Carnegie Mellon University
Heinz College

94-775 Unstructured Data Analytics for Policy

Lecture 11: Review of basic prediction concepts; intro to neural nets & deep learning

Slides by George H. Chen

Administrivia

- Reminder: HW2 is due today by 11:59pm
  (you are allowed to use late days if you still have those)
    - Solutions will be automatically released 48 hours (+ a few min) later in case anyone is using 2 late days
- Reminder: Quiz 2 is this Friday during your recitation slot
  (first 40 min)
    - Please see my Canvas announcement from this morning for the Spring 2025 Quiz 2 + more practice problems
      (from my other class 95-865)
- Reminder: Your Quiz 2 review session (run by your TA Shurui)
  is tomorrow 7pm-8pm over Zoom
  (check Canvas announcement for the Zoom link)
- Reminder: Some of your final project teams are on the hook for addressing concerns regarding your final project proposal by tomorrow night (by email)

Course Outline

Part I: Exploratory data analysis

Identify structure present in "unstructured" data

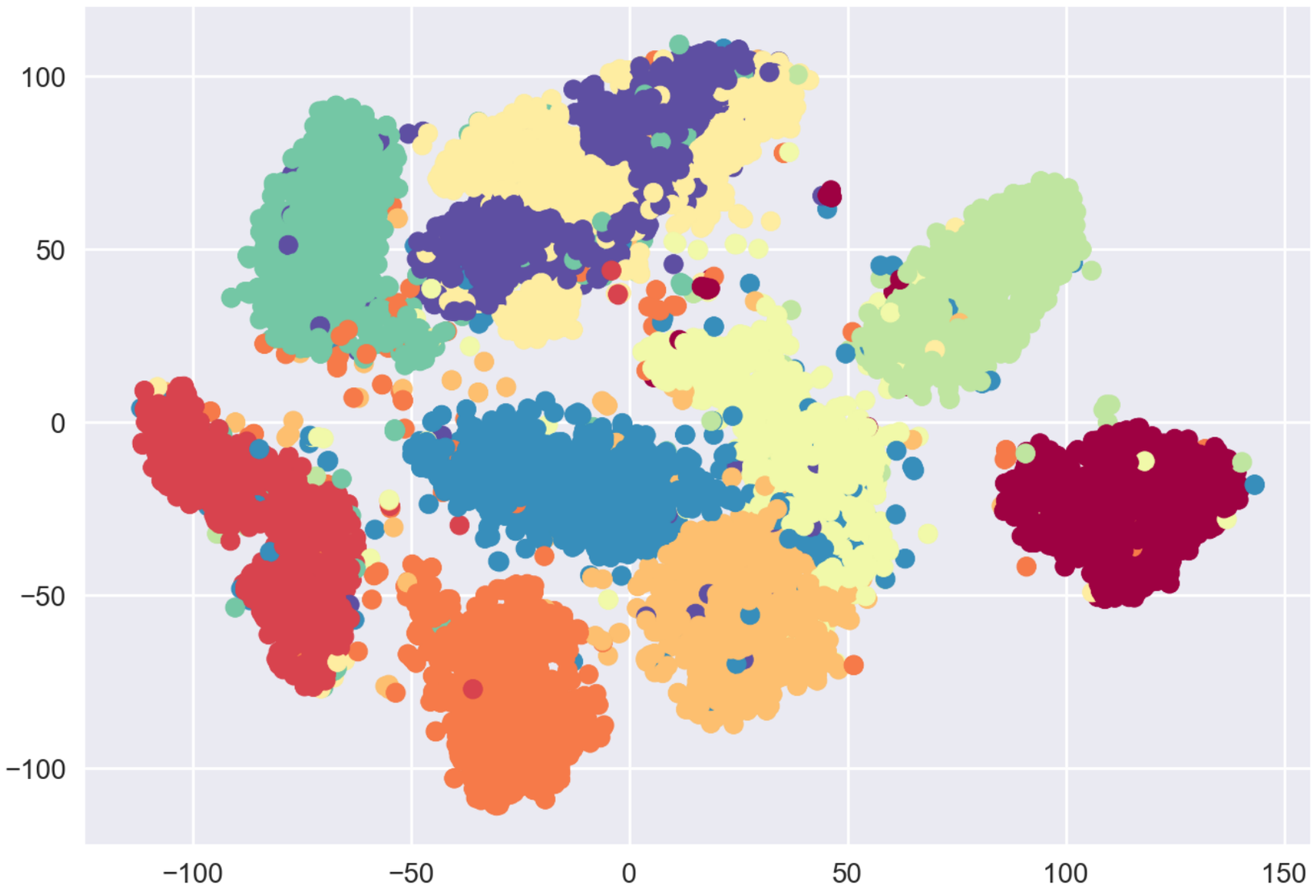
- Frequency and co-occurrence analysis
- Visualizing high-dimensional data/dimensionality reduction
- Clustering
- Topic modeling

Part II: Predictive data analysis

Make predictions using known structure in data

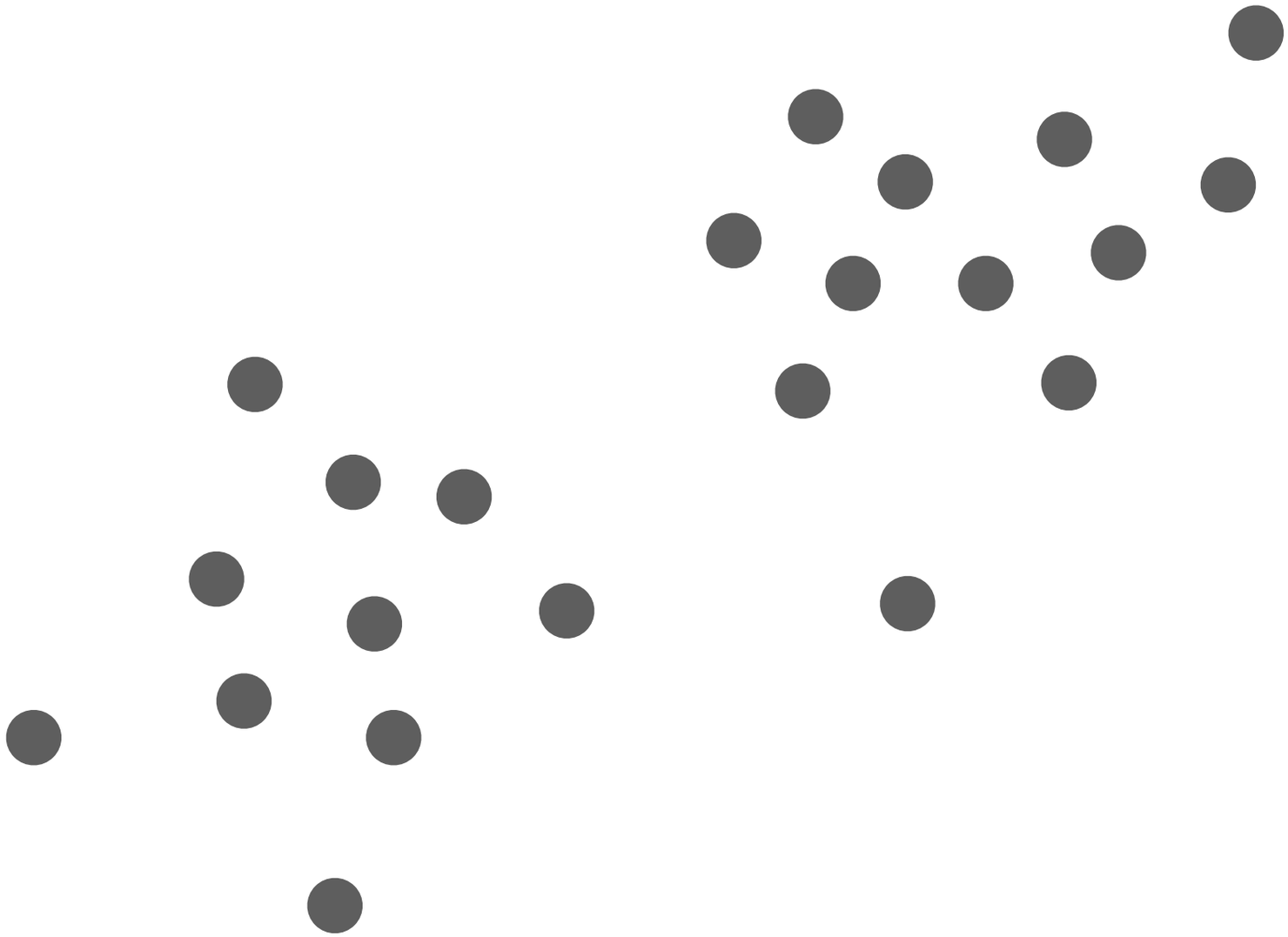
- Basic concepts and how to assess quality of prediction models
- Neural nets and deep learning for analyzing images and text

What if we have labels?



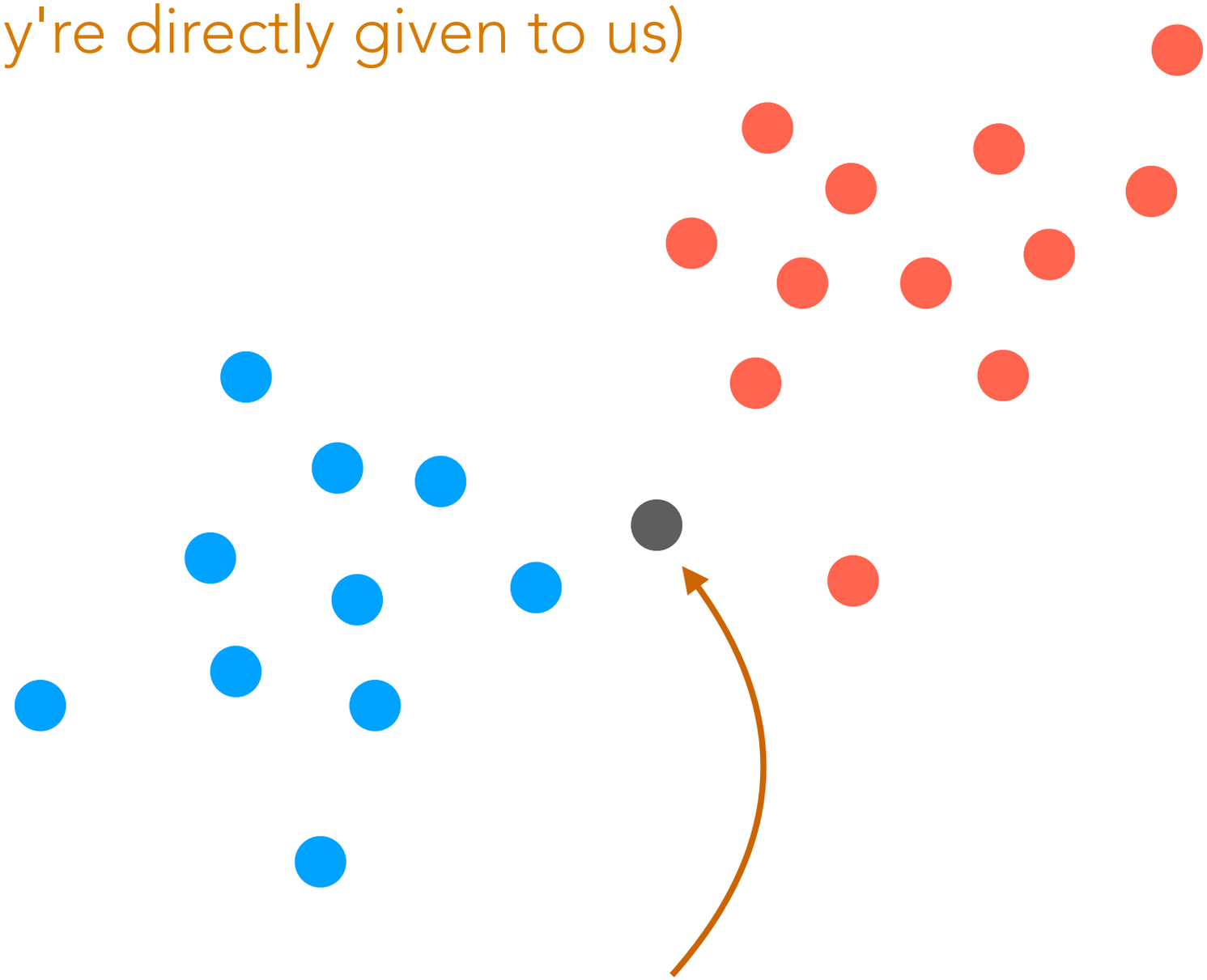
Example: MNIST handwritten digits have known labels

If the labels are known…



If the labels are known…

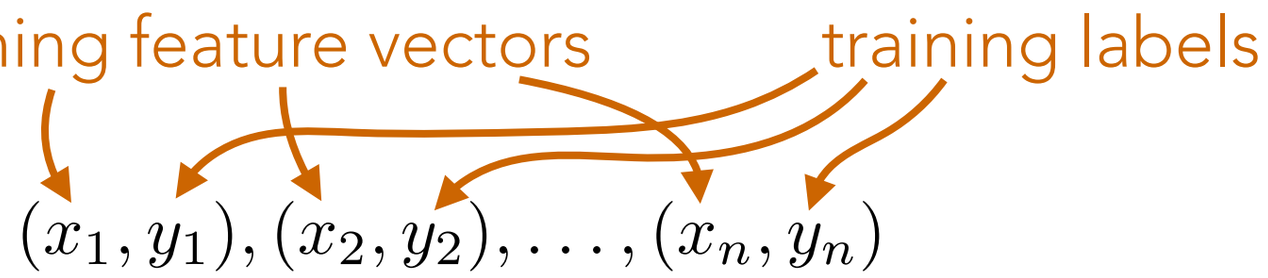
We no longer need to try to guess cluster assignments
(since they're directly given to us)



Prediction: for a test data point, what is its label?

Predictive Data Analysis

Training data



Goal: Given new test feature vector x, predict label y

- y is discrete (such as colors red and blue)
  ➔ prediction is referred to as **classification**
- y is continuous (such as a real number)
  ➔ prediction is referred to as **regression**

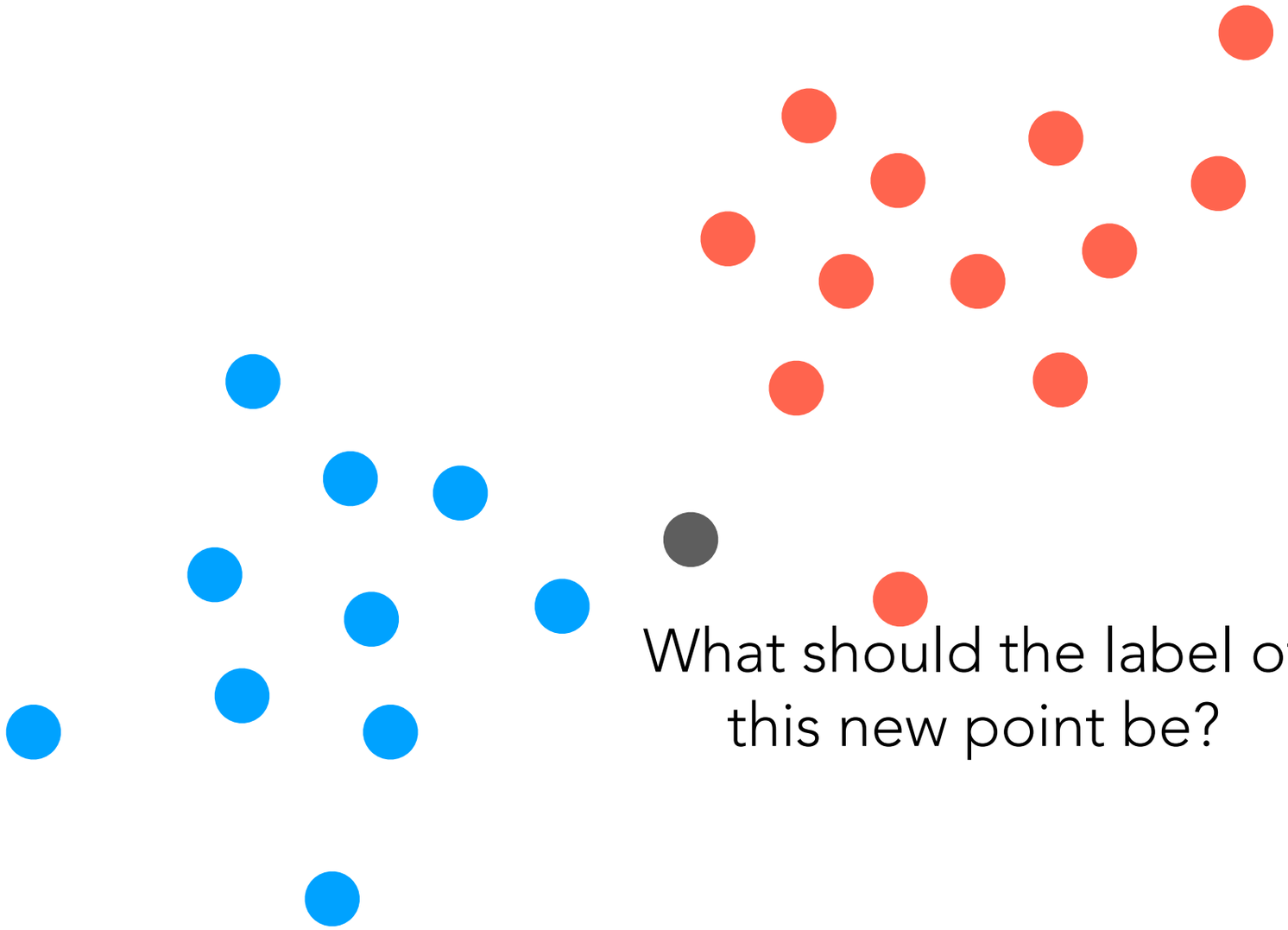
We could have many such test feature vectors, which we collectively refer to as *test data*

For handling unstructured data, far and away the most popular standard approach now is to use **neural nets & deep learning** for prediction

Before we start our coverage of neural nets & deep learning, we first cover basics of prediction

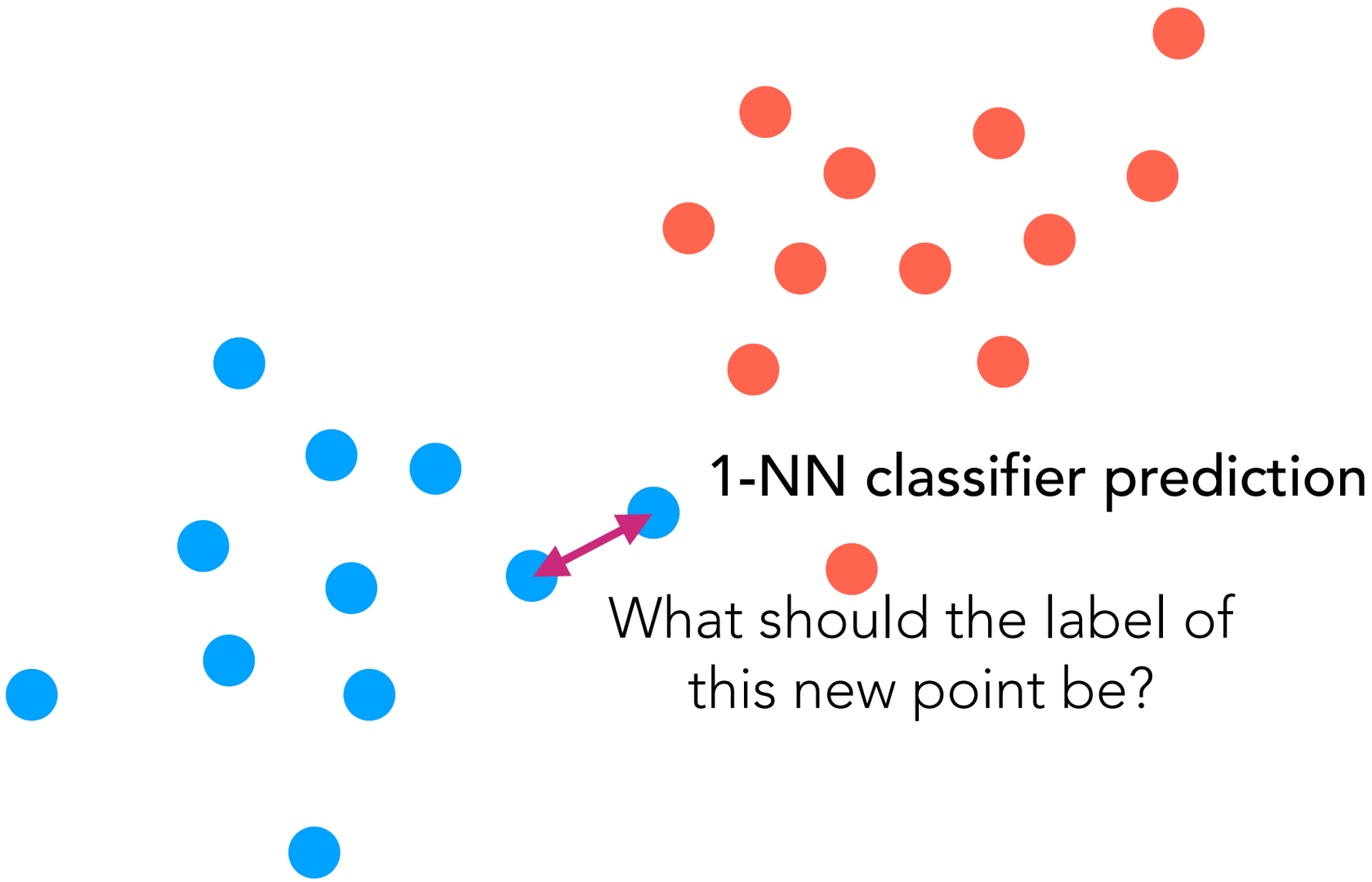
Example: *k*-NN Classification

Example: *k*-NN Classification

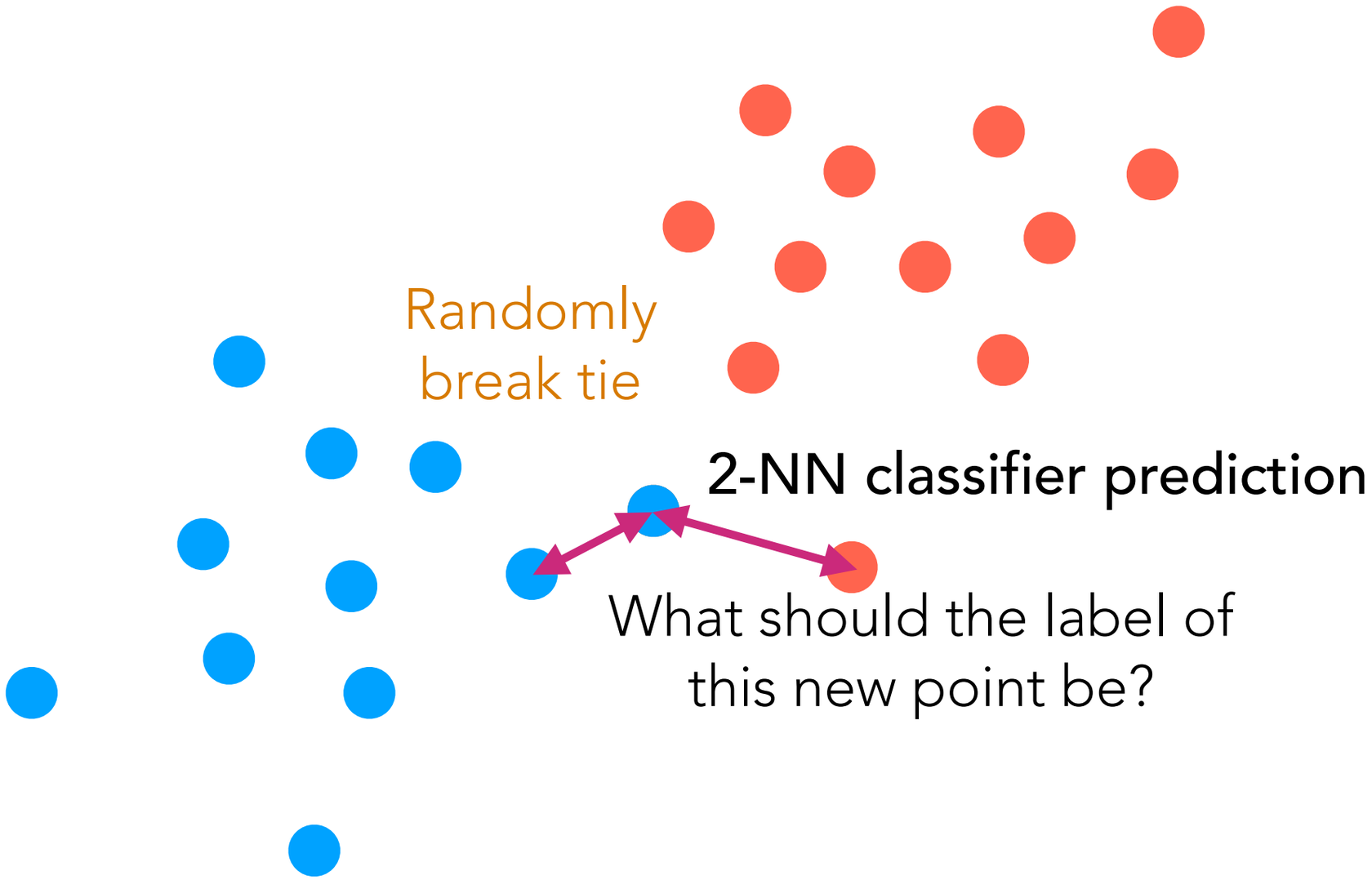


What should the label of this new point be?

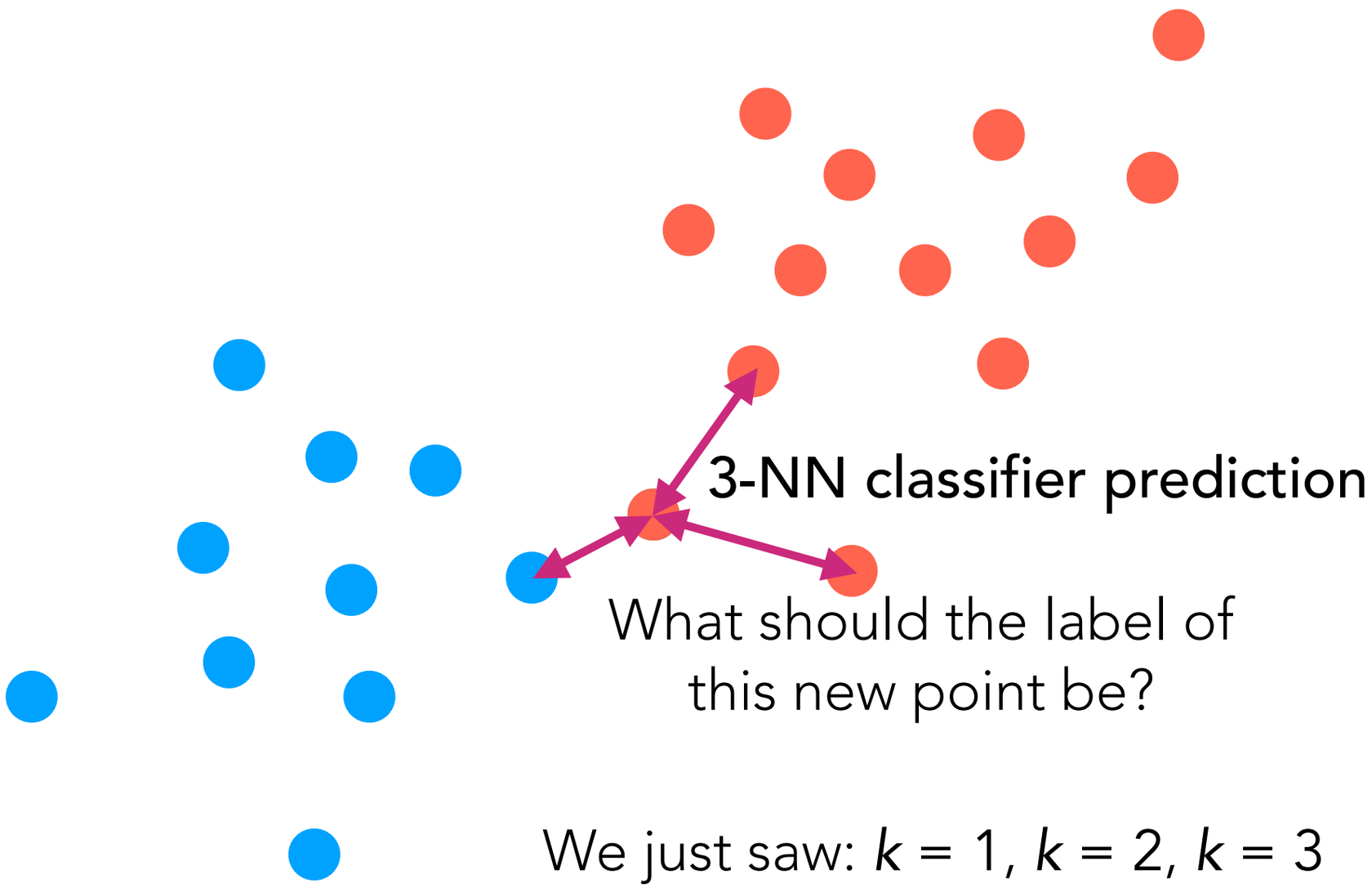
Example: *k*-NN Classification



Example: *k*-NN Classification



Example: *k*-NN Classification

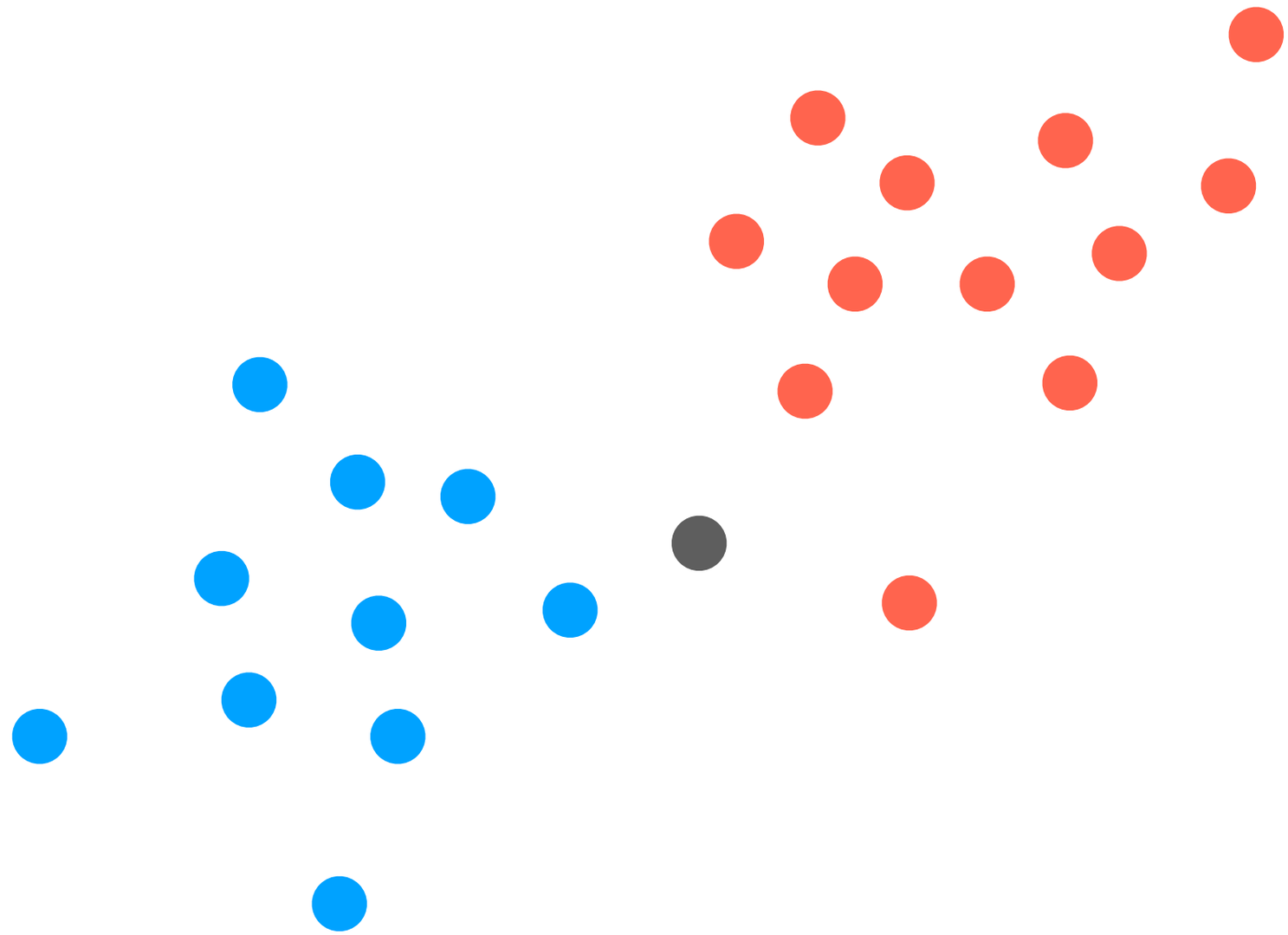


What should the label of this new point be?

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

What happens if $k = n$?

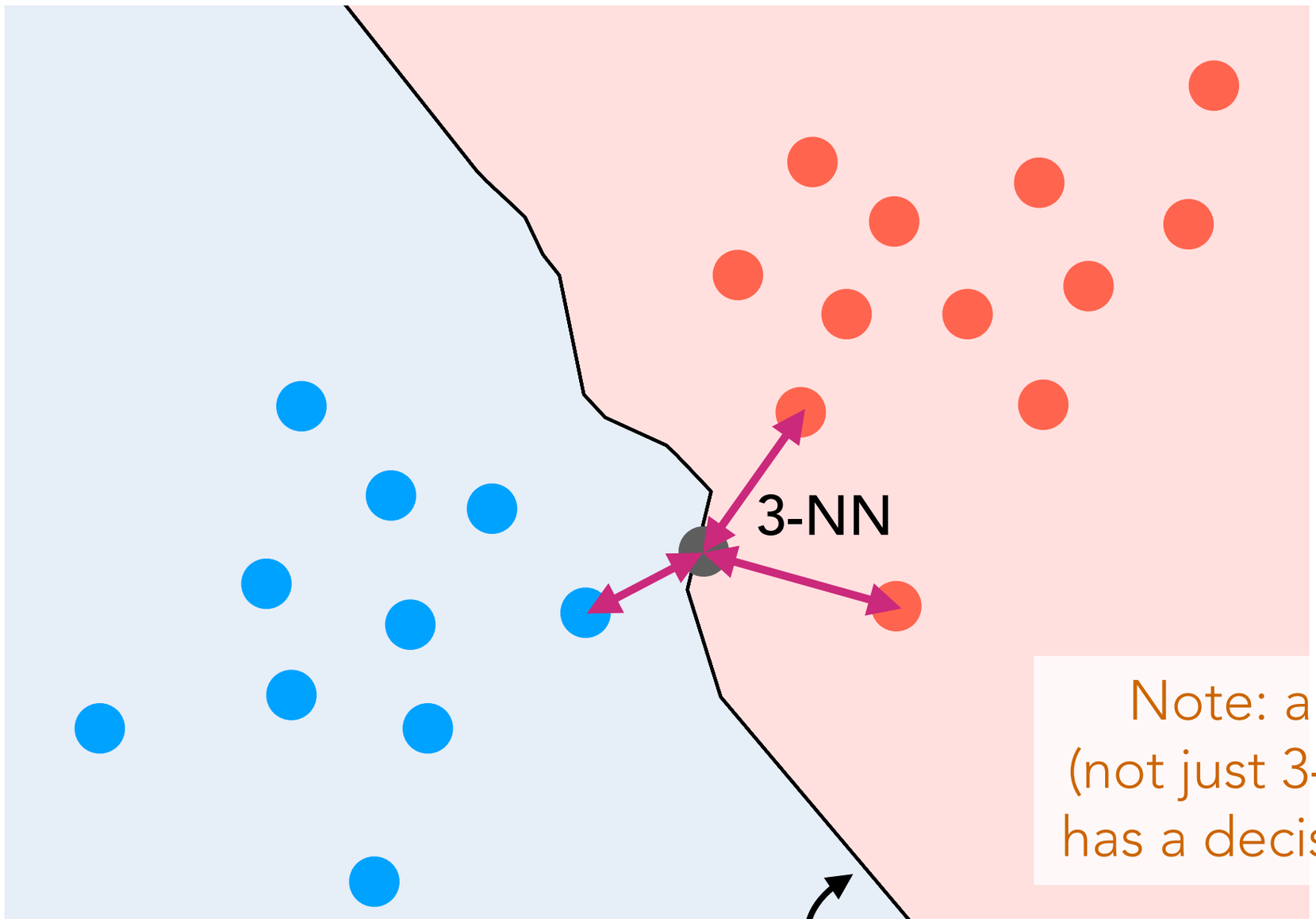
Example: *k*-NN Classification



For any test feature vector ***x***:

1. Find the ***k*** closest training feature vectors to ***x***
2. Use the most popular label among the ***k*** training points found (breaking ties arbitrarily) to use as the predicted label for ***x***

Example: *k*-NN Classification



If test point on right of decision boundary → 3-NN classifier predicts red

If test point on left of decision boundary → 3-NN classifier predicts blue

How do we choose *k*?

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

Fundamental question:
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* automatically get fit to training data

- Example: *k*-means
  - Hyperparameter: number of clusters *k*
  - Parameters: cluster centers

- Example: *k*-NN classification
  - Hyperparameter: number of nearest neighbors *k*
  - Parameters: N/A

Actually, there's another hyperparameter: distance function to use
(for simplicity, we assume Euclidean distance for now)

⚠️ Major assumption:
training and test data “look alike”
(technically: training and test data are i.i.d. sampled from the same underlying distribution)

Prediction is harder when training and test data appear quite different!

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

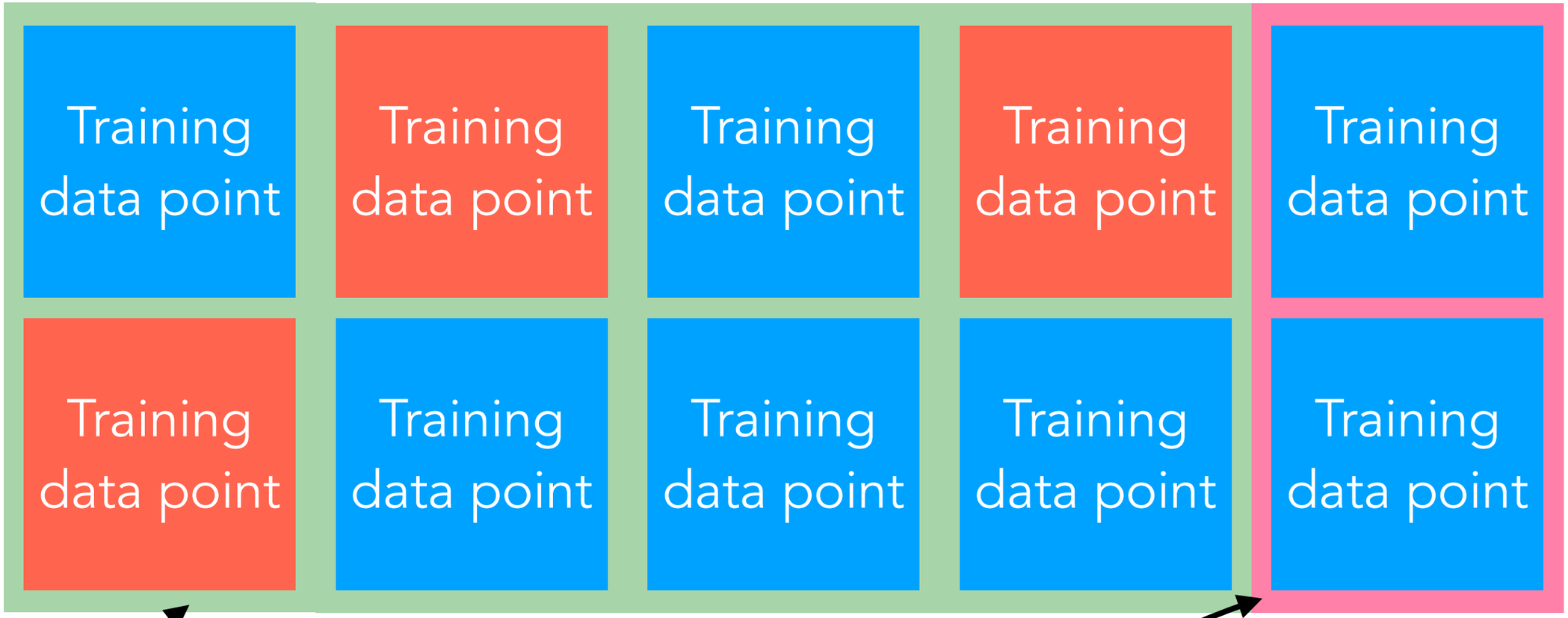
Test data point

Test data point

Test data point

Example: future emails to classify as spam/ham

(shuffling makes sense since we assume data are i.i.d.)



Train method on data in green

Predict on data in pink

Terminology for this class:
"**Proper training data**"
(the green box)

"**Validation data**"
(the pink box)

This is called data splitting/"train-validation split"

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

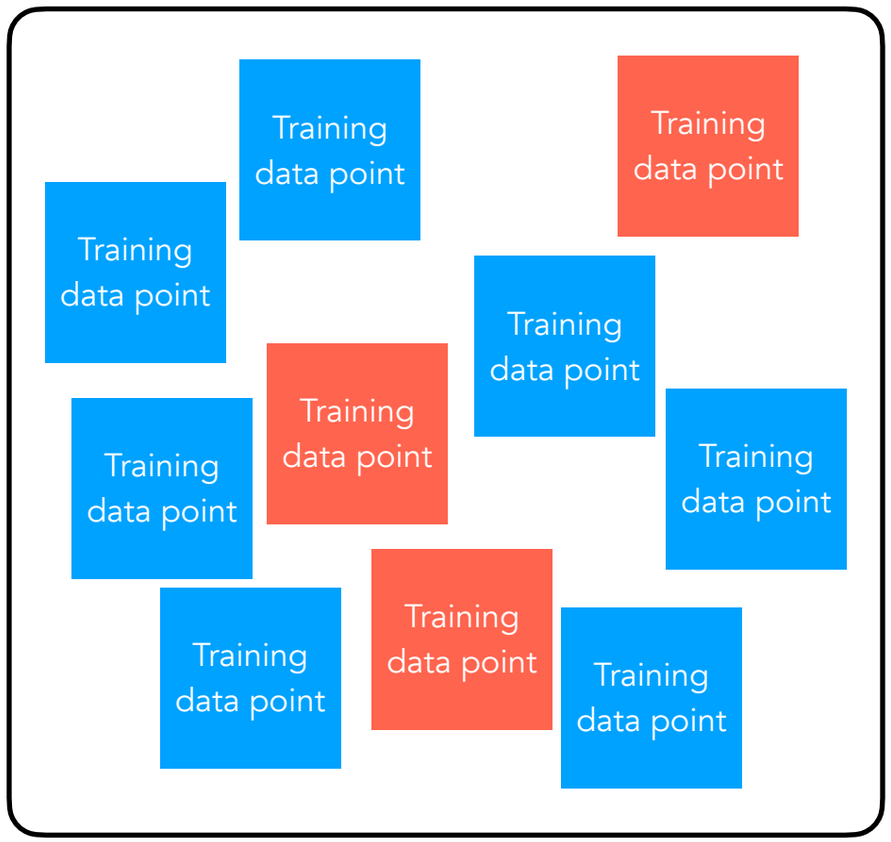
Predicted labels on validation data

Compute prediction accuracy

50%

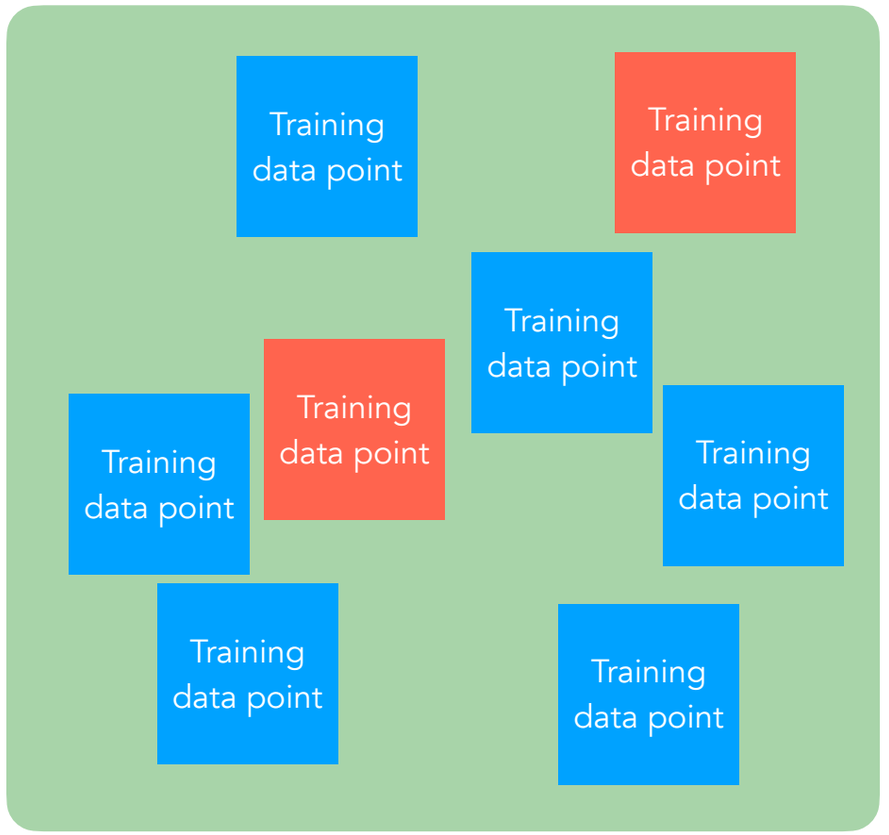
Some people, including sklearn, call this "train-test split" but in this class, we will use "test data" to refer to true test data that the training procedure does not see

Training data



Randomly split into two portions (example: 80% / 20%)

"Proper training data"



"Validation data"

For $k = 1, 2, \ldots,$ some user-specified max:

1. Train k-NN classifier on proper training data
2. Use a score function to evaluate how well the trained model predicts on validation data

Use whichever value of k achieves the best score

There are many score functions possible

Example: fraction of validation points correctly predicted (raw accuracy)

Terminology Remarks

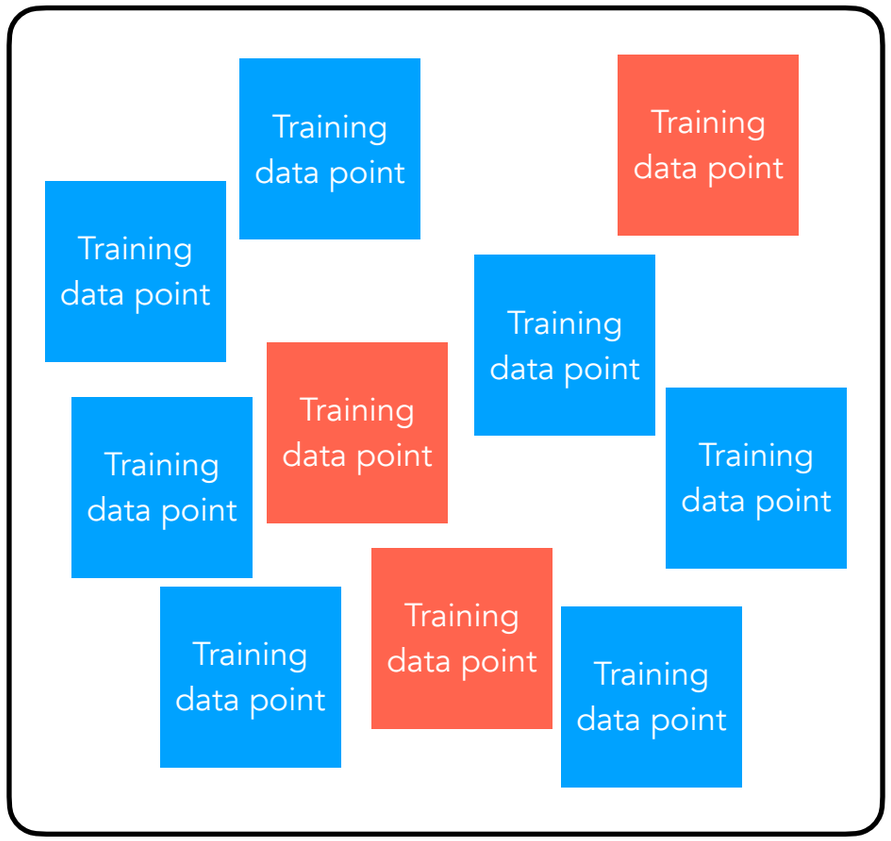
- What we're using is commonly called a **train/validation split**
  - If you also consider that there's a **test set** that's not part of train/validation data: division is called **train/validation/test split**

- **Warning:** in the machine learning community, what I'm calling the "proper training data"/"proper training set" is commonly also called the "training data"/"training set" even though it is typically a *subset* of the full training data (that we split into proper training/validation sets)
  - Put another way: what precisely the "training data" refers to can be ambiguous as it could mean the full training data or the full training data minus the validation data
  - In 95-865, to avoid confusion, we use the somewhat non-standard terminology "proper training set"/"proper training data" to refer to the the full training data minus the validation data

Hyperparameter Tuning for *k*-NN Classifier

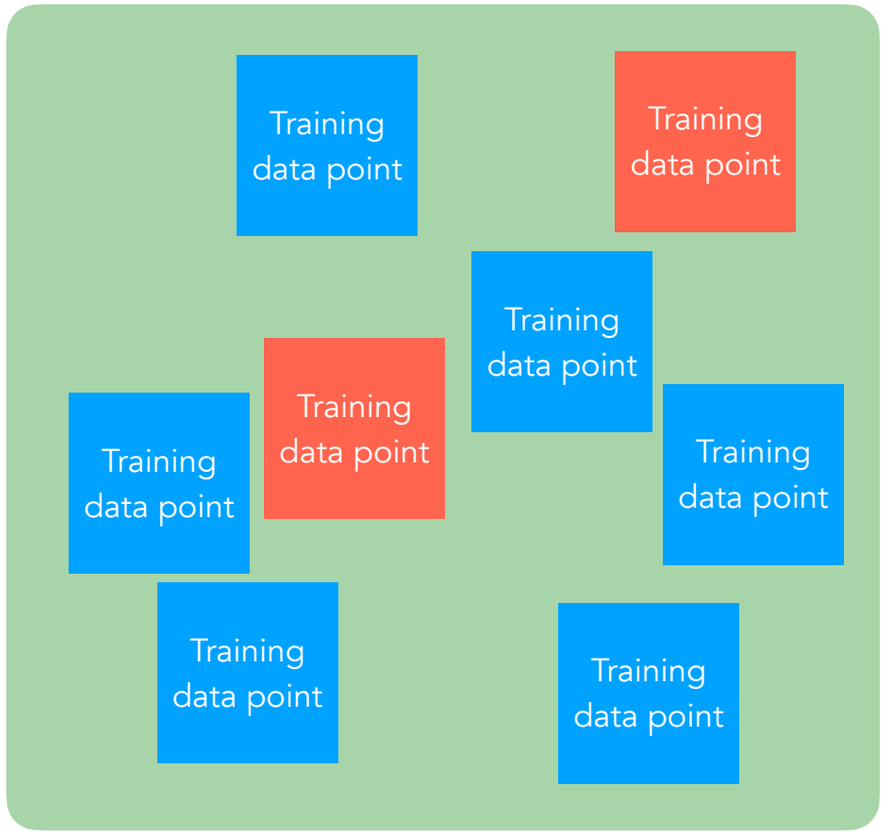
Demo

Training data



Randomly split into two portions (example: 80% / 20%)

"Proper training data"



"Validation data"

For $k = 1, 2, \ldots,$ some user-specified max:

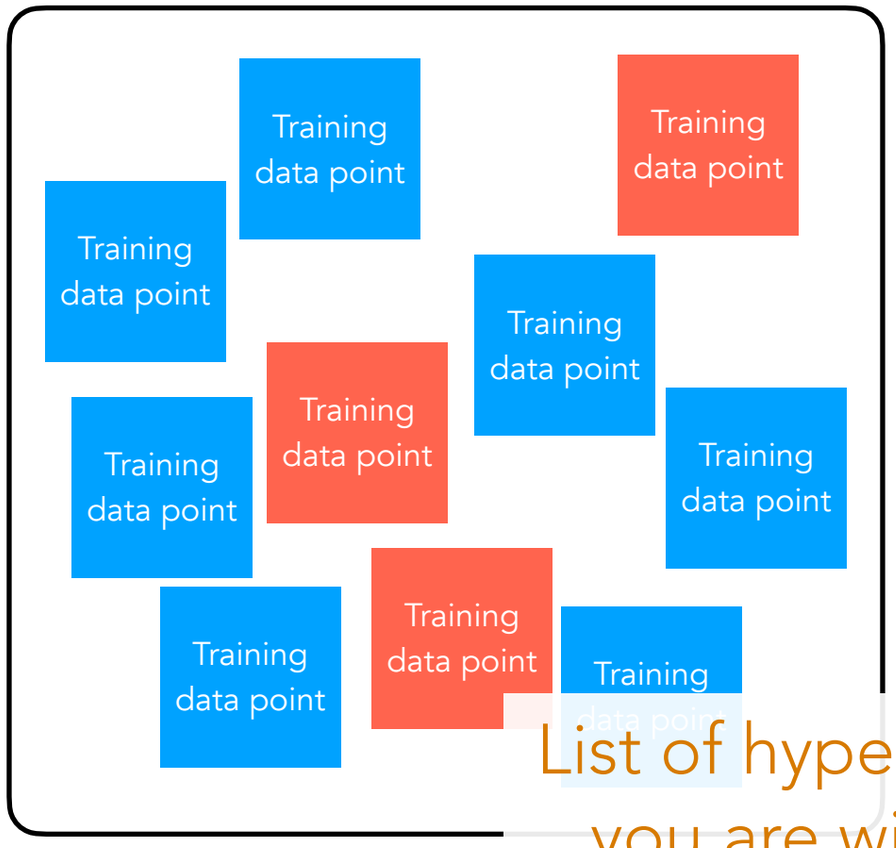
1. Train k-NN classifier on proper training data
2. Use a score function to evaluate how well the trained model predicts on validation data

Use whichever value of k achieves the best score

There are many score functions possible

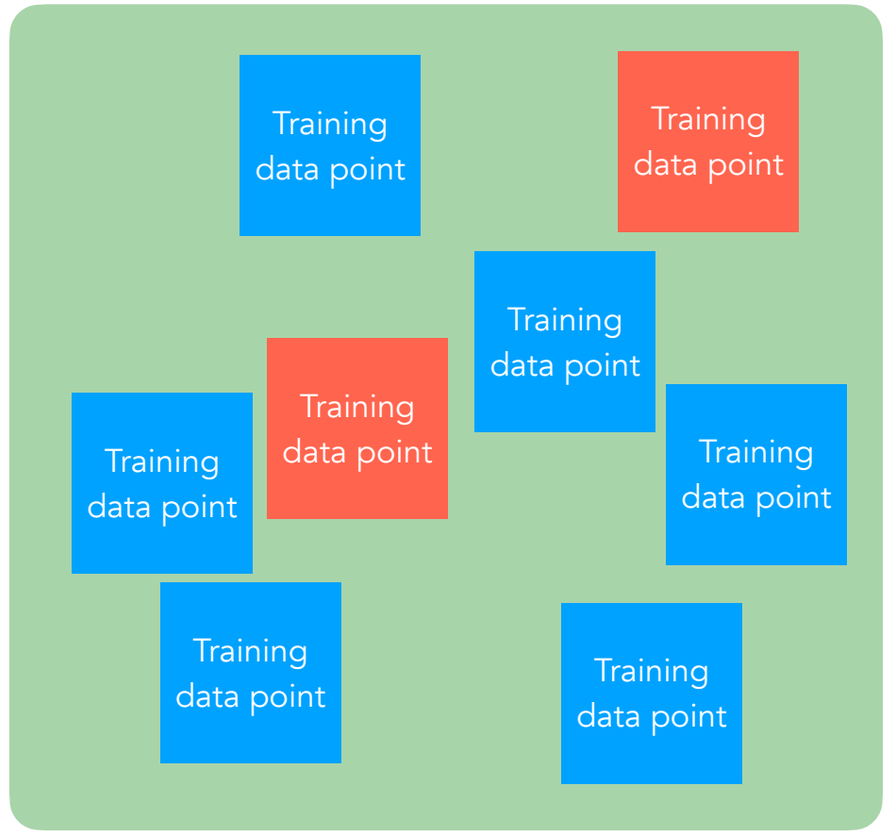
Example: fraction of validation points correctly predicted (raw accuracy)

Training data



Randomly split into two portions (example: 80% / 20%)

"Proper training data"



List of hyperparameters you are willing to try

For k, ρ = (1, "Euclidean"), (1, "Cosine"), …:

1. Train k-NN classifier on proper training data with distance dist ρ
2. Use a score function to evaluate how well the trained model predicts on validation data

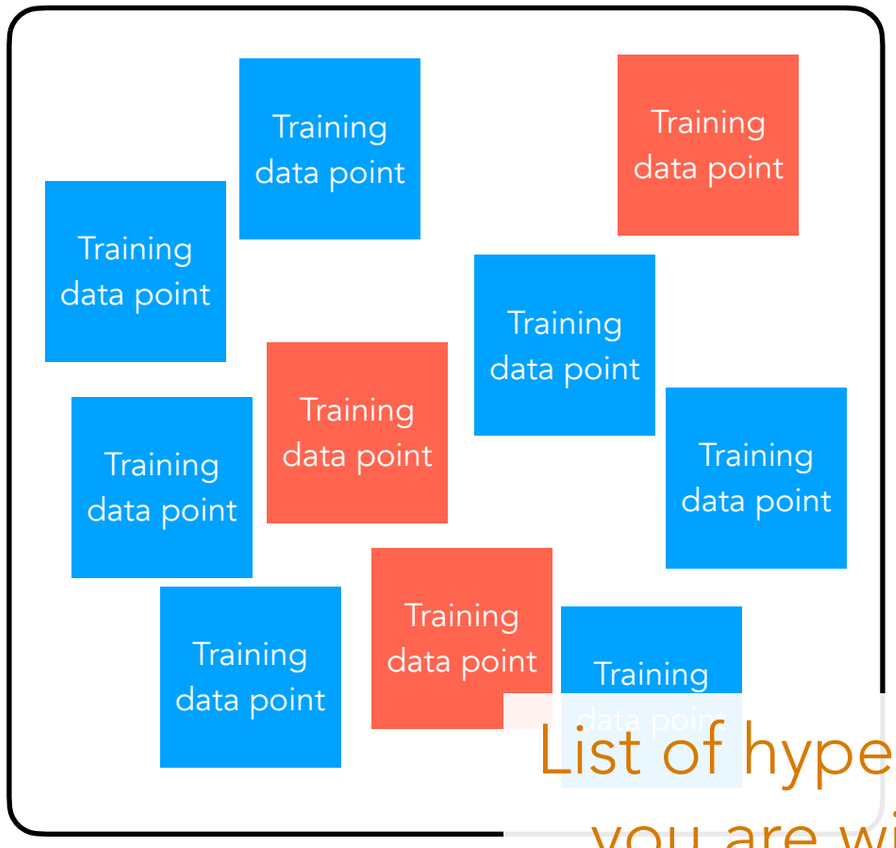
Use whichever value of k achieves the best score

"Validation data"

There are many score functions possible

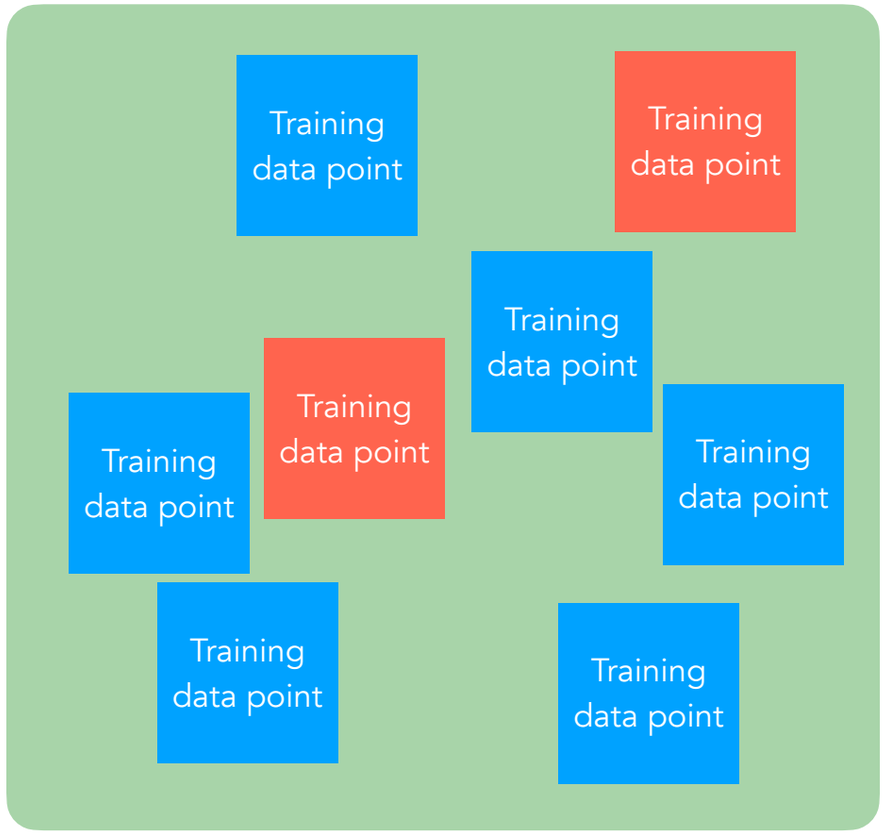
Example: fraction of validation points correctly predicted (raw accuracy)

Training data



Randomly split into two portions (example: 80% / 20%)

"Proper training data"



List of hyperparameters you are willing to try

For θ in Θ

1. Train prediction model on proper training data with hyperparameter setting θ
2. Use a score function to evaluate how well the trained model predicts on validation data

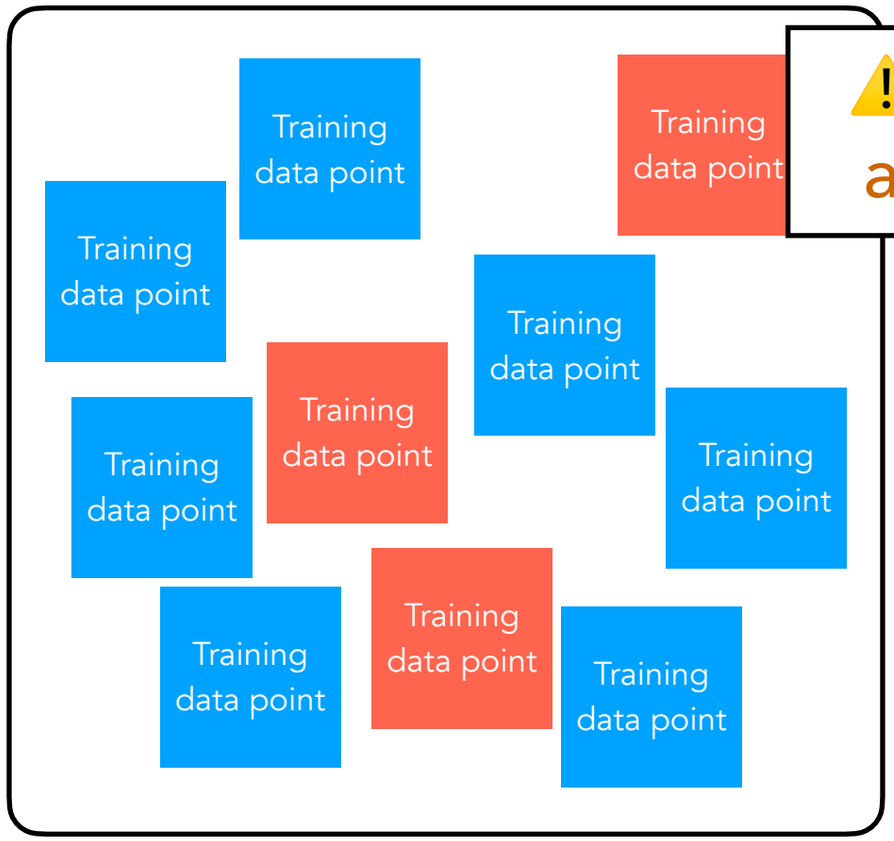
Use whichever value of θ achieves the best score

"Validation data"

There are many score functions possible

Example: fraction of validation points correctly predicted (raw accuracy)

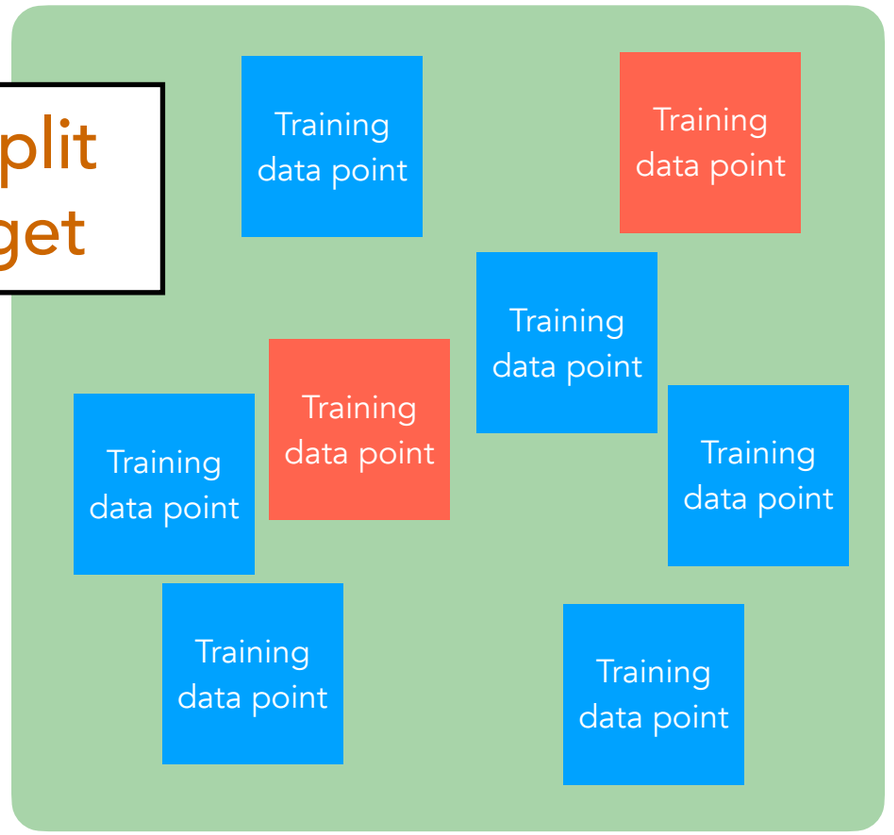
Training data



⚠ How we randomly split affects the scores we get

Randomly split into two portions (example: 80% / 20%)

"Proper training data"



"Validation data"

For θ in Θ

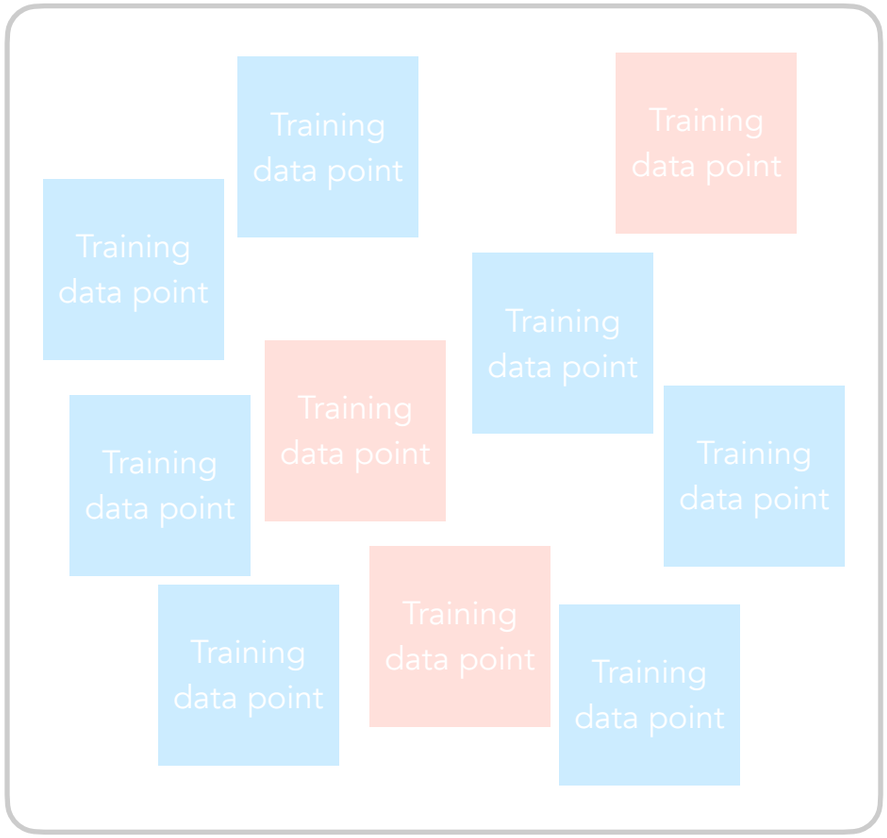
1. Train prediction model on proper training data with hyperparameter setting θ
2. Use a score function to evaluate how well the trained model predicts on validation data

Use whichever value of θ achieves the best score

There are many score functions possible

Example: fraction of validation points correctly predicted (raw accuracy)

Training data



Randomly split into two portions (example: 80% / 20%)

"Proper training data"

"Validation data"

For θ in Θ

1. Train prediction model on proper training data with hyperparameter setting θ
2. Use a score function to eval… trained model predicts on v…

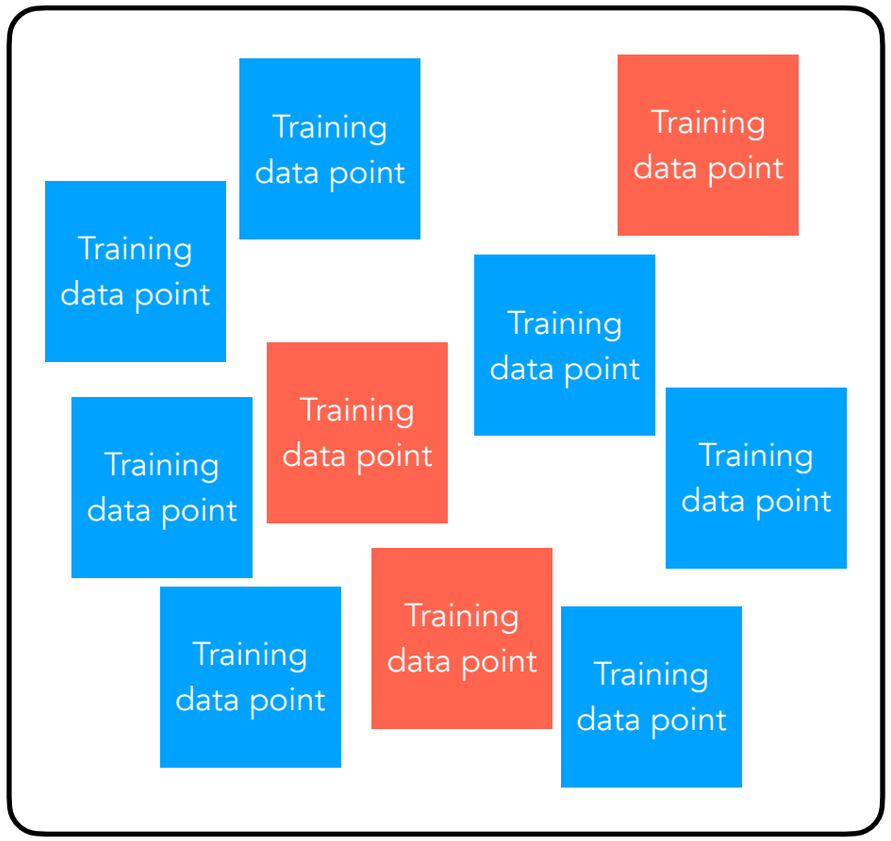
⚠ Randomness in model training affects the scores we get

Use whichever value of θ achieves the best score

There are many score functions possible

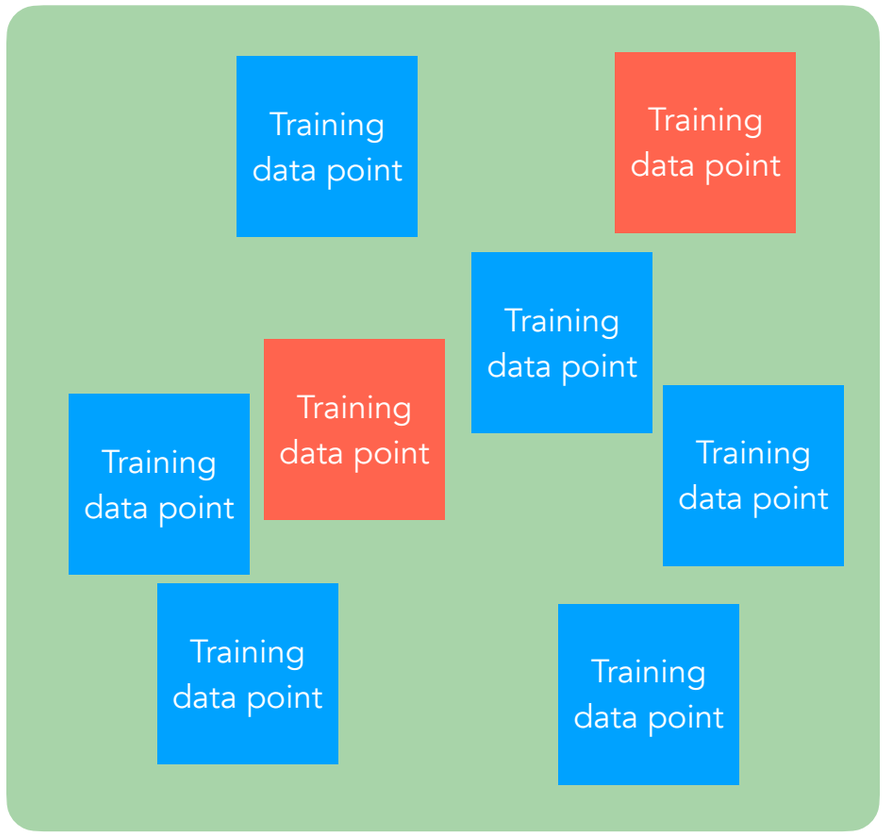
Example: fraction of validation points correctly predicted (raw accuracy)

Training data



Randomly split into two portions (example: 80% / 20%)

"Proper training data"



"Validation data"

For θ in Θ

1. Train prediction model on proper training data with hyperparameter setting θ
2. Use a score function to evaluate how well the trained model predicts on validation data

Use whichever value of θ achieves the best score

There are many score functions possible

Example: fraction of validation points correctly predicted (raw accuracy)

The rest of the prediction models we consider in UDA will be based on *neural nets* **(which commonly have hyperparameters!)**

Neural net models can be tuned in the same manner we just saw for k-NN classification

Important: you may have seen *cross-validation* before

- If you don't know/remember what this is, don't worry about it
- Cross-validation is commonly too expensive for neural net training so we stick to the train/val split strategy

Neural Nets & Deep Learning

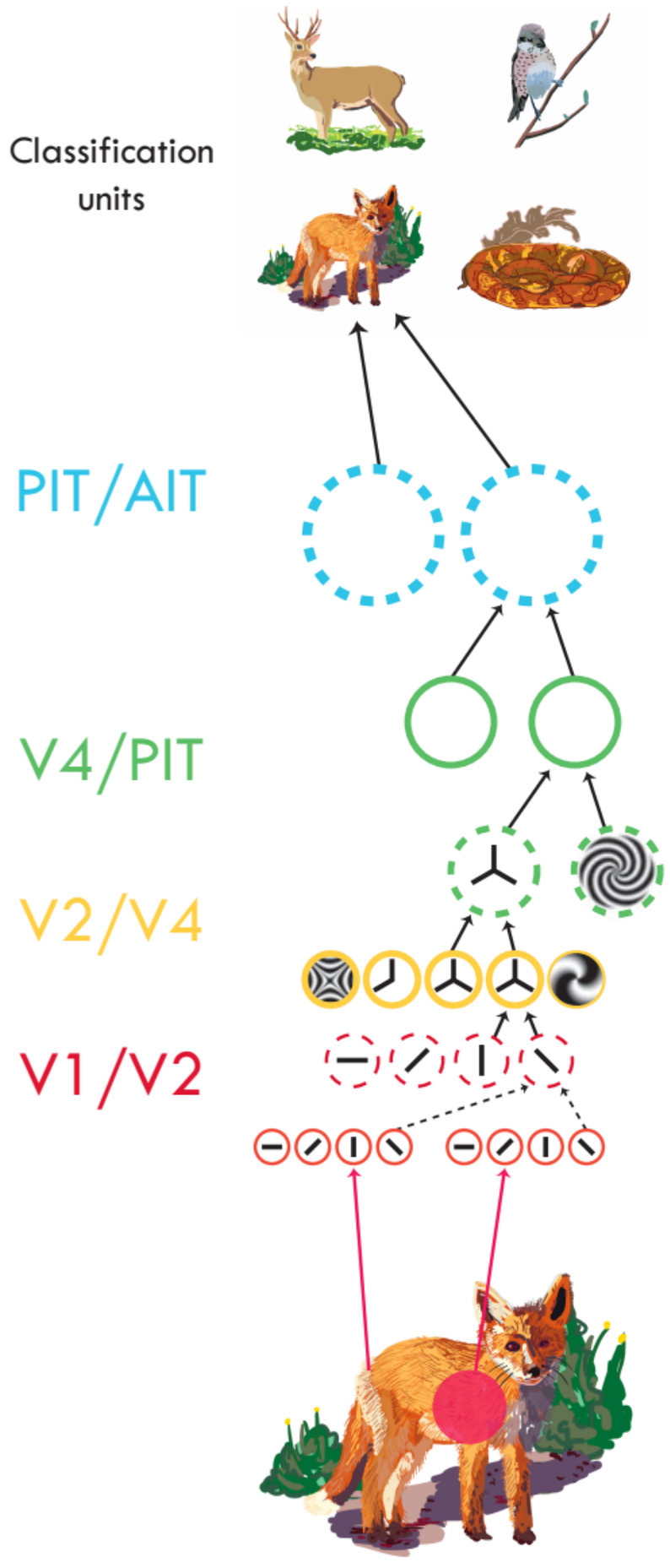
Extremely useful in practice:

- Human-level image classification
- Human-level speech recognition
- Human-level in machine translation, text-to-speech
- Self-driving cars
- Better than humans at playing Go and many other games
- Capable of generating fake images, video, and audio that look real
- Human-level chatbots (ChatGPT, GPT4.0, Gemini, Claude, …)

⚠️ We don't fully understand when many of these technologies fail or how best to prevent their misuse

⚠️ All of this technology will get better over time

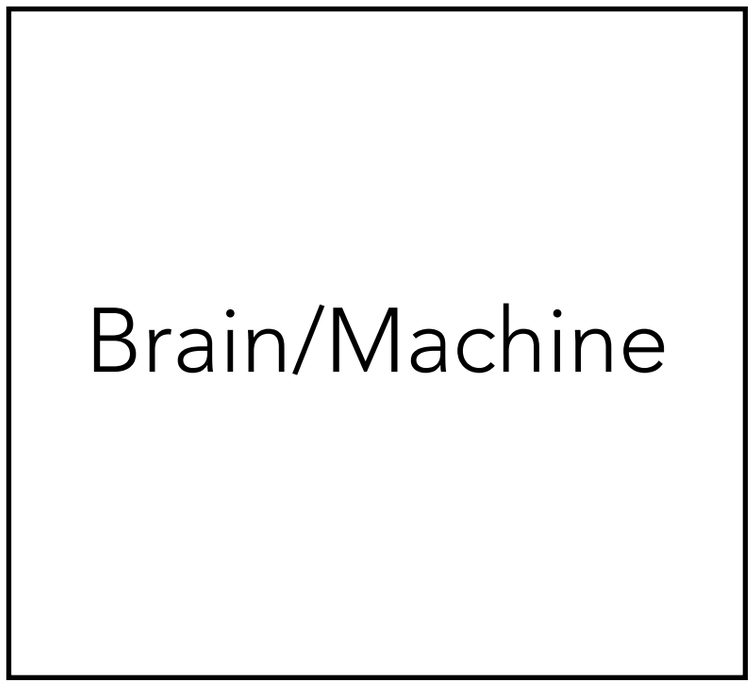
What is deep learning?



Slide by Phillip Isola

Serre, 2014

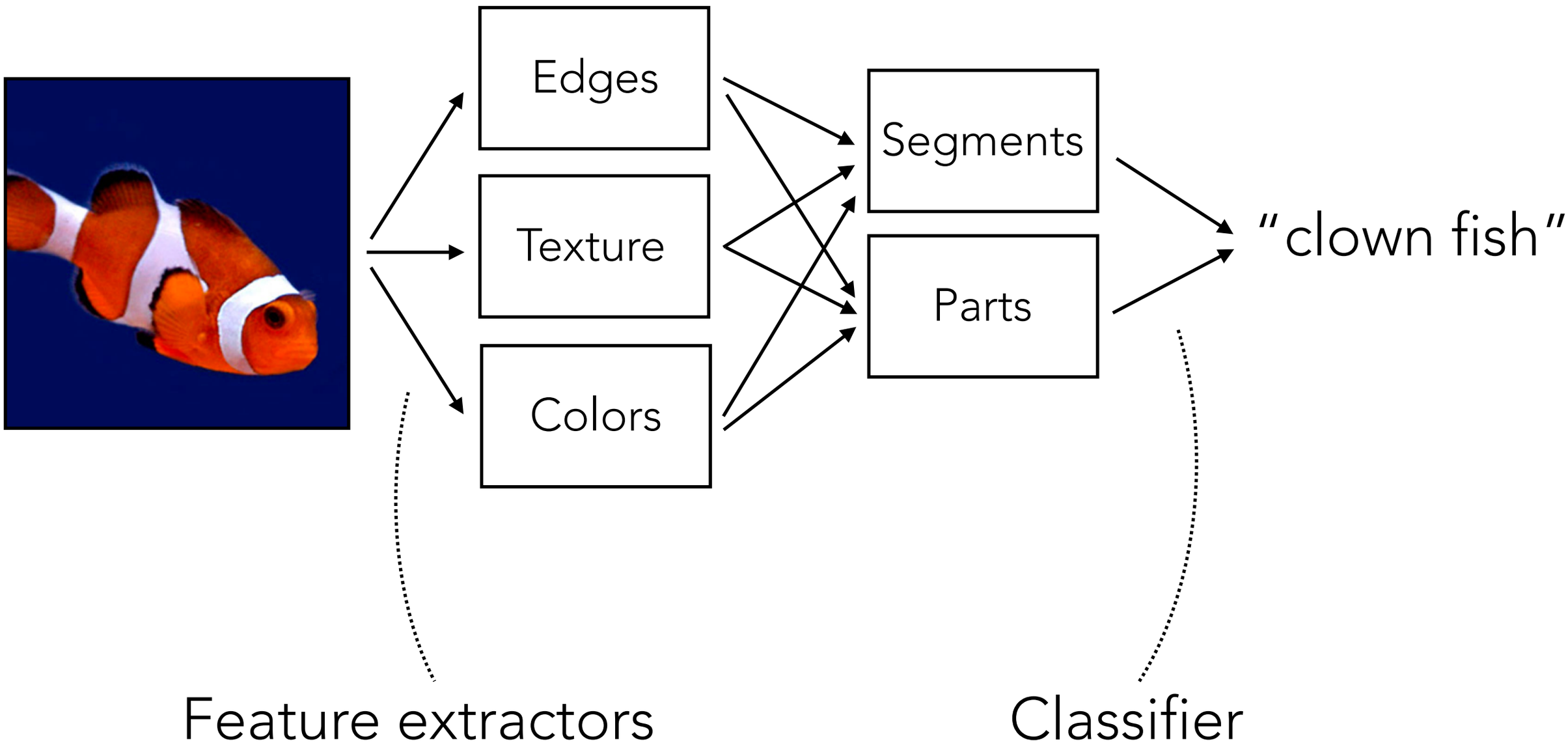
Basic Idea



“clown fish”

Slide by Phillip Isola

Classical Object Recognition



Slide by Phillip Isola

Classical Object Recognition

Learned

Edges

Texture

Colors

Segments

Parts

"clown fish"

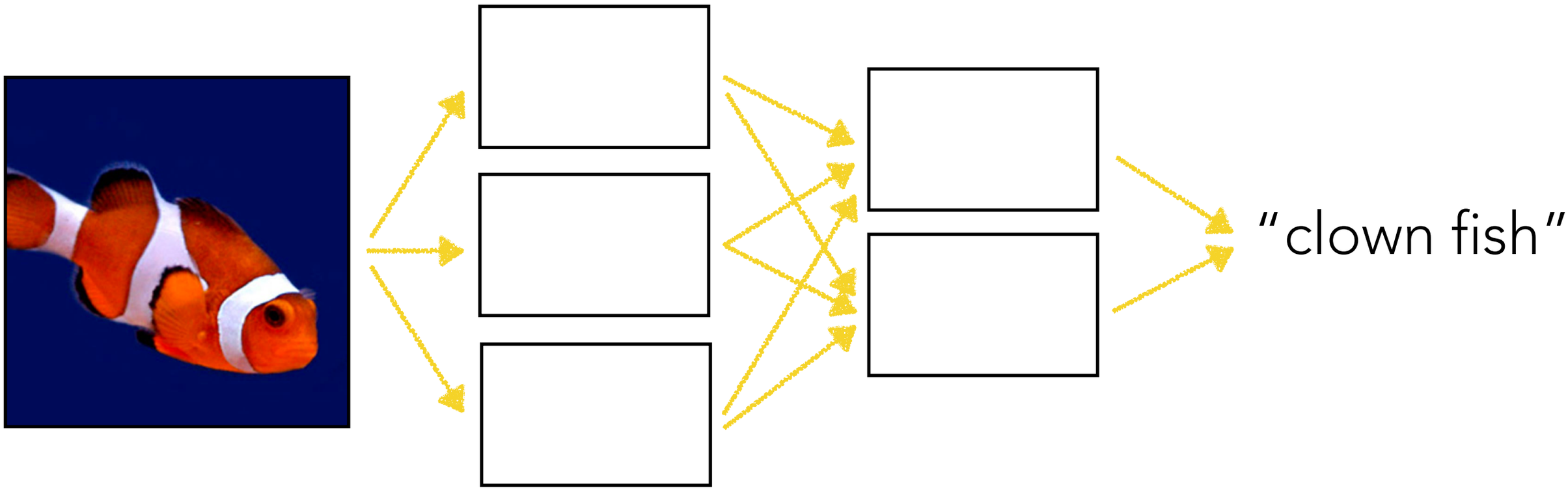
Feature extractors

Classifier

Slide by Phillip Isola

Neural Network

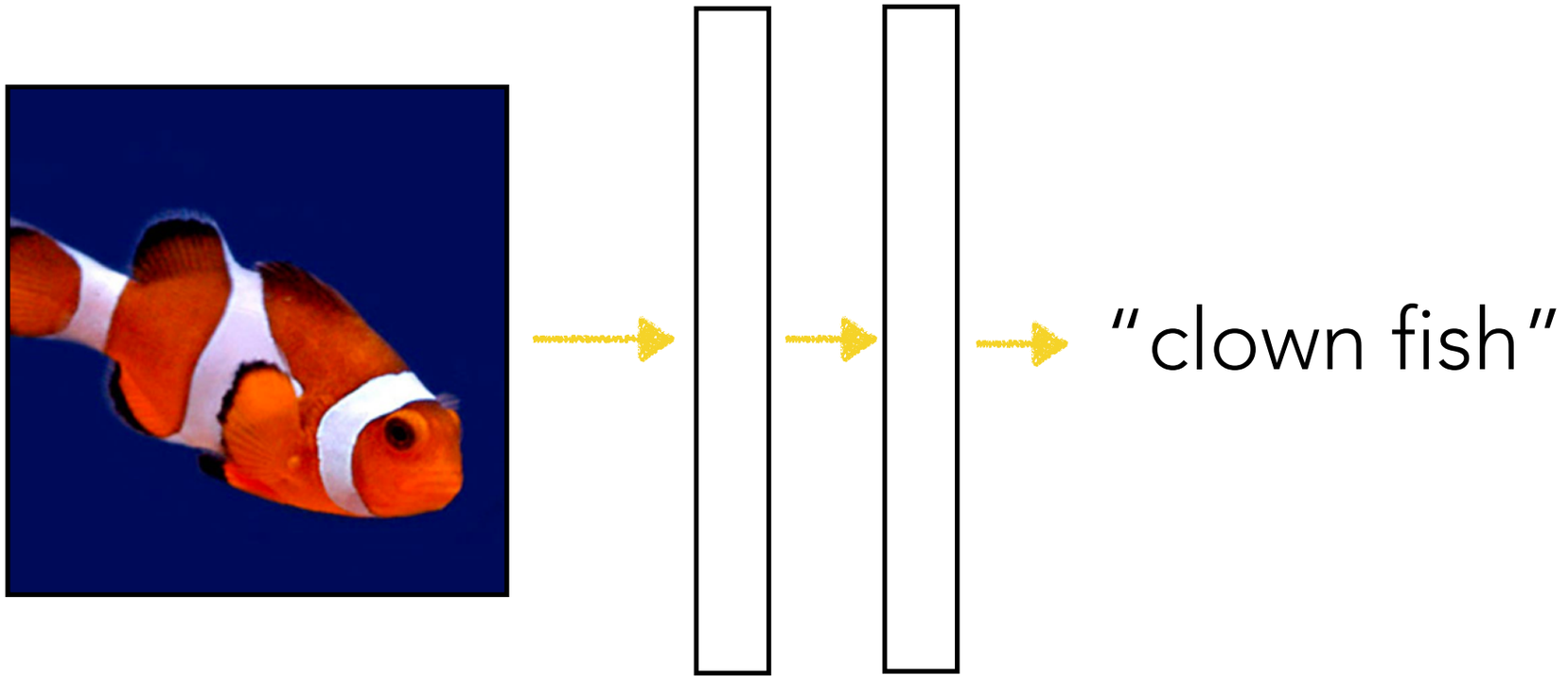
Learned



Slide by Phillip Isola

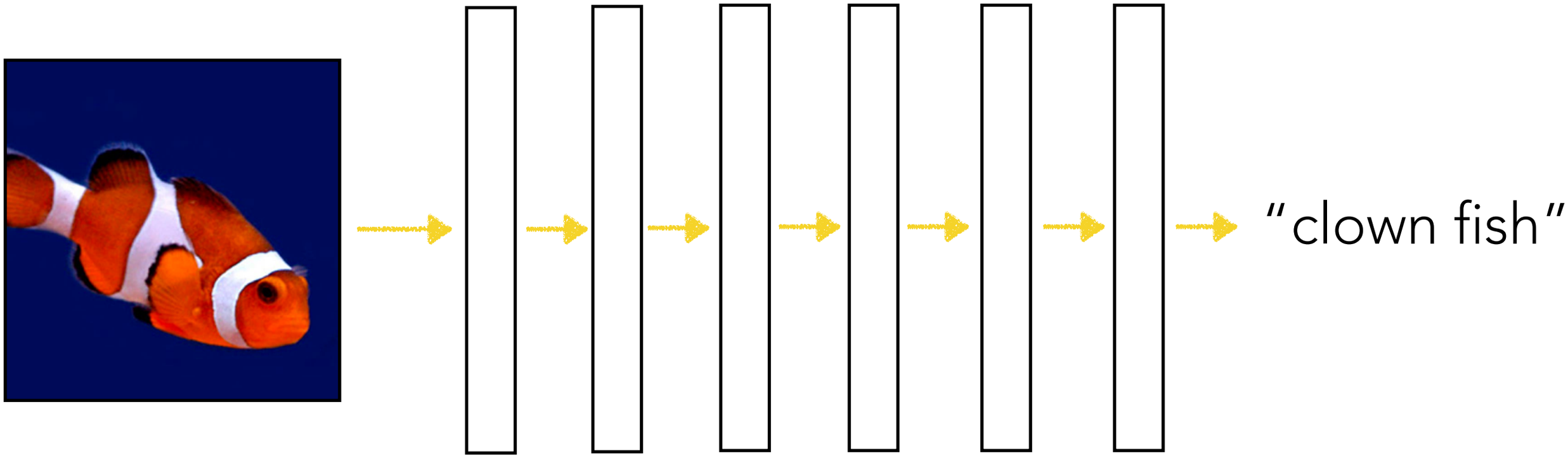
Neural Network

Learned



Slide by Phillip Isola

Deep Neural Network

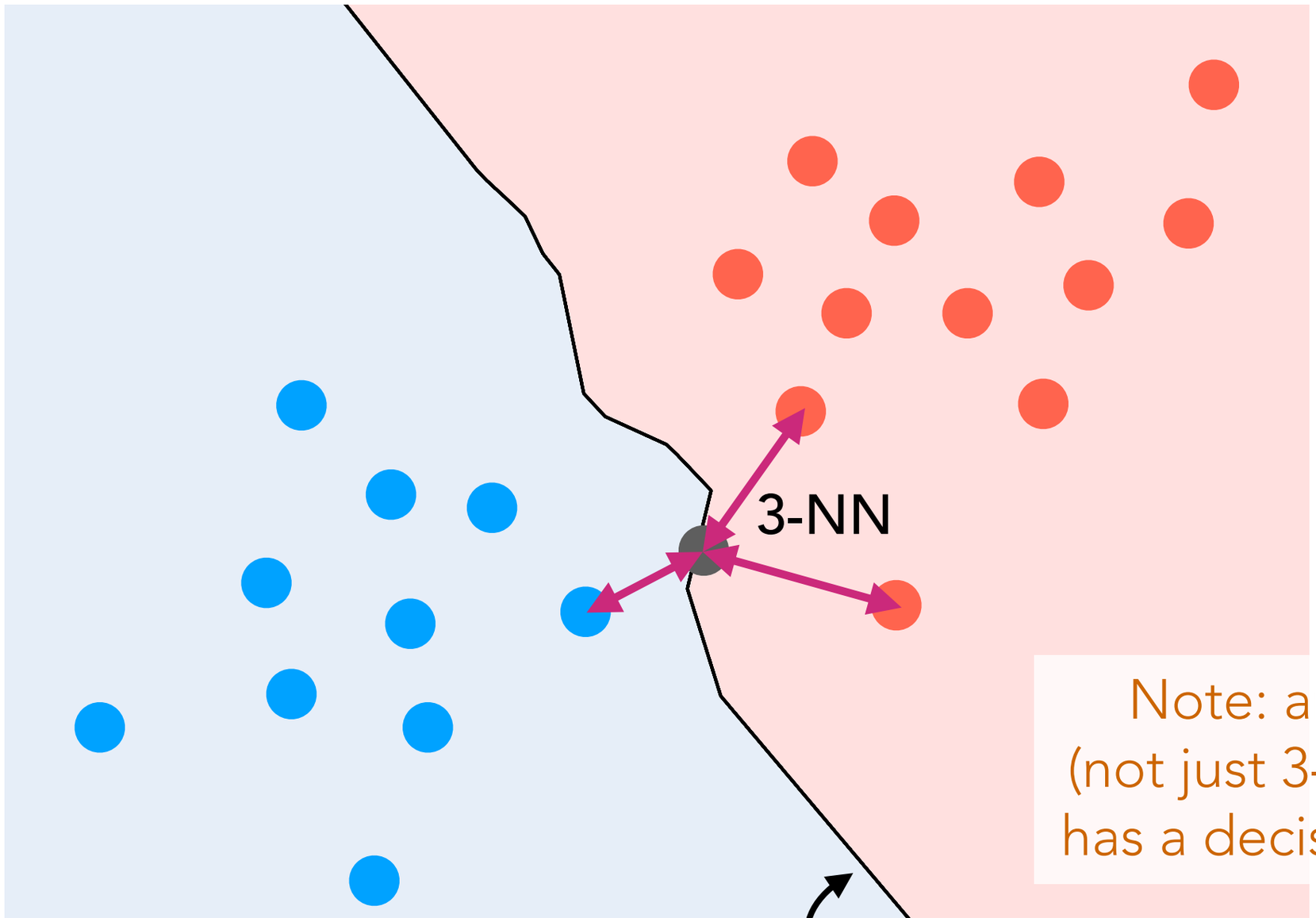


Deep learning just refers to learning deep neural nets

Slide by Phillip Isola

The Crumpled Paper Analogy

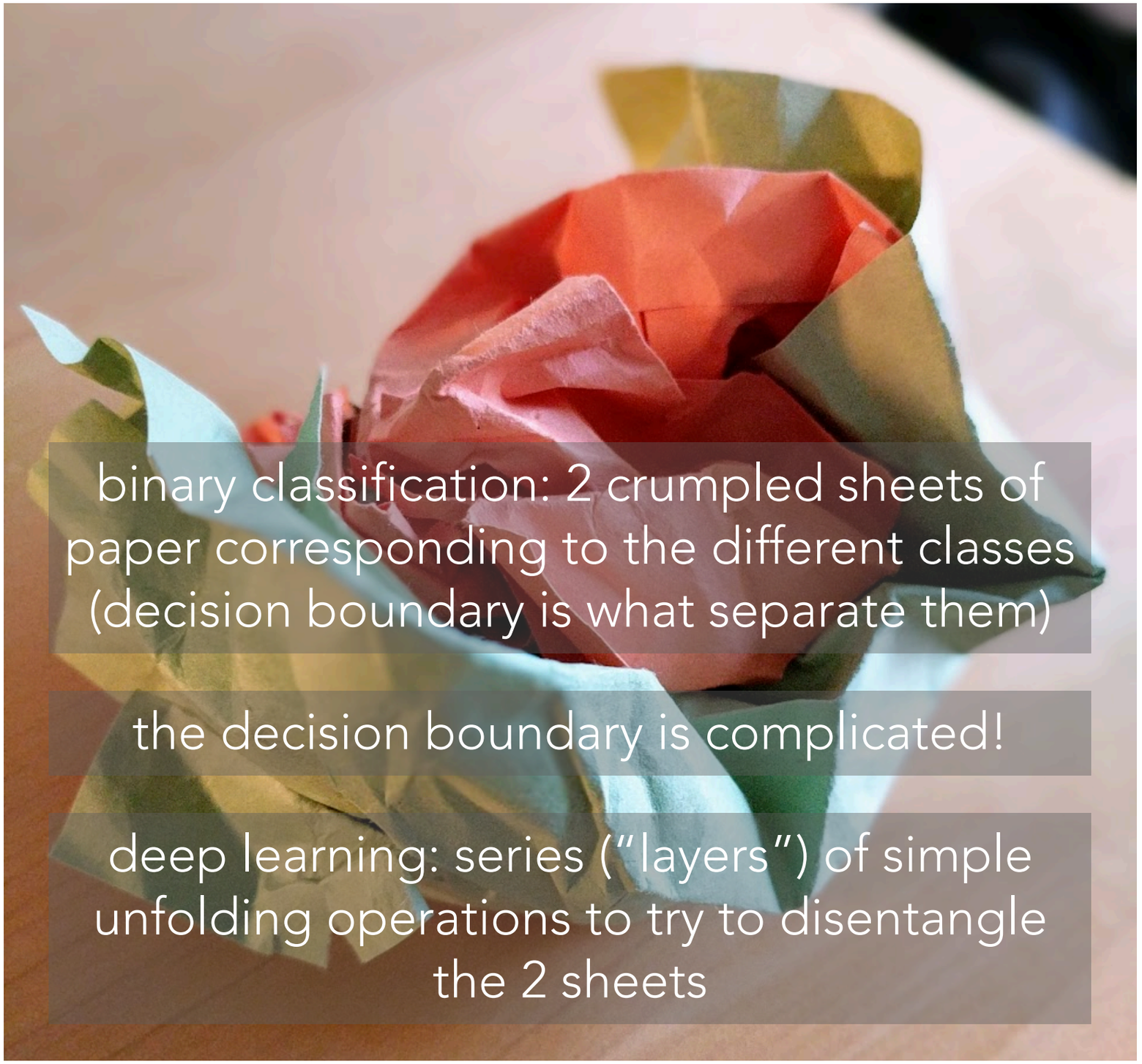
(Flashback) Example: *k*-NN Classification



If test point on right of decision boundary → 3-NN classifier predicts red

If test point on left of decision boundary → 3-NN classifier predicts blue

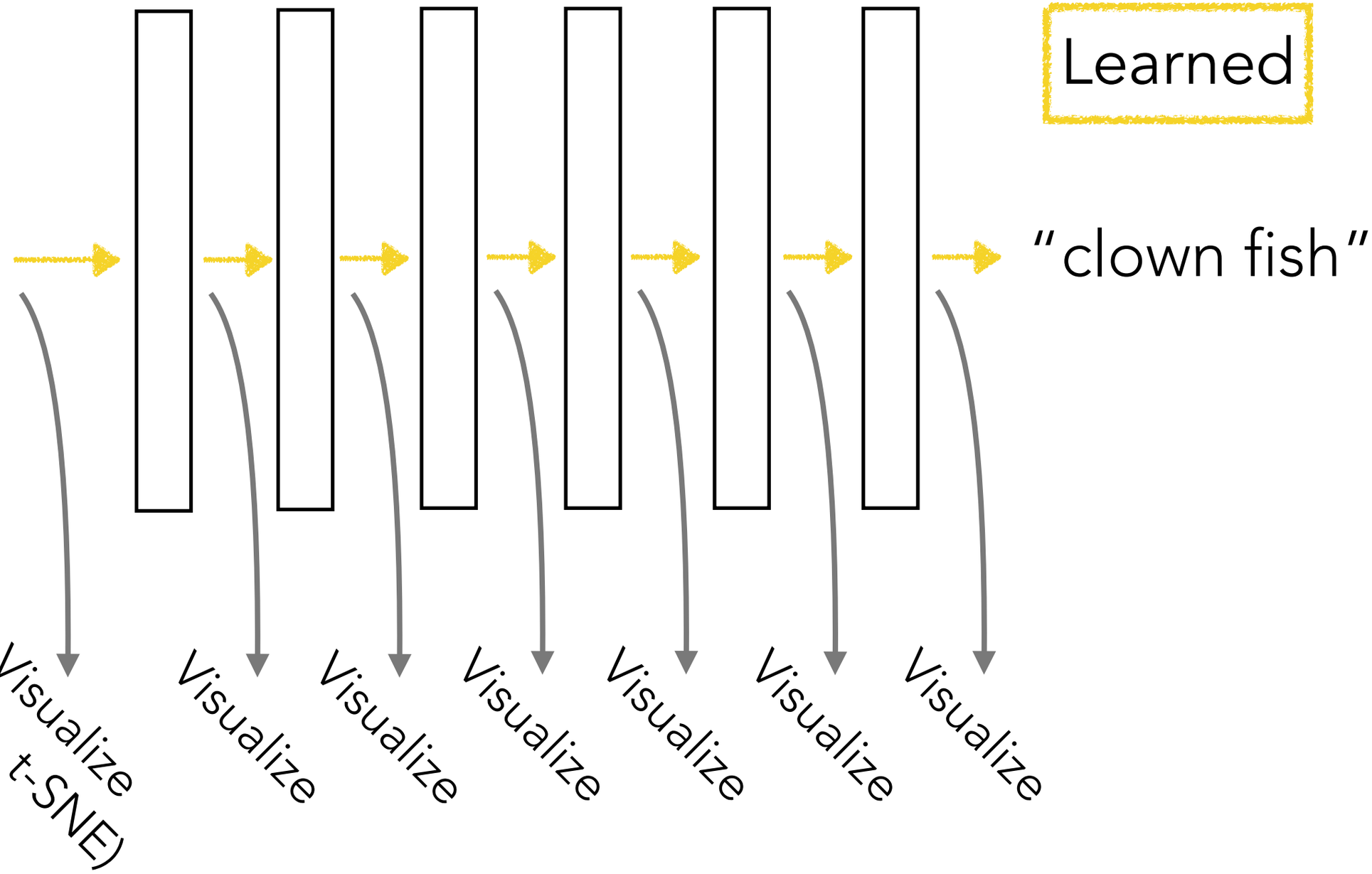
Crumpled Paper Analogy



Analogy: Francois Chollet, photo: George Chen

Representation Learning

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



Representation Learning

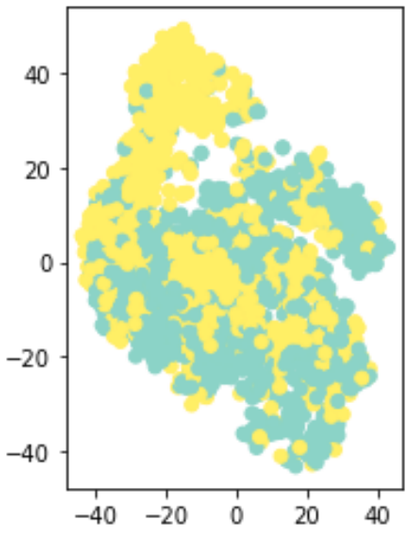
Each layer's output is *another way we could represent the input data*

Learned

classifier

"clown fish"

Visualize (e.g., t-SNE)



Visualize