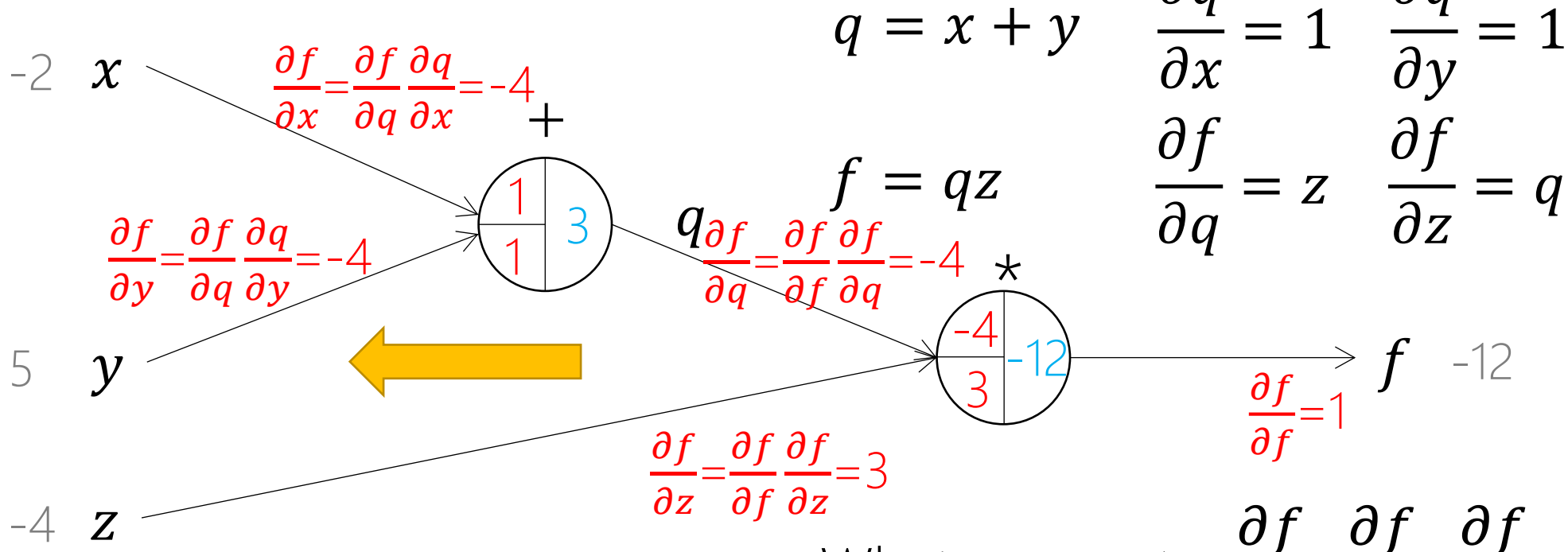
$q = x + y$ $\quad \frac{\partial q}{\partial x} = 1$ $\quad \frac{\partial q}{\partial y} = 1$

$f = qz$ $\quad \frac{\partial f}{\partial q} = z$ $\quad \frac{\partial f}{\partial z} = q$

What we want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation

- A simple example: $f(x, y, z) = (x + y)z$
  - Computational graph:

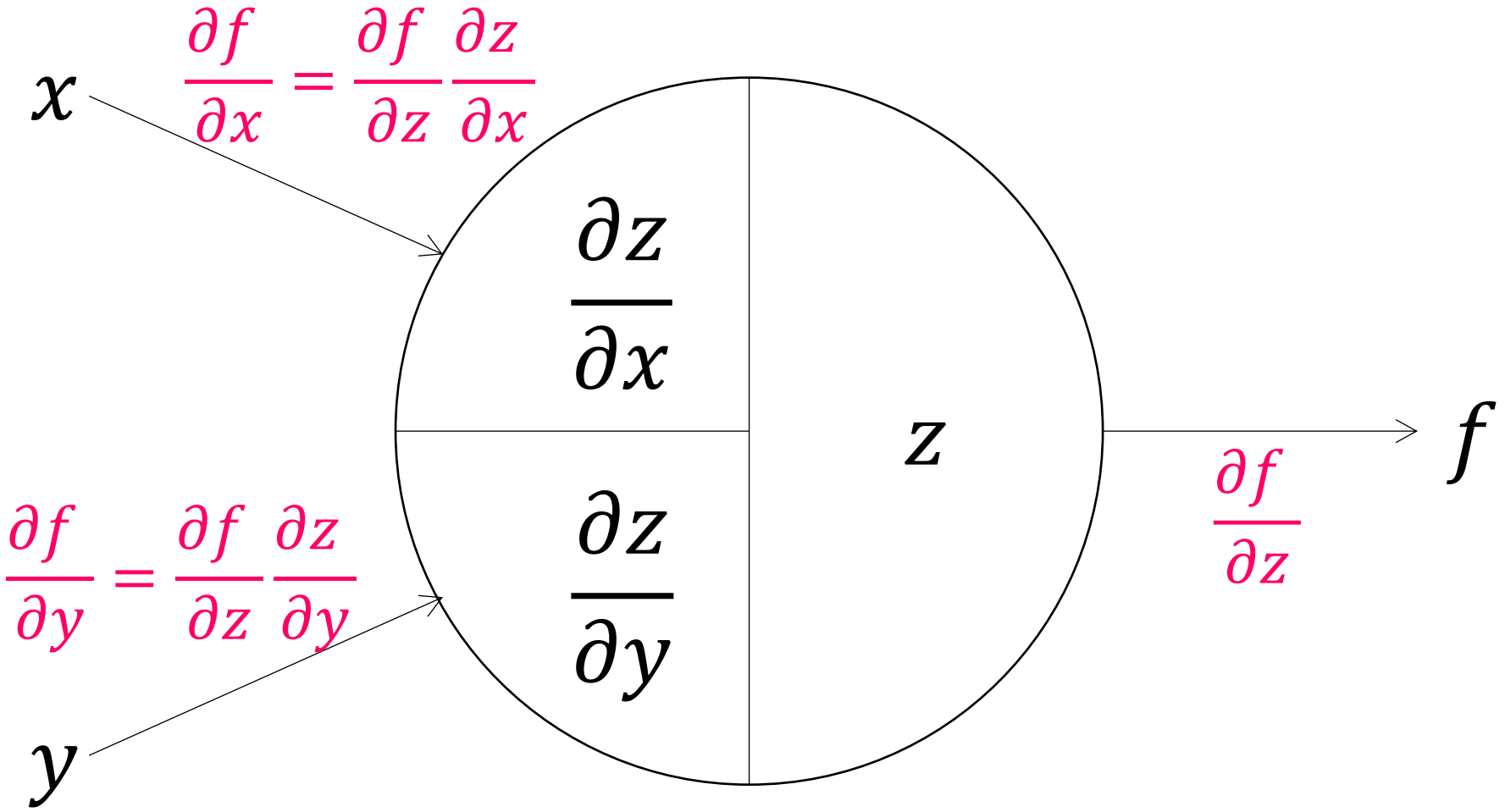


$q = x + y$ $\quad \frac{\partial q}{\partial x} = 1$ $\quad \frac{\partial q}{\partial y} = 1$

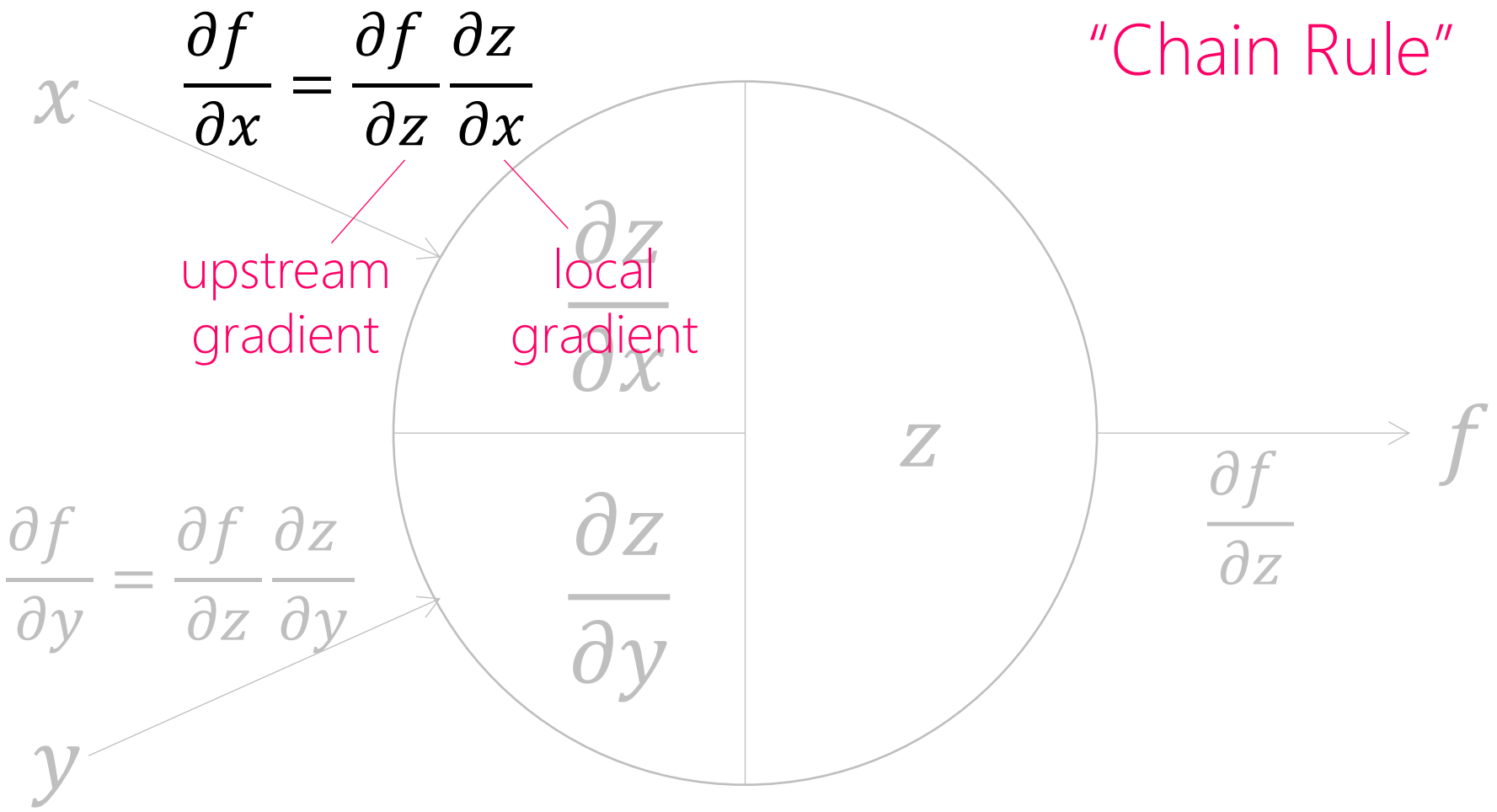
$f = qz$ $\quad \frac{\partial f}{\partial q} = z$ $\quad \frac{\partial f}{\partial z} = q$

What we want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

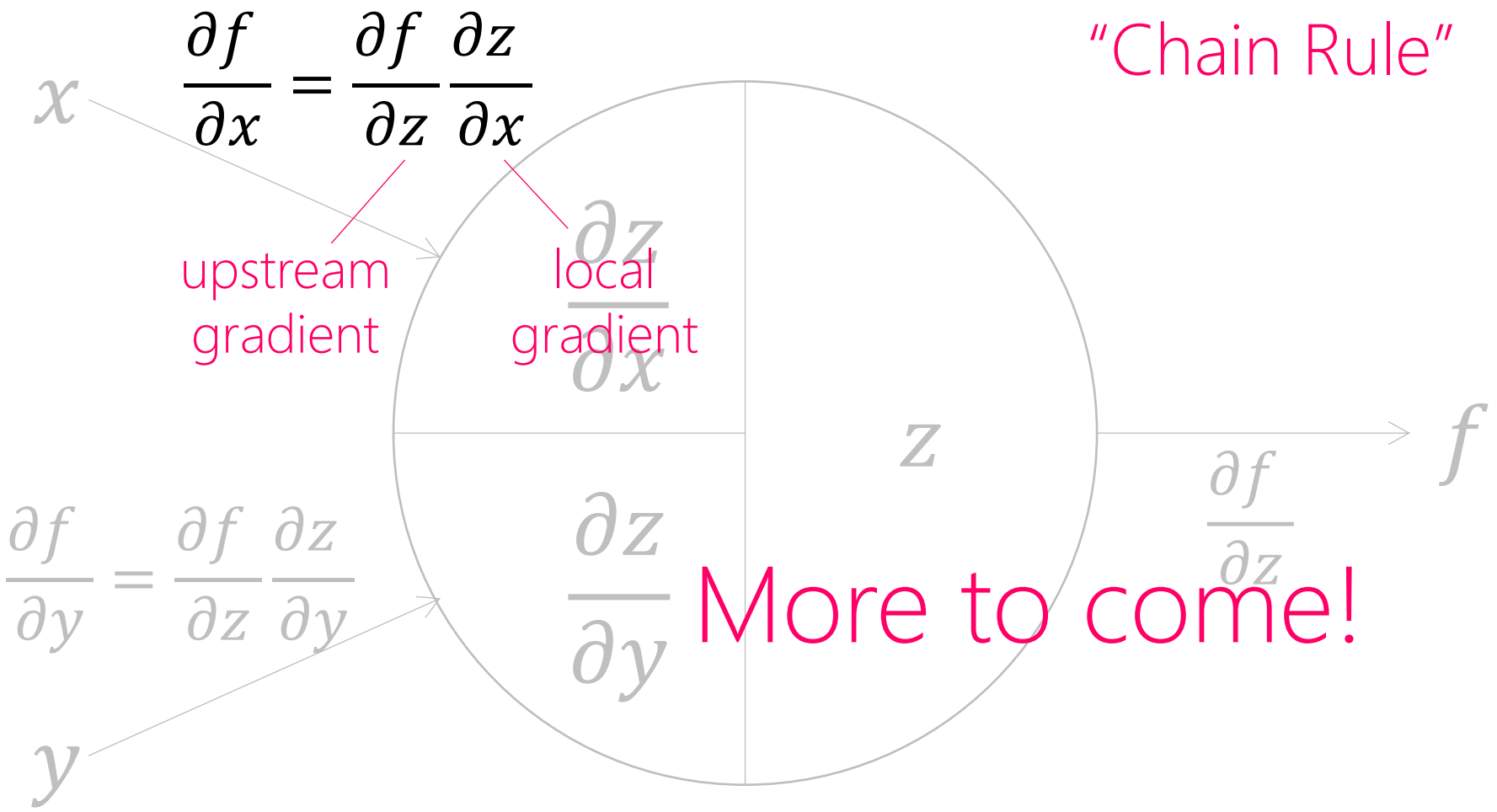
Backpropagation



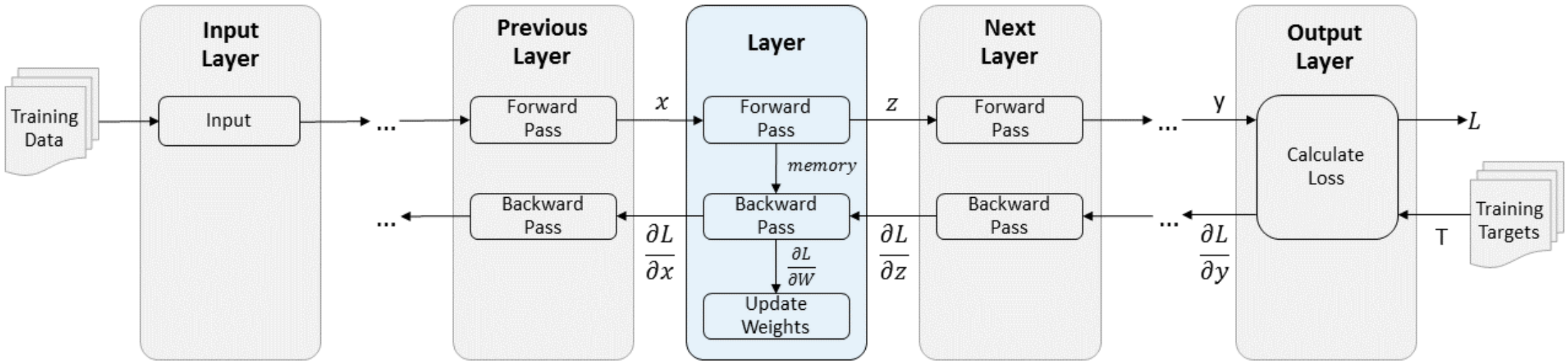
Backpropagation



Backpropagation



Putting them all together



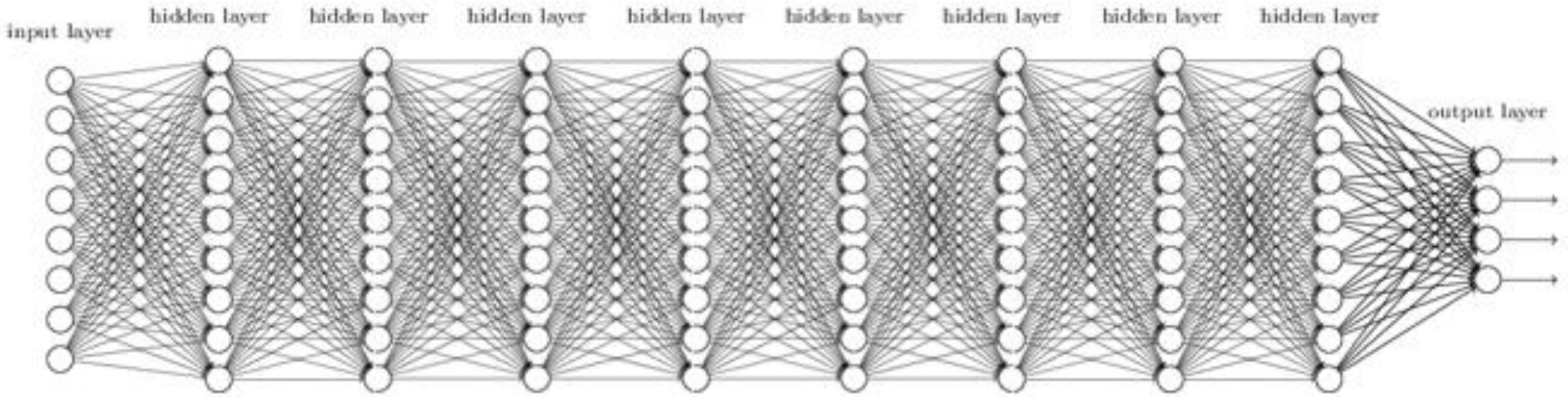
https://www.mathworks.com/help/nnet/ug/define-custom-deep-learning-layers.html

Putting them all together

- Linear model: $f = Wx + b$
- Neural network with a single hidden layer: $f = W_2 \max(0, W_1 x + b_1) + b_2$
- Neural network with two hidden layers:

$$f = W_3 \max(0, W_2 \max(0, W_1 x + b_1) + b_2) + b_3$$

- … stacking up hidden layers → "deep" neural networks



Deep Neural Network

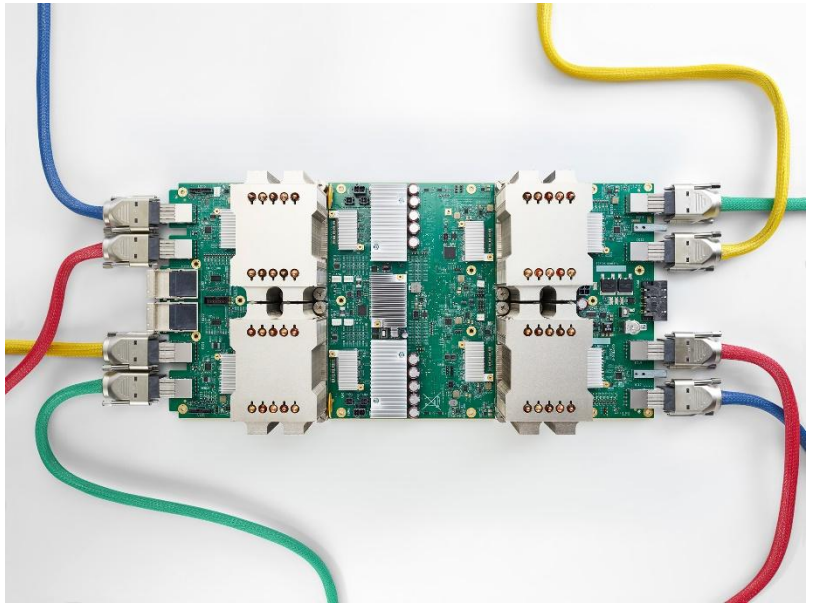
5MB Hard Drive Being Shipped by IBM, 1956


A FEW
MOMENTS LATER

40 years later…

NVIDIA GeForce Quadro GV100

NVIDIA Jetson TX2



Google Tensor Processing Unit (TPU)


IMAGENET

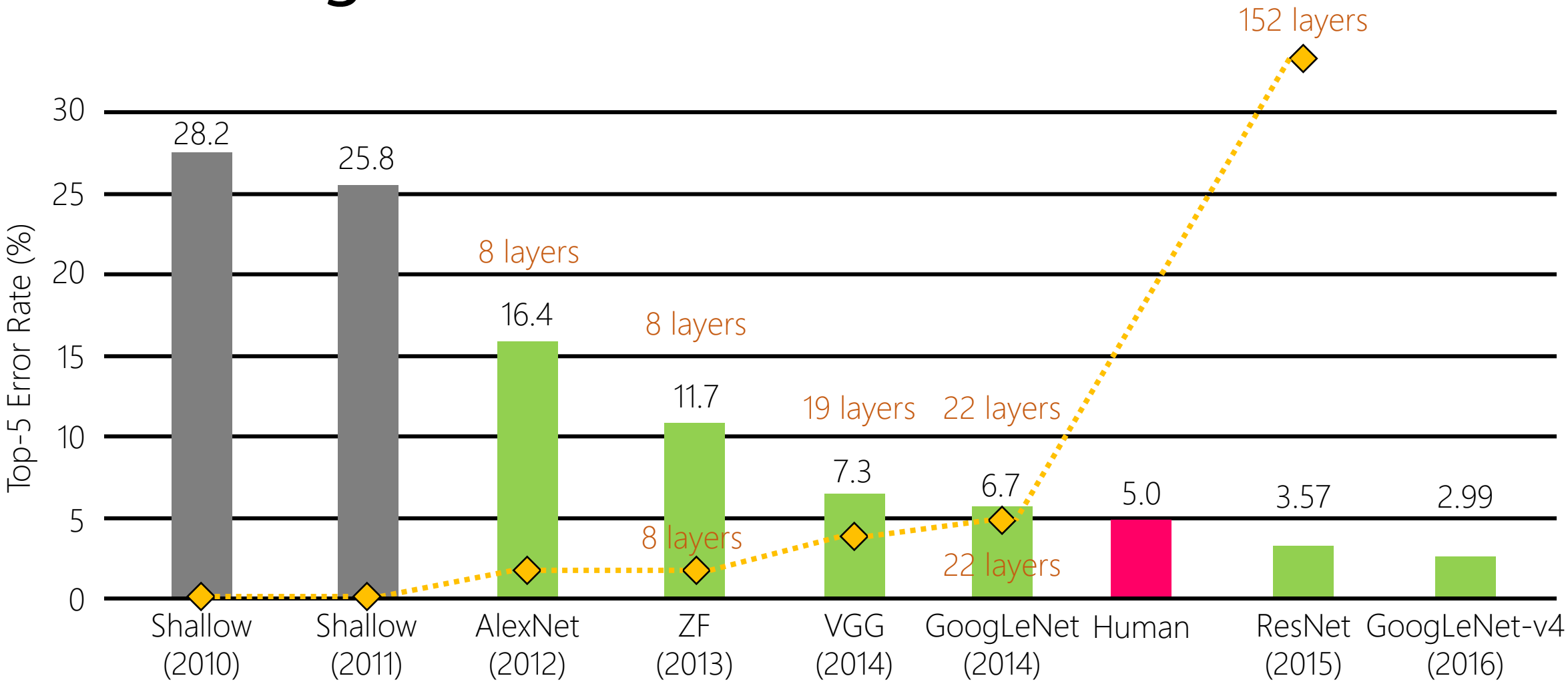
ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- ImageNet (First appearance as a poster at CVPR 2009)
  - Over 14 million images with hand-annotated labels (crowdsourced via M-turk)
  - Over 20 thousand categories
  - 1M+ images with bounding boxes.

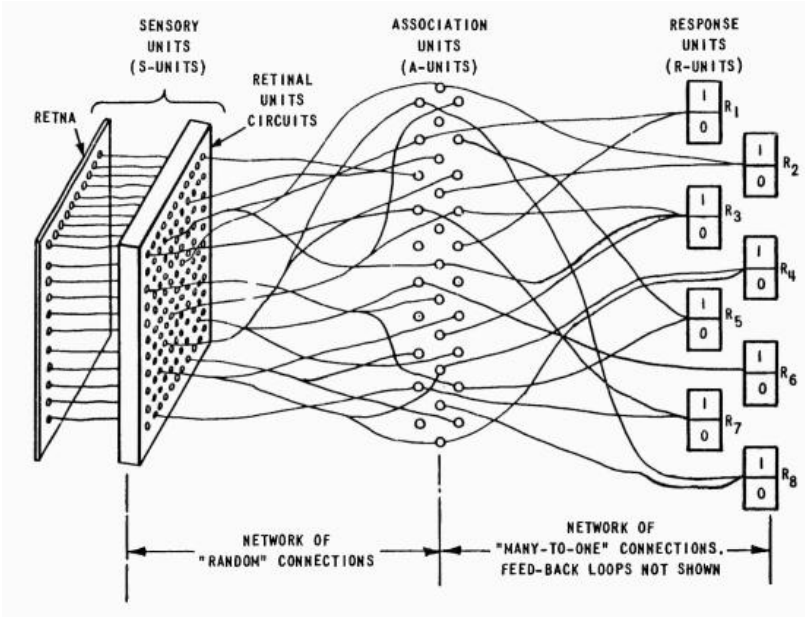
- ImageNet Challenge
  - Since 2010
  - Uses a trimmed set of thousand non-overlapping classes.
  - Humans error is about 5%*
  - Playground of computer vision researchers.
  - Marked the start of the current AI boom.

Russakovsky et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252.

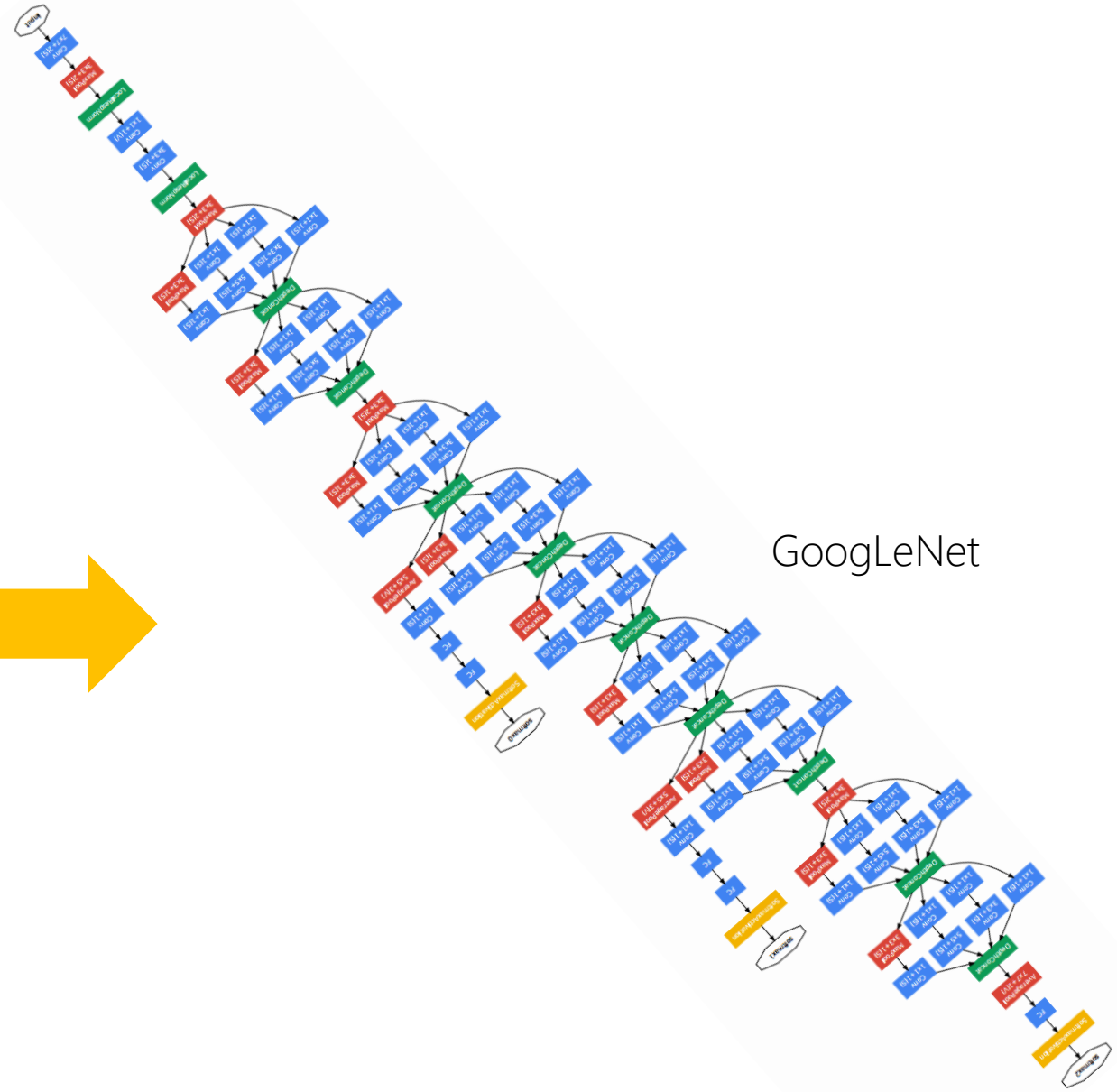
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



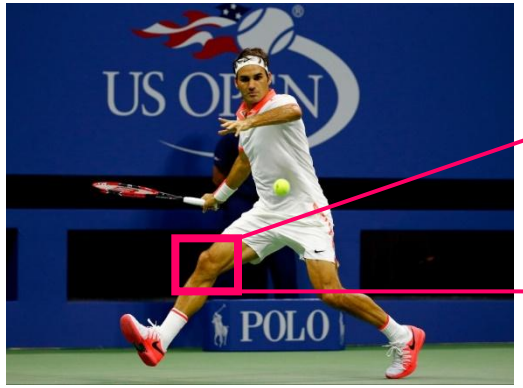
Going Deeper



Perceptron



GoogLeNet

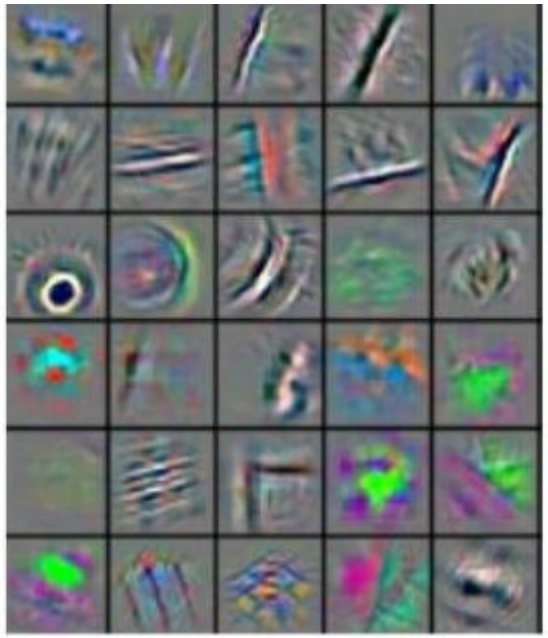


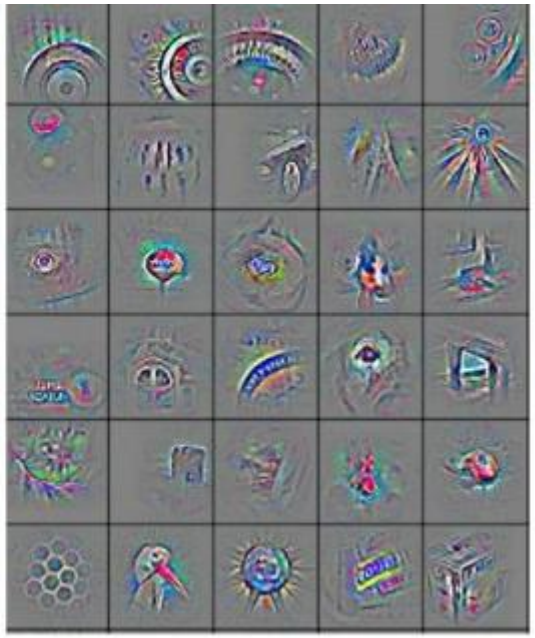
Low-level Features

Mid-level Features

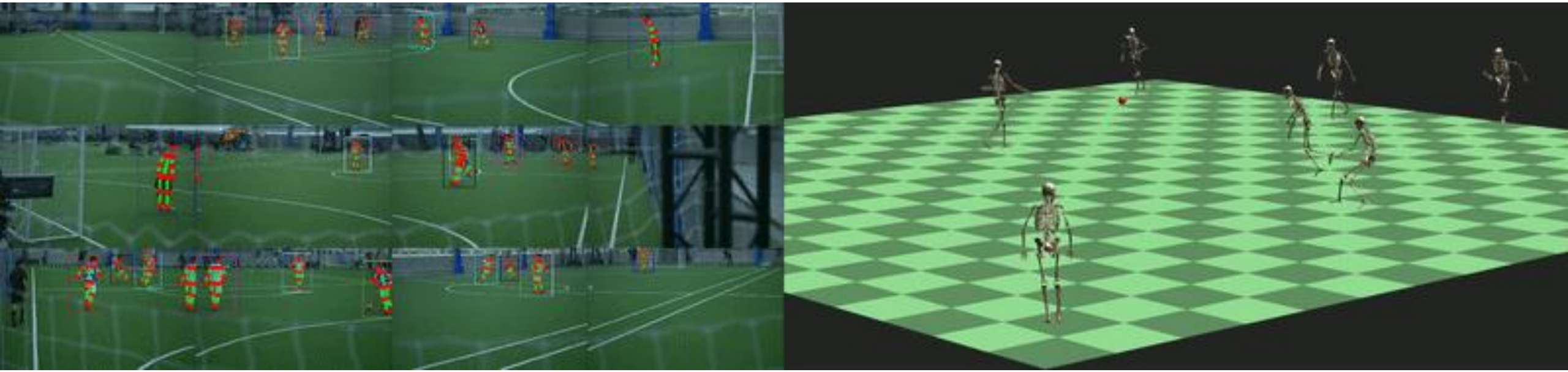
High-level Features

Shoulder: 0.01
Elbow: 0.12
Wrist: 0.01
Knee: 0.74
Ankle: 0.03
...



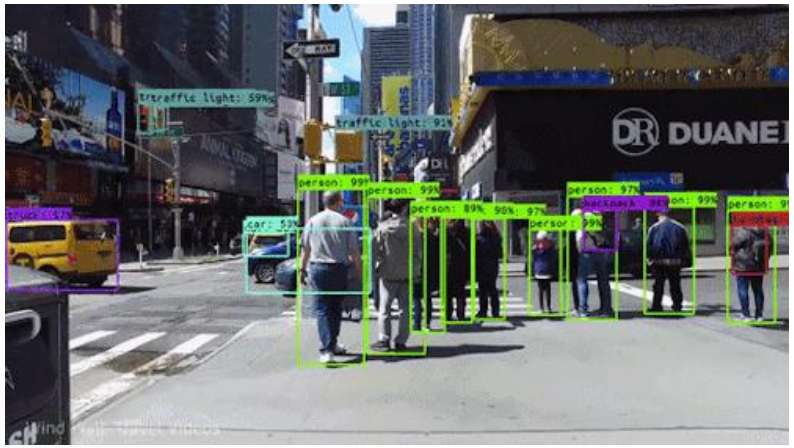


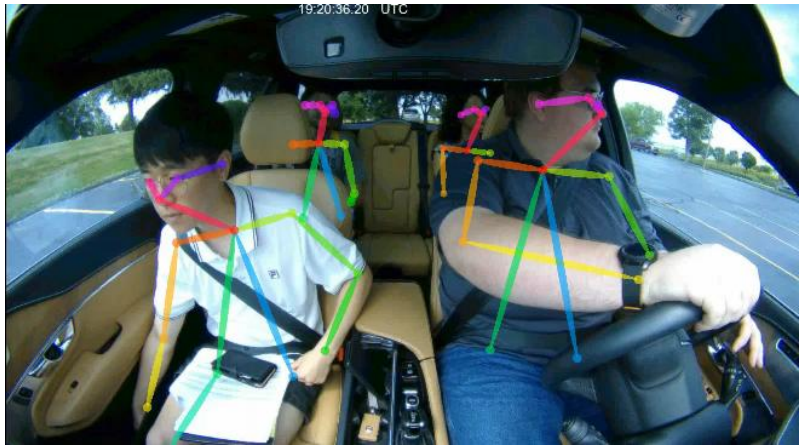
e.g. Pose Estimation

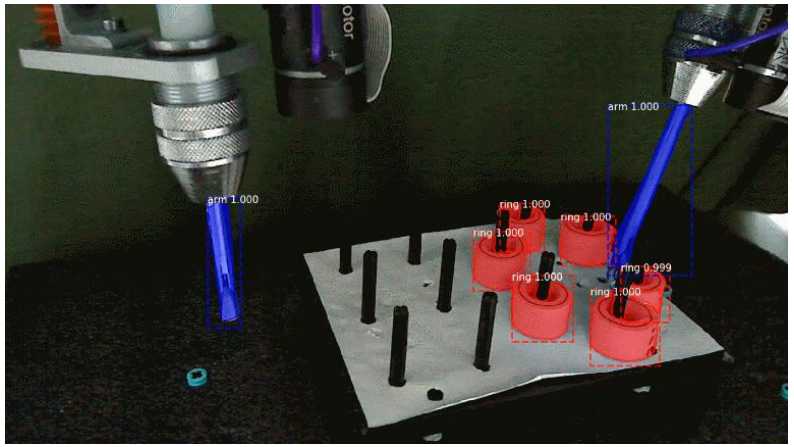


Ohashi et al. (2020). arXiv:2001.05613

…and many others

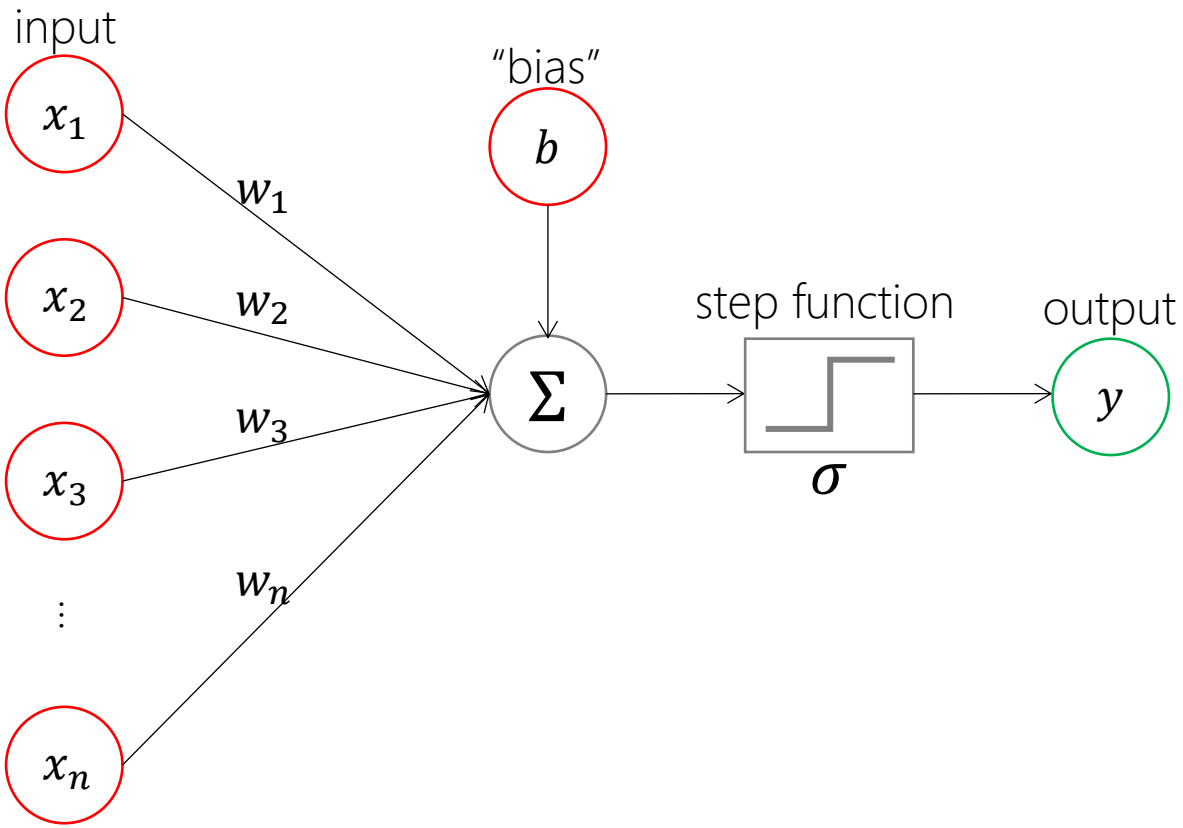






Closer Look into Neural Nets

- Back to perceptron…

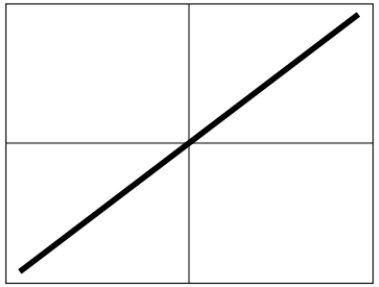


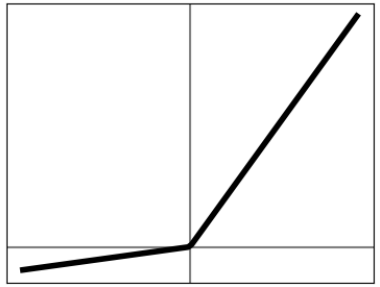
$$z = \sum_j w_j x_j + b$$

$$y = \sigma(z)$$

Linear (No Activation)

$$\sigma(z) = z$$





Leaky ReLU

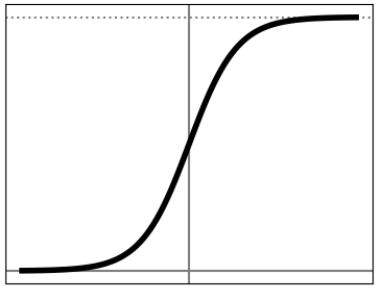
$$\sigma(z) = \max(0.1z, z)$$

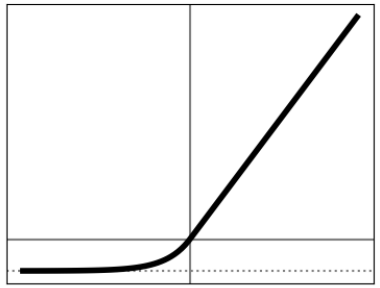
Parametric ReLU (PReLU)

$$\sigma(z) = \max(\alpha z, z)$$

Sigmoid

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



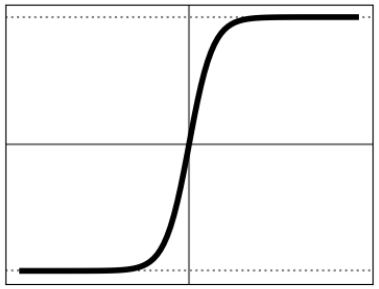


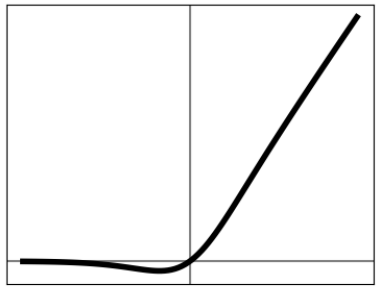
Exponential ReLU (ELU)

$$\sigma(z) = \begin{cases} z, & z \geq 0 \\ \alpha(e^z - 1), & z < 0 \end{cases}$$

Hyperbolic Tangent

$$\sigma(z) = \tanh(z)$$





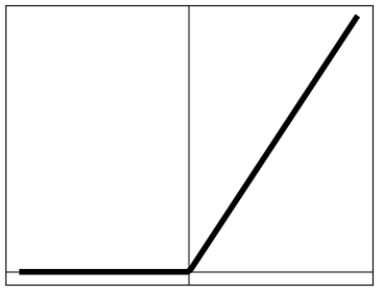
Swish

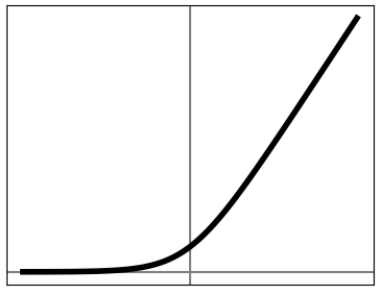
$$\sigma(z) = \frac{z}{1 + e^{-z}}$$

$$= z * \text{sigmoid}(z)$$

Rectified Linear Unit (ReLU)

$$\sigma(z) = \max(0, z)$$



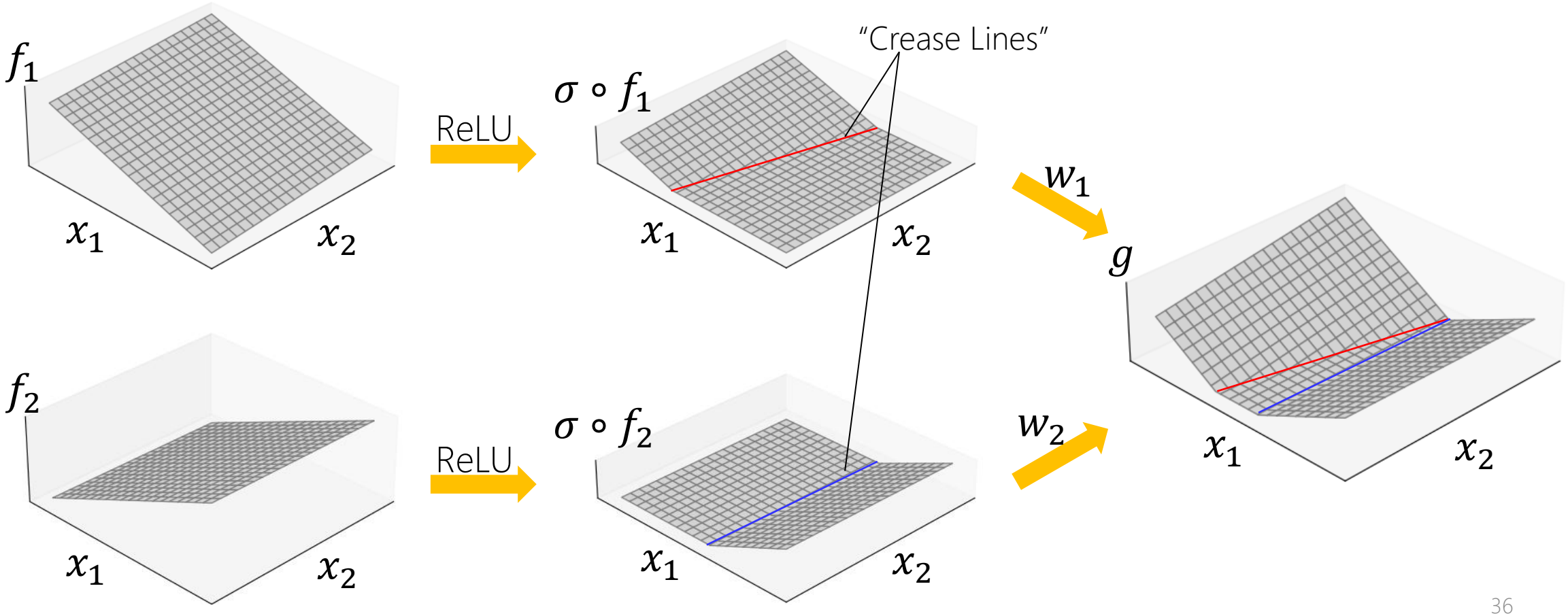


SoftPlus

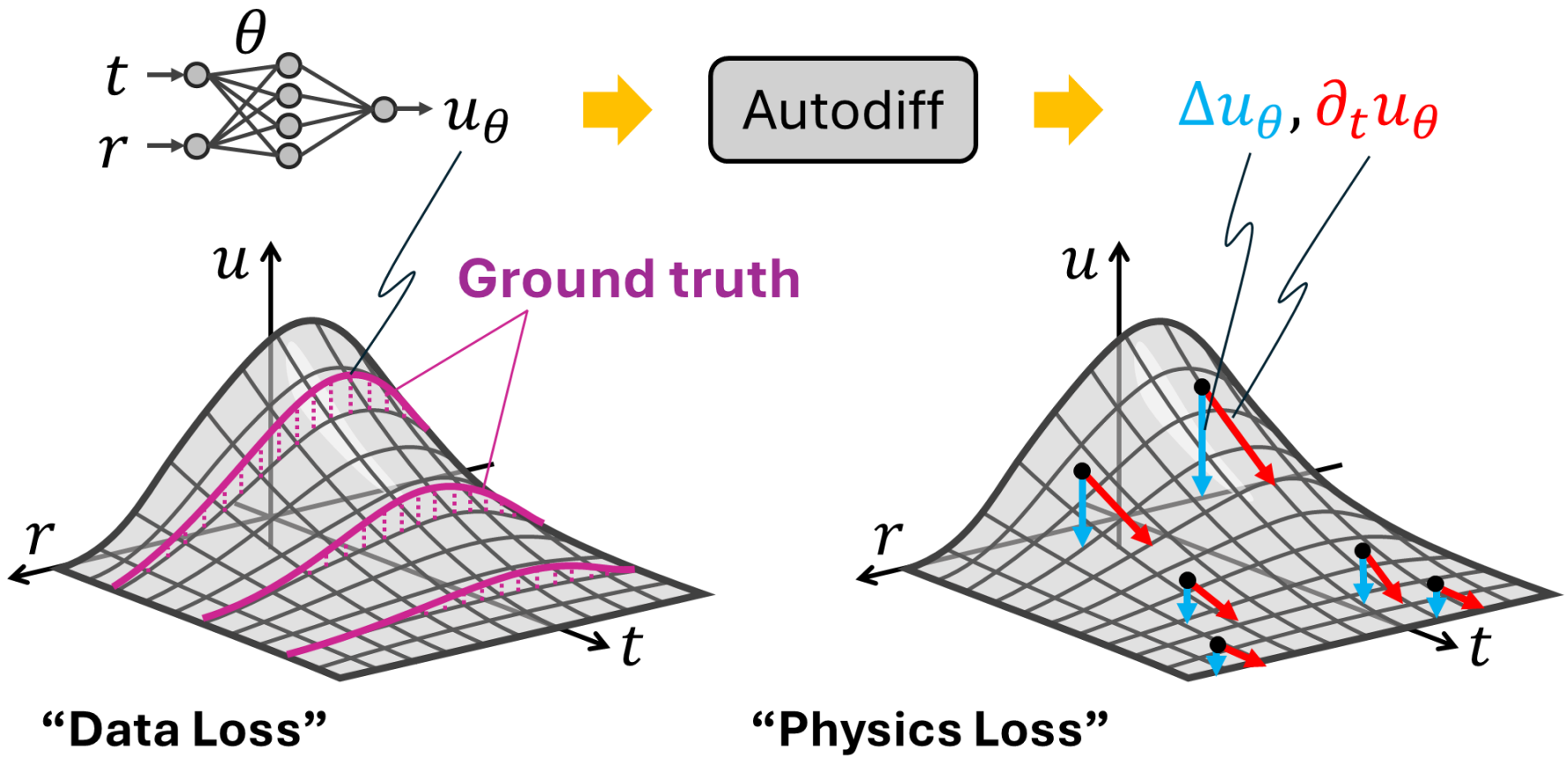
$$\sigma(z) = \log(1 + e^z)$$

Closer Look into Neural Nets

- Geometric view of perceptron?



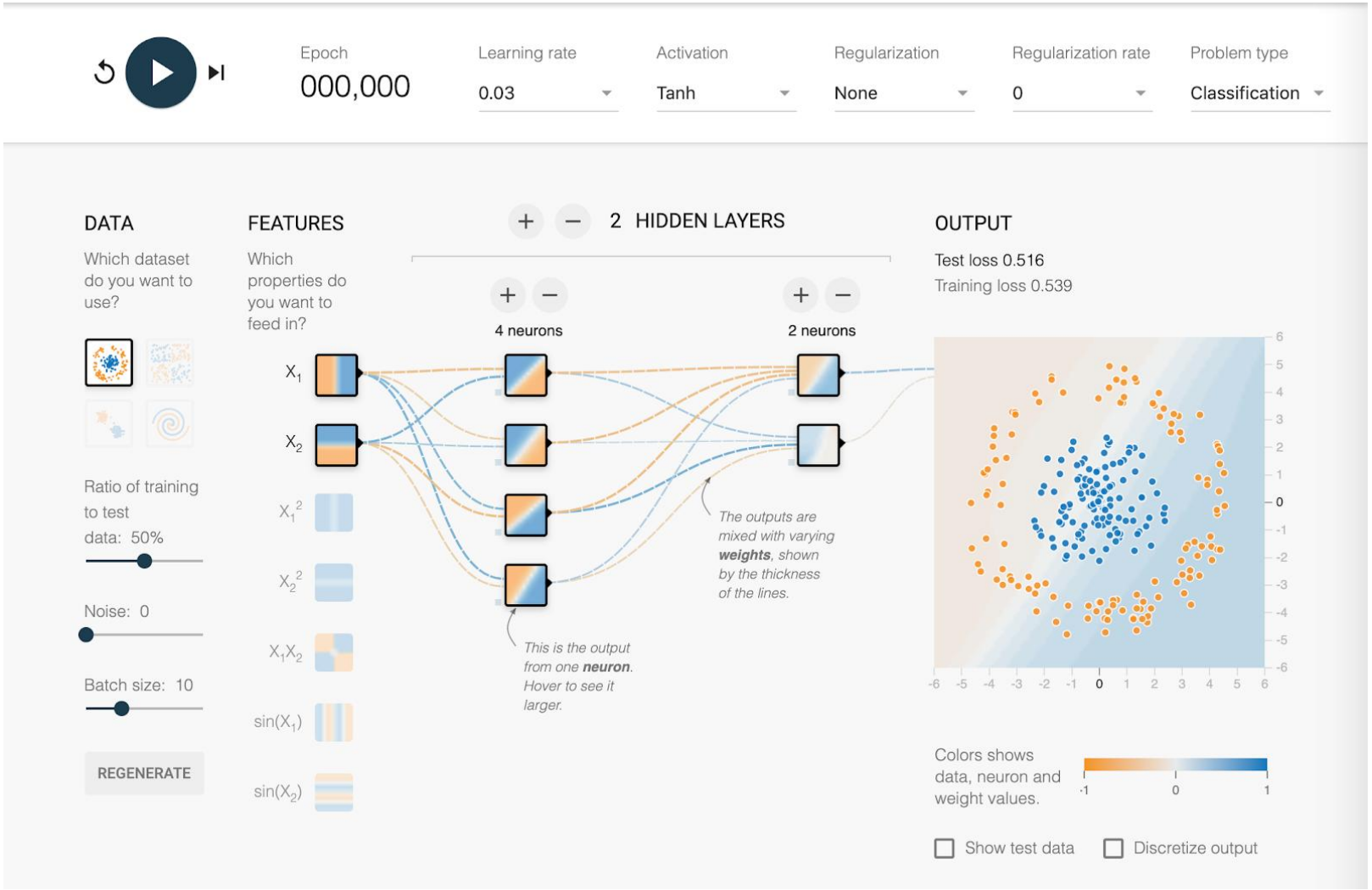
Peek on Physics-Informed Neural Nets



Universal Approximation Theorem (UAT)

- In a nutshell: “

TensorFlow Playground



https://playground.tensorflow.org

Formulating NN Problems

Image Classification

- Images are more straightforward and intuitive subject to study deep networks
- Easy to generalize to other forms of data
- Major breakthroughs in deep learning research were in computer vision
- Image classification = core task in computer vision

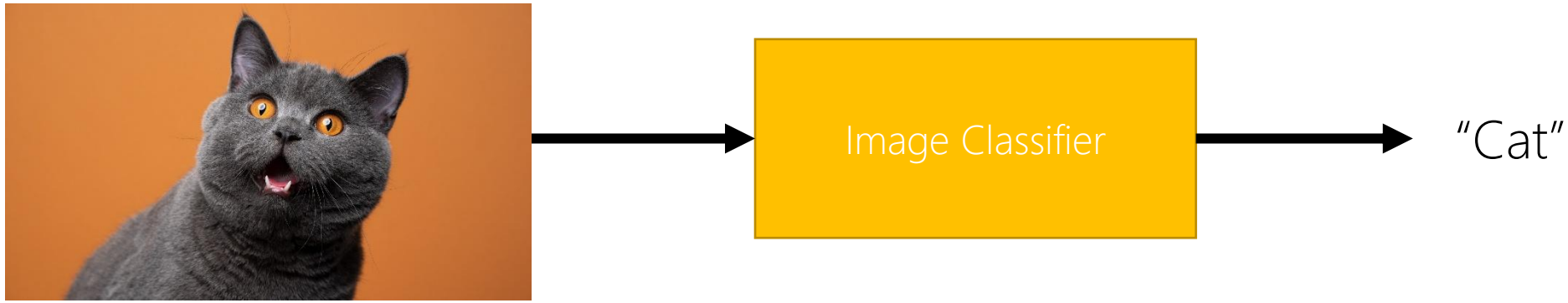
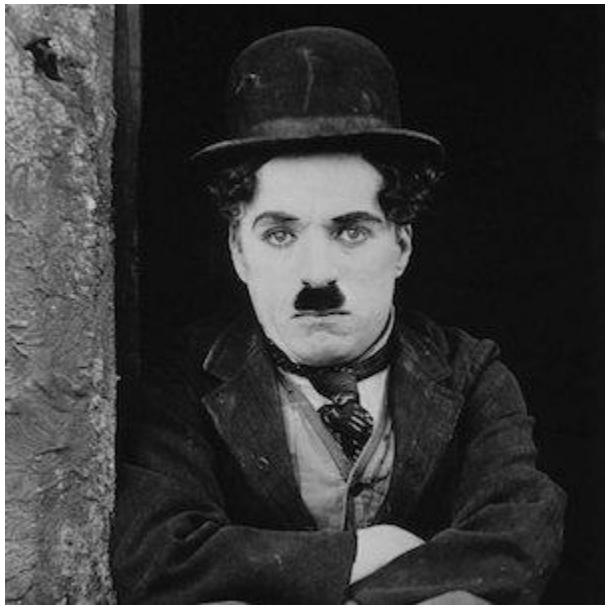


Image Classification is Challenging!

- Problem 1. Semantic Challenge



What we see

127	123	101	40	12	13	14	22	26	18	13	13	13	13	13	15
114	81	98	38	13	14	28	36	34	29	17	13	13	13	13	15
142	102	87	33	13	15	31	38	34	36	22	13	13	14	15	17
132	112	87	24	14	22	34	52	46	37	26	20	14	15	16	18
120	103	83	19	15	18	57	158	161	131	24	19	14	15	15	17
117	110	84	18	14	21	114	163	171	128	49	17	15	15	15	17
117	112	89	21	14	26	155	163	174	157	117	15	16	16	16	17
117	103	84	21	15	13	120	174	106	182	87	11	15	15	16	17
117	103	86	21	16	16	56	158	125	156	55	19	21	16	16	17
112	102	94	23	17	29	33	86	97	80	44	38	40	26	15	16
107	91	95	25	19	33	39	98	77	105	65	42	35	40	17	16
98	95	97	26	25	29	39	101	108	70	70	39	29	46	20	16
111	92	91	34	34	23	27	64	144	89	54	23	30	42	31	16
113	96	90	36	32	28	21	31	92	83	47	21	35	42	42	17
119	109	87	34	31	33	45	51	45	134	128	36	33	40	35	17
98	90	78	29	28	34	28	29	39	70	51	65	64	35	23	16

What computers see
= just a grid of numbers (e.g. 1024 x 768 x 3)

Image Classification is Challenging!

- Problem 2. Varying Perspectives

Image Classification is Challenging!

- Problem 3. Illumination

http://www.ifp.illinois.edu/~tchen5/vlfr.html

Image Classification is Challenging!

- Problem 4. Occlusion

Image Classification is Challenging!

- Problem 5. Deformation, Shape & Posture Change

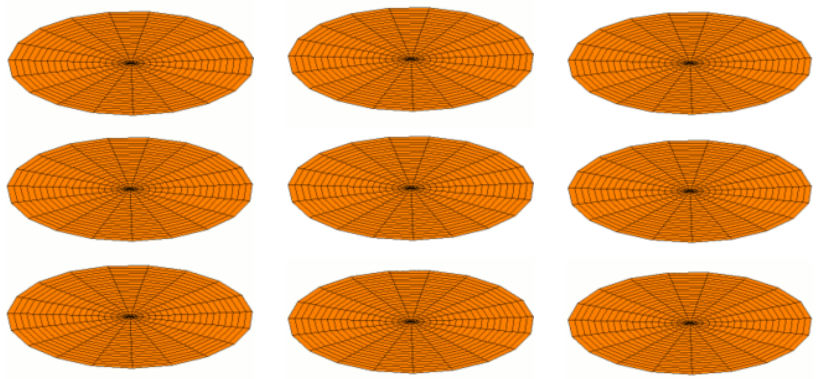
The same guy in front of you in 2011

Image Classification is Challenging!

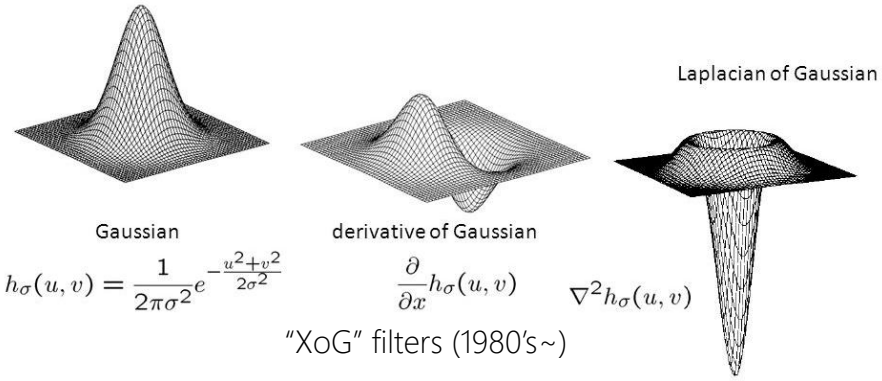
- Problem 6. Intraclass Variation

Arman, Alarm Clocks (Reveils), 1960, MCA Chicago

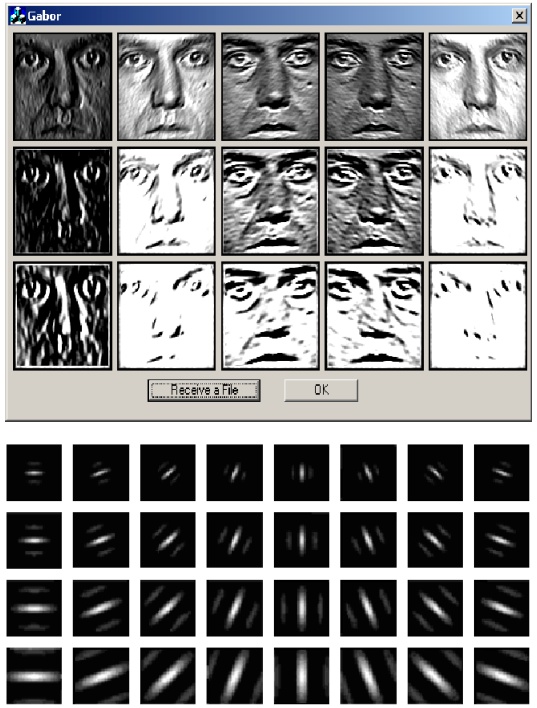
Conventional Approach (Model-Driven)



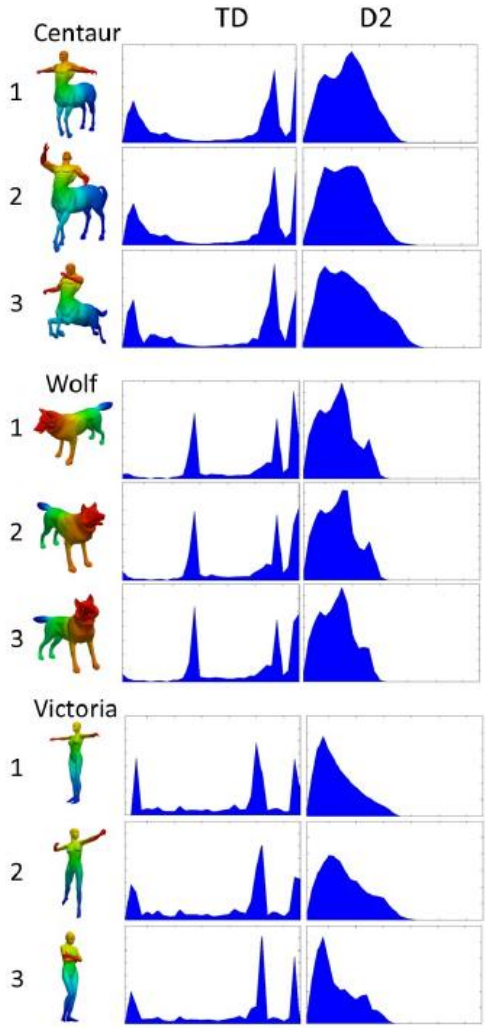
Laplace-Beltrami Spectra (2000's~)



"XoG" filters (1980's~)



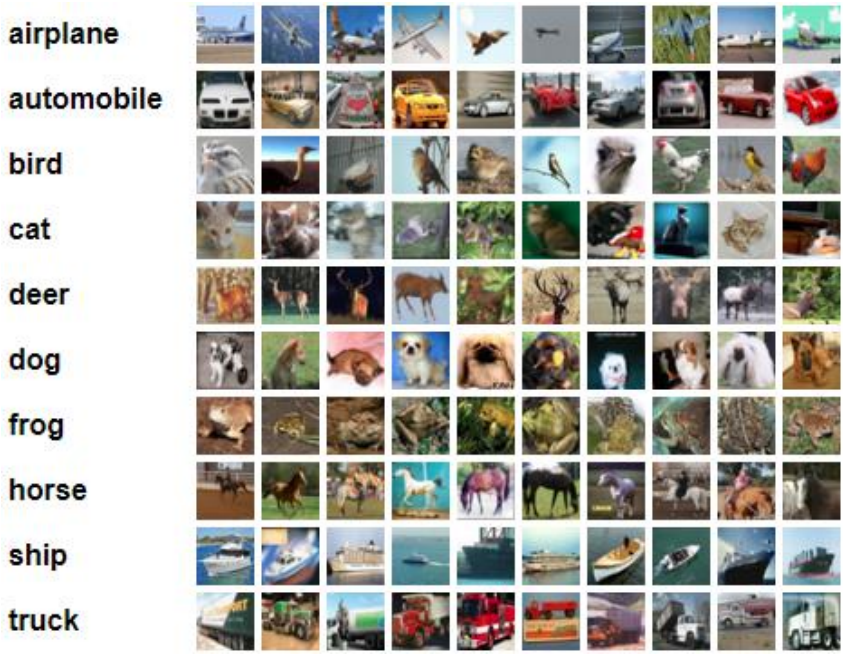
Gabor Filter (1998~)



Distance Histograms (2002~)

Machine Learning (Data-Driven)

Data with Labels



training

Image Classifier

Image Classification with Perceptron

- Idea: Formulate the image classification problem as $y = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\mathbf{x}$ is an image and $\boldsymbol{y}$ is the image label.

Image Classification with Perceptron

- Idea: Formulate the image classification problem as $\boldsymbol{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\mathbf{x}$ is an image and $\boldsymbol{y}$ is the image label.

- Our model "$\boldsymbol{f}$" can be written formally as $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta}) \coloneqq \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\boldsymbol{\theta} \coloneqq (\mathbf{W}, \mathbf{b})$.

Image Classification with Perceptron

- Idea: Formulate the image classification problem as $\boldsymbol{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\mathbf{x}$ is an image and $\boldsymbol{y}$ is the image label.

- Our model "$\boldsymbol{f}$" can be written formally as $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta}) \coloneqq \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\boldsymbol{\theta} \coloneqq (\mathbf{W}, \mathbf{b})$.

- Goal: Find $\boldsymbol{\theta}$ ($\mathbf{W}$ and $\mathbf{b}$) that minimize the "*difference*" between the model prediction $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta})$ and the ground truth $\boldsymbol{y}$

Image Classification with Perceptron

- Idea: Formulate the image classification problem as $\boldsymbol{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\mathbf{x}$ is an image and $\boldsymbol{y}$ is the image label.

- Our model "$\boldsymbol{f}$" can be written formally as $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta}) \coloneqq \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\boldsymbol{\theta} \coloneqq (\mathbf{W}, \mathbf{b})$.

- Goal: Find $\boldsymbol{\theta}$ ($\mathbf{W}$ and $\mathbf{b}$) that minimize the "*difference*" between the model prediction $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta})$ and the ground truth $\boldsymbol{y}$
  - In English: We want the predicted label of our model $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta})$ to be as close as possible to the true label $\boldsymbol{y}$ and we're going to accomplish that by carefully "tuning $\mathbf{W}$ and $\mathbf{b}$".

Image Classification with Perceptron

- Idea: Formulate the image classification problem as $\boldsymbol{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\mathbf{x}$ is an image and $\boldsymbol{y}$ is the image label.

- Our model "$\boldsymbol{f}$" can be written formally as $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta}) \coloneqq \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\boldsymbol{\theta} \coloneqq (\mathbf{W}, \mathbf{b})$.

- Goal: Find $\boldsymbol{\theta}$ ($\mathbf{W}$ and $\mathbf{b}$) that minimize the "*difference*" between the model prediction $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta})$ and the ground truth $\boldsymbol{y}$
  - In English: We want the predicted label of our model $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta})$ to be as close as possible to the true label $\boldsymbol{y}$ and we're going to accomplish that by carefully "tuning $\mathbf{W}$ and $\mathbf{b}$".
  - Based on what, though?

Image Classification with Perceptron

- Idea: Formulate the image classification problem as $\boldsymbol{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\mathbf{x}$ is an image and $\boldsymbol{y}$ is the image label.

- Our model "$\boldsymbol{f}$" can be written formally as $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta}) \coloneqq \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\boldsymbol{\theta} \coloneqq (\mathbf{W}, \mathbf{b})$.

- Goal: Find $\boldsymbol{\theta}$ ($\mathbf{W}$ and $\mathbf{b}$) that minimize the "*difference*" between the model prediction $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta})$ and the ground truth $\boldsymbol{y}$
  - In English: We want the predicted label of our model $\boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta})$ to be as close as possible to the true label $\boldsymbol{y}$ and we're going to accomplish that by carefully "tuning $\mathbf{W}$ and $\mathbf{b}$".
  - Based on what, though? Lots of labeled data! $(\mathbf{x}_i, y_i)_{i=1,\cdots,N}$

Image Classifier: Step-by-Step

1. Collect a bunch of data $(\mathbf{x}_i, y_i)_{i=1,\cdots,N}$
2. Split them into two sets: training set and test set
3. Solve $\min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{N_{\mathbf{train}}} d(y_i,\ f(\mathbf{x}_i \mid \boldsymbol{\theta}))$
4. See if $d(y,\ f(\mathbf{x} \mid \boldsymbol{\theta}))$ is "close enough to zero" on the test set
   - If it does: profit!
   - If it doesn't: sit and cry (or build a better model, collect more data, etc.: Next class!)

Example: Cat/Dog Classifier

- The cats and dogs dataset: https://www.microsoft.com/en-us/download/details.aspx?id=54765
- 12,500 cat images & 12,500 dog images from internet





Example: Cat/Dog Classifier

( , −1)

( , −1)

( , −1)

. . .

( , 1)

(

, 1)

( , 1)

. . .

Example: Cat/Dog Classifier

width = 400
height = 500
channels = 3

Example: Cat/Dog Classifier

width = 400
height = 500
channels = 3

gray + resize

64 x 64

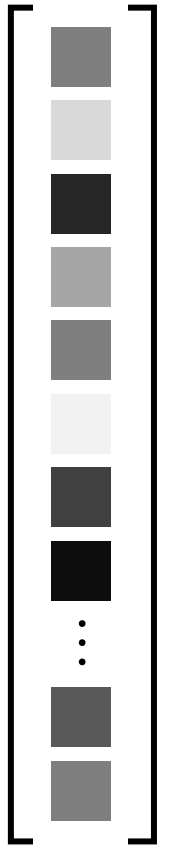
Example: Cat/Dog Classifier

width = 400
height = 500
channels = 3

gray + resize

64 x 64

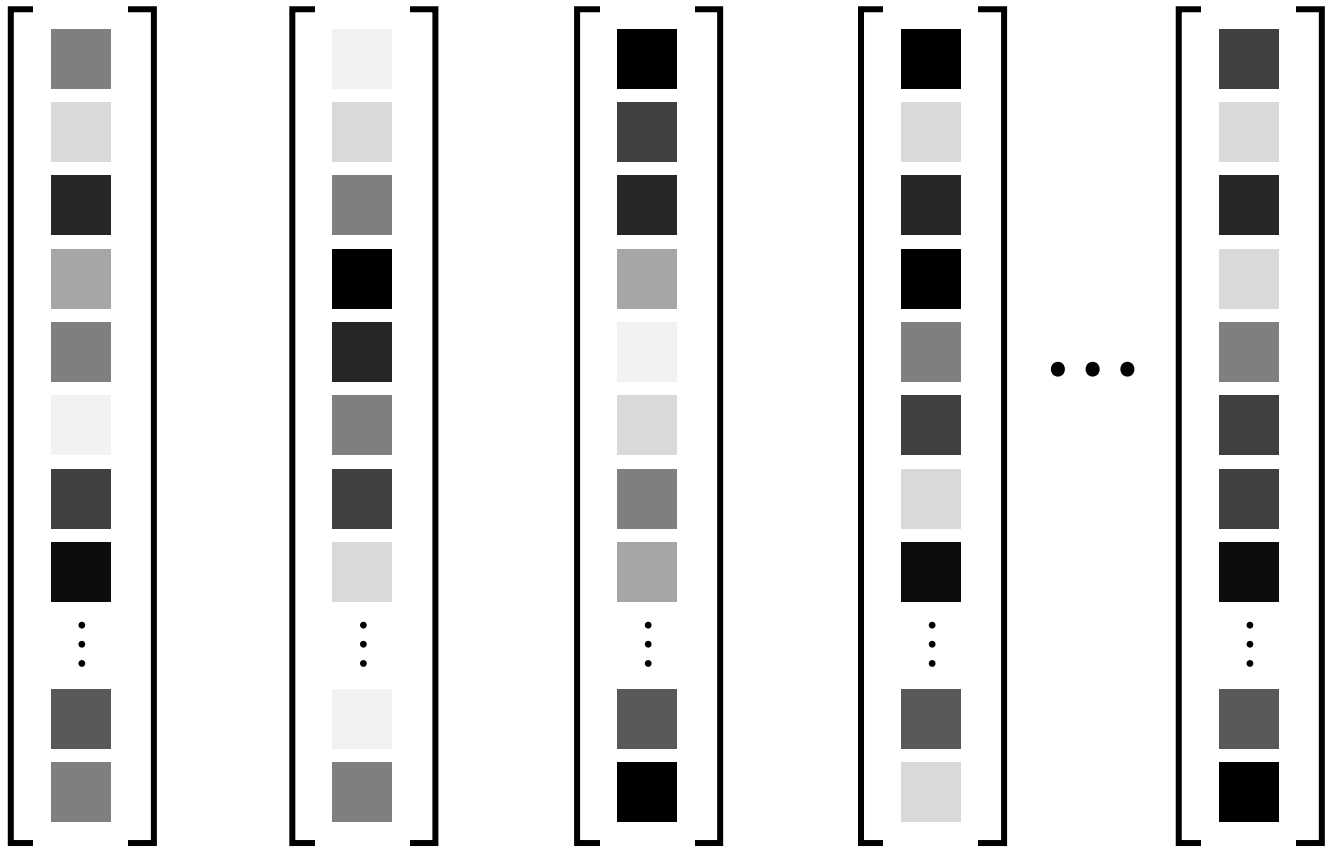
flatten



4096 x 1

Example: Cat/Dog Classifier

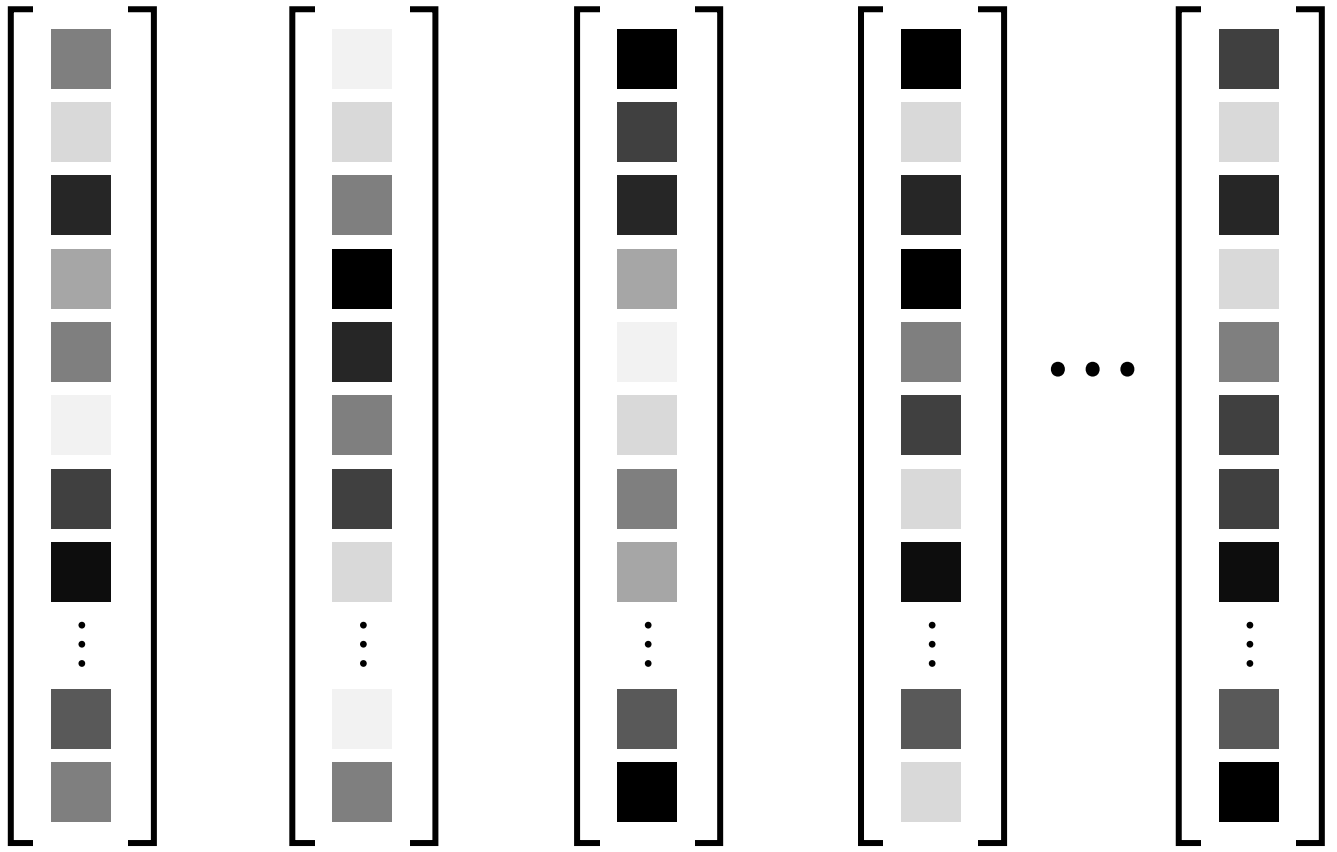
$\mathbf{x}$:



y: -1 $\quad$ 1 $\quad$ 1 $\quad$ -1 $\quad$ -1

Example: Cat/Dog Classifier

$\mathbf{x}$:



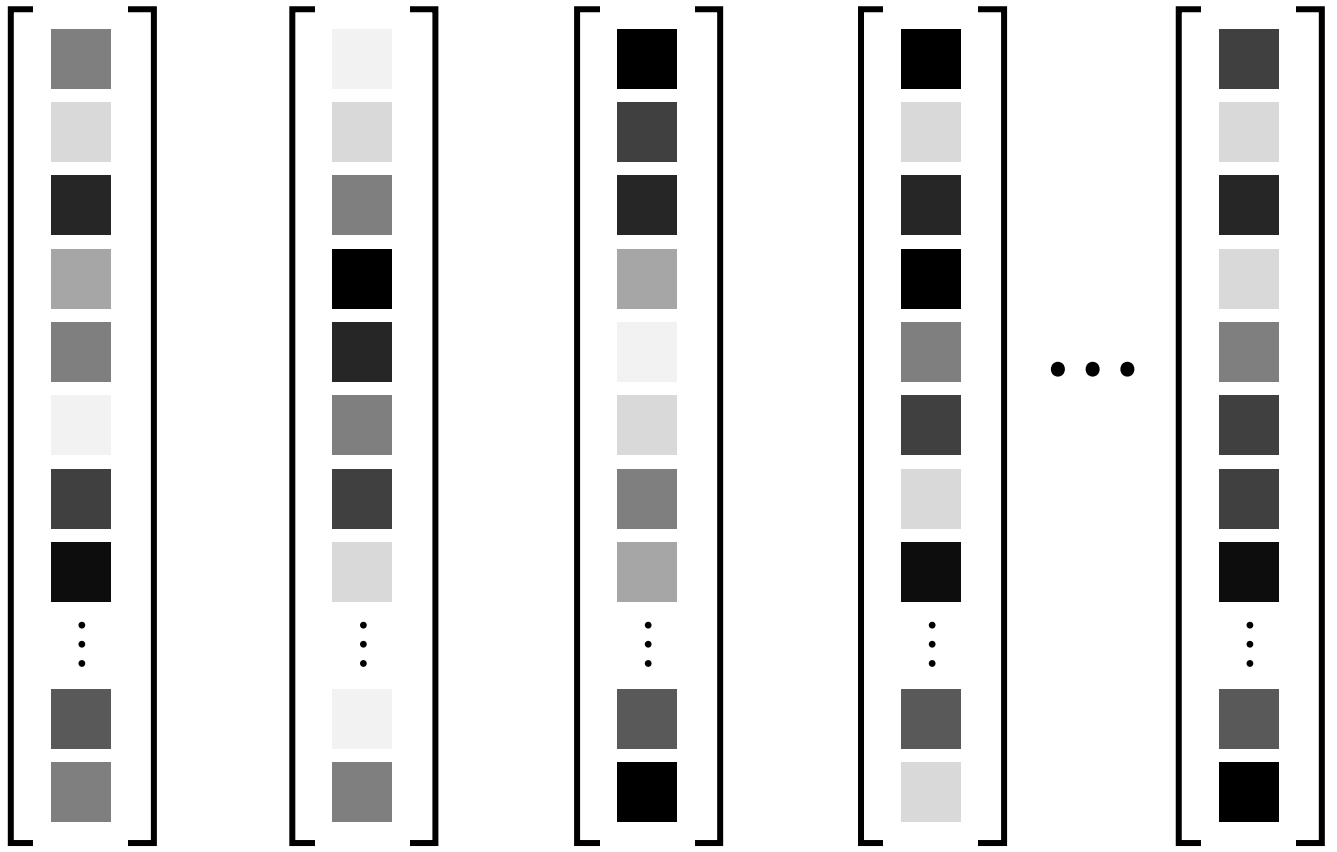
y: -1 $\quad 1 \quad 1 \quad -1 \quad -1$

$$\min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{N} |\mathbf{W}\mathbf{x}_i + \mathbf{b} - y_i|^2$$

1 x 4096

1 x 1

Example: Cat/Dog Classifier

$\mathbf{x}$: 

y: -1 $\quad$ 1 $\quad$ 1 $\quad$ -1 $\quad$ -1

$$\min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{N} |\mathbf{W}\mathbf{x}_i + \mathbf{b} - y_i|^2$$

$$\mathbf{W}^*, \mathbf{b}^*$$

Example: Cat/Dog Classifier

(Online)

Query

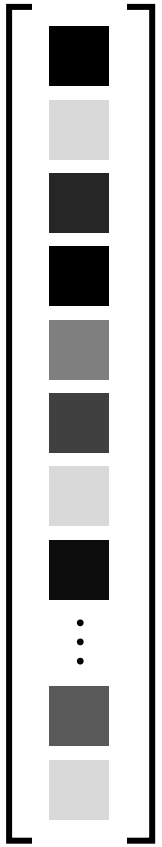
(*Never seen by the model)

Example: Cat/Dog Classifier

(Online)

Query

Grayscale
+
Resize
+
Flatten



Example: Cat/Dog Classifier

(Online)

Query

Grayscale
+
Resize
+
Flatten

$$\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} \rightarrow \begin{bmatrix} \\ \mathbf{W}^* \\ \\ \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \mathbf{b}^* \\ \\ \end{bmatrix}$$

Example: Cat/Dog Classifier

(Online)

Query

Grayscale
+
Resize
+
Flatten

$$\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} \rightarrow \begin{bmatrix} & \mathbf{W}^* & \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} \mathbf{b}^* \end{bmatrix} \rightarrow 0.47$$

Example: Cat/Dog Classifier

(Online)

Query

Grayscale
+
Resize
+
Flatten

$$\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{W}^* \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} \mathbf{b}^* \end{bmatrix} \rightarrow 0.47$$

> 0

$\therefore$ dog

Example: Cat/Dog Classifier

(Online)

Query

Grayscale
+
Resize
+
Flatten

$$\begin{bmatrix} \\ \mathbf{W}^* \\ \\ \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \mathbf{b}^* \\ \\ \end{bmatrix} \rightarrow -0.7$$

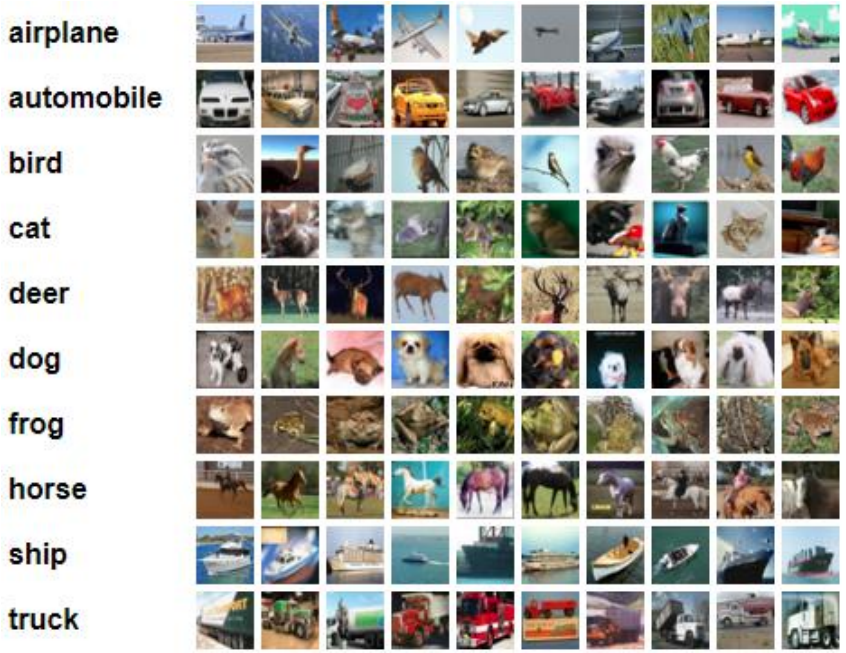
< 0

$\therefore$ cat

Example: 10-way Classifier (CIFAR-10 Dataset)

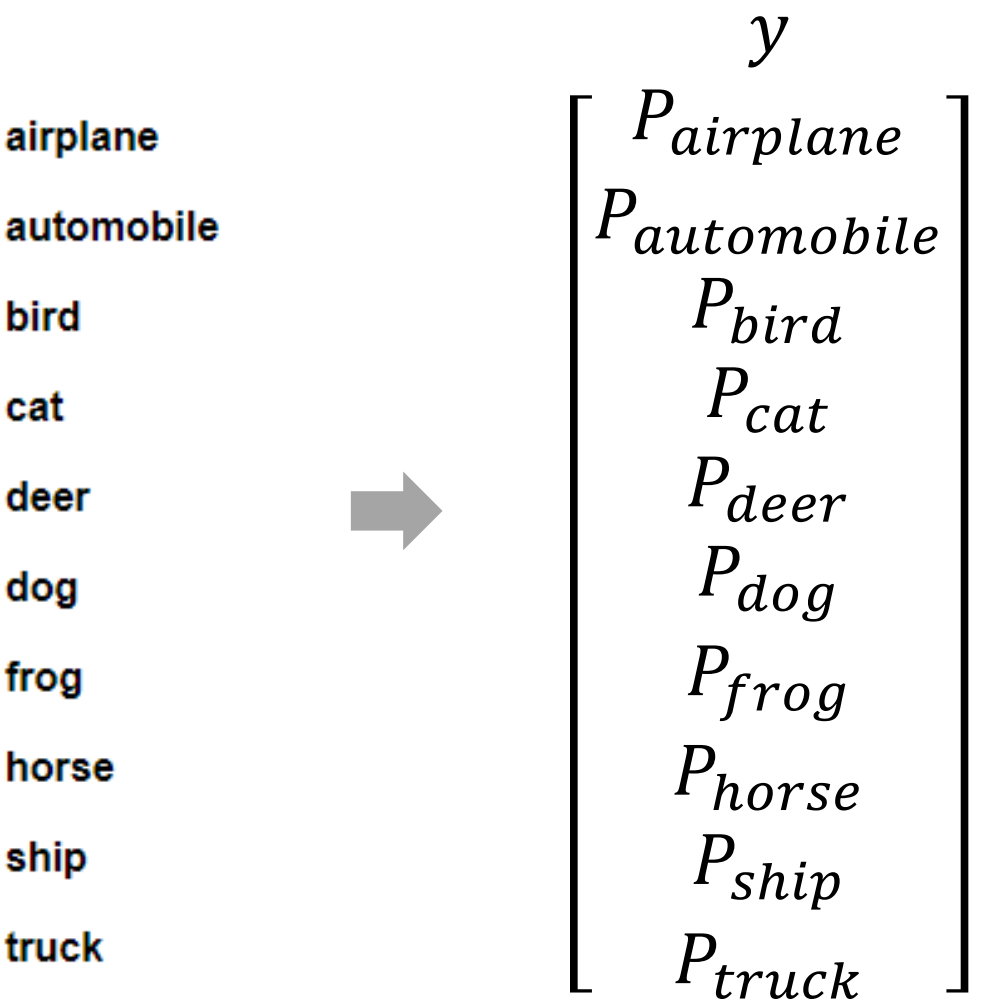
- What if there were more than two classes (e.g. CIFAR, ImageNet)?

CIFAR-10: 60,000 images in 10 classes



ImageNet: 14,197,122 images in 21,841 classes
ILSVRC: 1.4M images in 1,000 classes

Probability Mass Function



$$0 \leq P_i \leq 1, \forall i$$

$$\sum_i P_i = 1$$

“One-hot” Encoding

True PMF

$$y$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

“One-hot” Encoding

True PMF y

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

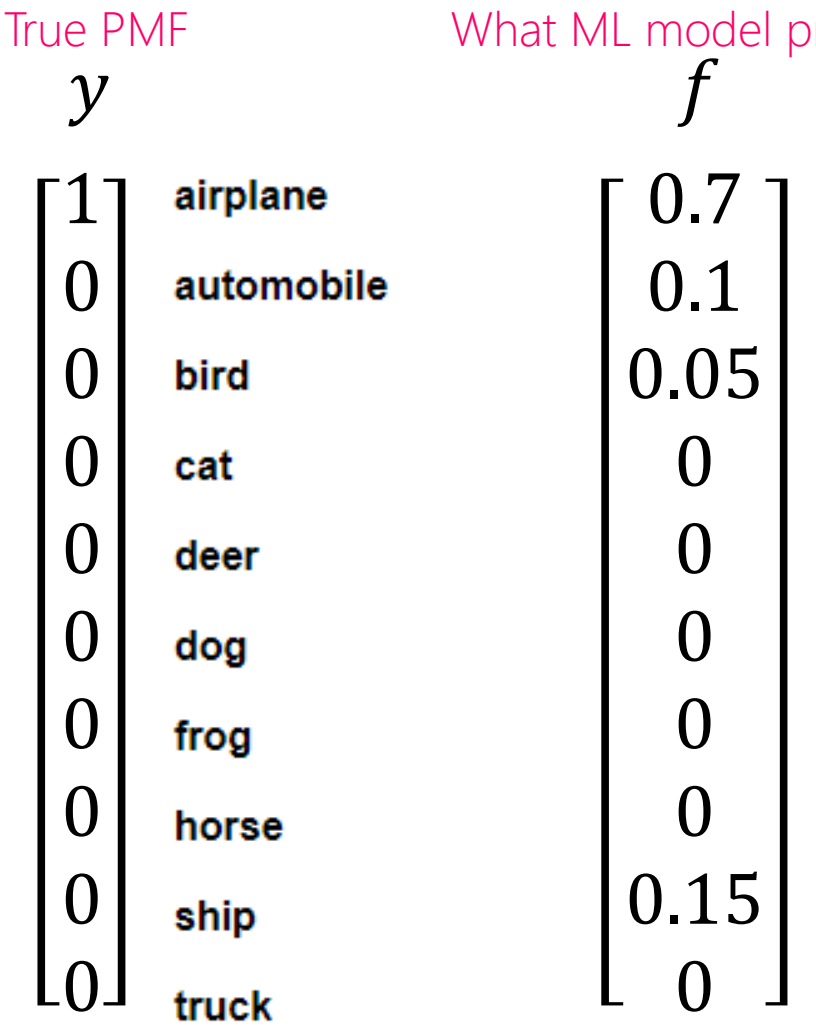
What ML model predicts f

$$\begin{bmatrix} 0.7 \\ 0.1 \\ 0.05 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.15 \\ 0 \end{bmatrix}$$

Cross-entropy

$$-\sum_i y_i \log(f_i)$$

In English: How similar are the two distributions?
(We will re-visit this later to learn how and why.)



Example: 10-way Classifier (CIFAR-10 Dataset)

$\mathbf{x}_1 =$ 

airplane automobile bird cat deer dog frog horse ship truck

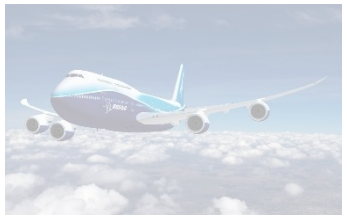
$y_1 = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

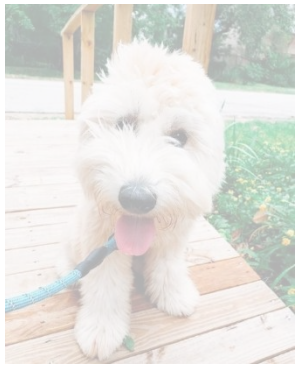
$\mathbf{x}_2 =$ 

$y_2 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]$

⋮ ⋮ ⋮

Example: 10-way Classifier (CIFAR-10 Dataset)





airplane automobile bird cat deer dog frog horse ship truck

$\mathbf{x}_1 =$ ⋯ $y_1 = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

Minimize $-\sum_i y_i \log(f(\mathbf{x}_i|\mathbf{W}, \mathbf{b}))$ ➡ $\mathbf{W}^*, \mathbf{b}^*$

$\mathbf{x}_2 =$ ⋯ $y_2 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]$

⋮ ⋮ ⋮

Example: 10-way Classifier (CIFAR-10 Dataset)

(online)

Query

$$\underset{10 \times 4096}{\begin{bmatrix} \mathbf{W}^* \end{bmatrix}} \underset{4096 \times 1}{\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}} + \underset{10 \times 1}{\begin{bmatrix} \mathbf{b}^* \end{bmatrix}}$$

Example: 10-way Classifier (CIFAR-10 Dataset)

(online)

Query

$$\begin{bmatrix} & 10 \times 4096 & \\ & \mathbf{W}^* & \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} 10 \times 1 \\ \mathbf{b}^* \end{bmatrix} \rightarrow \begin{bmatrix} -0.1 \\ 0 \\ 0 \\ 0.05 \\ 0.9 \\ 0.1 \\ 0 \\ 0.25 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \text{airplane} \\ \text{automobile} \\ \text{bird} \\ \text{cat} \\ \text{deer} \\ \text{dog} \\ \text{frog} \\ \text{horse} \\ \text{ship} \\ \text{truck} \end{matrix}$$

4096 x 1

Example: 10-way Classifier (CIFAR-10 Dataset)

(online)

Query

$$\begin{bmatrix} & \overset{10 \times 4096}{\mathbf{W}^*} & \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \vdots \\ \\ \end{bmatrix} + \begin{bmatrix} \overset{10 \times 1}{\mathbf{b}^*} \end{bmatrix} \Rightarrow \begin{bmatrix} -0.1 \\ 0 \\ 0 \\ 0.05 \\ 0.9 \\ 0.1 \\ 0 \\ 0.25 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \text{airplane} \\ \text{automobile} \\ \text{bird} \\ \text{cat} \\ \text{deer} \\ \text{dog} \\ \text{frog} \\ \text{horse} \\ \text{ship} \\ \text{truck} \end{matrix} \Rightarrow \text{“deer”}$$

4096×1

Example: 10-way Classifier (CIFAR-10 Dataset)

(online)

Query

$$\begin{bmatrix} & \overset{10 \times 4096}{\mathbf{W}^*} & \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \vdots \\ \\ \end{bmatrix} + \begin{bmatrix} \overset{10 \times 1}{\mathbf{b}^*} \end{bmatrix} \Rightarrow \begin{bmatrix} -0.1 \\ 0 \\ 0 \\ 0.05 \\ 0.9 \\ 0.1 \\ 0 \\ 0.25 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \text{airplane} \\ \text{automobile} \\ \text{bird} \\ \text{cat} \\ \text{deer} \\ \text{dog} \\ \text{frog} \\ \text{horse} \\ \text{ship} \\ \text{truck} \end{matrix} \Rightarrow \text{“deer”}$$

4096×1

Wait a second, anything wrong?

Example: 10-way Classifier (CIFAR-10 Dataset)

(online)

Query

$$0 \leq P_i \leq 1, \forall i$$

$$\sum_i P_i = 1$$

$$\begin{bmatrix} \mathbf{W}^* \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} + \begin{bmatrix} \mathbf{b}^* \end{bmatrix} \Rightarrow \begin{bmatrix} -0.1 \\ 0 \\ 0 \\ 0.05 \\ 0.9 \\ 0.1 \\ 0 \\ 0.25 \\ 0 \\ 0 \end{bmatrix}$$

10 x 1

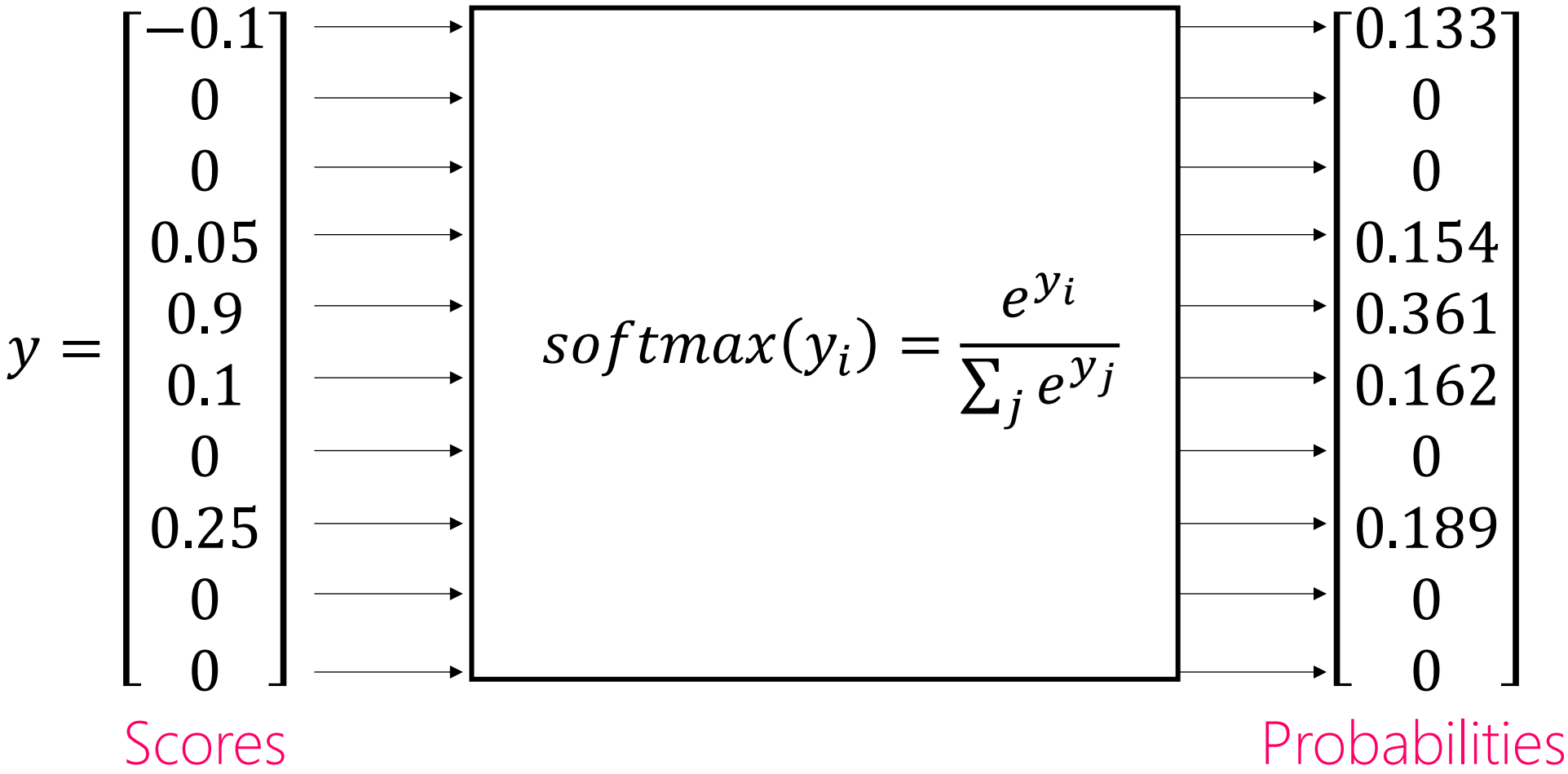
4096 x 1

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

⇒ "deer"

Wait a second, anything wrong?

Softmax!

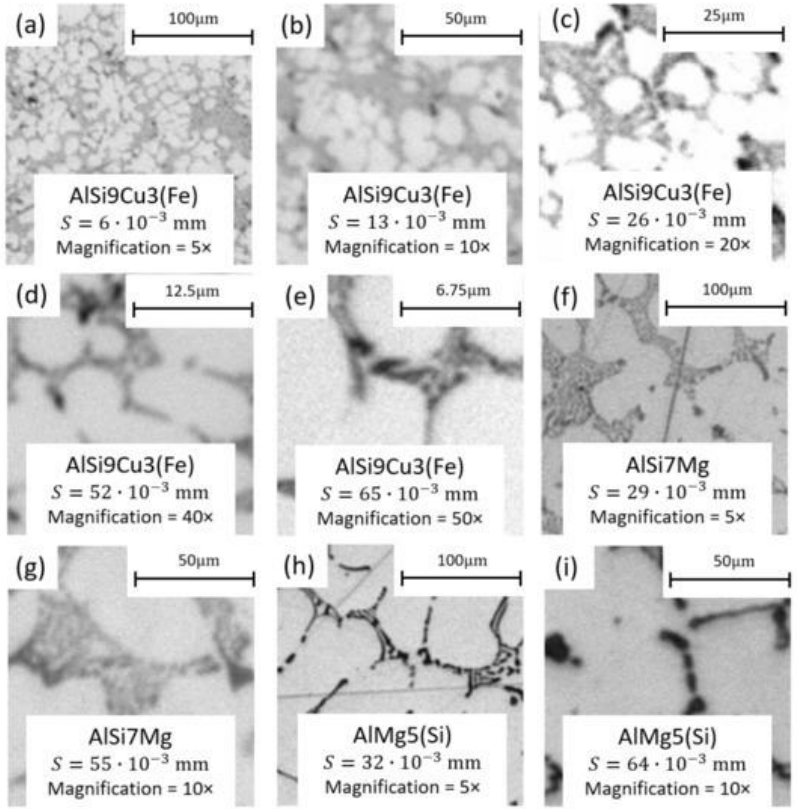

$y = \begin{bmatrix} -0.1 \\ 0 \\ 0 \\ 0.05 \\ 0.9 \\ 0.1 \\ 0 \\ 0.25 \\ 0 \\ 0 \end{bmatrix}$
$softmax(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$
$\begin{bmatrix} 0.133 \\ 0 \\ 0 \\ 0.154 \\ 0.361 \\ 0.162 \\ 0 \\ 0.189 \\ 0 \\ 0 \end{bmatrix}$
Scores
Probabilities

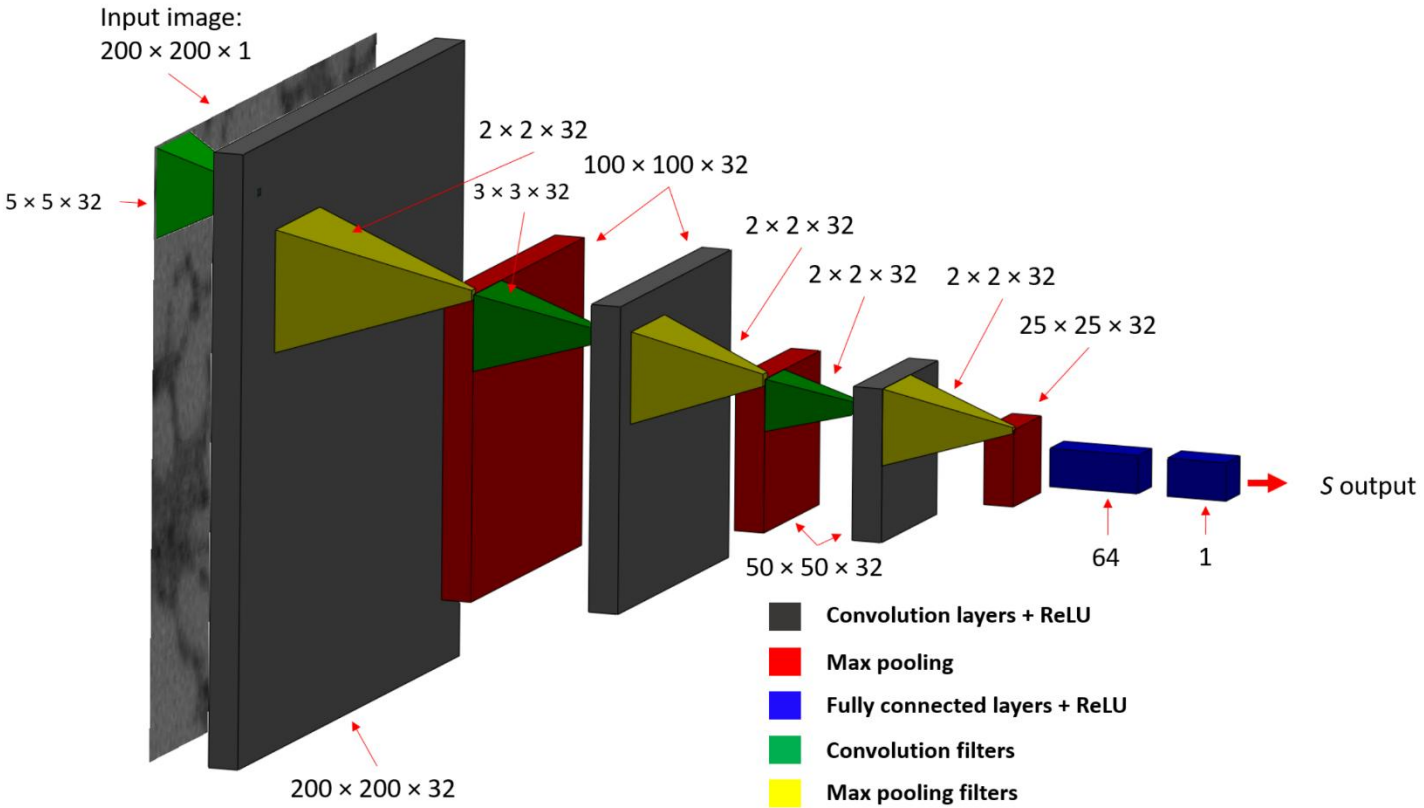
n-way Softmax Classifier: Step-by-Step

1. Collect a bunch of data $(\mathbf{x}_i, y_i)_{i=1,\cdots,N}$
2. Encode y_i in one-hot encoding
3. Split the data into two sets: training set and test set
4. Solve $\min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{N\mathbf{train}} -y_i \log(f_i)$ where $f_i = softmax\big(\sigma(\mathbf{W}\mathbf{x}_i + \mathbf{b})\big)$
5. Test the trained model $f(\mathbf{x} \mid \mathbf{W}^*, \mathbf{b}^*) = softmax\big(\sigma(\mathbf{W}^*\mathbf{x} + \mathbf{b}^*)\big)$ on the test set

Generalizing this formulation…

- Instead of $\boldsymbol{y}$ being a class probability…

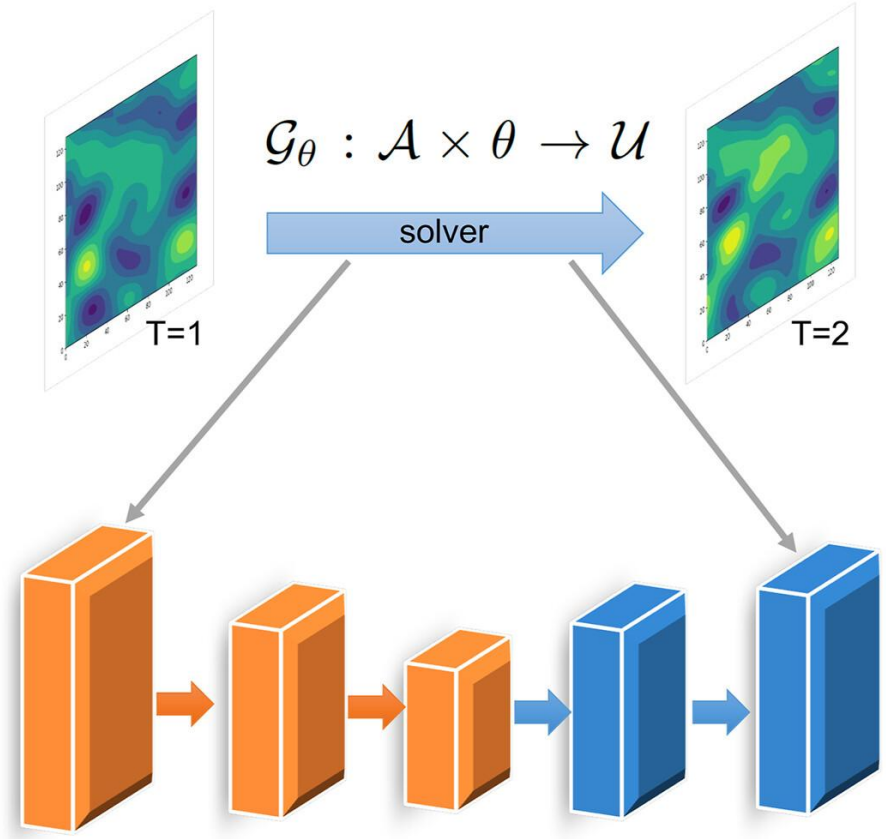


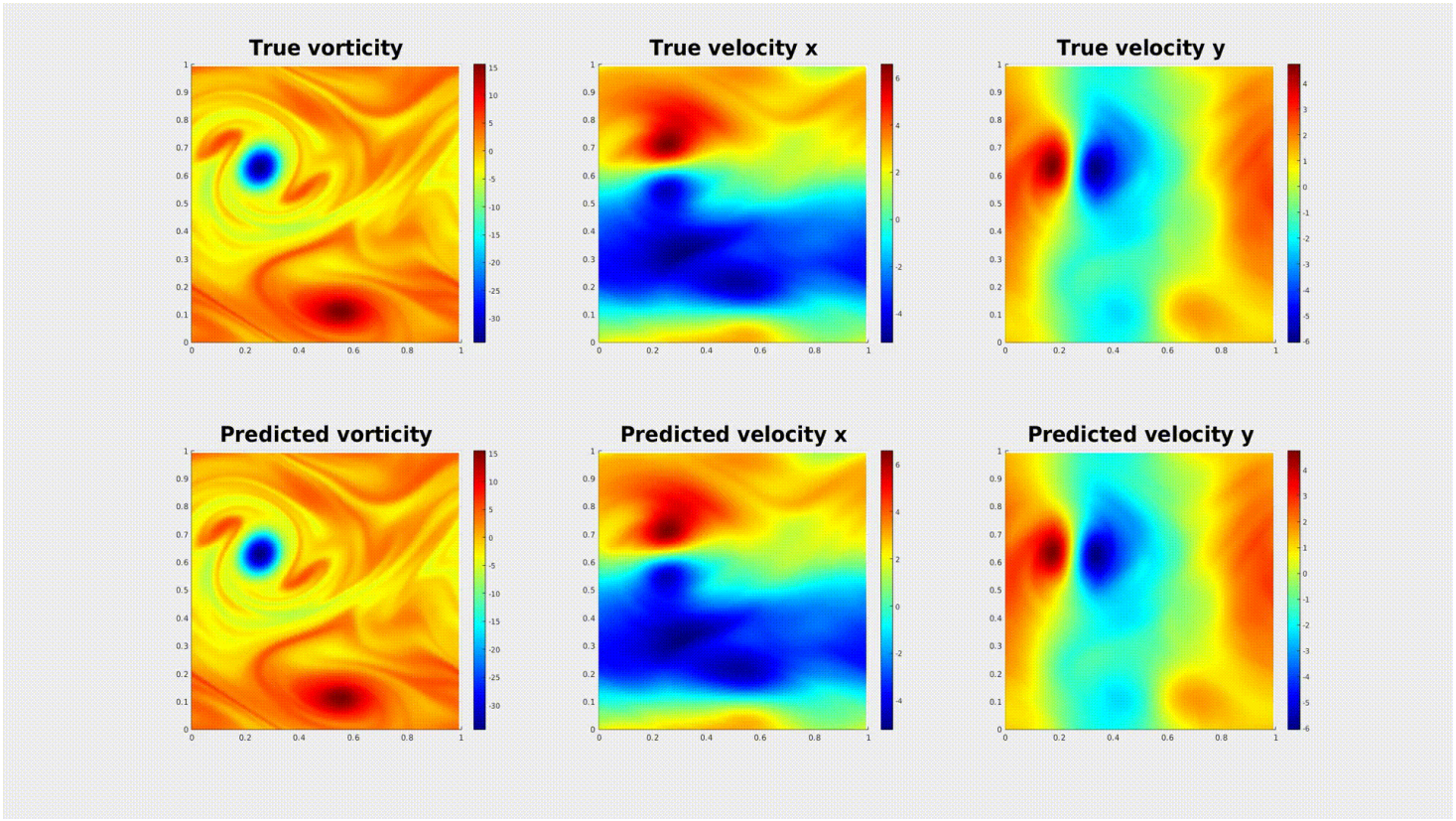


Nikolić et al. (2021). Casting Microstructure Inspection Using Computer Vision: Dendrite Spacing in Aluminum Alloys, Metals, 11(5): 756.

Generalizing this formulation…

- Instead of $\boldsymbol{y}$ being a class probability…





(Left) Zhang et al. (2024). Emulator of PR-DNS: Accelerating Dynamical Fields With Neural Operators in Particle-Resolved Direct Numerical Simulation, Adv. Modeling Earth Sys., 16(2): e2023MS003898.
(Right) https://zongyi-li.github.io/neural-operator/

More details?

- Coming up:
  - Lecture 3: Training neural networks: gradient descent, backpropagation, and more
  - Lab 1: Introduction to PyTorch—Build your first neural network

- Fun videos to watch:
  - https://www.3blue1brown.com/topics/neural-networks