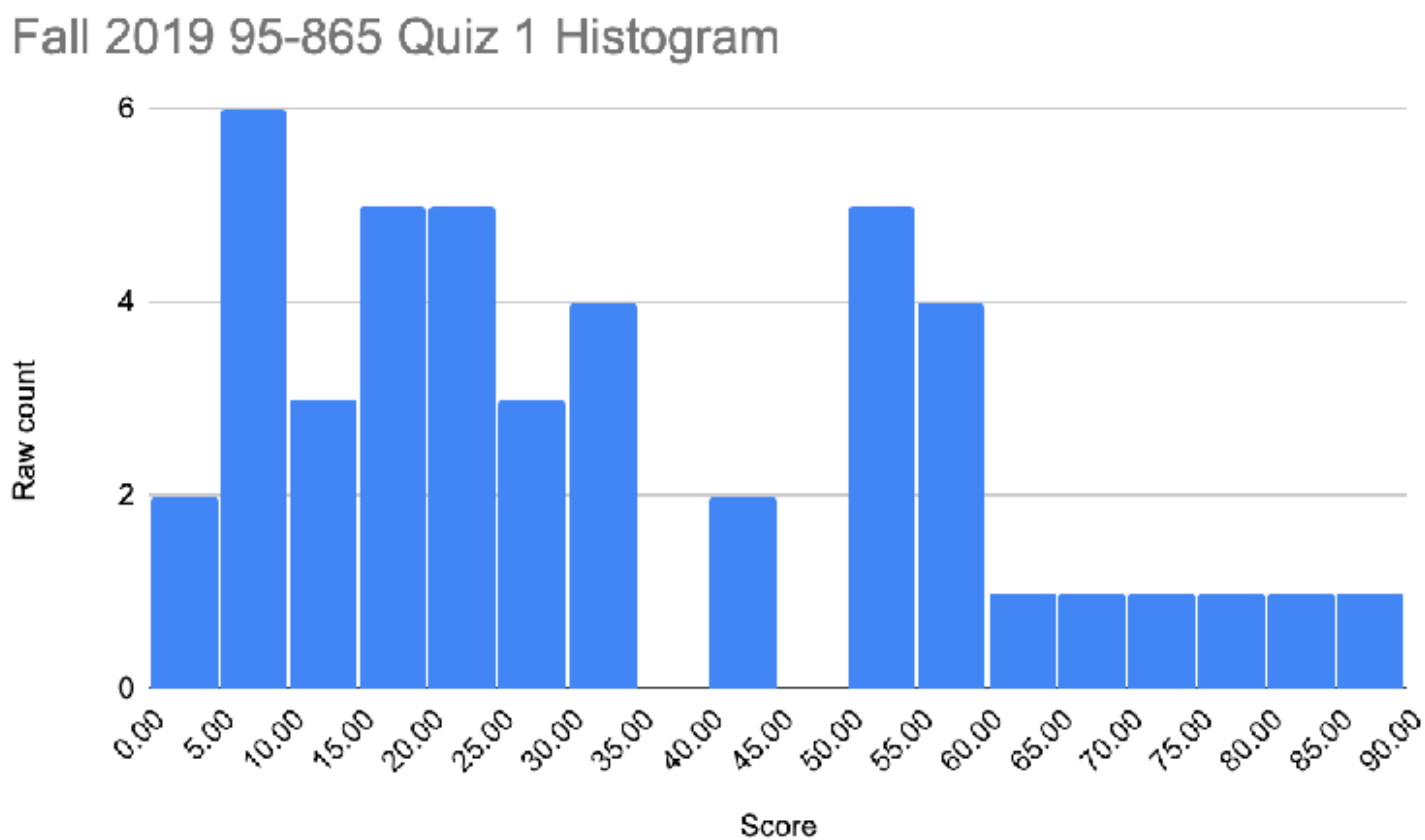
Carnegie Mellon University
Heinz College

95-865 Unstructured Data Analytics

Week 5: Intro to predictive data analytics, neural nets, and deep learning

George Chen

Quiz 1



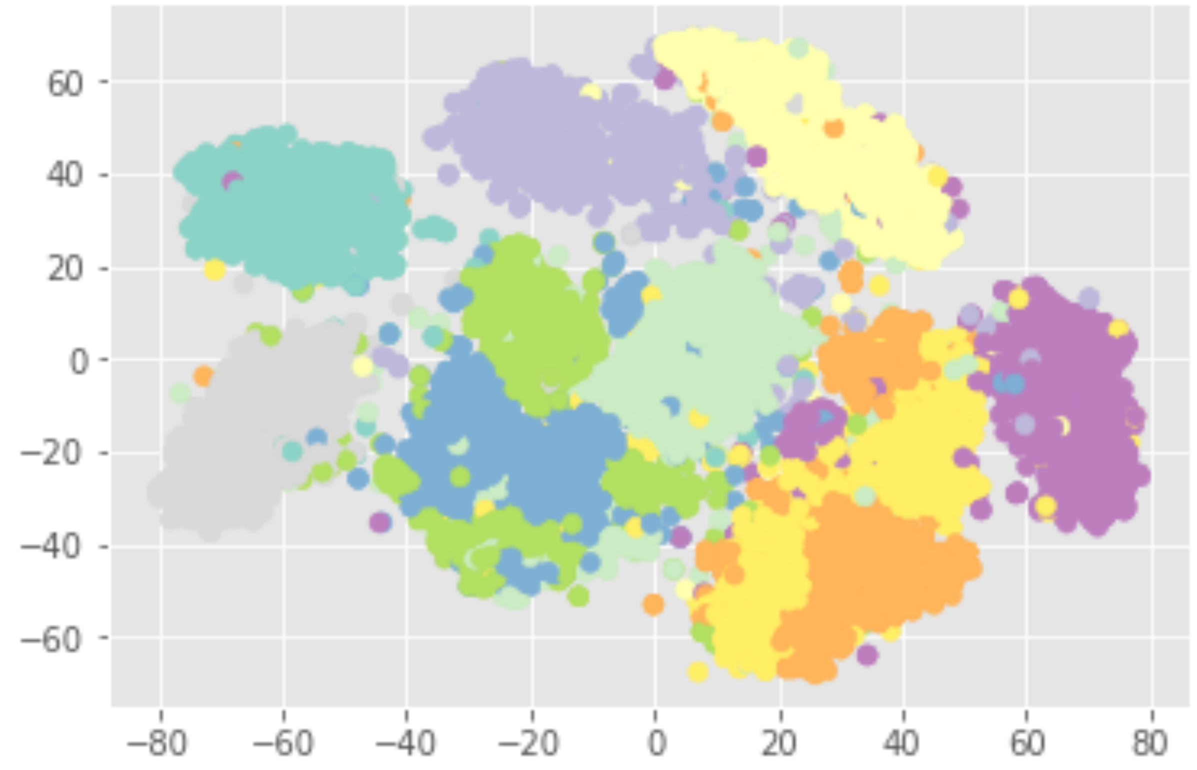
Mean: 33.3, std dev: 23.6, max achieved: 87

Quiz 1 Regrade Requests

- How regrades work:
  1. Study solutions (already posted in Canvas under “Files”) very carefully
  2. If you think there’s a mistake, send me an email and be very specific about what was incorrectly graded and how many points are at stake
  3. We will regrade your whole quiz 1 (the version that you submitted to Canvas on the quiz day)
  4. Your score can go up, go down, or stay the same, and the regraded result is final
- Due this Friday 11:59pm Pittsburgh time

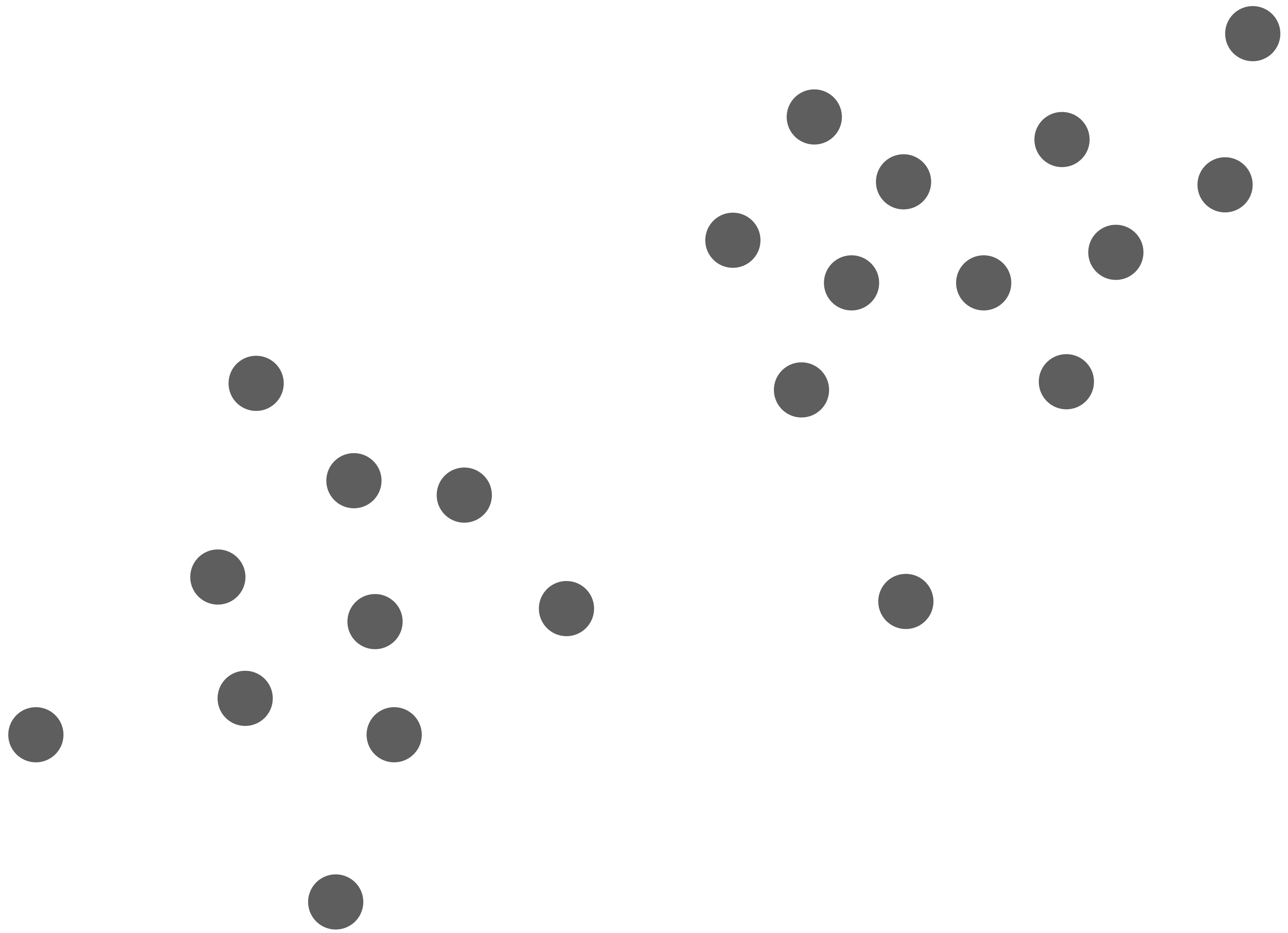
Disclaimer: unfortunately “k” means many things

What if we have labels?



Example: MNIST handwritten digits have known labels

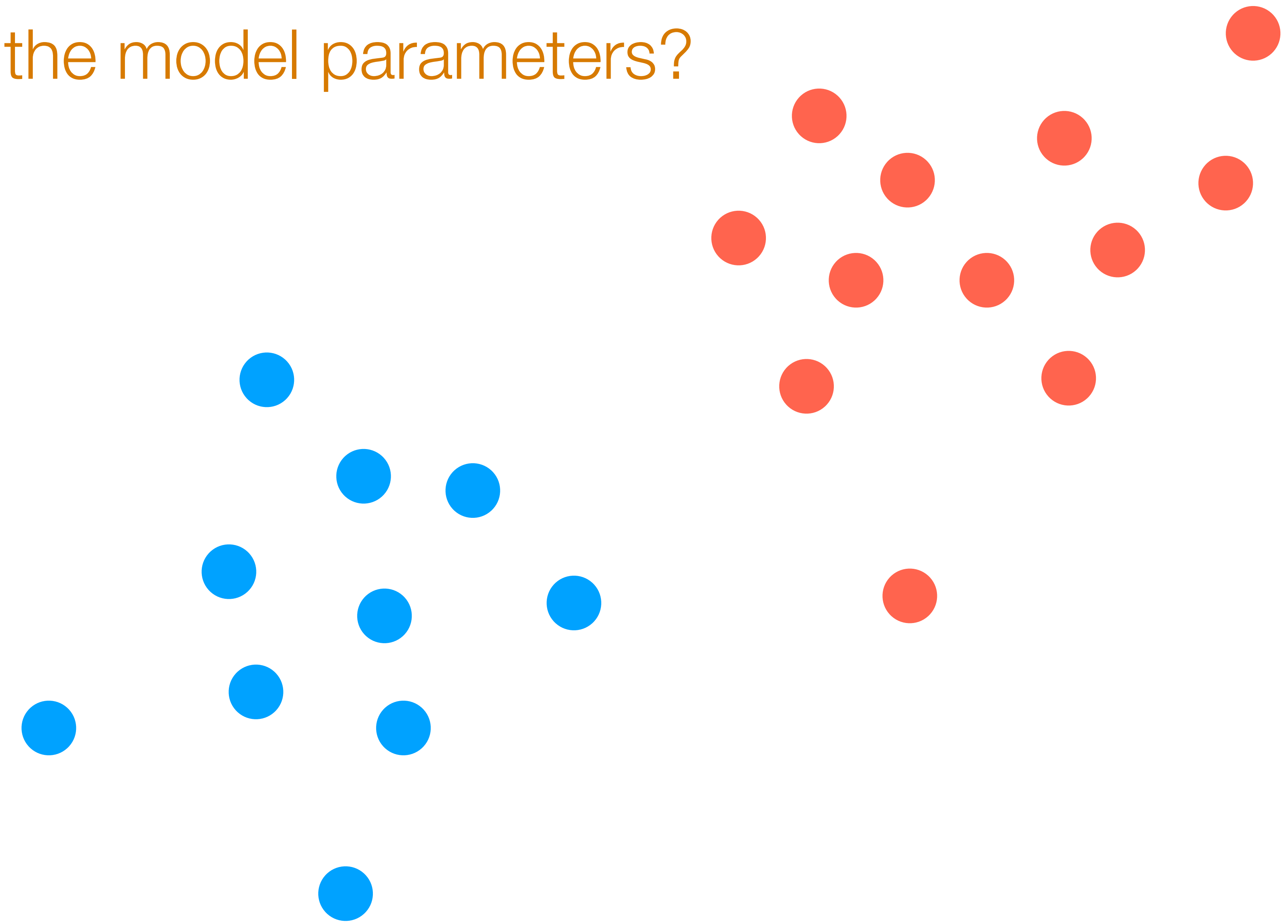
If the labels are known…



If the labels are known…

And we assume data generated by GMM…

What are the model parameters?



Flashback: Learning a GMM

Don't need this top part if we know the labels!

~~Step 0: Pick *k*~~

~~Step 1: Pick <u>guesses</u> for **cluster probabilities, means, and covariances** (often done using *k*-means)~~

~~**Repeat until convergence:**~~

~~Step 2: Compute probability of each point belonging to each of the *k* clusters~~

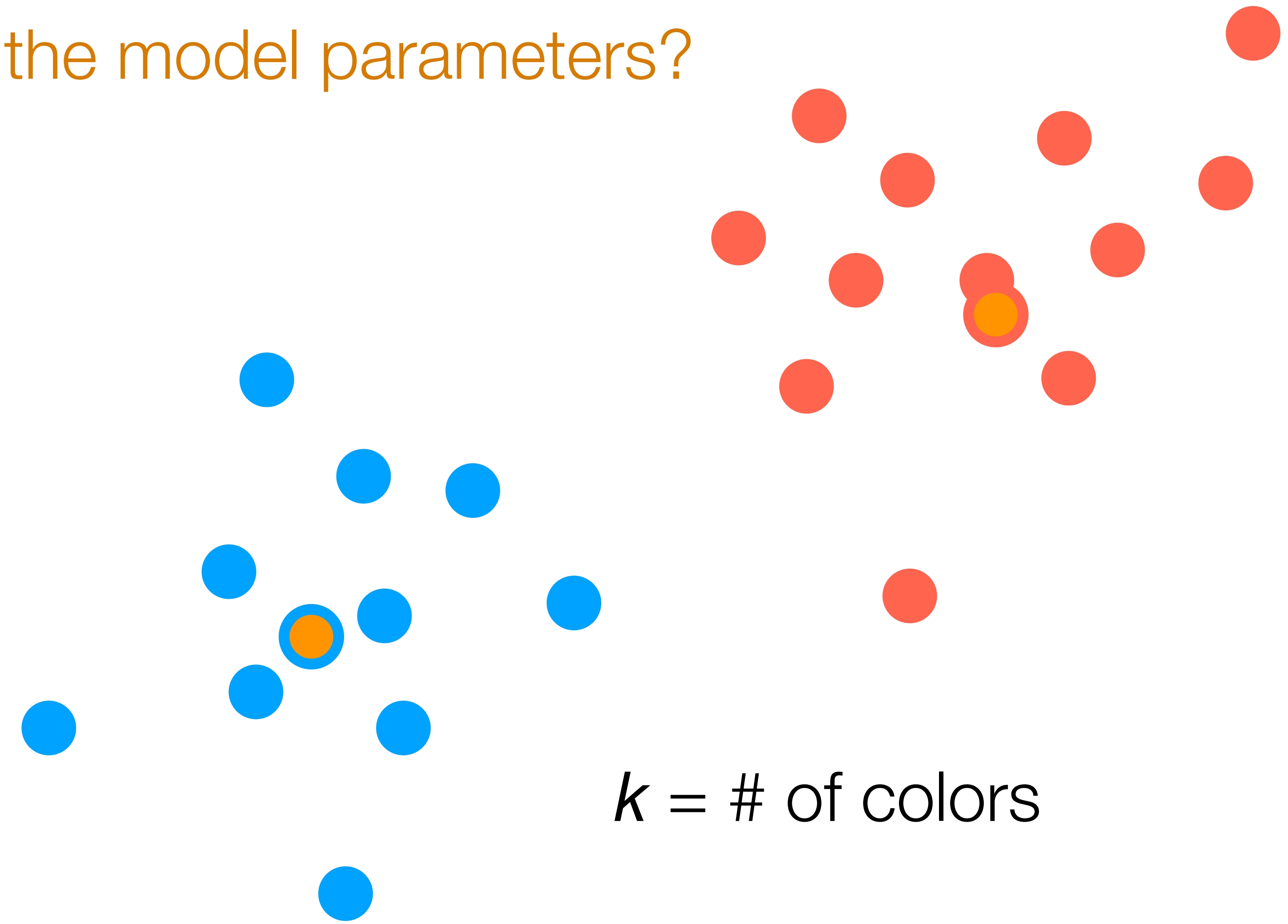
Step 3: Update **cluster probabilities, means, and covariances** carefully accounting for probabilities of each point belonging to each of the clusters

We don't need to repeat until convergence

If the labels are known…

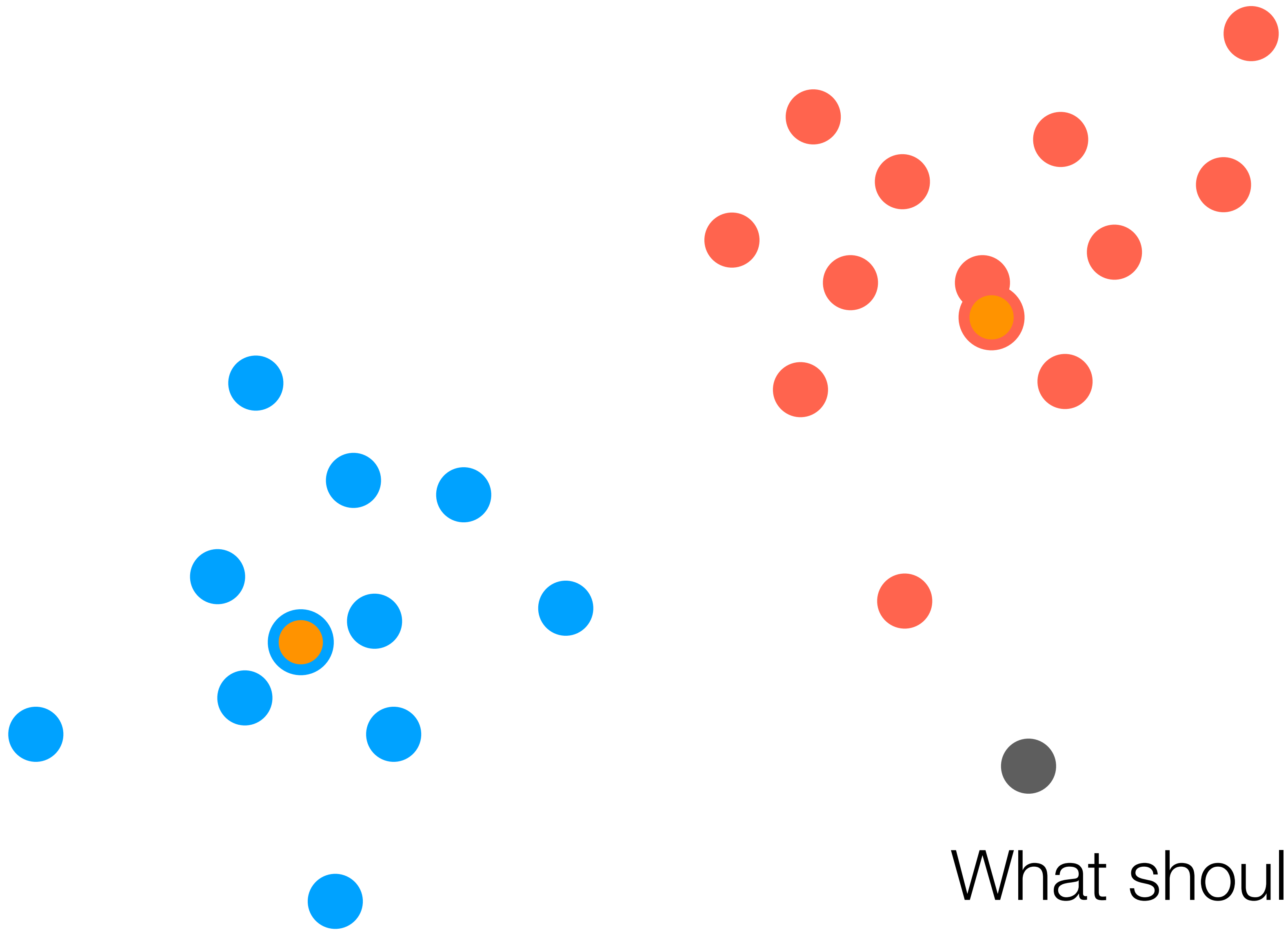
And we assume data generated by GMM…

What are the model parameters?



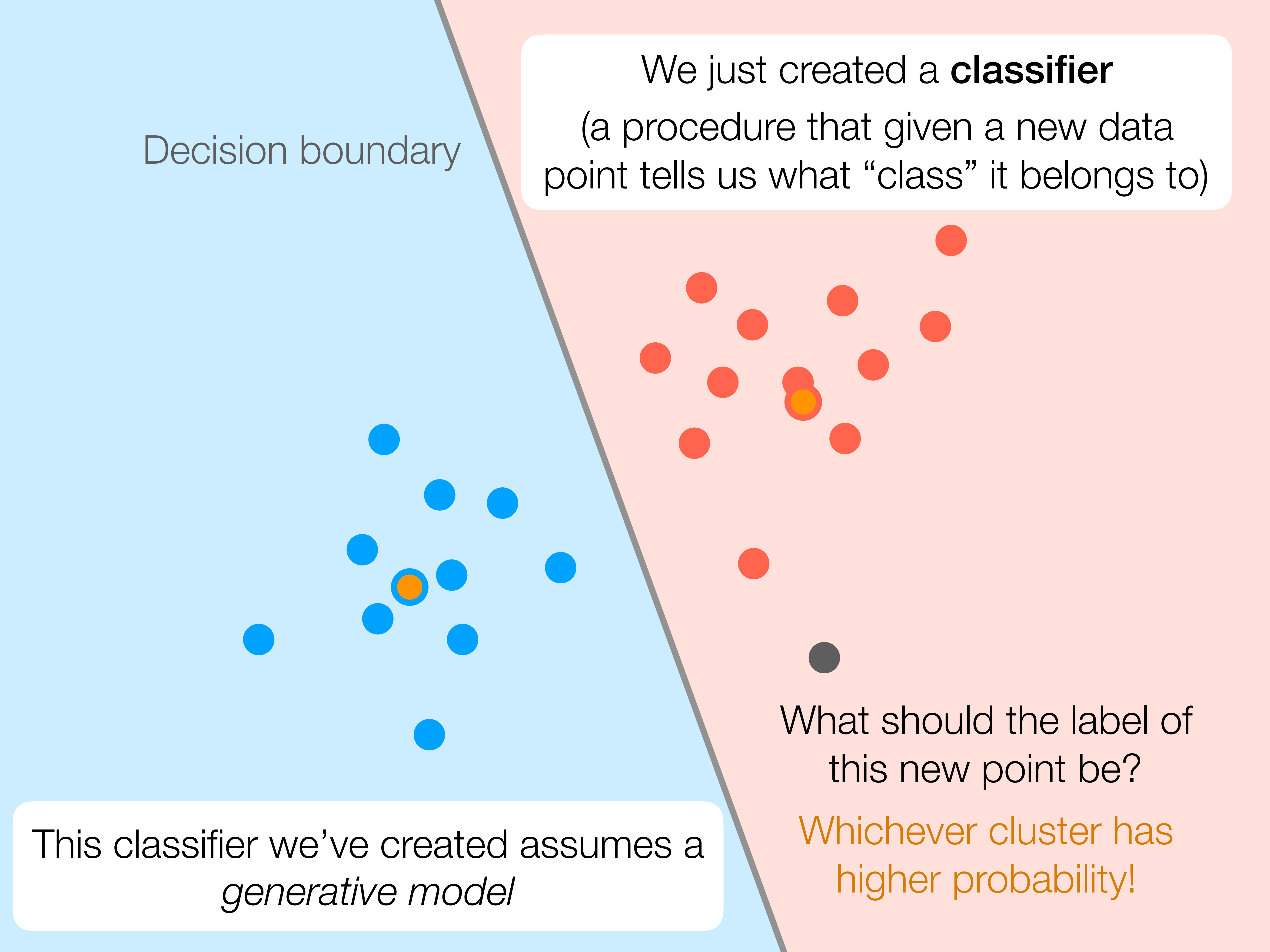
k = # of colors

We can directly estimate cluster means, covariances



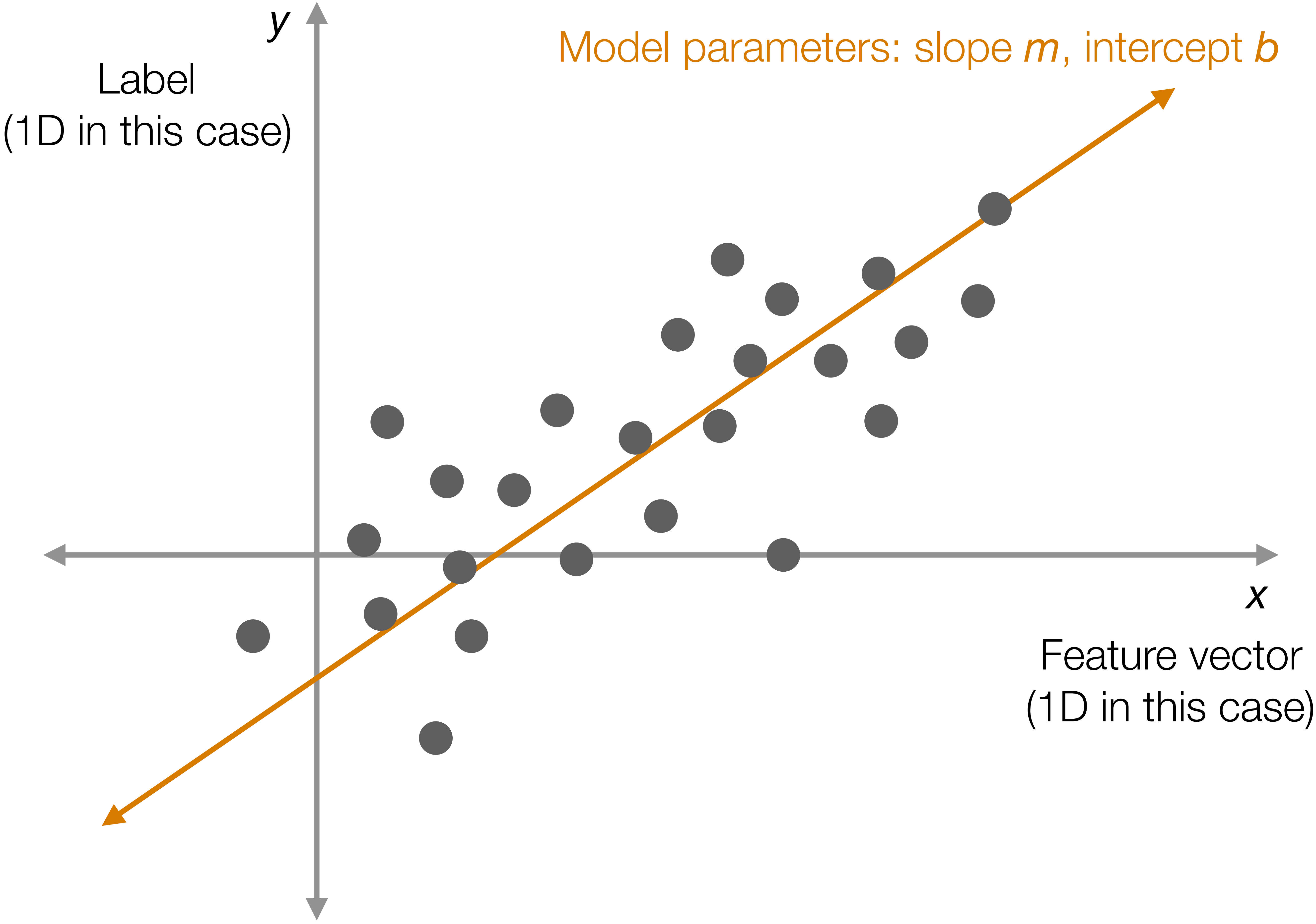
What should the label of this new point be?

Whichever cluster has higher probability!

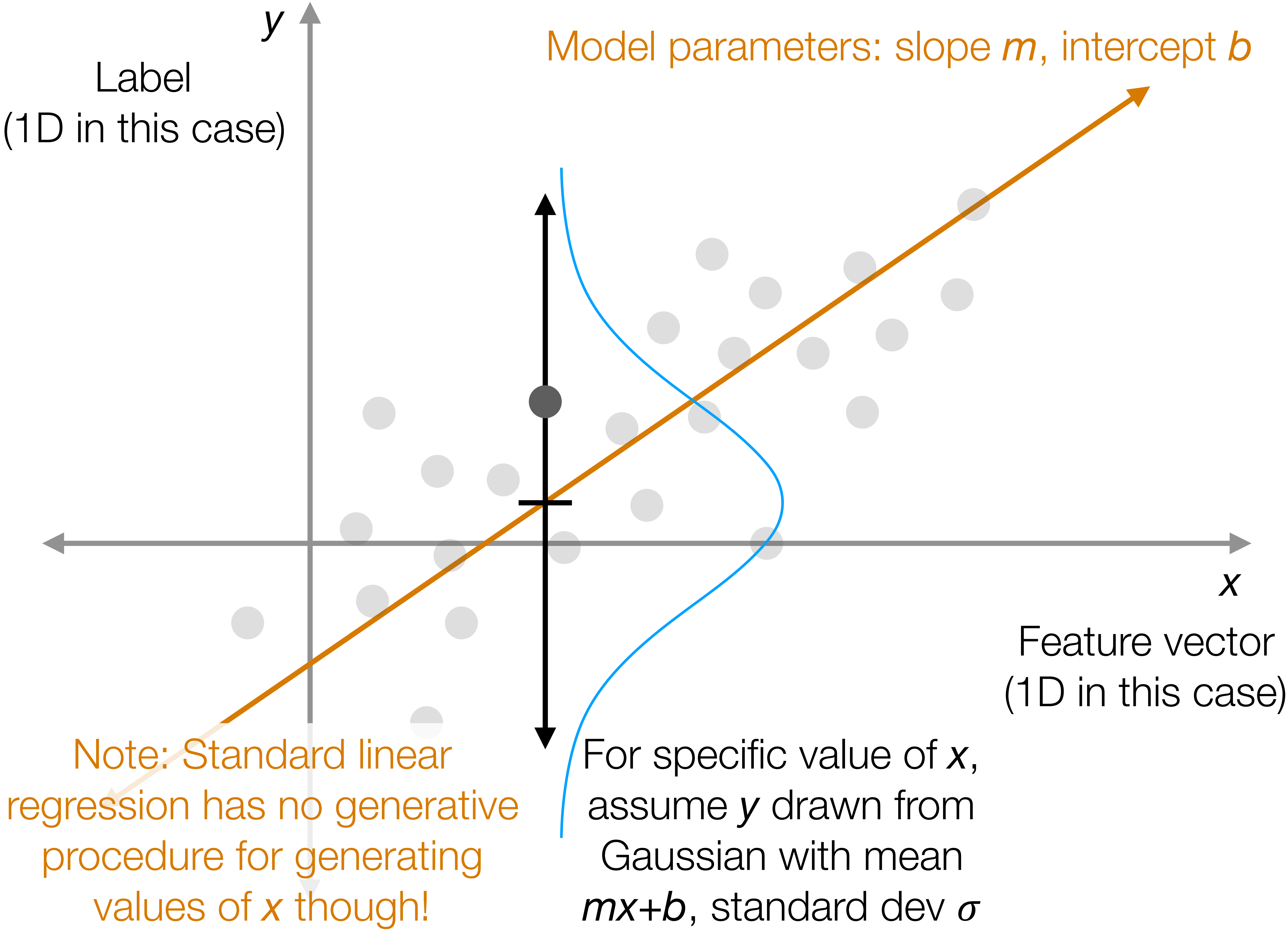

We just created a **classifier**
(a procedure that given a new data point tells us what "class" it belongs to)
Decision boundary
What should the label of this new point be?
Whichever cluster has higher probability!
This classifier we've created assumes a *generative model*

You’ve seen a prediction model that is partly a generative model

Linear regression!



y
Label
(1D in this case)
Model parameters: slope m, intercept b
x
Feature vector
(1D in this case)



y
Label
(1D in this case)
Model parameters: slope m, intercept b
x
Feature vector
(1D in this case)
Note: Standard linear regression has no generative procedure for generating values of x though!
For specific value of x, assume y drawn from Gaussian with mean mx+b, standard dev σ

Predictive Data Analysis

Training data

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

Goal: Given new feature vector x, predict label y

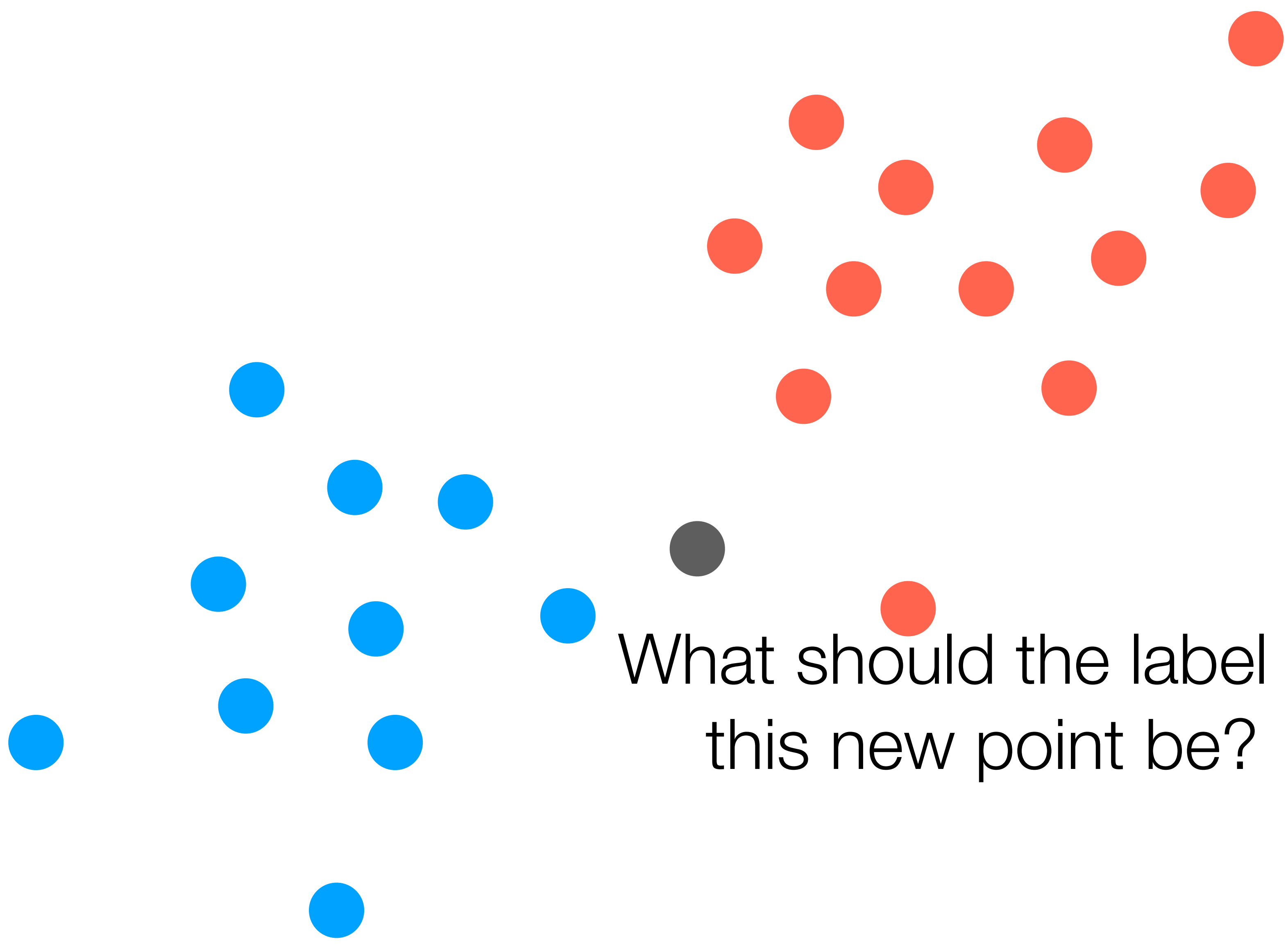
- y is discrete (such as colors red and blue)
  ➔ prediction method is called a **classifier**
- y is continuous (such as a real number)
  ➔ prediction method is called a **regressor**

A giant zoo of methods

- Generative models (like what we just described)
- Discriminative methods (just care about learning prediction rule *without* assuming generative model)

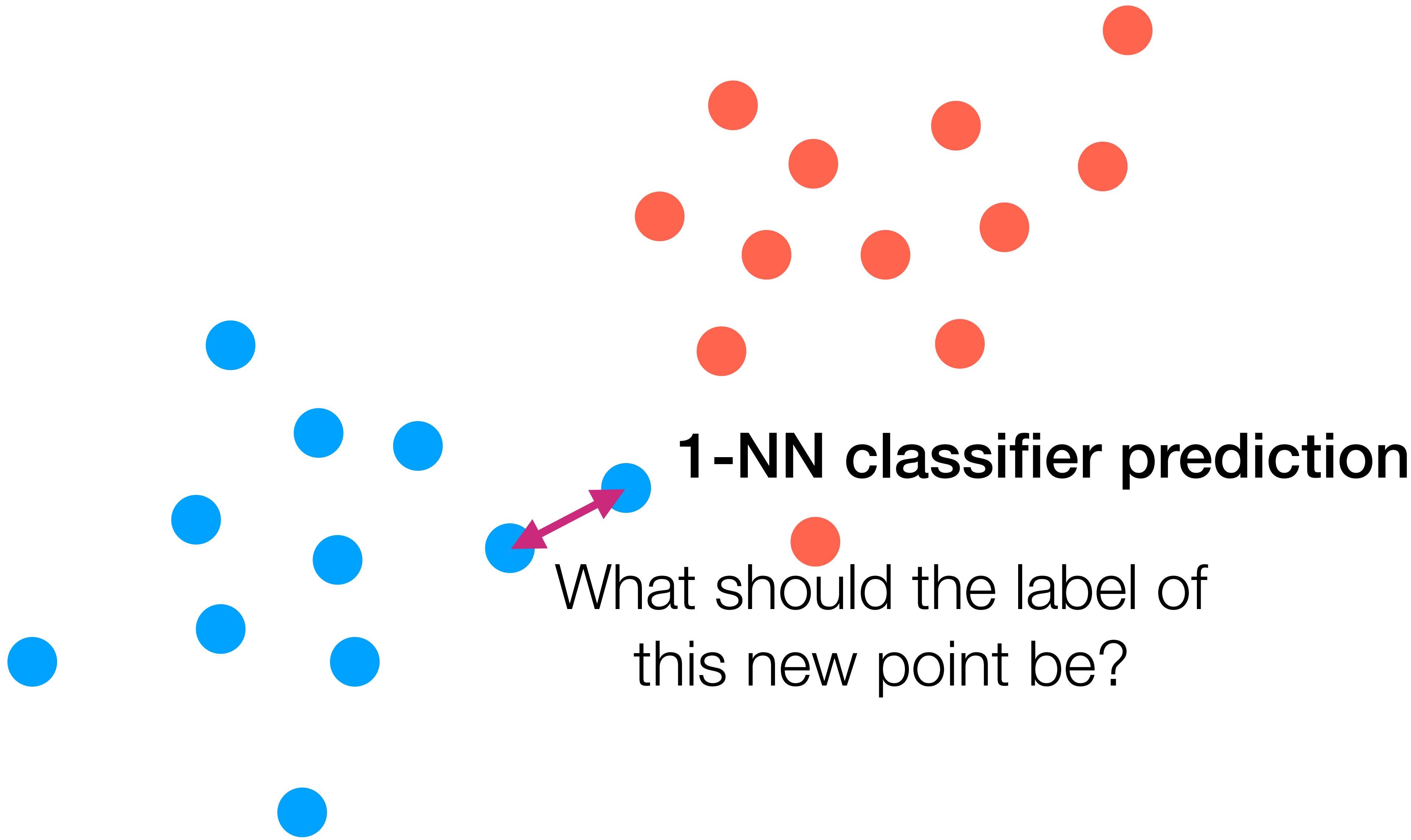
Example of a Discriminative Method: *k*-NN Classification

Example: *k*-NN Classification

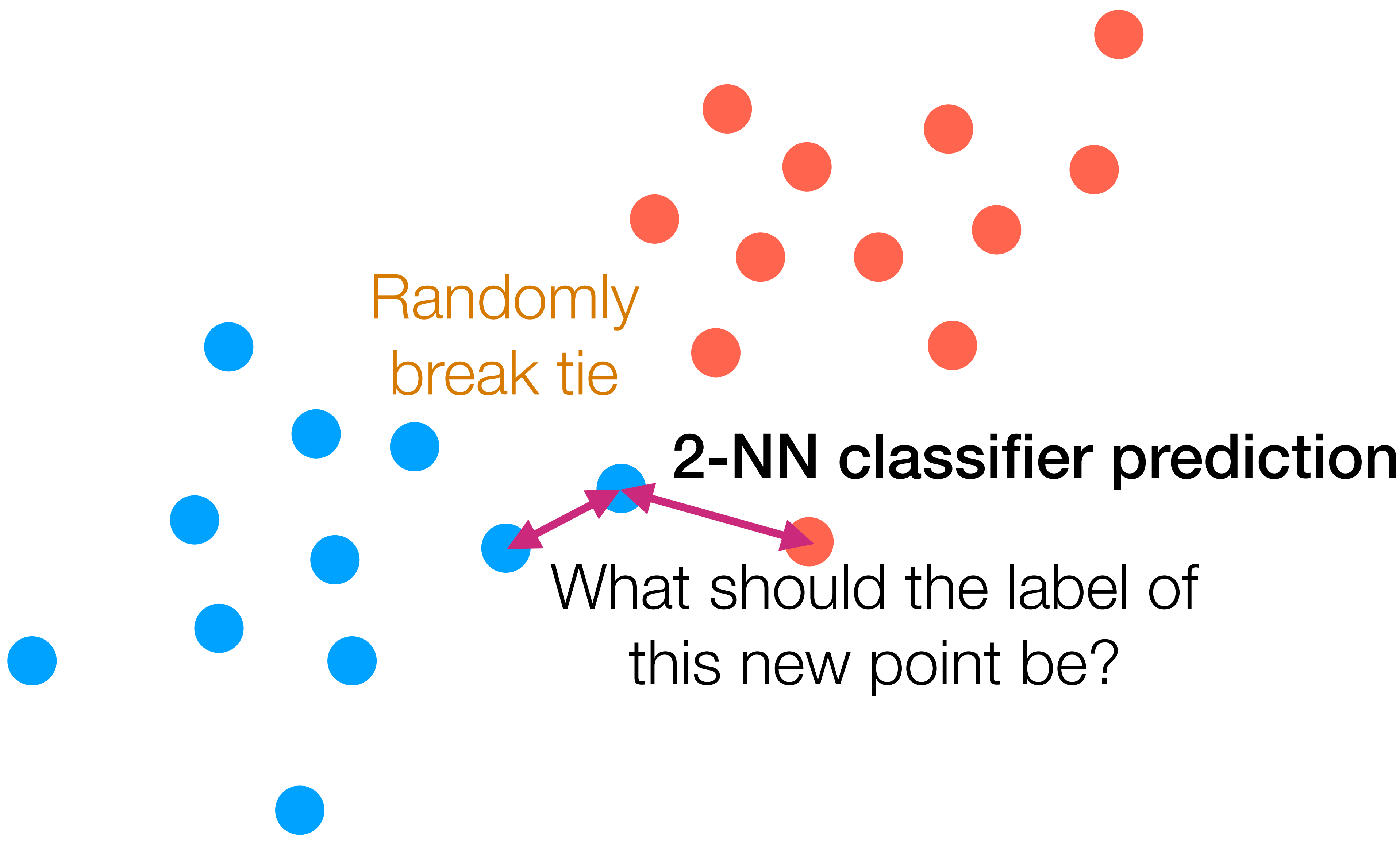


What should the label of
this new point be?

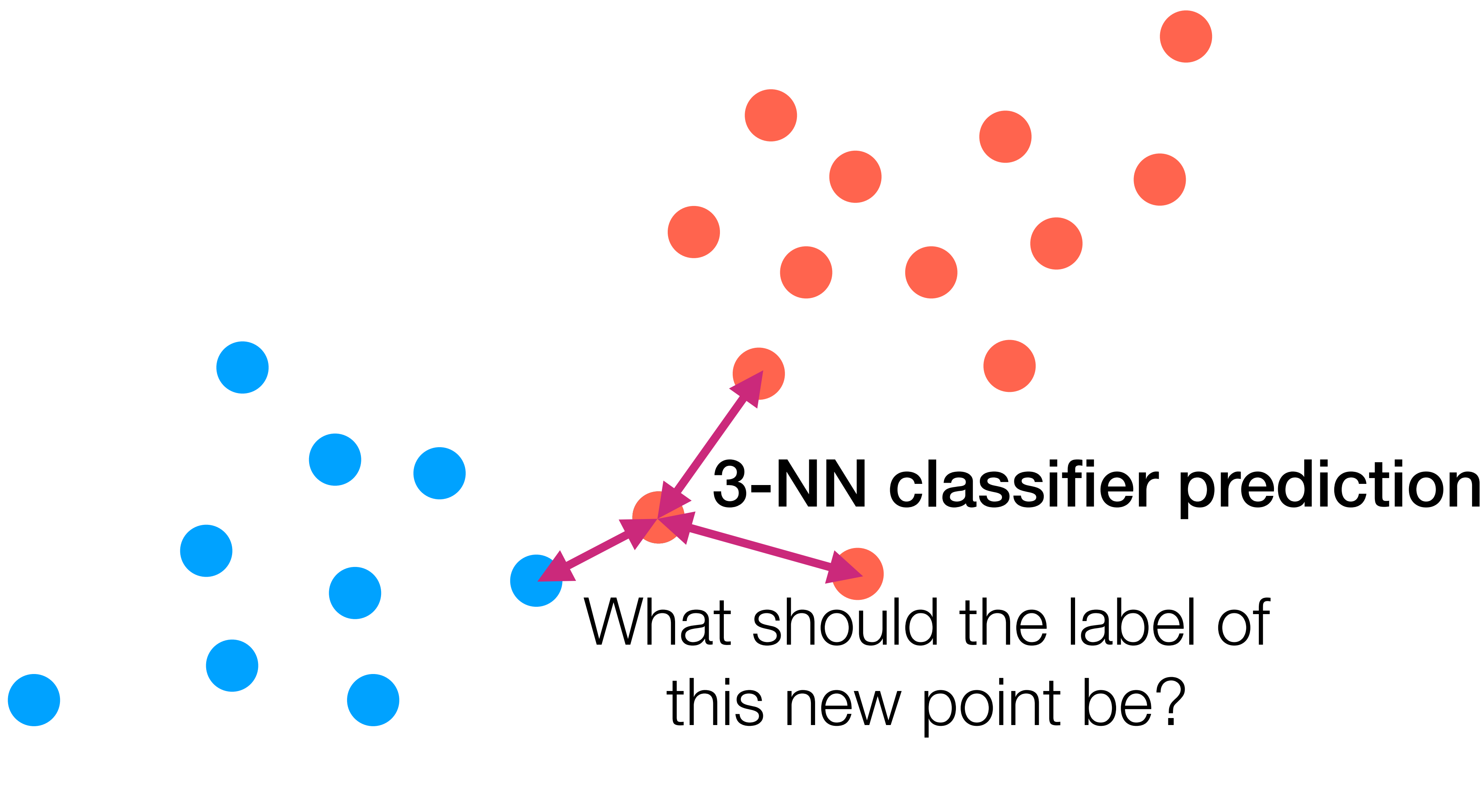
Example: *k*-NN Classification



Example: *k*-NN Classification



Example: *k*-NN Classification



We just saw: k = 1, k = 2, k = 3

What happens if $k = n$?

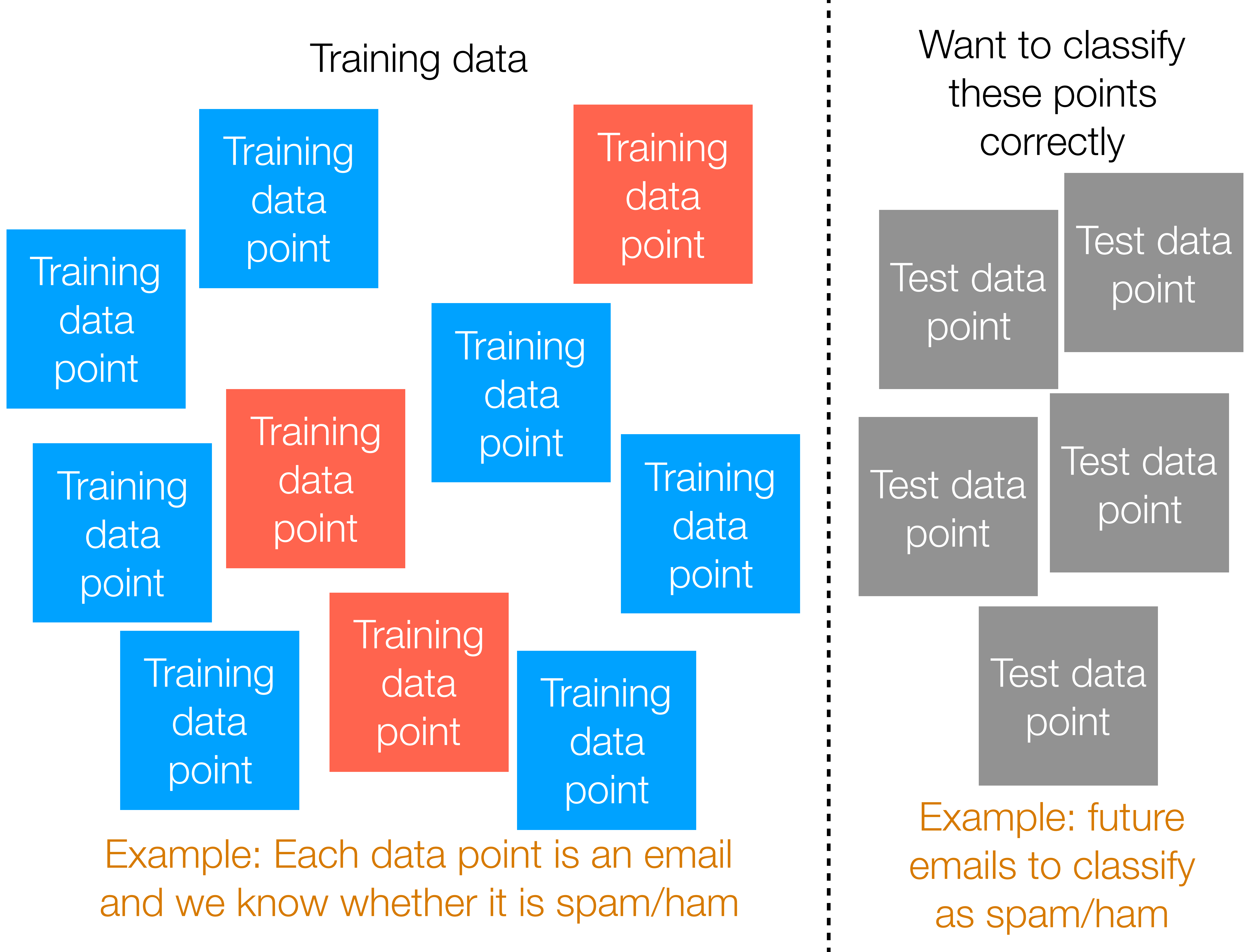
How do we choose k?

What I'll describe next can be used to select hyperparameter(s) for any prediction method

First: How do we assess how good a prediction method is?

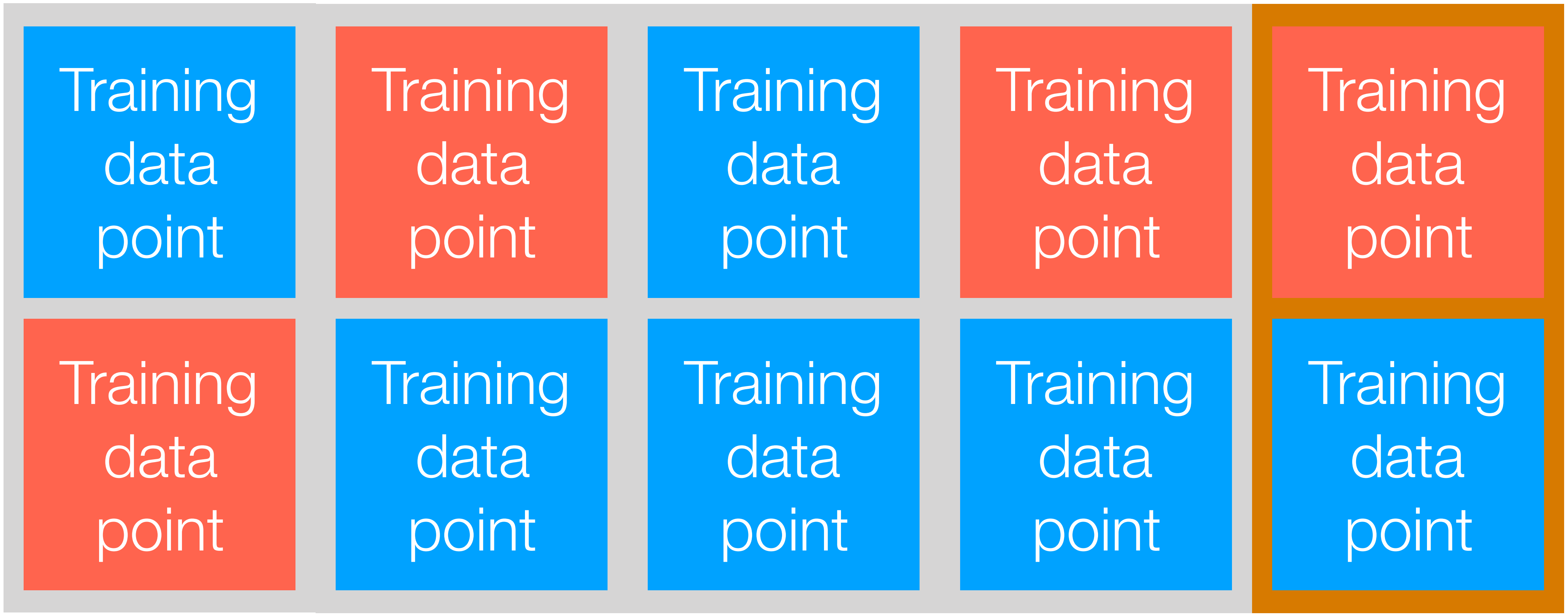
Hyperparameters vs. Parameters

- We fit a model's parameters to training data (terminology: we "learn" the parameters)
- We pick values of hyperparameters and they do *not* get fit to training data
- Example: Gaussian mixture model
  - Hyperparameter: number of clusters k
  - Parameters: cluster probabilities, means, covariances
- Example: k-NN classification
  - Hyperparameter: number of nearest neighbors k
  - Parameters: N/A



Training data
Training data point
Training data point
Training data point
Training data point
Training data point
Training data point
Training data point
Training data point
Training data point
Training data point
Example: Each data point is an email and we know whether it is spam/ham
Want to classify these points correctly
Test data point
Test data point
Test data point
Test data point
Test data point
Example: future emails to classify as spam/ham

Predicted labels



Train method on data in gray

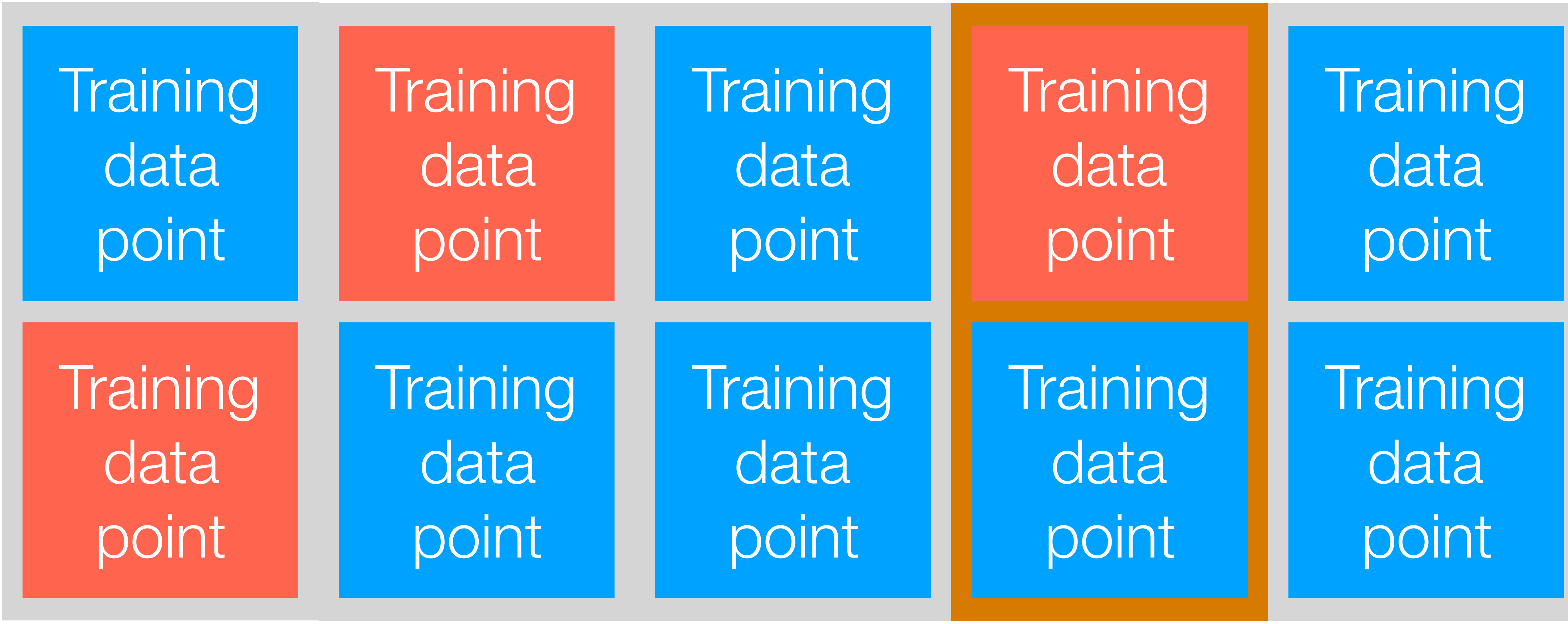
Predict on data in orange

Compute prediction error

50%

Simple data splitting (commonly called train/test split)

In this example: we did a 80%-20% split



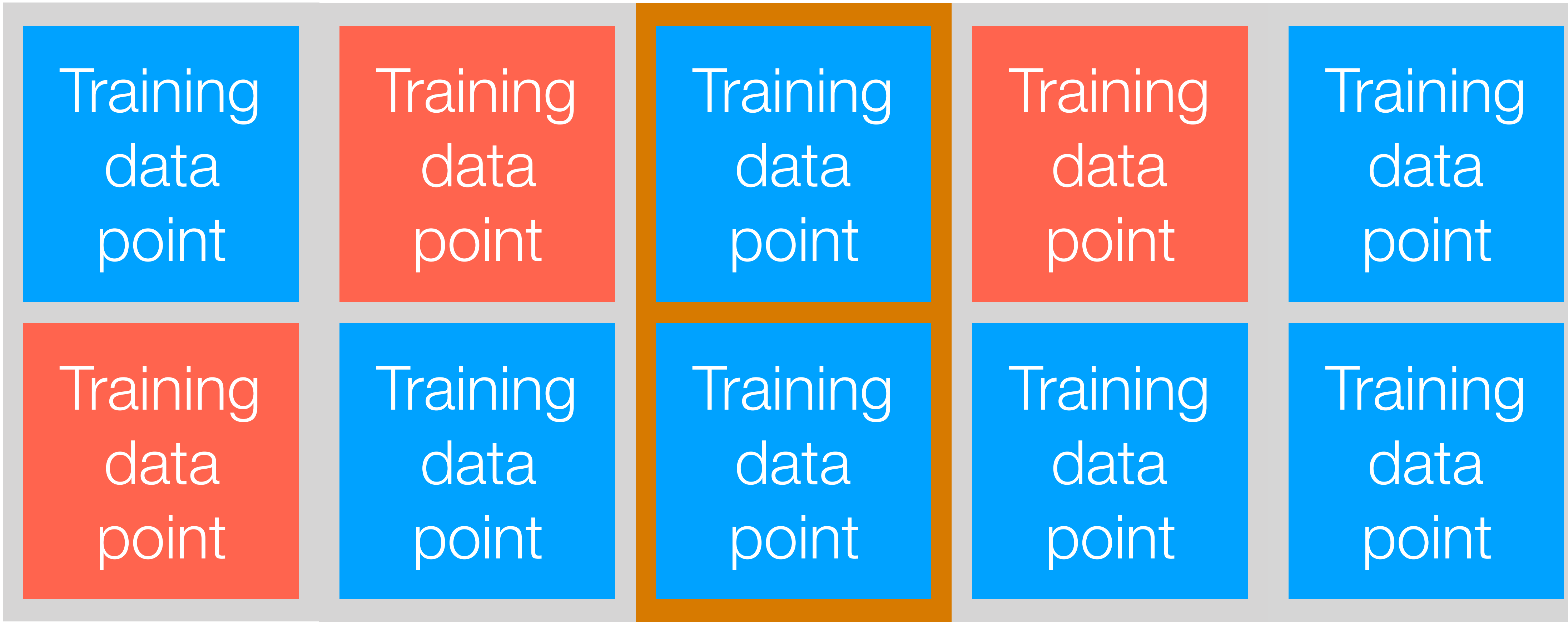
Train method on data in gray

Predict on data in orange

Compute prediction error

0%

50%

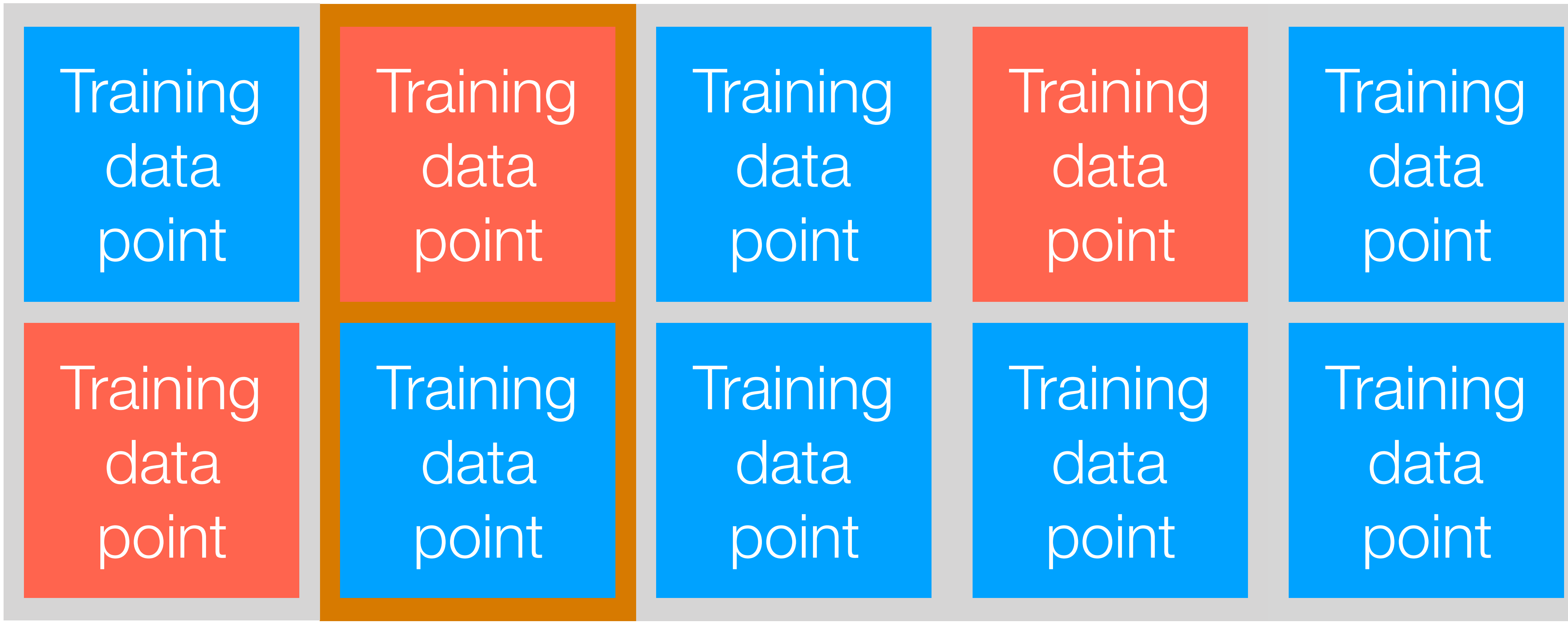


Train method on data in gray

Predict on data in orange

Compute prediction error

50% 0% 50%

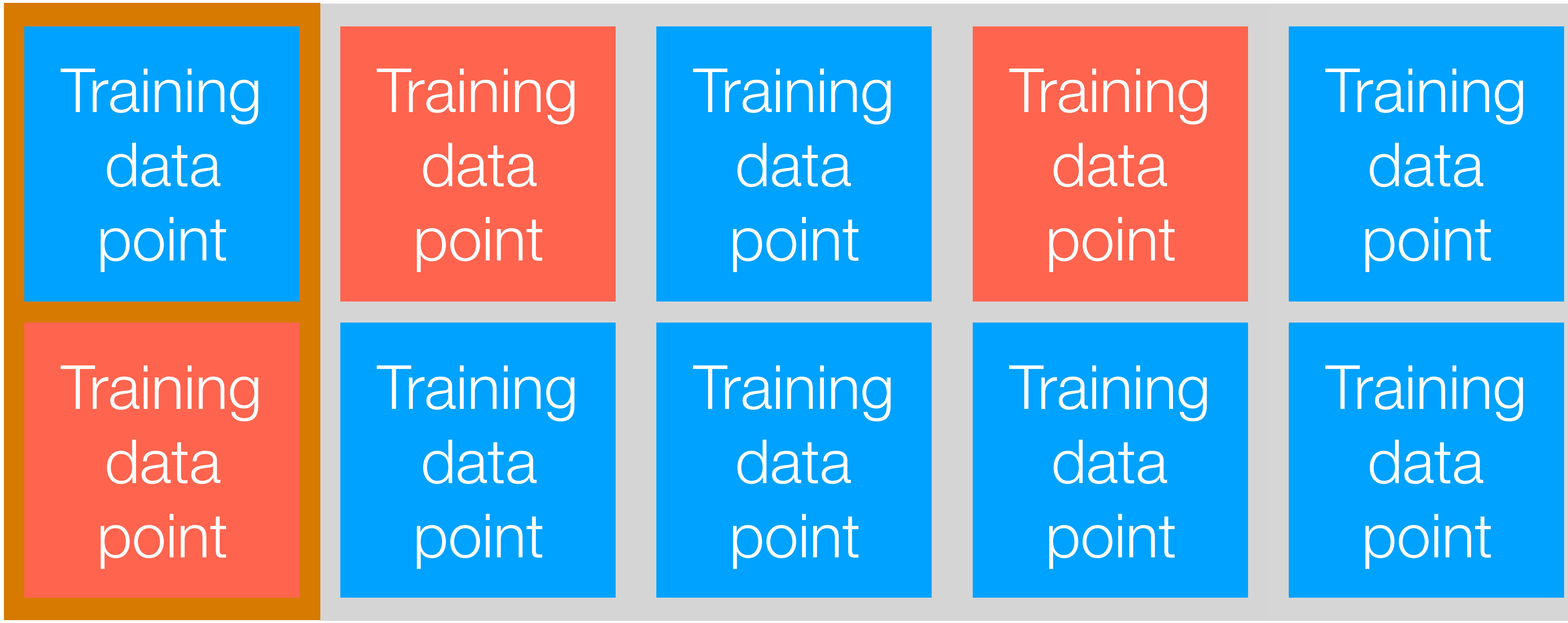


Train method on data in gray

Predict on data in orange

Compute prediction error

0% 50% 0% 50%



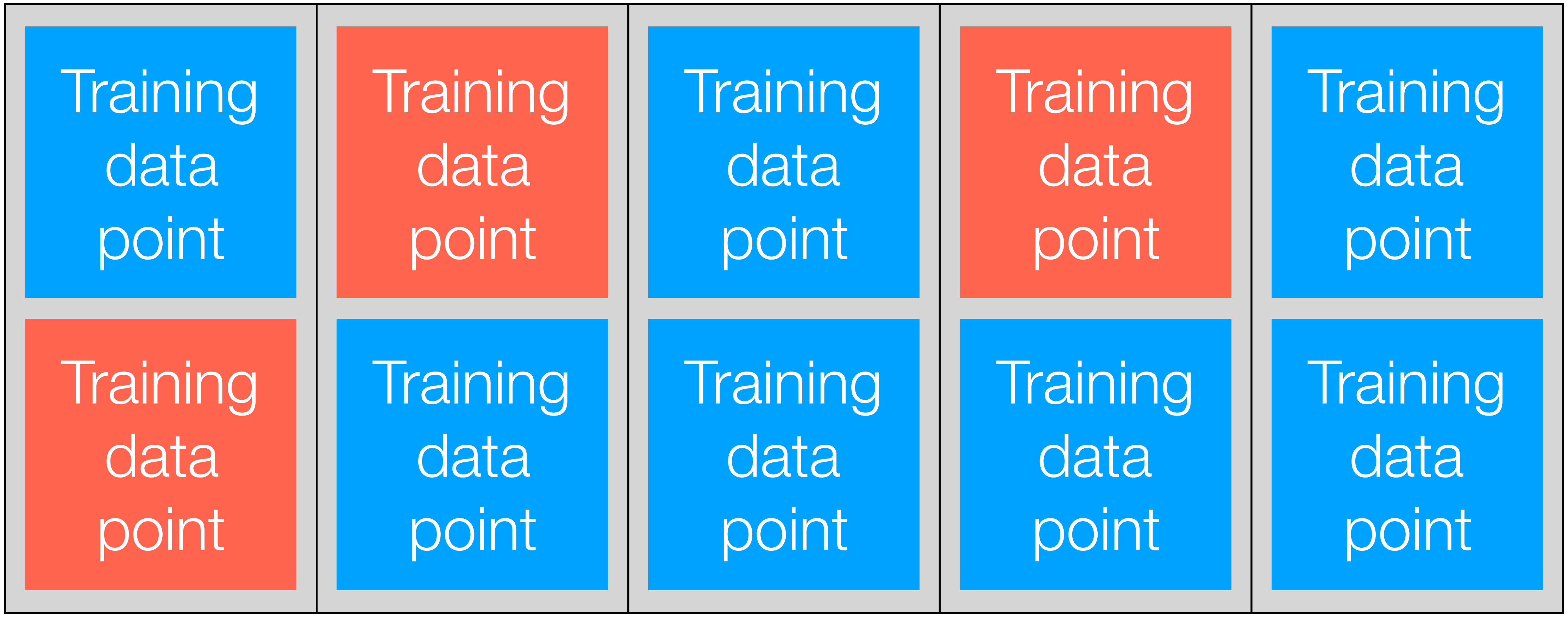
Train method on data in gray

Predict on data in orange

Compute prediction error

0% 0% 50% 0% 50%

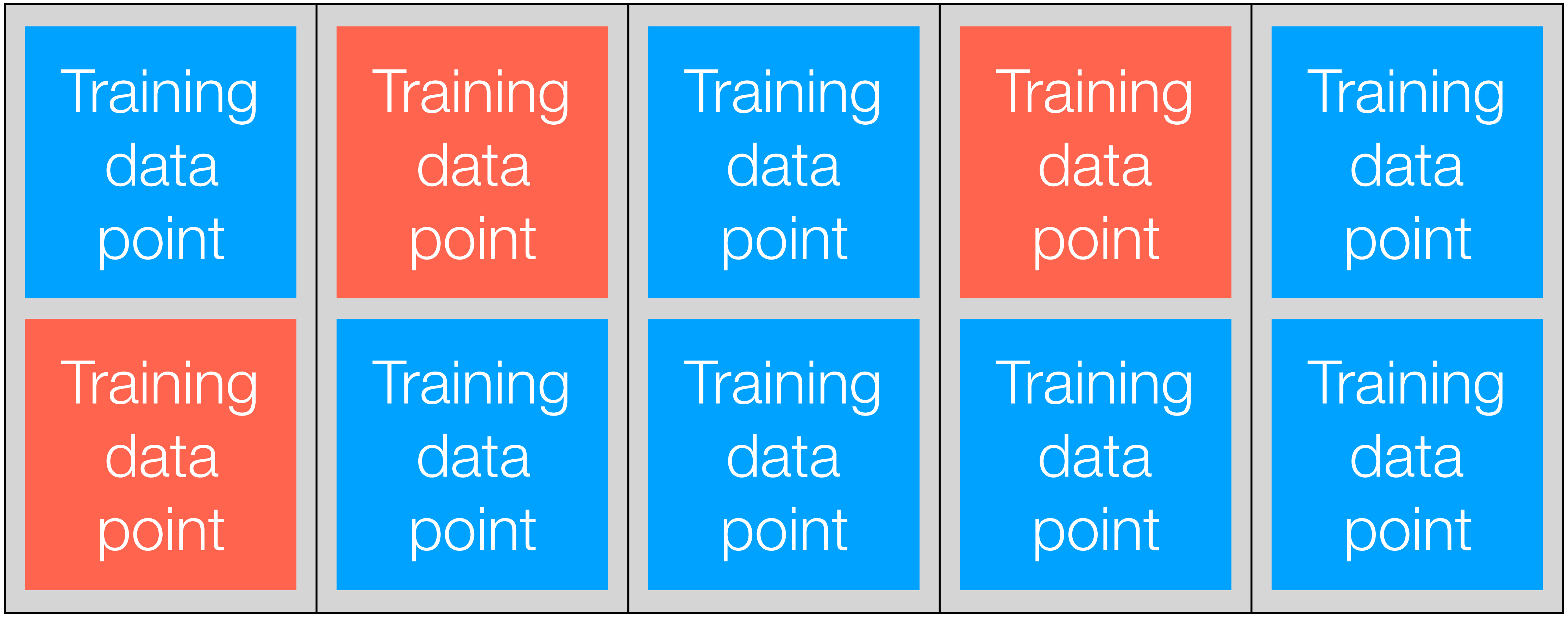
Average error: (0+0+50+0+50)/5 = 20%



1. Shuffle data and put them into "folds" (5 folds in this example)
2. For each fold (which consists of its own train/validation sets):
   (a) Train on fold's training data, test on fold's validation data
   (b) Compute prediction error
3. Compute average prediction error across the folds

not the same *k* as in *k*-means or *k*-NN classification

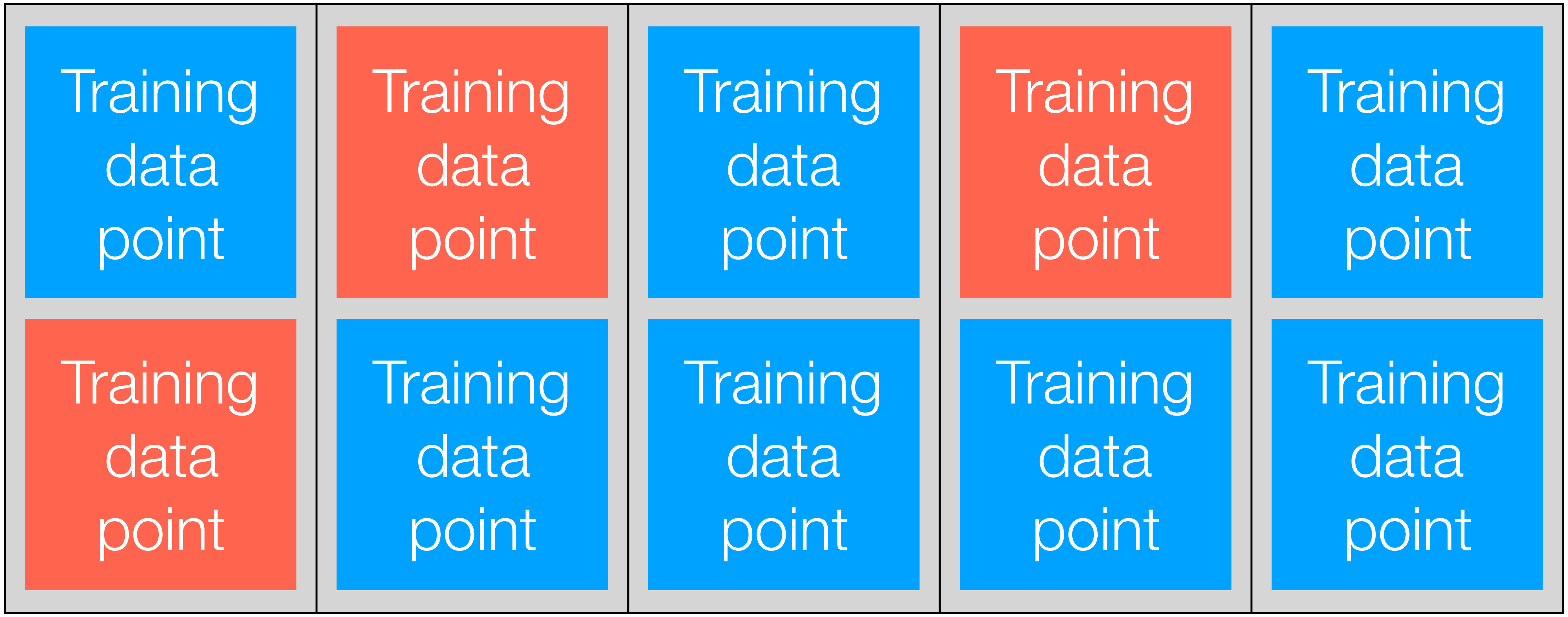
k-fold Cross Validation



1. Shuffle data and put them into "folds" (*k*=5 folds in this example)
2. For each fold (which consists of its own train/validation sets):
   (a) Train on fold's training data, test on fold's validation data
   (b) Compute prediction error
3. Compute average prediction error across the folds

not the same *k* as in *k*-means or *k*-NN classification

k-fold Cross Validation



1. Shuffle data and put them into “folds” (*k*=5 folds in this example)

2. For each fold (which consists of its own train/validation sets):
   (a) Train on fold’s training data, test on fold’s validation data
   (b) Compute **some sort of prediction score**

3. Compute **average prediction score** across the folds
   “cross validation score”

Choosing *k* in *k*-NN Classification

Note: *k*-NN classifier has a single hyperparameter *k*

For each *k* = 1, 2, 3, …, the maximum *k* you are willing to try:

- Compute 5-fold cross validation score using *k*-NN classifier as prediction method

Use whichever *k* has the best cross validation score

Automatic Hyperparameter Selection

Suppose the prediction algorithm you're using has hyperparameters θ

For each hyperparameter setting θ you are willing to try:

> Compute 5-fold cross validation score using your algorithm with hyperparameters θ

Use whichever θ has the best cross validation score

Why 5?

People have found using 10 folds or 5 folds to work well in practice but it's just empirical — there's no deep reason

Training data

Important: the errors from simple data splitting and cross-validation are *estimates* of the true error on test data!

Example: earlier, we got a cross validation score of 20% error

This is a guess for the error we will get on test data

This guess is <u>not</u> always accurate!

Example: Each data point is an email and we know whether it is spam/ham

Want to classify these points correctly

Test data point

Test data point

Test data point

Test data point

Test data point

Example: future emails to classify as spam/ham

Cross-Validation Remarks

- k-fold cross-validation is a randomized procedure
    - Re-running CV results in different cross-validation scores!
- Suppose there are n training data points and k folds
    - If we are trying 10 different hyperparameter settings, how many times do we do model fitting? $10k$
        - If this number is similar in size to n, CV can overfit!
    - How many training data are used in each model fit during cross-validation? $[(k-1)/k]n$
        - Smaller # folds typically means faster training
- If $k = n$, would re-running cross-validation result in different cross-validation scores? What about $k = 2$?
    - For deterministic training procedure: same CV result for $k = n$ (since shuffling doesn't matter), different for $k = 2$

Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

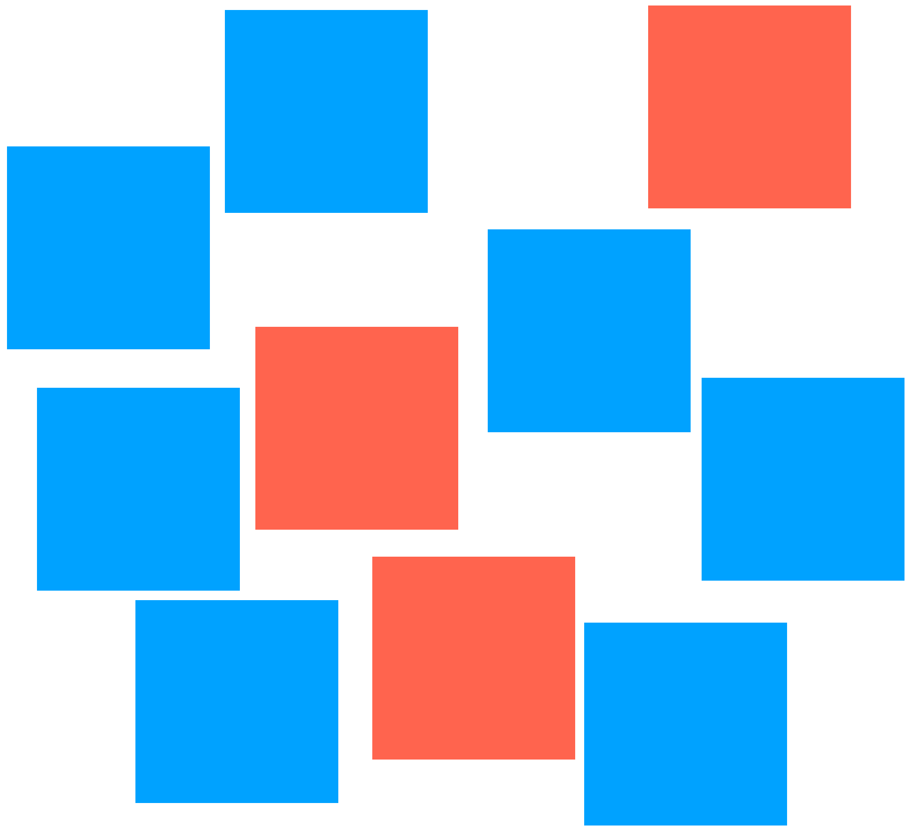
In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”:

Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In "binary" classification (there are 2 labels such as spam/ham) when 1 label is considered "positive" and the other "negative":

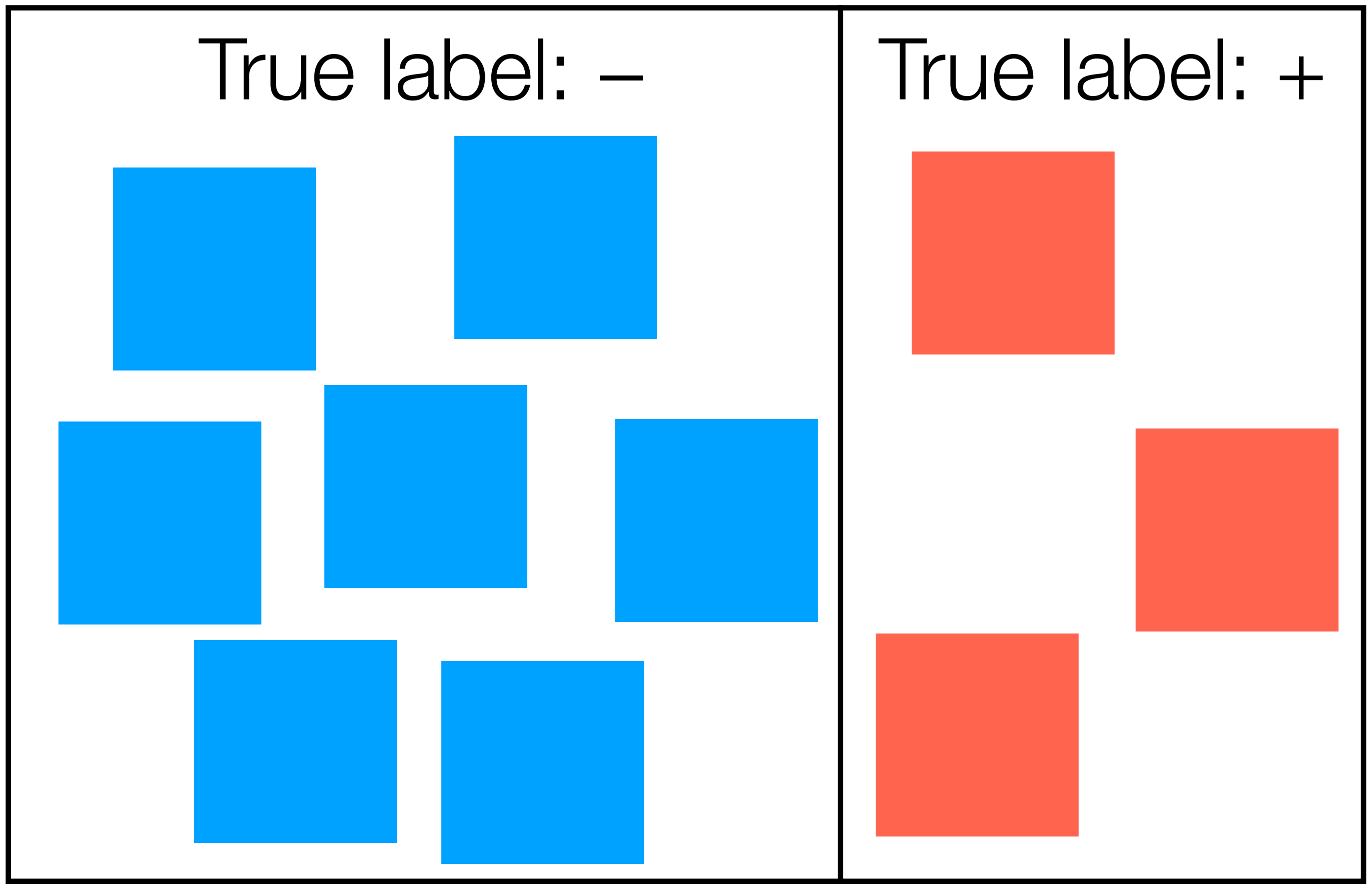


Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”:



Different Ways to Measure Accuracy

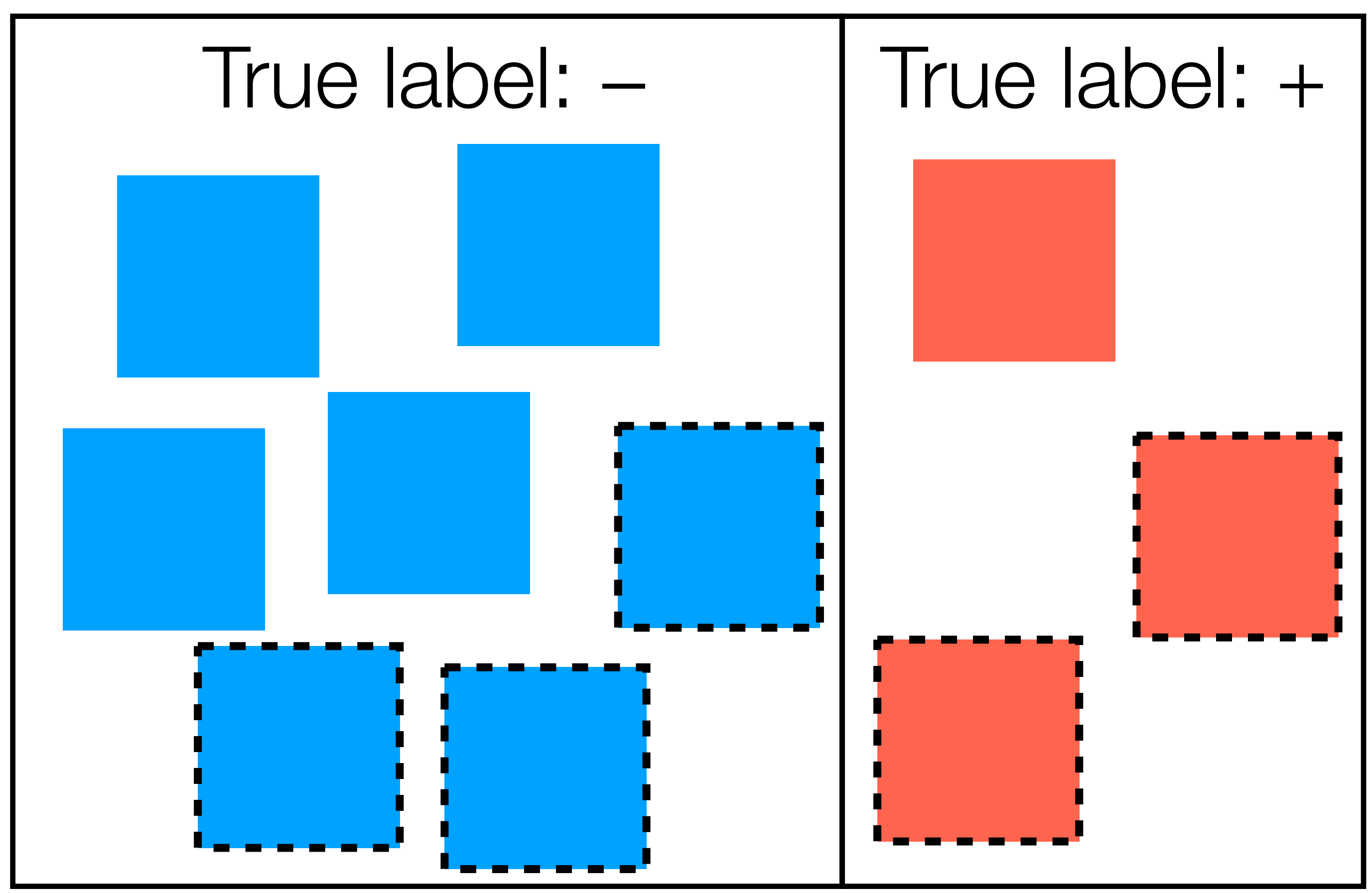
Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In "binary" classification (there are 2 labels such as spam/ham) when 1 label is considered "positive" and the other "negative":

Outlined in dotted black: predicted label +

(all other points predicted to be –)



Recall/True Positive Rate: fraction of dotted line in true label +

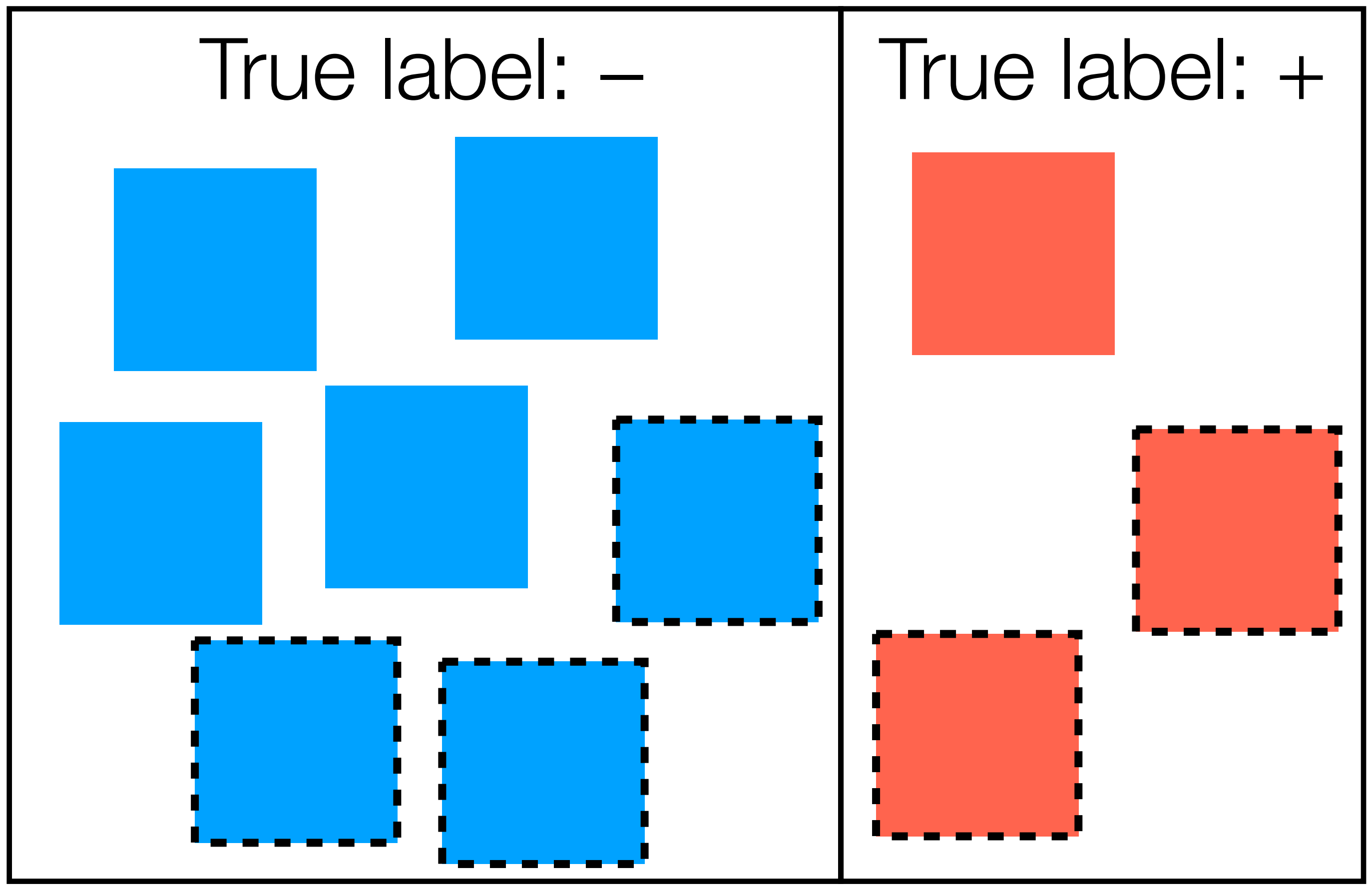
Precision: fraction of + in dotted line

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”:

Outlined in dotted black: predicted label +

(all other points predicted to be –)



Recall/True Positive Rate: fraction of dotted line in true label + = 2/3

Precision: fraction of + in dotted line = 2/5

False Positive Rate: fraction of dotted line in true label – = 3/7

F1 score: $\dfrac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$ = 1/2

Prediction and Model Validation

Demo

Deep Learning

IMAGENET

Over 10 million images, 1000 object classes

2011: Traditional computer vision achieves accuracy ~74%

2012: Initial deep neural network approach accuracy ~84%

2015 onwards: Deep learning achieves accuracy 96%+

Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. IJCV 2015.

Deep Learning Takeover

Academia:

- Top computer vision conferences (CVPR, ICCV, ECCV) are now nearly all about deep learning
- Top machine learning conferences (ICML, NeurIPS) have *heavily* been taken over by deep learning

Heavily dominated by industry now!

Extremely useful in practice:

- Near human level image classification (including handwritten digit recognition)
- Near human level speech recognition
- Improvements in machine translation, text-to-speech
- Self-driving cars
- *Better* than humans at playing Go

Google DeepMind’s AlphaGo vs Lee Sedol, 2016

GAMING TECH ARTIFICIAL INTELLIGENCE

DeepMind's StarCraft 2 AI is now better than 99.8 percent of all human players

16

AlphaStar is now grandmaster level in the real-time strategy game

By Nick Statt | @nickstatt | Oct 30, 2019, 2:00pm EDT

SHARE

Is it all hype?

Should you as a human be afraid of robots taking your job?!?

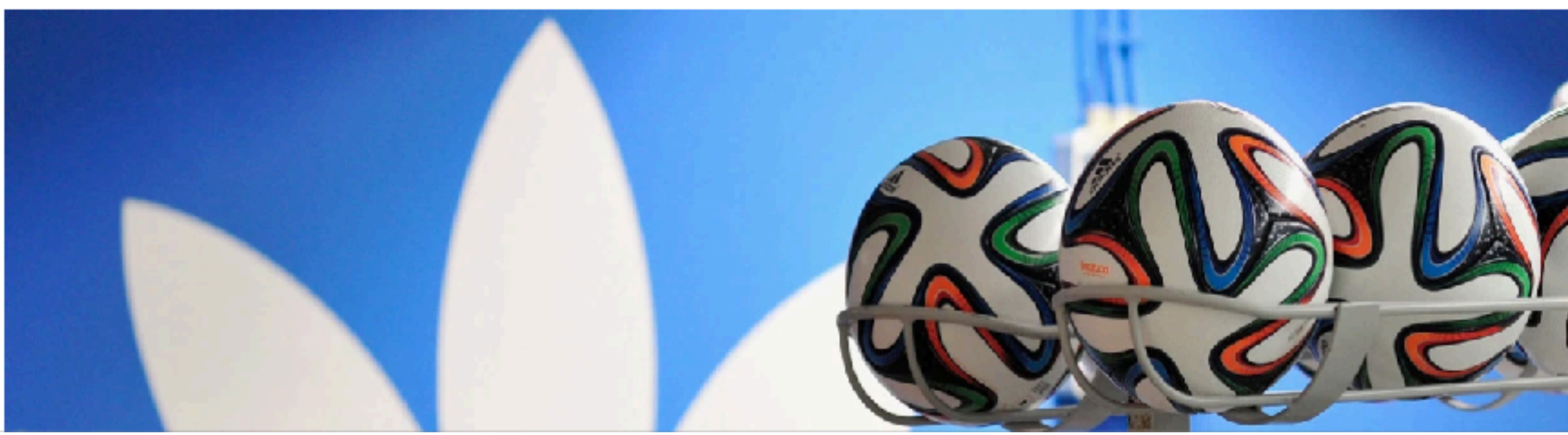
NEW YORK POST

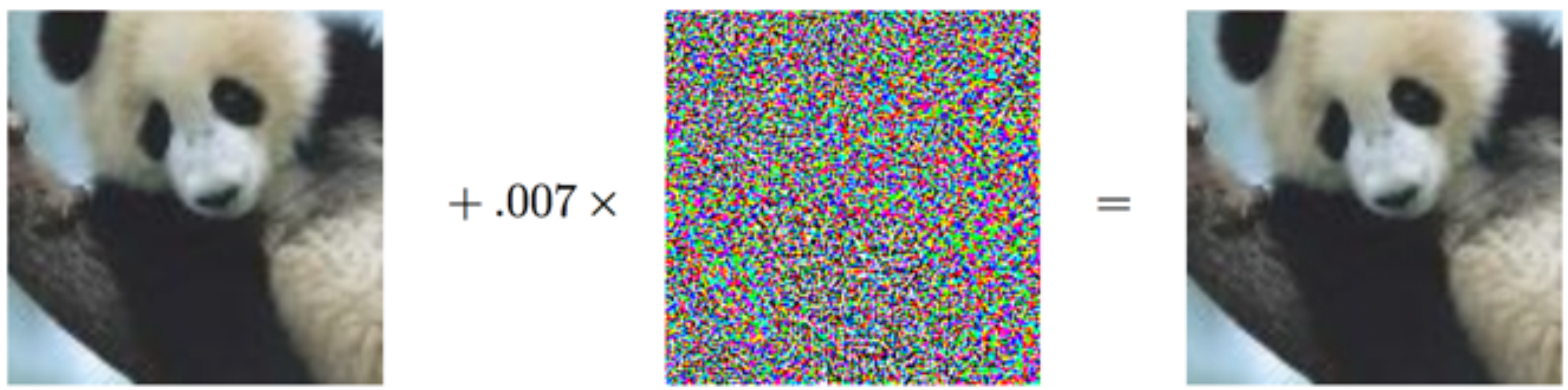
BUSINESS

American robots lose jobs to Asian robots as Adidas shifts manufacturing

By Reuters

November 11, 2019 | 9:13am | Updated



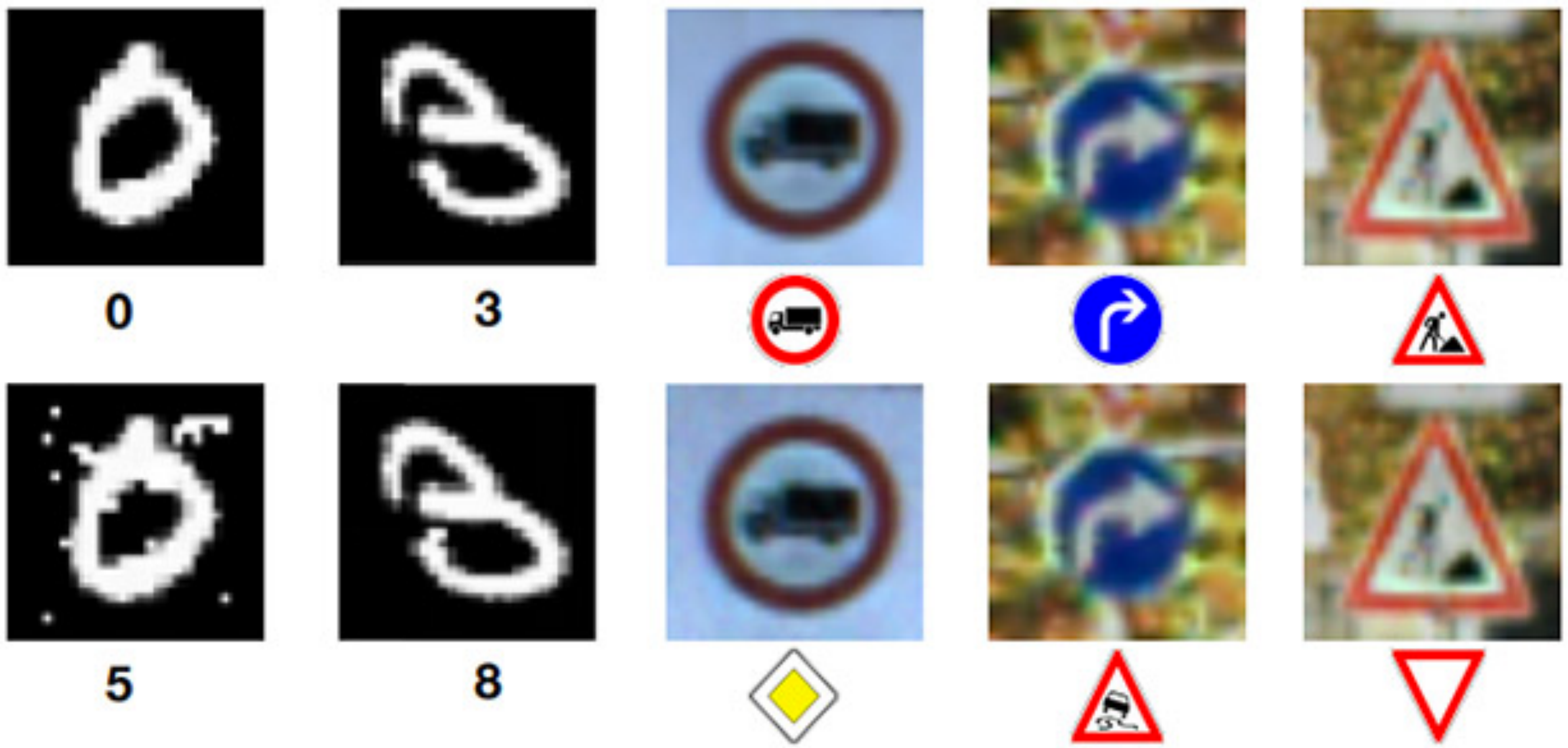


panda
~58% confidence

adversarial
noise

gibbon
~99% confidence

Source: Goodfellow, Shlens, and Szegedy. Explaining and Harnessing Adversarial Examples. ICLR 2015.

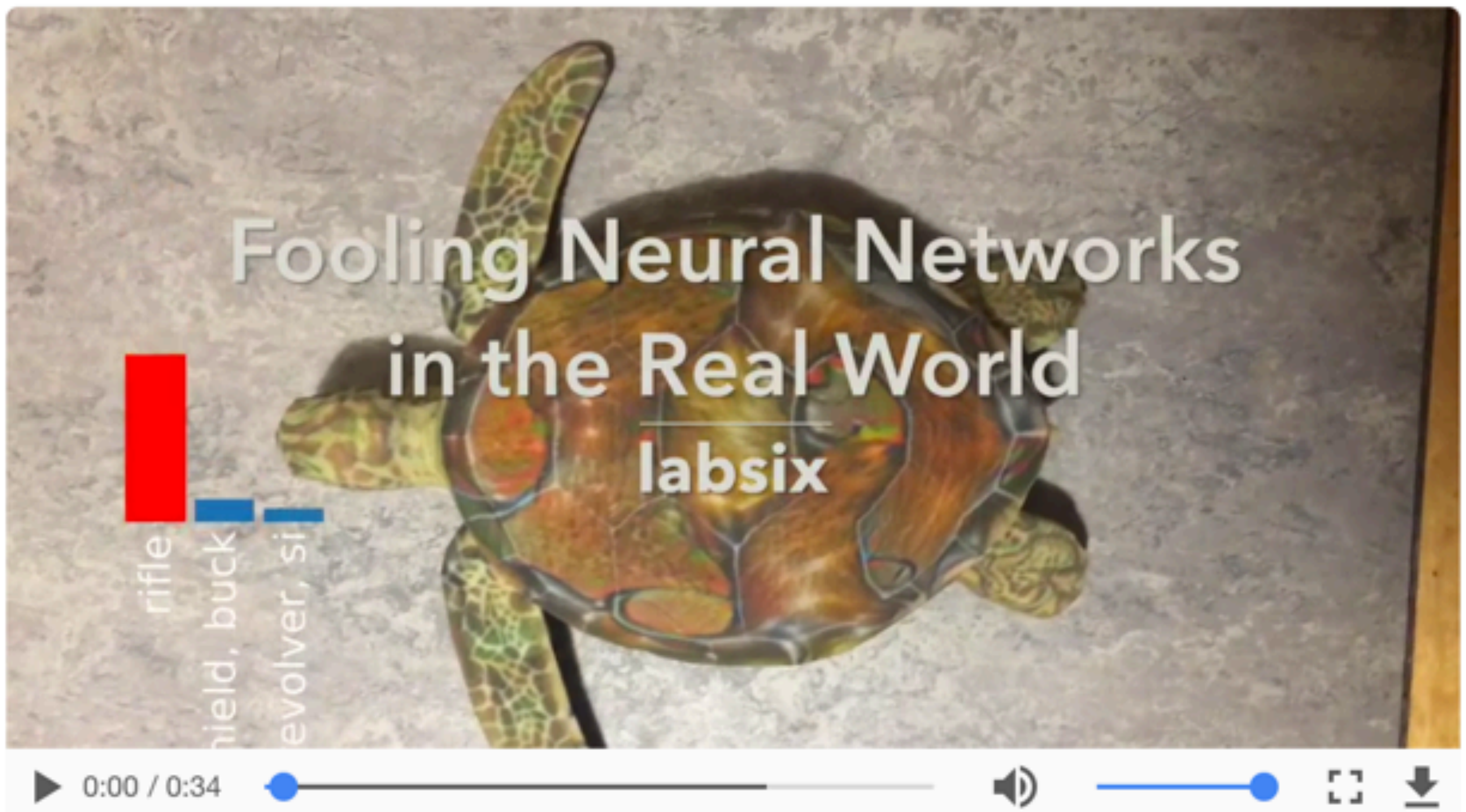


Source: Papernot et al. Practical Black-Box Attacks against Machine Learning. Asia Conference on Computer and Communications Security 2017.

Fooling Neural Networks in the Physical World with 3D Adversarial Objects

31 Oct 2017 · 3 min read — shared on Hacker News, Lobsters, Reddit, Twitter

We've developed an approach to generate *3D adversarial objects* that reliably fool neural networks in the real world, no matter how the objects are looked at.

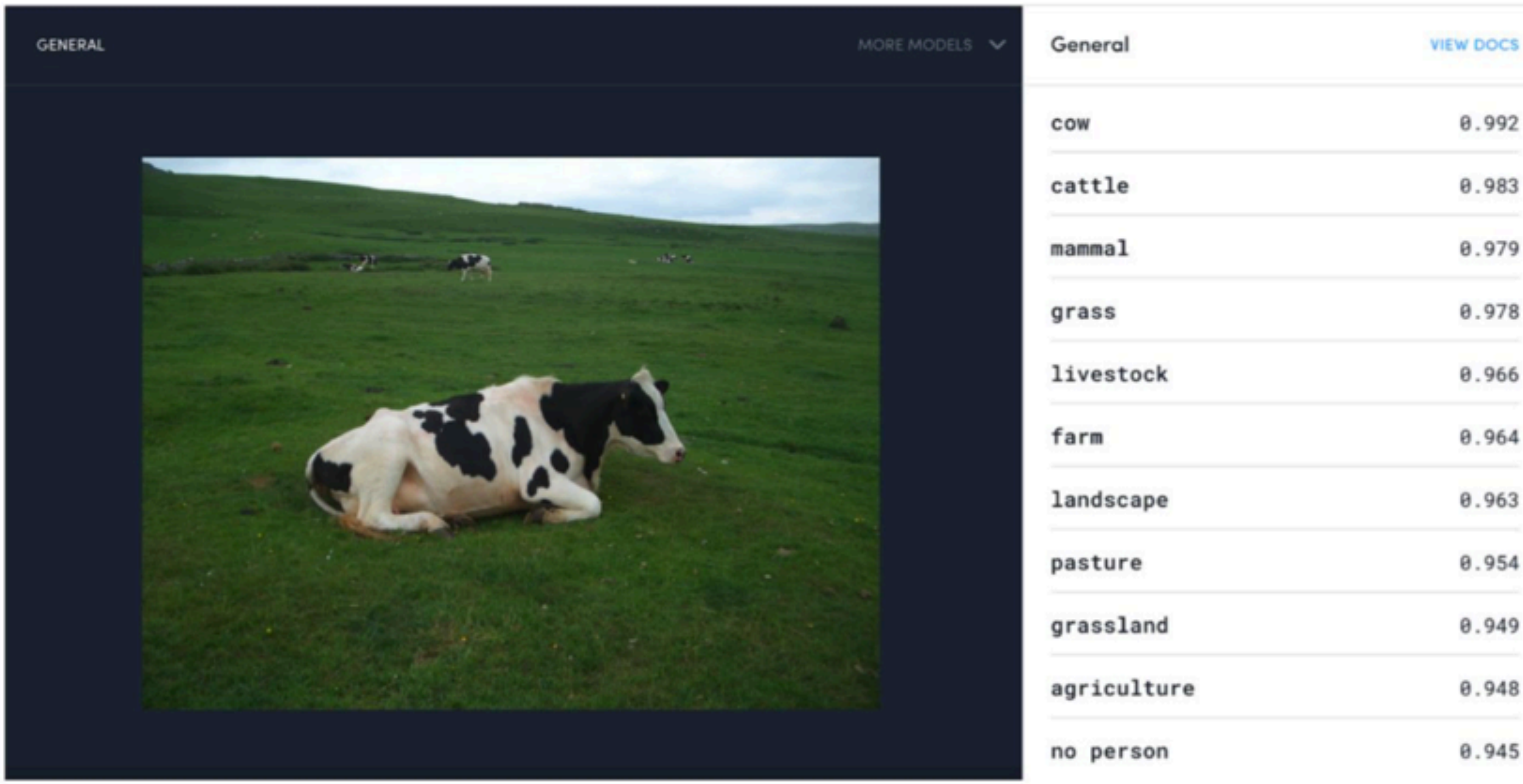


Neural network based classifiers reach near-human performance in many tasks, and they're used in high risk, real world systems. Yet, these same neural networks are particularly vulnerable to *adversarial examples*, carefully perturbed inputs that cause

Source: labsix

Source: https://www.cc.gatech.edu/news/611783/erasing-stop-signs-shapeshifter-shows-self-driving-cars-can-still-be-manipulated

Source: Gizmodo article “This Neural Network's Hilariously Bad Image Descriptions Are Still Advanced AI”. September 16, 2015. (They’re using the NeuralTalk image-to-caption software.)

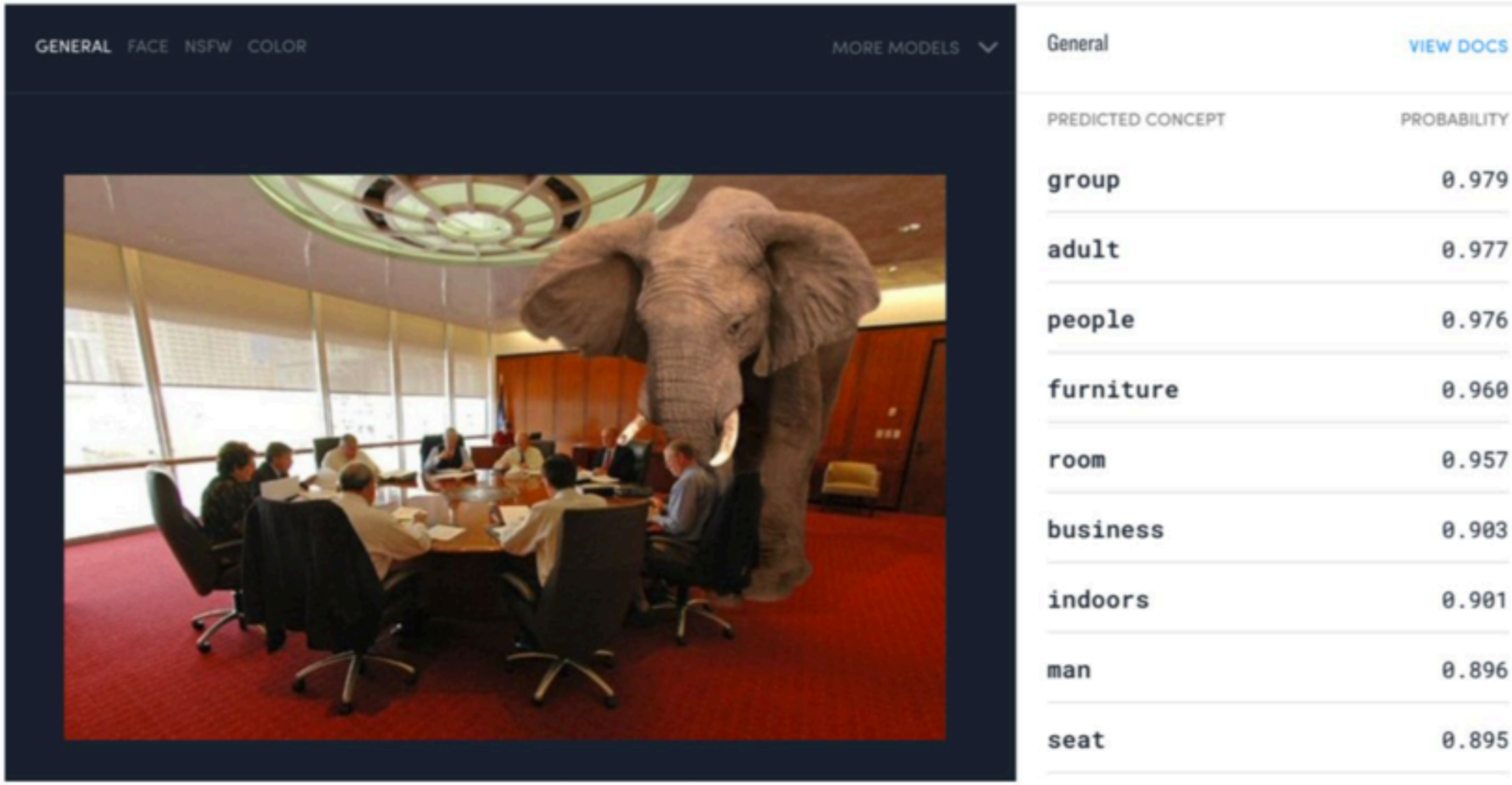


Source: Pietro Perona

General	VIEW DOCS
no person	0.991
beach	0.990
water	0.985
sand	0.981
sea	0.980
travel	0.978
seashore	0.972
summer	0.954
sky	0.946
outdoors	0.944
ocean	0.936

cow is not among top objects found!

Source: Pietro Perona



elephant is not among top objects found!

Source: David Lopez-Paz

Another AI Winter?

~1970’s: First AI winter over symbolic AI

~1980’s: Second AI winter over “expert systems”

Every time: Lots of hype, explosion in funding, then bubble bursts

About membership

Medium

Sign in | Get started

Michael Jordan Follow

Michael I. Jordan is a Professor in the Department of Electrical Engineering and Computer Sciences and the Department of Statistics at UC Berkeley.

Apr 18 · 16 min read

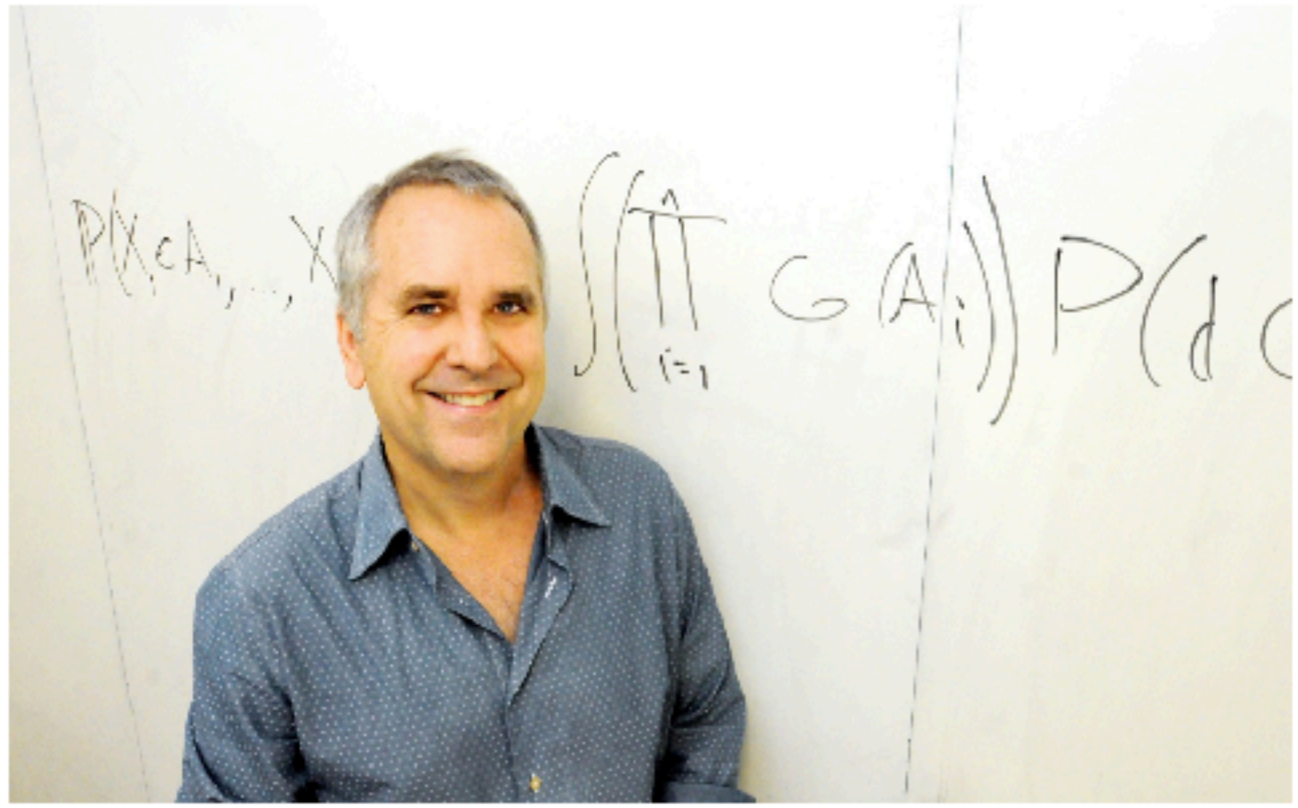


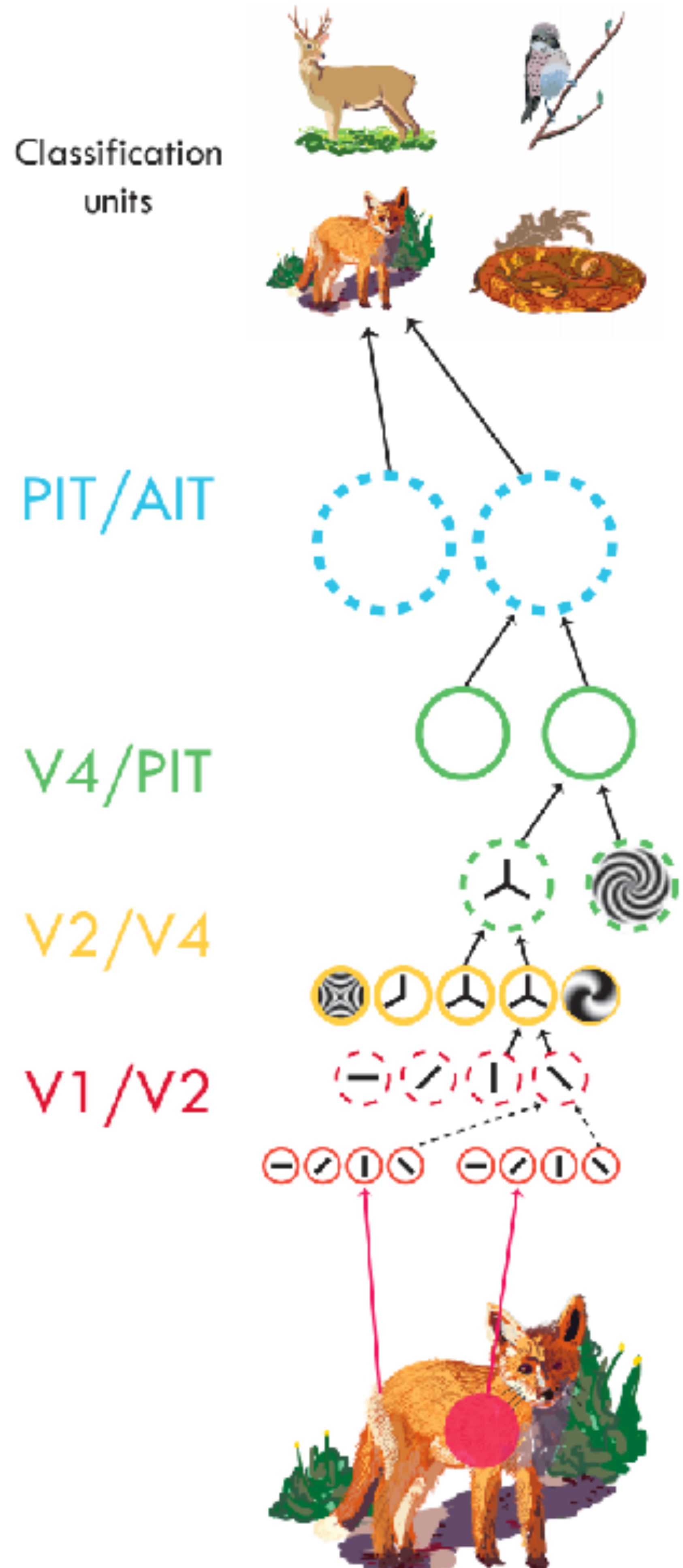
Photo credit: Peg Skorpinski

Artificial Intelligence—The Revolution Hasn't Happened Yet

Artificial Intelligence (AI) is the mantra of the current era. The phrase is intoned by technologists, academicians, journalists and venture capitalists

https://medium.com/@mijordan3/artificial-intelligence-the-revolution-hasnt-happened-yet-5e1d5812e1e7

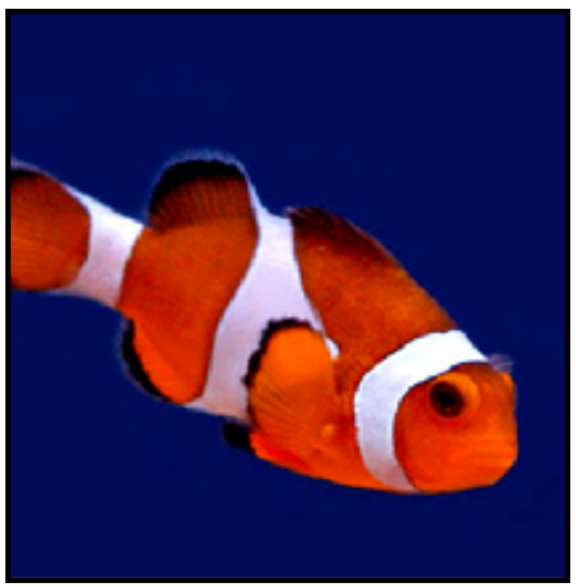
What is deep learning?



Slide by Phillip Isola

Serre, 2014

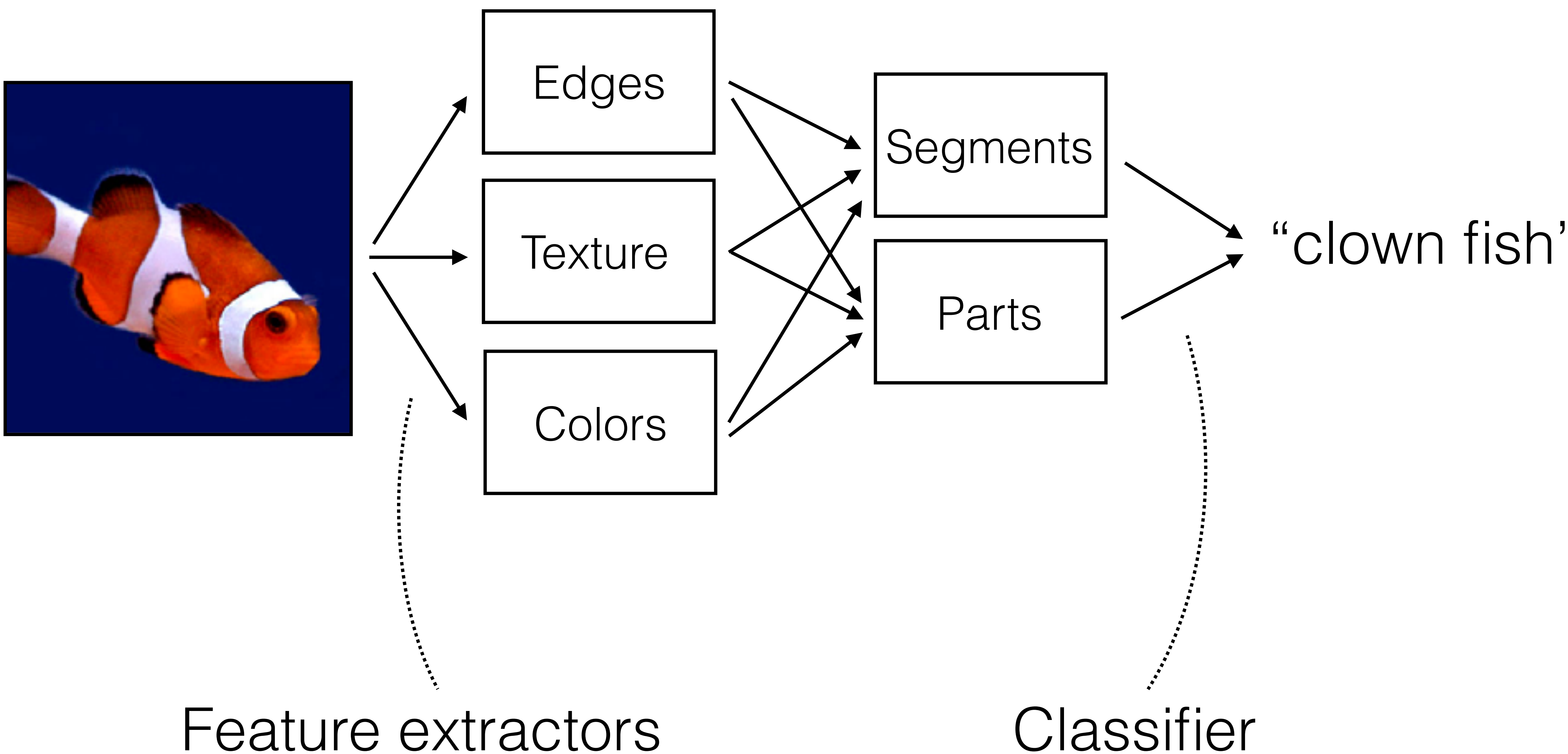
Basic Idea



→ Brain/Machine → "clown fish"

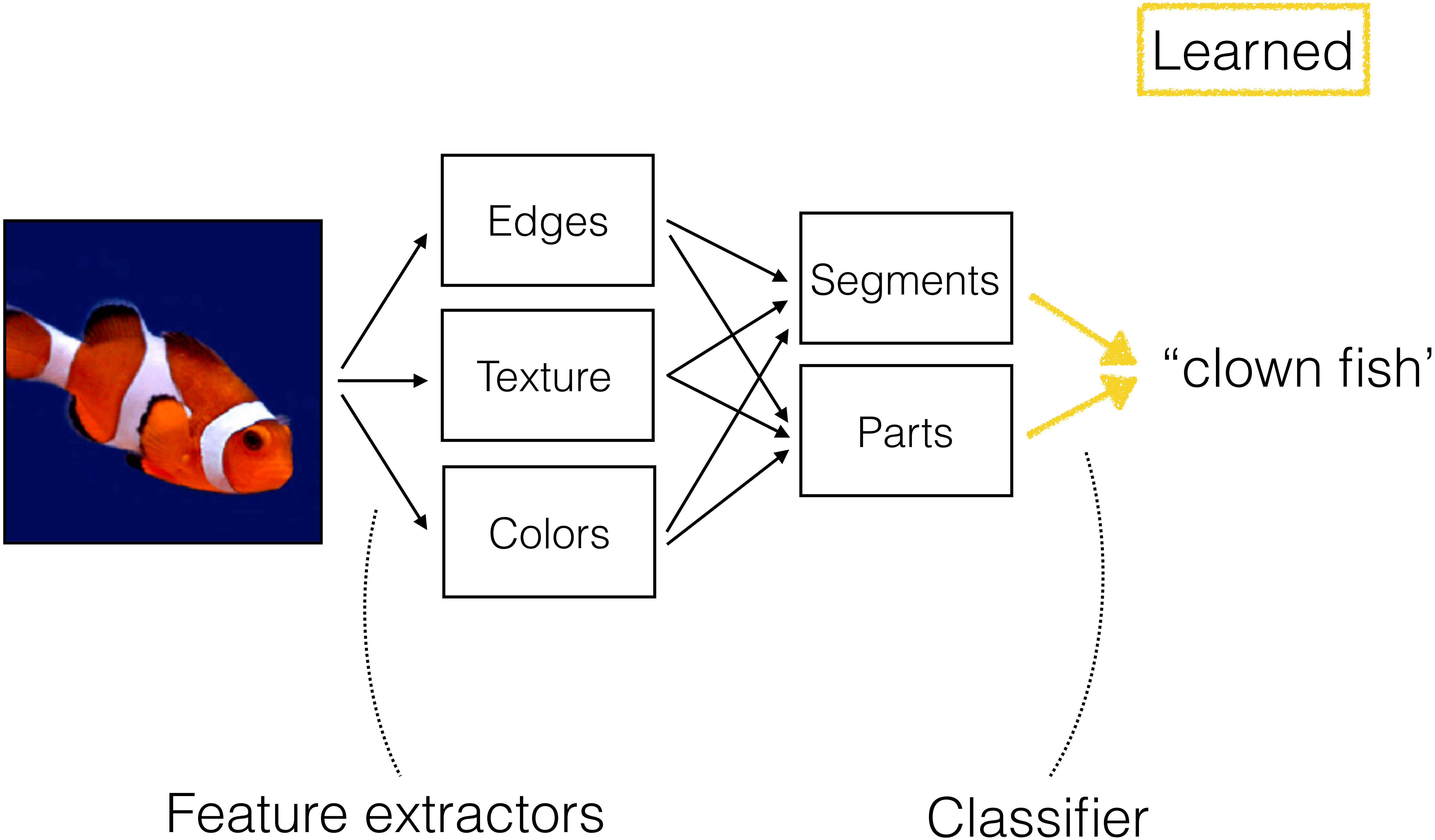
Slide by Phillip Isola

Object Recognition

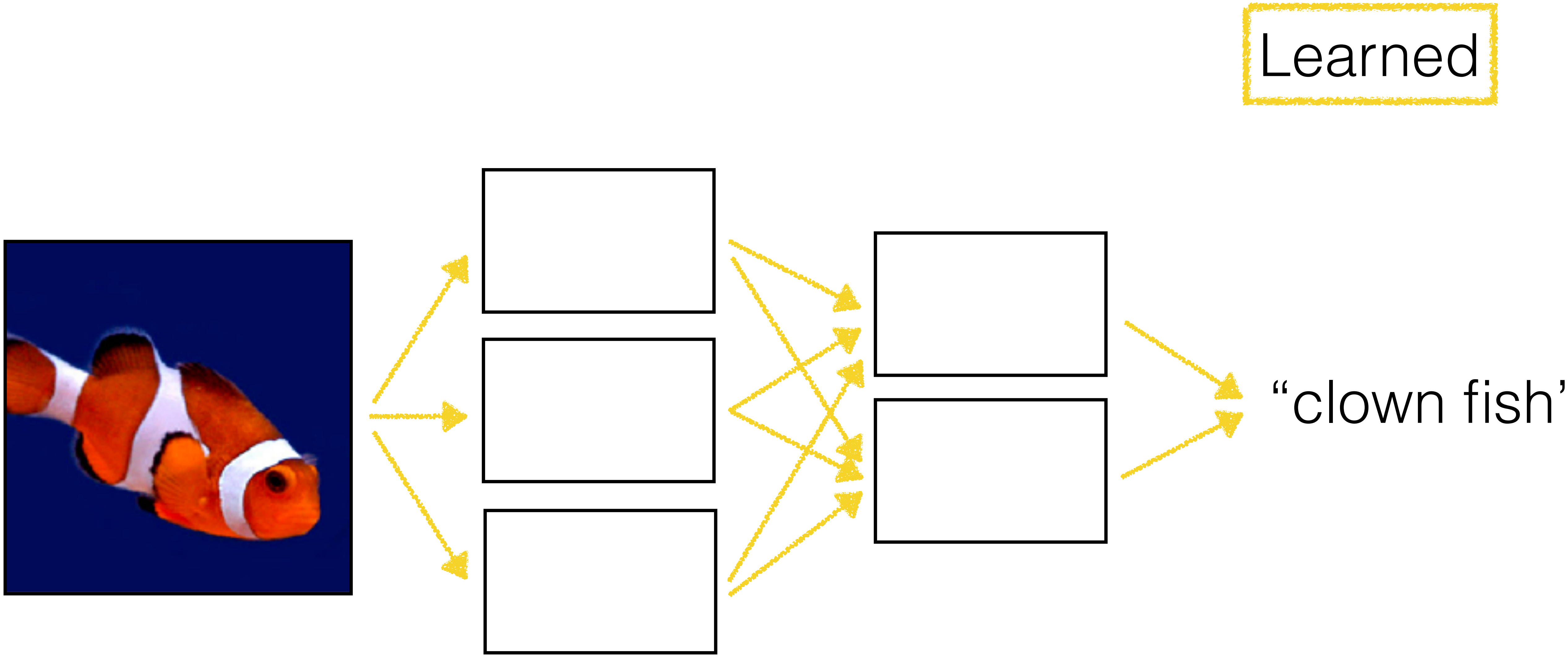


Slide by Phillip Isola

Object Recognition



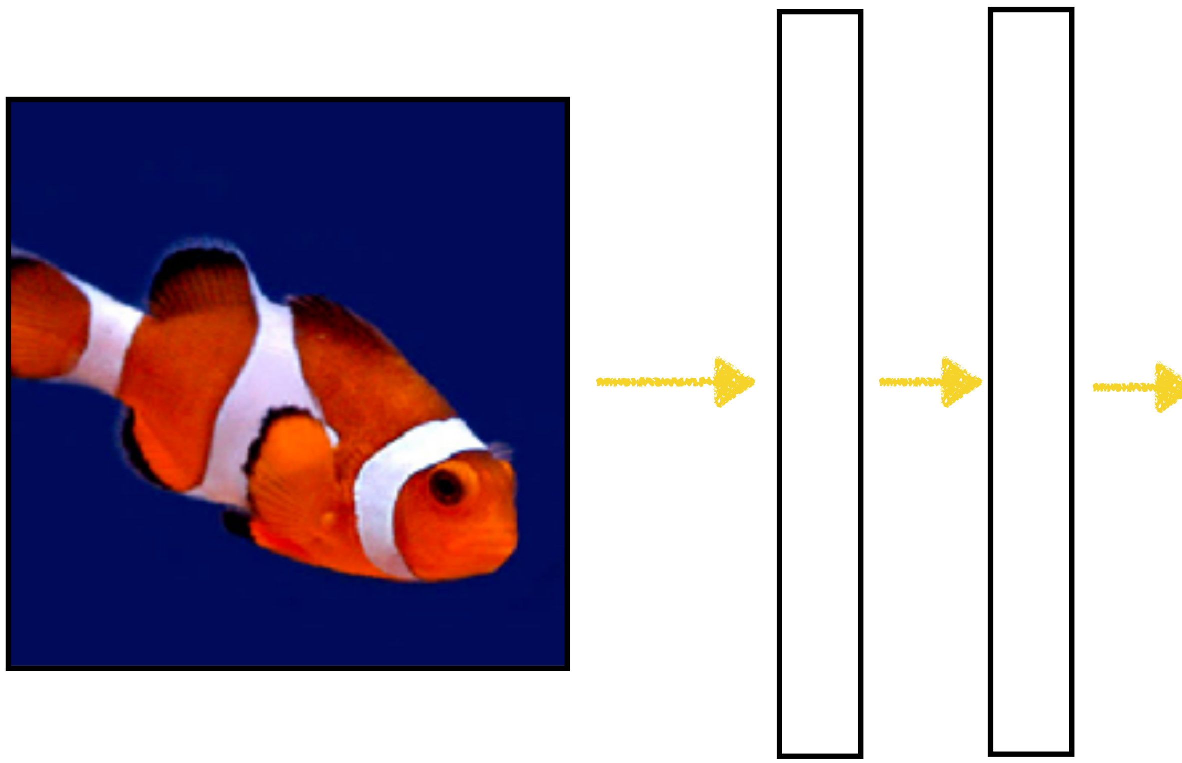
Neural Network



Slide by Phillip Isola

Neural Network

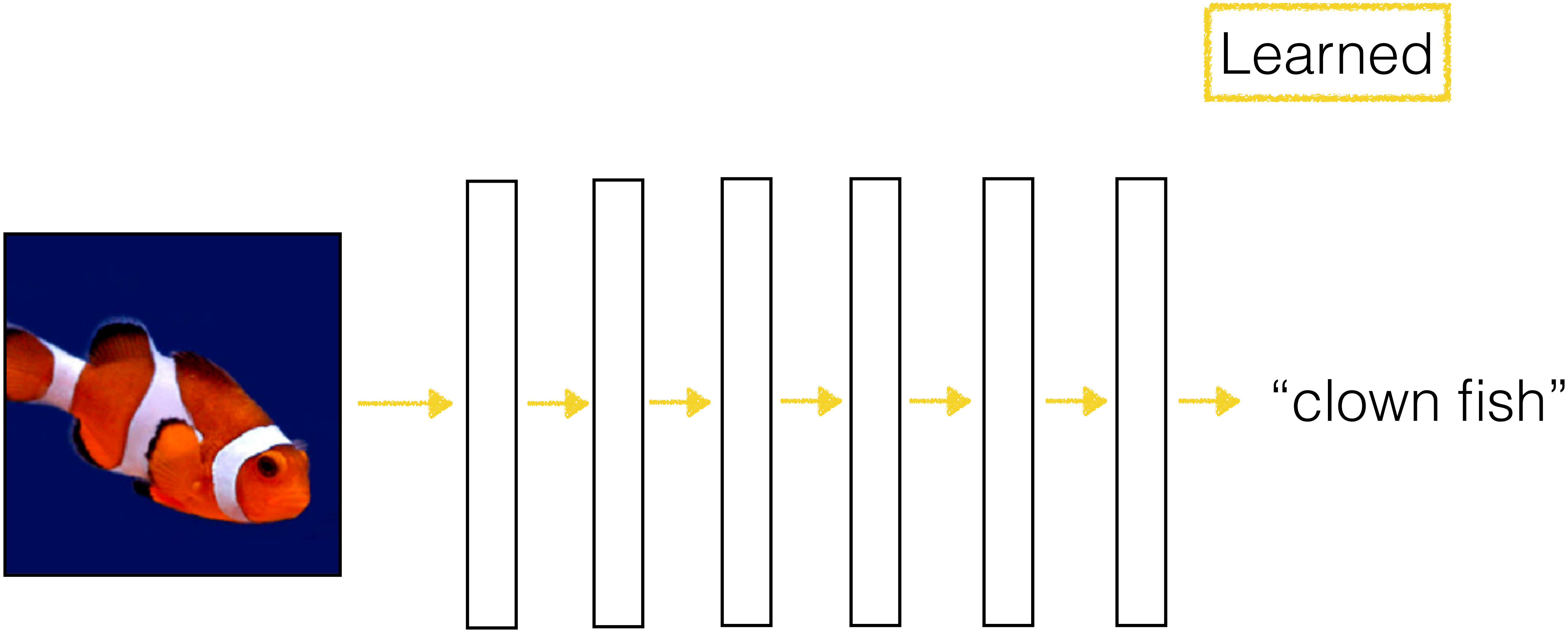
Learned



“clown fish”

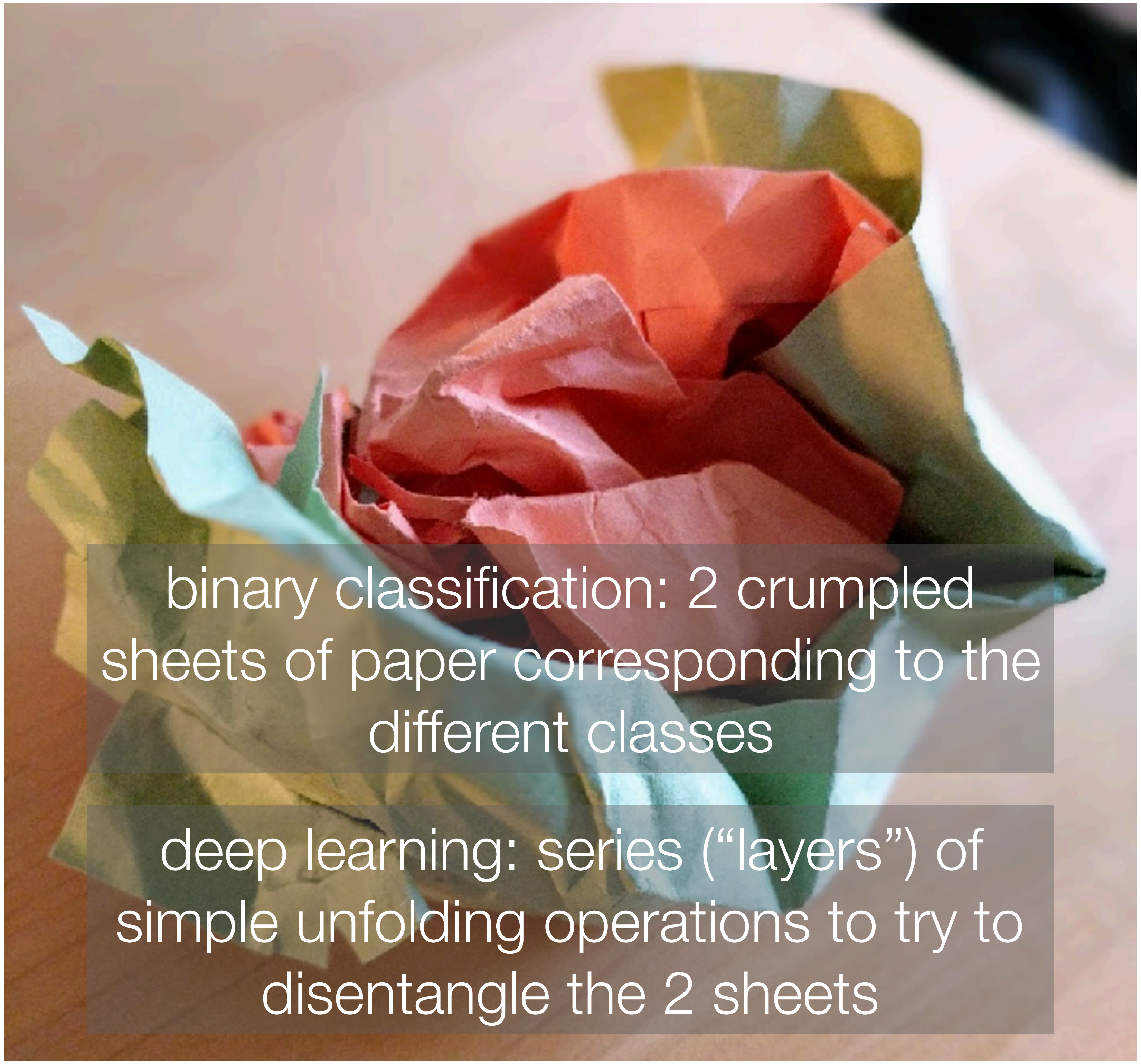
Slide by Phillip Isola

Deep Neural Network



Slide by Phillip Isola

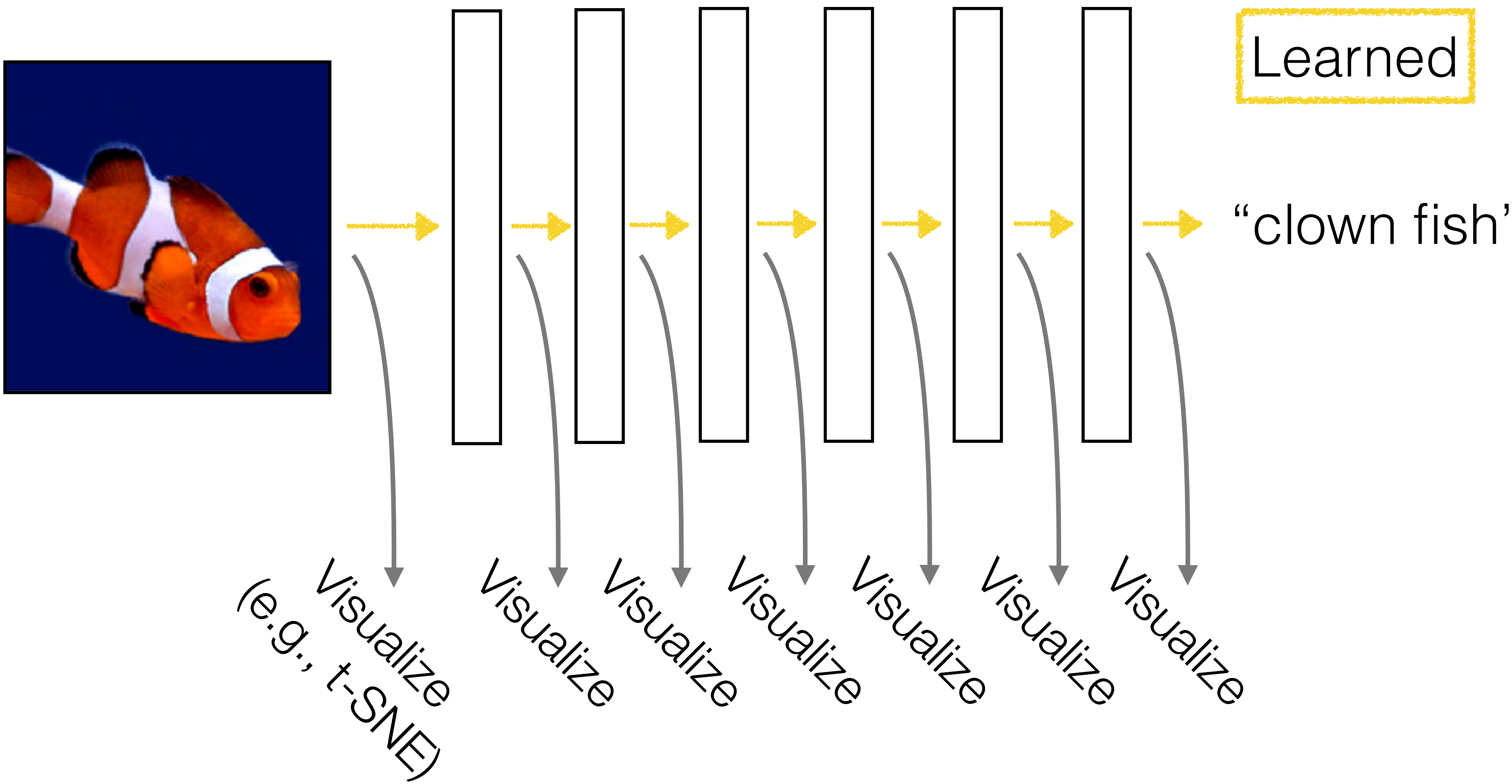
Crumpled Paper Analogy



Analogy: Francois Chollet, photo: George Chen

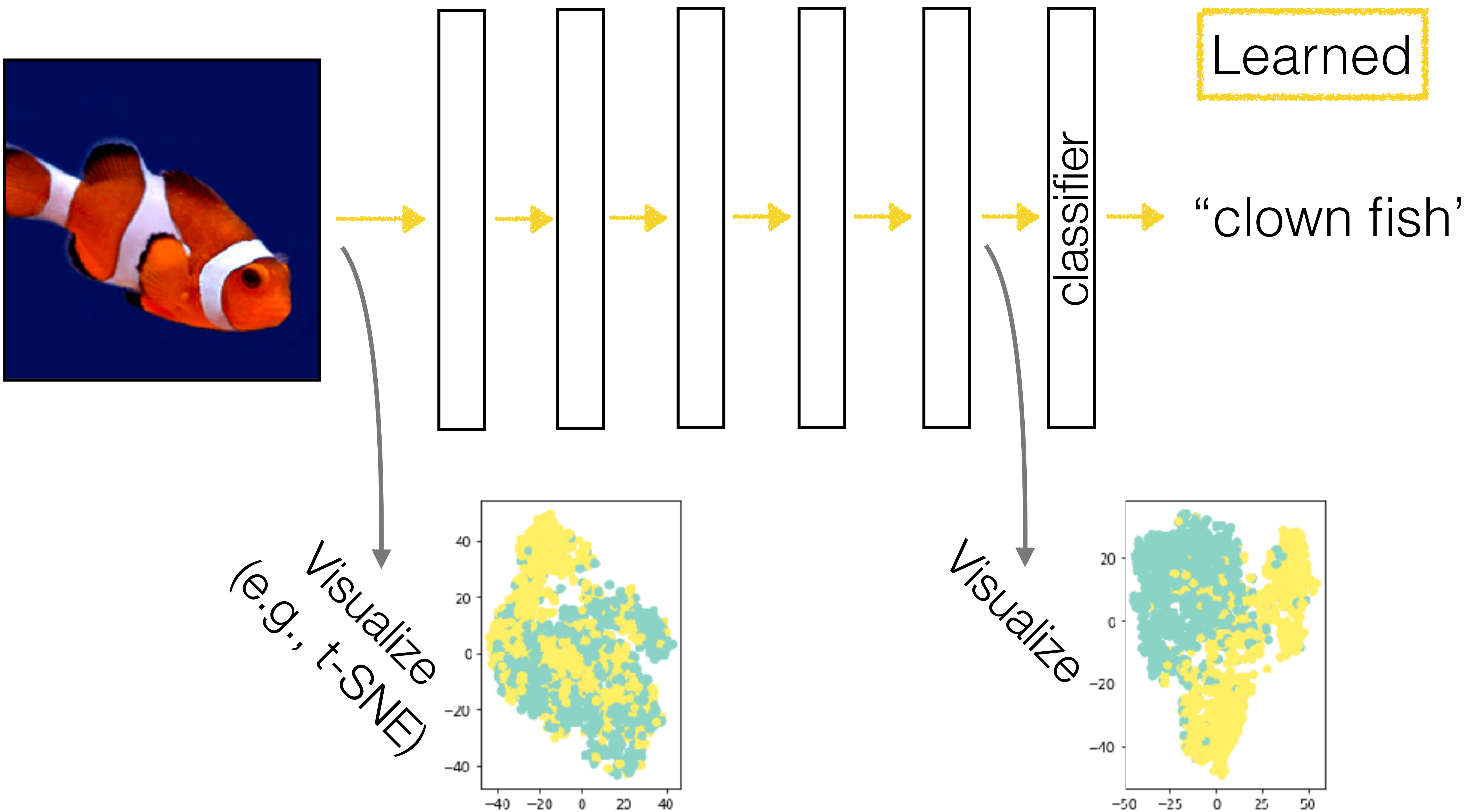
Representation Learning

Each layer's output is *another way we could represent the input data*



Representation Learning

Each layer's output is *another way we could represent the input data*



Why Does Deep Learning Work?

Actually the ideas behind deep learning are old (~1980's)

- Big data

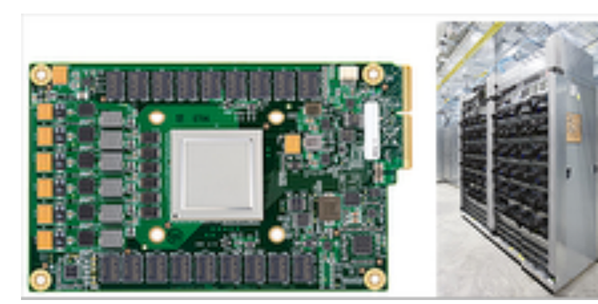
amazon.com fitbit NETFLIX UPMC LIFE CHANGING MEDICINE

- Better hardware

intel AMD

CPU's
& Moore's law

GPU's



TPU's

- Better algorithms

Structure Present in Data Matters

Neural nets aren’t doing black magic

- **Image analysis:** convolutional neural networks (convnets) neatly <u>incorporates basic image processing structure</u>

- **Time series analysis:** recurrent neural networks (RNNs) <u>incorporates ability to remember and forget things over time</u>

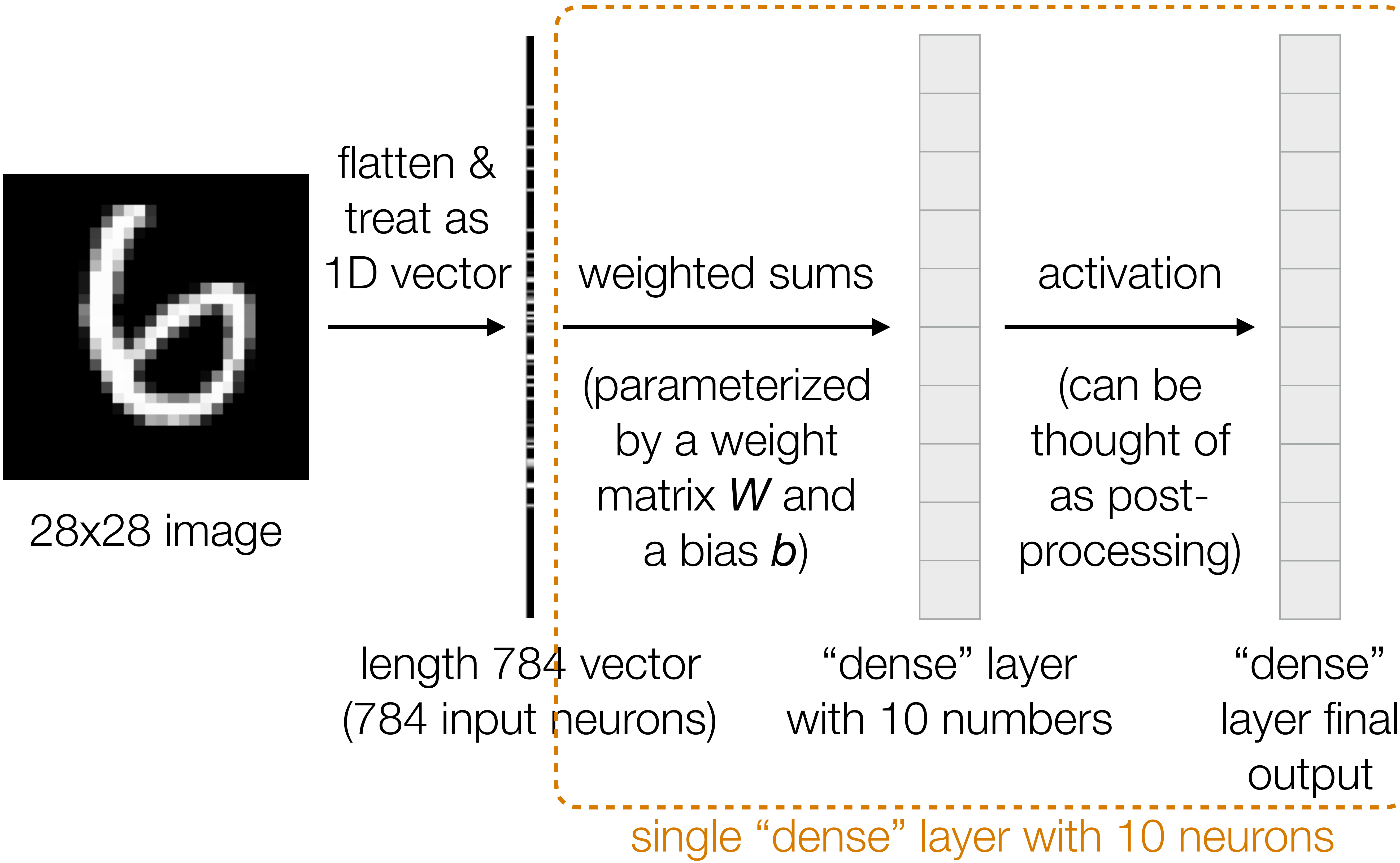
    - Note: text is a time series

    - Note: video is a time series

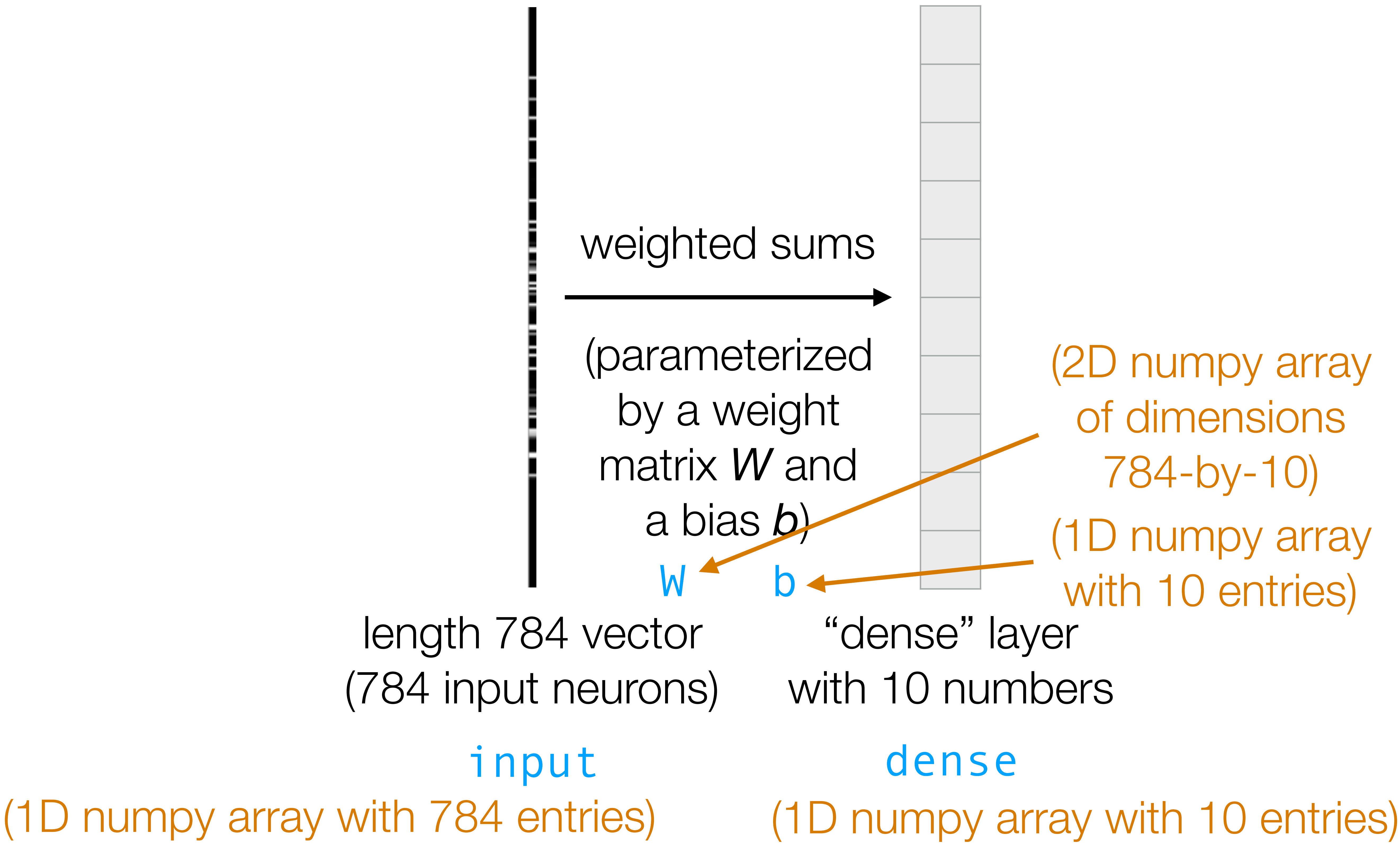
Handwritten Digit Recognition Example

Walkthrough of building a 1-layer and then a 2-layer neural net

Handwritten Digit Recognition



Handwritten Digit Recognition



Handwritten Digit Recognition

```
dense[0] = np.dot(input, W[:, 0]) + b[0]
dense[1] = np.dot(input, W[:, 1]) + b[1]
```

⋮

$$\texttt{dense[j]} = \sum_{i=0}^{783} \texttt{input[i]} \times \texttt{W[i, j]} + \texttt{b[j]}$$

weighted sums

(parameterized by a weight matrix *W* and a bias *b*)

`W` (2D numpy array of dimensions 784-by-10)

`b` (1D numpy array with 10 entries)

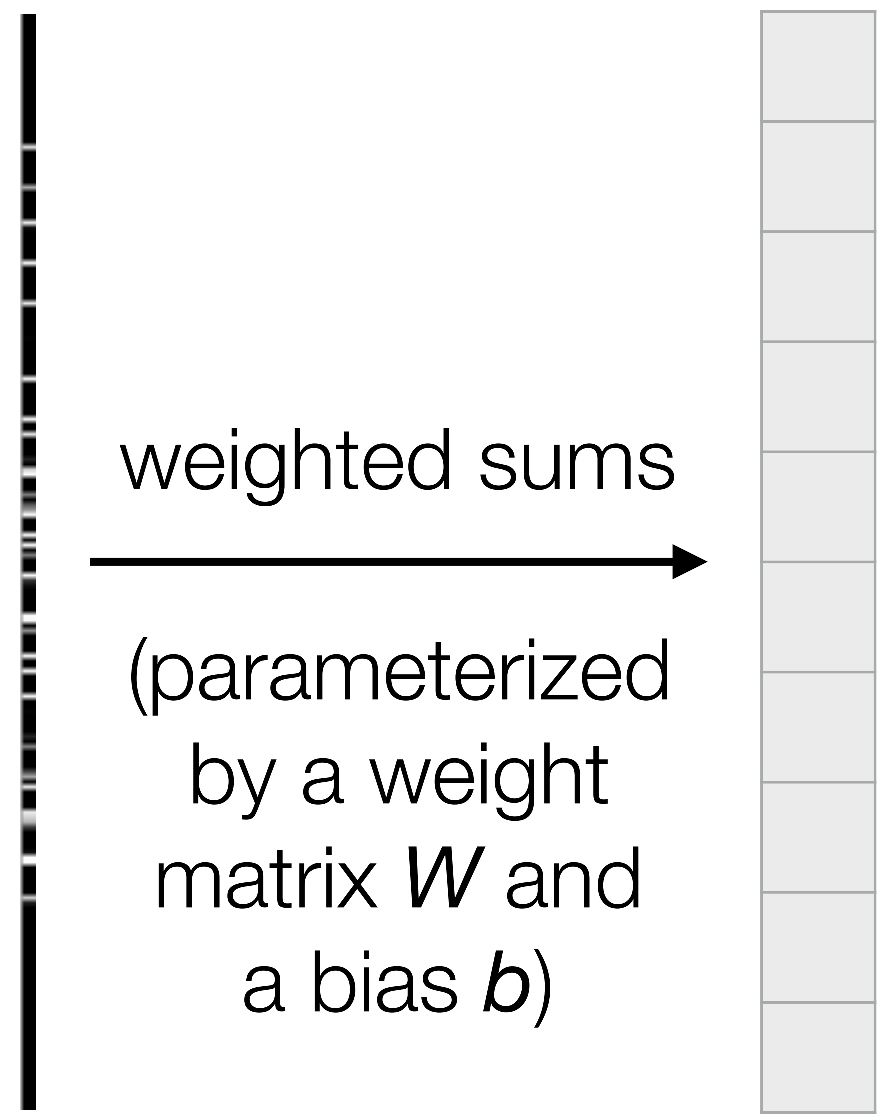
n 784 vector
nput neurons)

`input`
34 entries)

“dense” layer
with 10 numbers

`dense`
(1D numpy array with 10 entries)

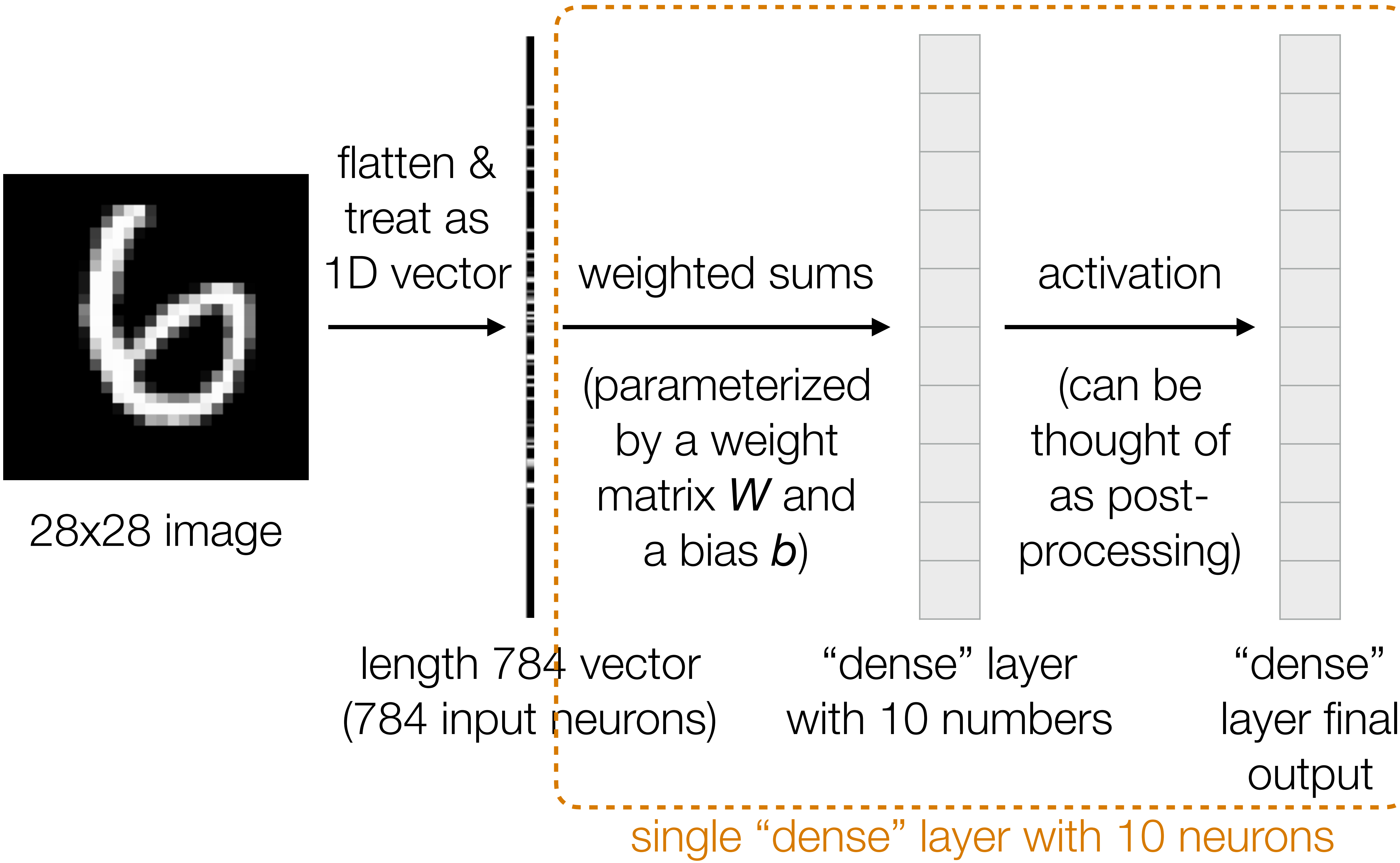
Handwritten Digit Recognition



length 784 vector
(784 input neurons)

“dense” layer
with 10 numbers

Handwritten Digit Recognition



Handwritten Digit Recognition

Many different activation functions possible

Example: **Rectified linear unit (ReLU)**
zeros out entries that are negative

```
dense_final = np.maximum(0, dense)
```

4	4
3.5	3.5
4	4
-1	0
0.5	0.5
2	2
-4	0
3	3
-2	0
5	5

ReLU →

(can be thought of as post-processing)

"dense" layer
with 10 numbers

`dense`

"dense"
layer final
output

`dense_final`

Handwritten Digit Recognition

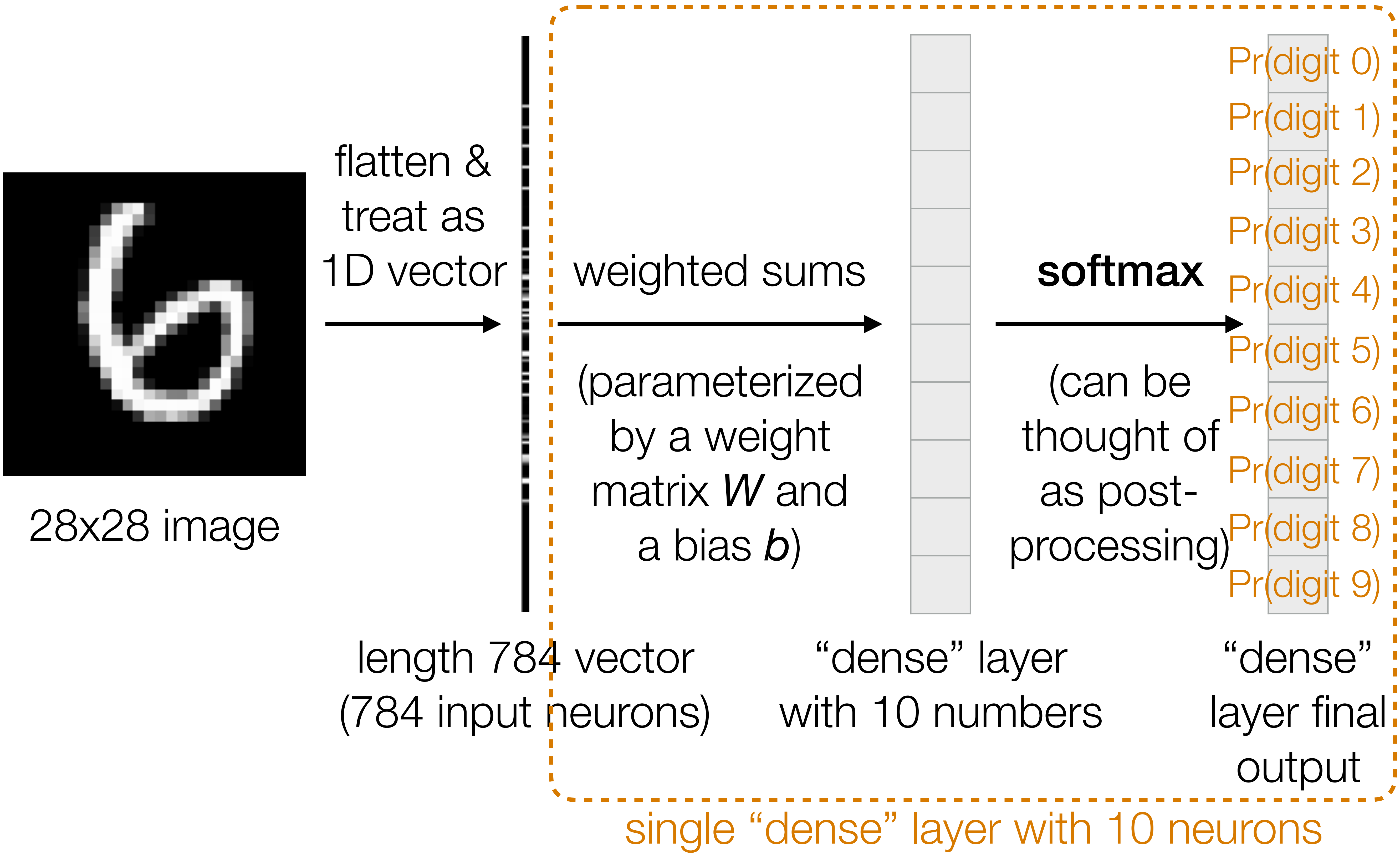
Many different activation functions possible

Example: **softmax** turns the entries in the dense layer (prior to activation) into a probability distribution (using the “softmax” transformation)

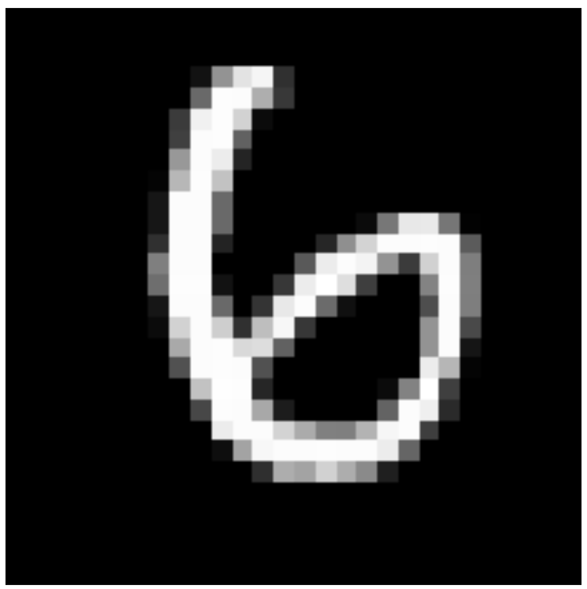
```
dense_exp = np.exp(dense)
dense_exp /= np.sum(dense_exp)
dense_final = dense_exp
```

“dense” layer with 10 numbers (`dense`)	**softmax** → (can be thought of as post-processing)	“dense” layer final output (`dense_final`)
4		0.17
3.5		0.10
4		0.17
-1		0.00
0.5		0.01
2		0.02
-4		0.00
3		0.06
-2		0.00
5		0.46

Handwritten Digit Recognition



Handwritten Digit Recognition



28x28 image

flatten &
treat as
1D vector

length 784 vector
(784 input neurons)

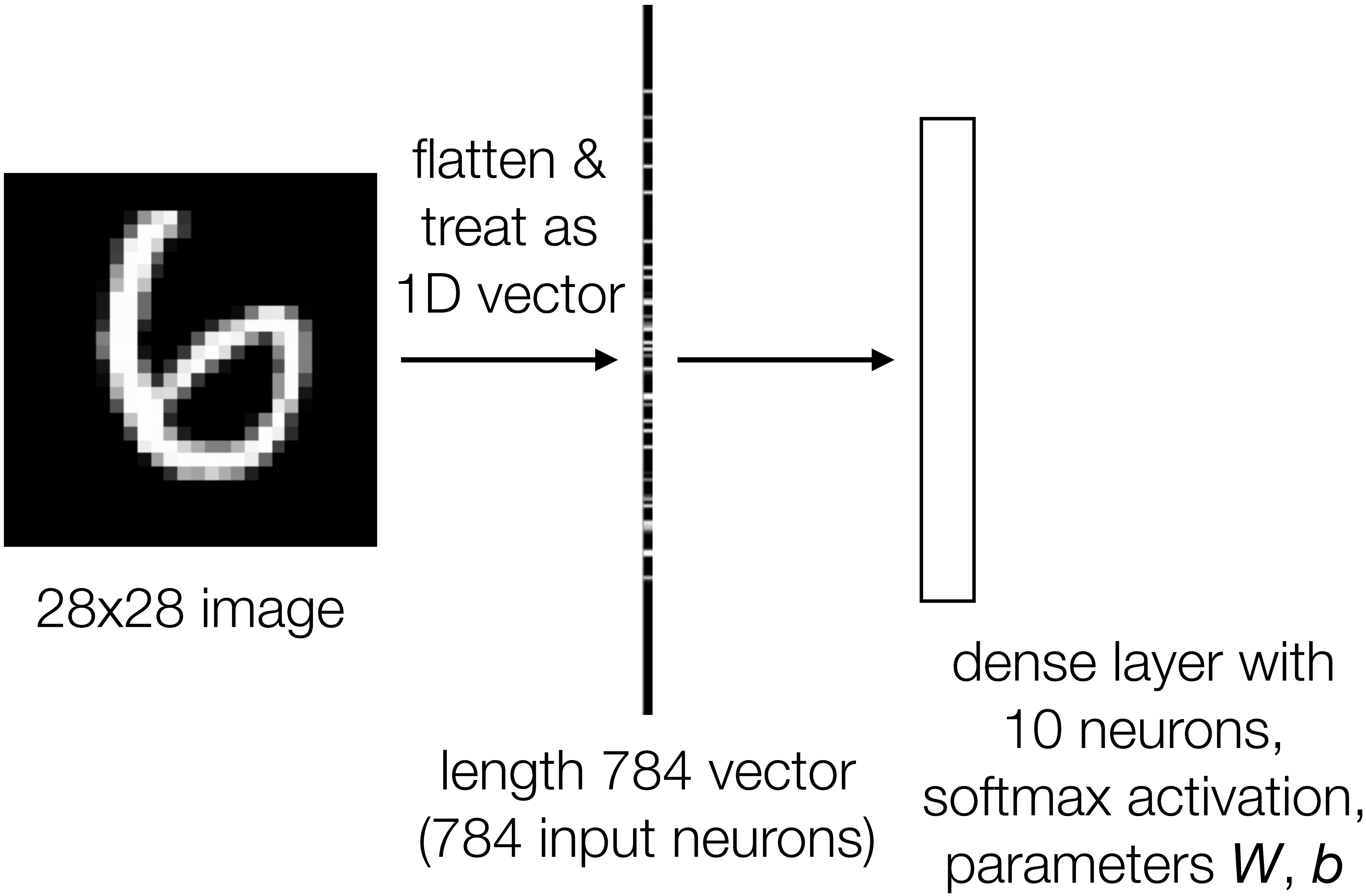
dense layer with
10 neurons,
softmax activation,
parameters *W*, *b*

We want the output of the
dense layer to encode
probabilities for whether the
input image is a 0, 1, 2, …, 9
*but as of now we aren't
providing any sort of
information to enforce this*

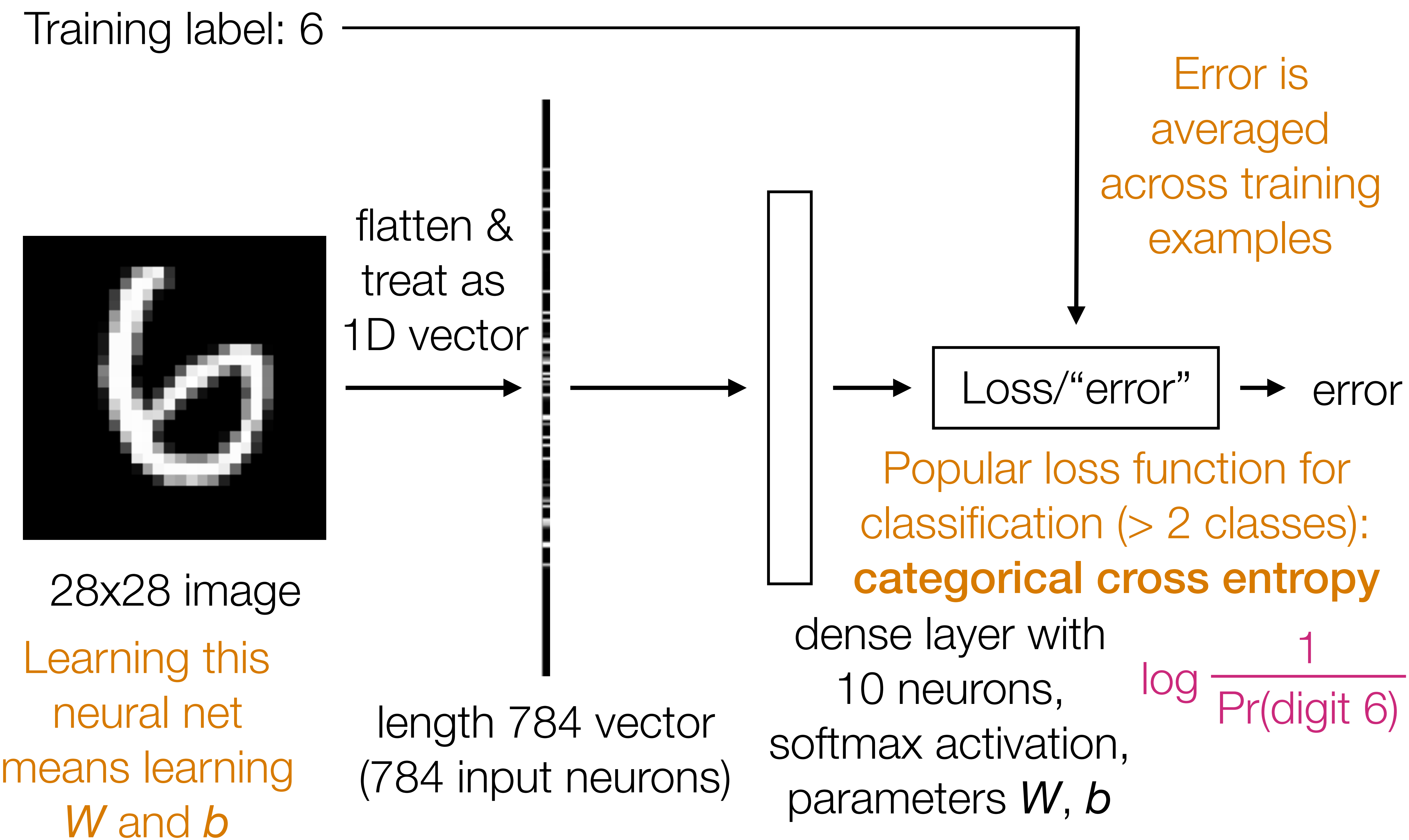
Handwritten Digit Recognition

Demo part 1

Handwritten Digit Recognition



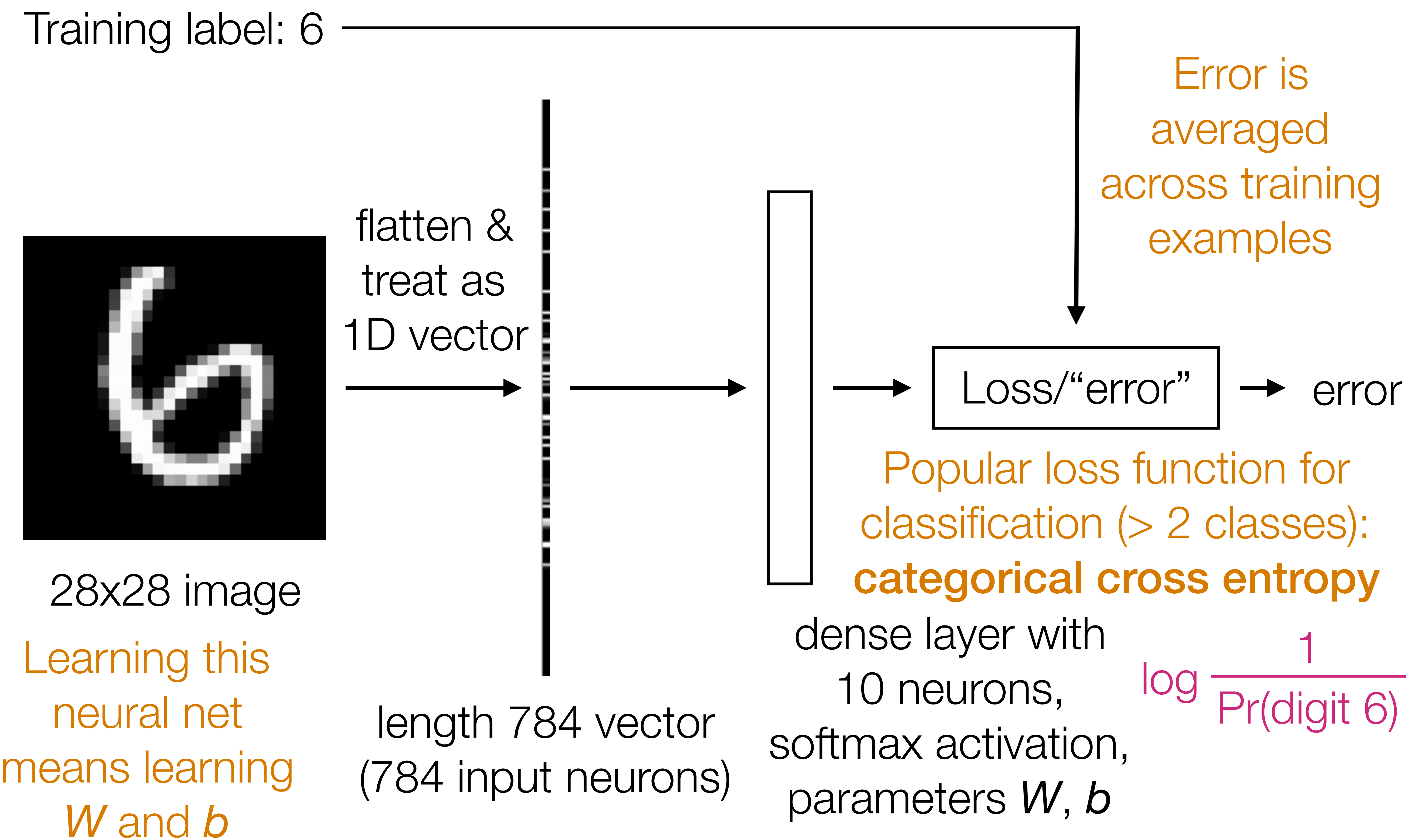
Handwritten Digit Recognition



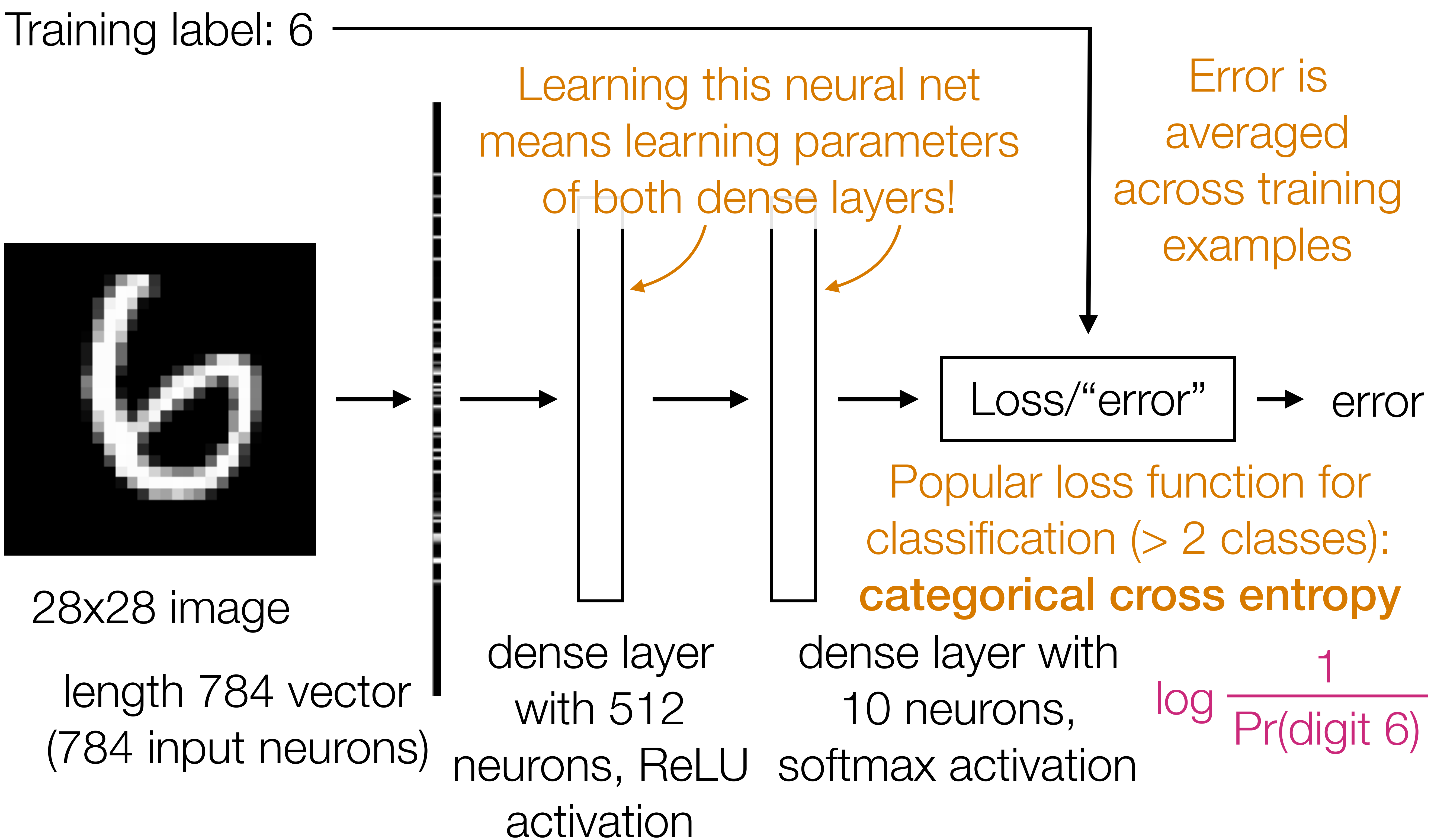
Handwritten Digit Recognition

Demo part 2

Handwritten Digit Recognition



Handwritten Digit Recognition



Handwritten Digit Recognition

Demo part 3