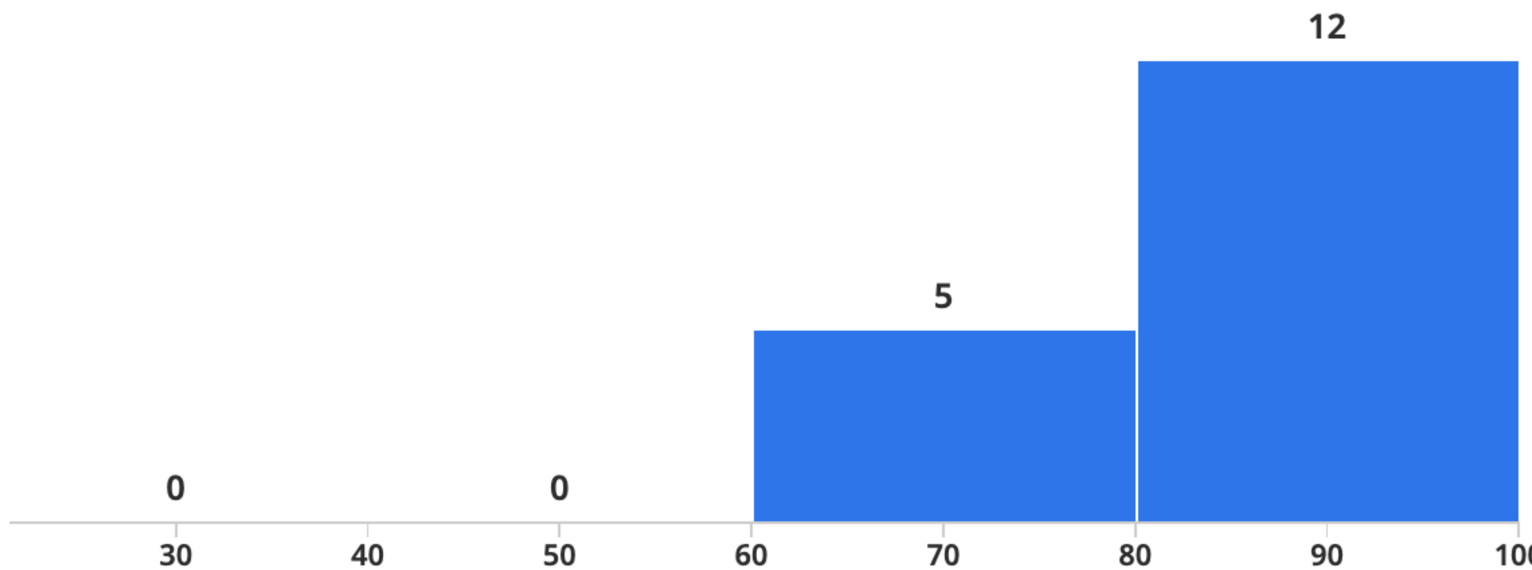
Carnegie Mellon University
Heinz College

94-775 Unstructured Data Analytics

Lecture 11: Hyperparameter tuning; intro to neural nets & deep learning

Nearly all slides by George H. Chen
with a few by Phillip Isola

Quiz 2



Median	Maximum	Mean	Std Dev
90.0	100.0	86.88	10.48

As I intended, Quiz 2 was easier than Quiz 1

Remember: letter grades are assigned based on a curve

Solutions are in Canvas -> Files -> "Quiz 2 solutions.pdf"

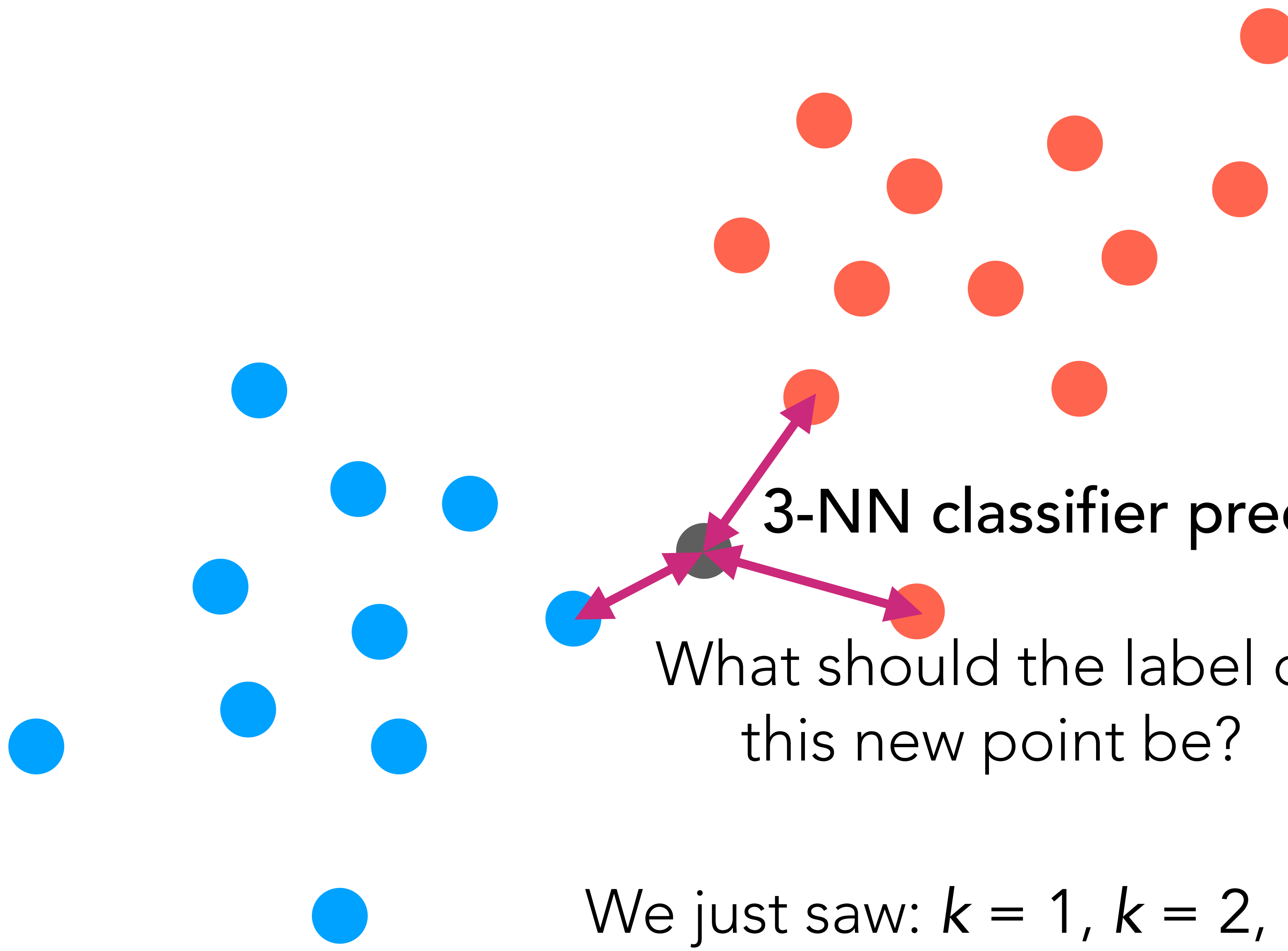
Regrade requests (use Gradescope's regrade request feature)
are due **Friday April 18, 11:59pm**
(for if you think there's a genuine grading error)

Reminder: if you get instructor-endorsed posts in Piazza, you could earn up to 20 bonus points on your Quiz 2!

We plan on shutting down the Piazza forum on
Monday April 28, 11:59pm

Please get your instructor endorsements by then!

(Flashback) Example: *k*-NN Classification



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

What happens if $k = n$?

(Flashback)

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?

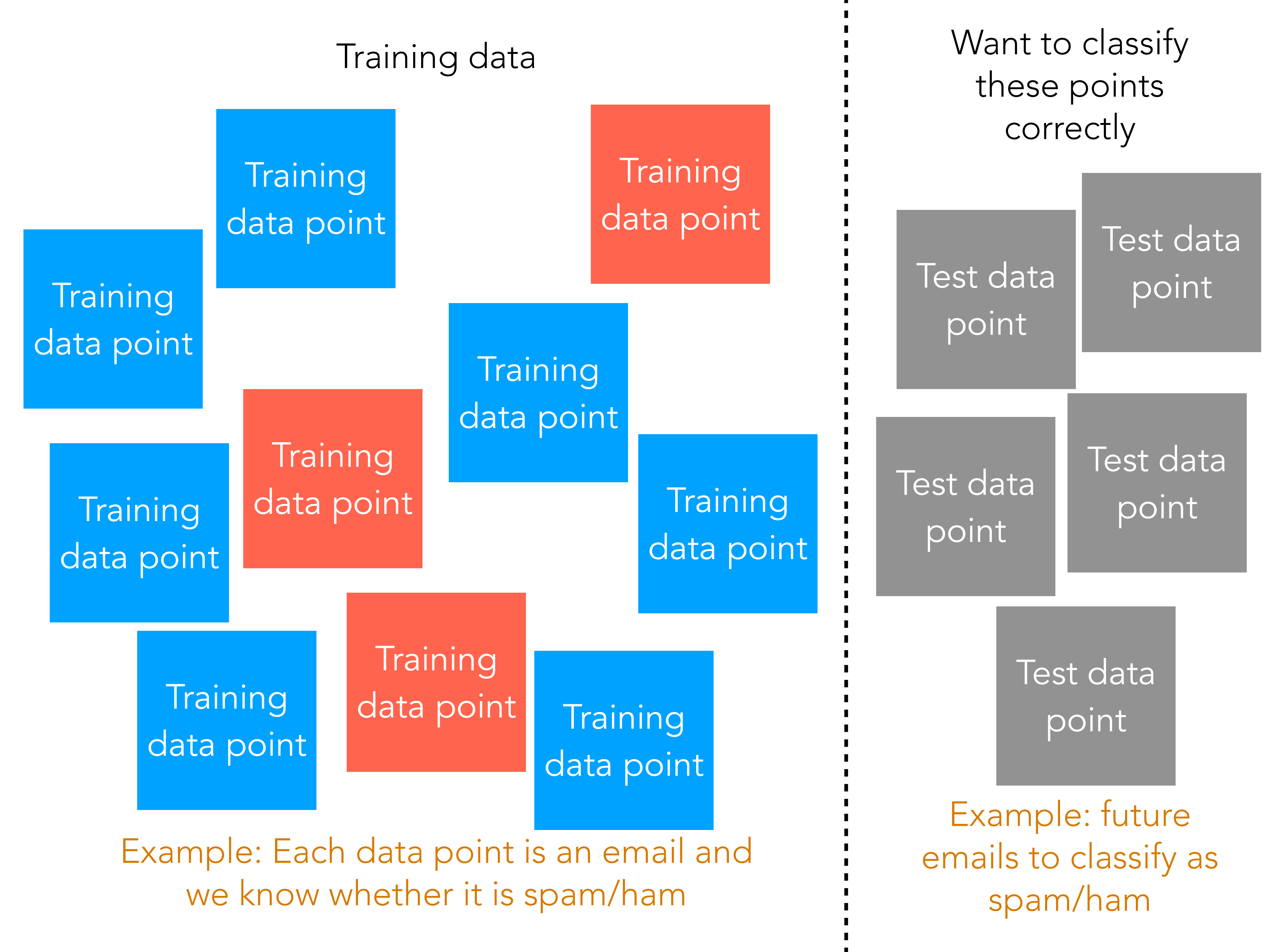
(Flashback) Hyperparameters vs. Parameters

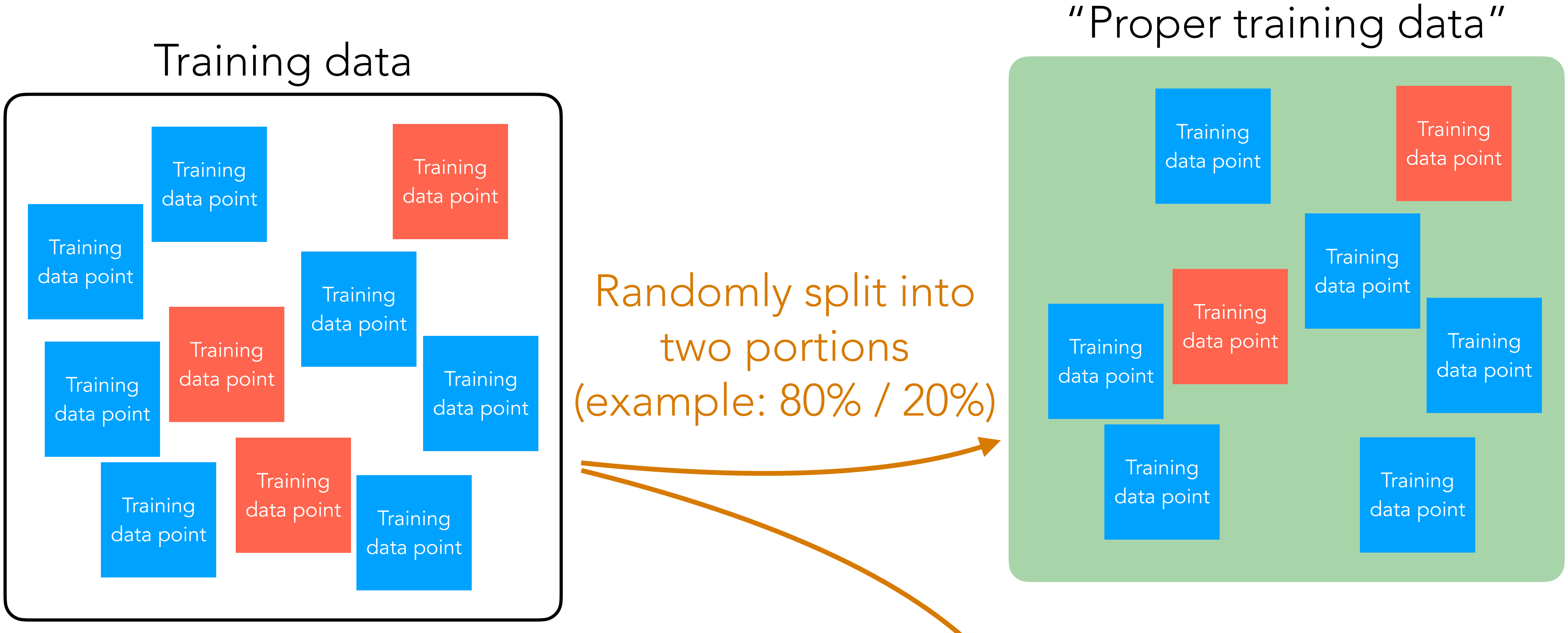
- We fit a model's parameters to training data (terminology: we "learn" the parameters)
- We pick values of hyperparameters and they do *not* automatically get fit to training data
- Example: Gaussian mixture model
  - Hyperparameter: number of clusters *k*
  - Parameters: cluster probabilities, means,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

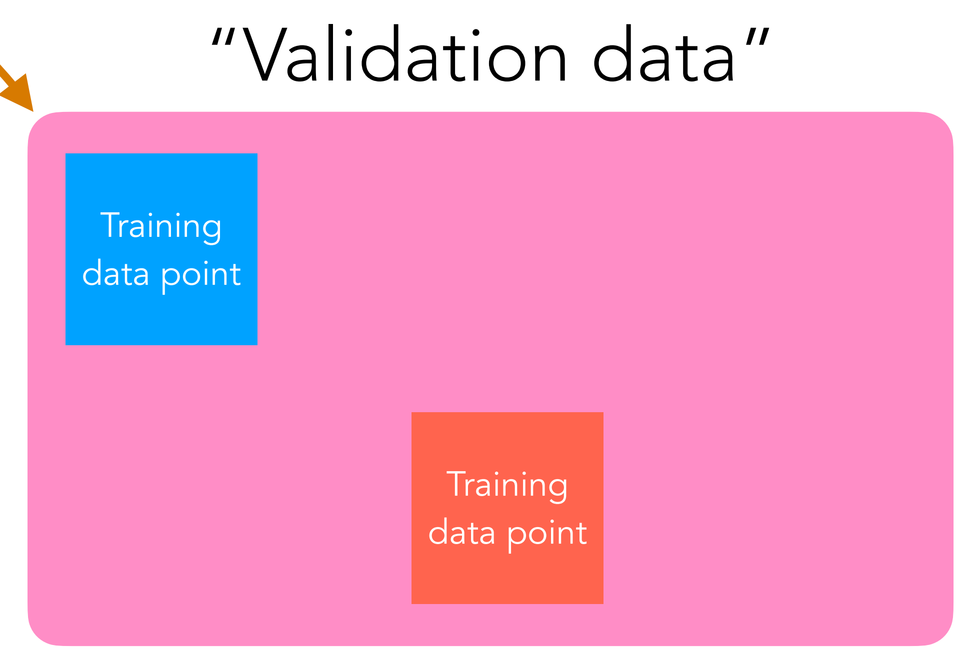
Actually, there's another hyperparameter: distance function to use
(for simplicity, we assume Euclidean distance for now)

⚠️ 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





For k = 1, 2, …, 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

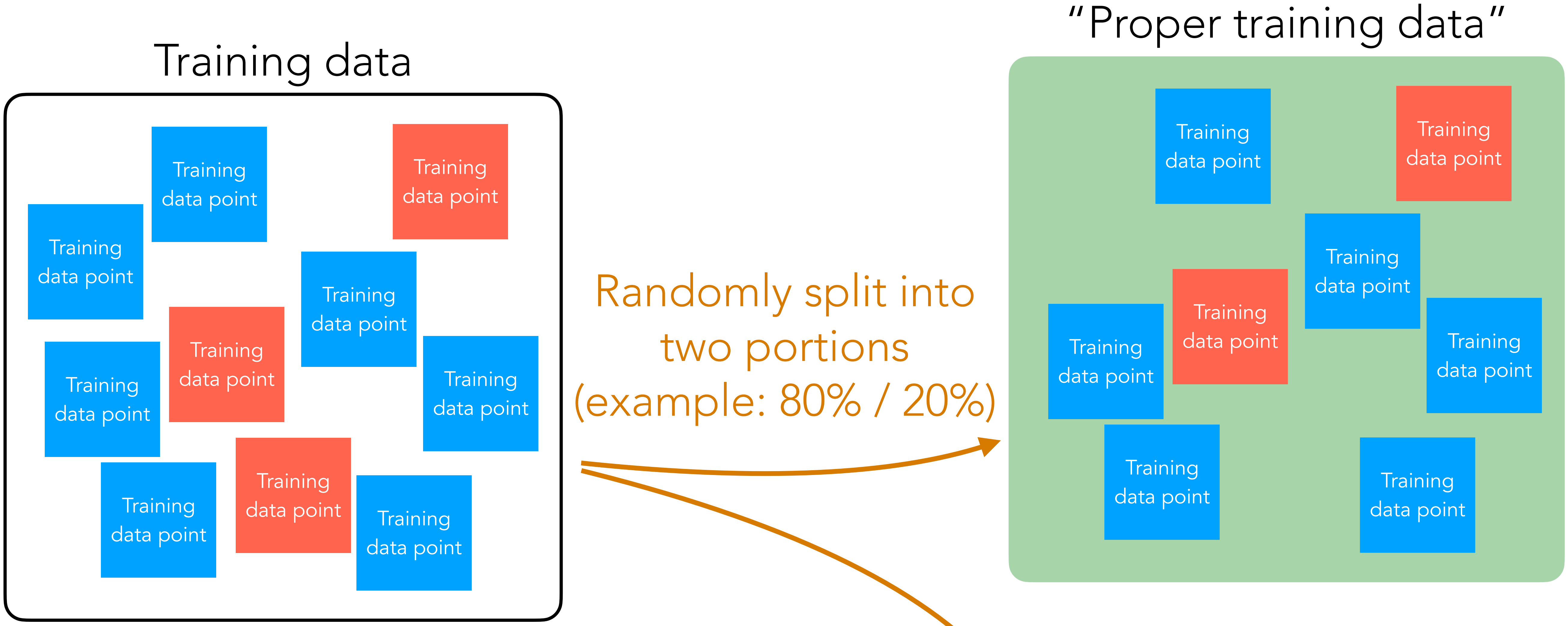
There are many score functions possible

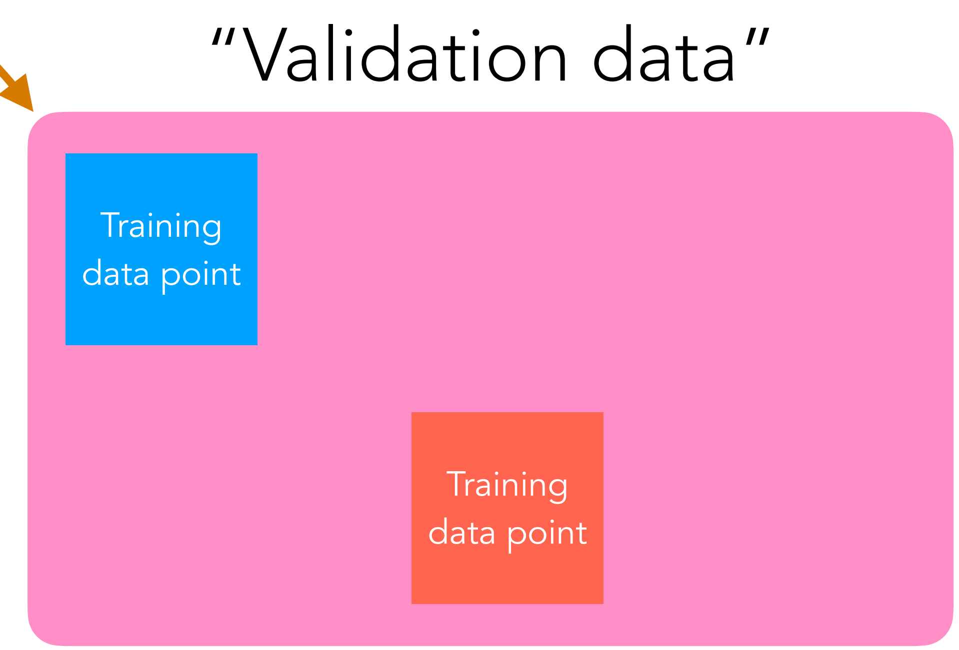
Examples: raw accuracy, true positive rate/recall, false positive rate, precision

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 <u>not</u> 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 <u>full training data</u> or the <u>full training data minus the validation data</u>
  - In 94-775, to avoid confusion, we use the somewhat non-standard terminology "proper training set"/"proper training data" to refer to the the <u>full training data minus the validation data</u>





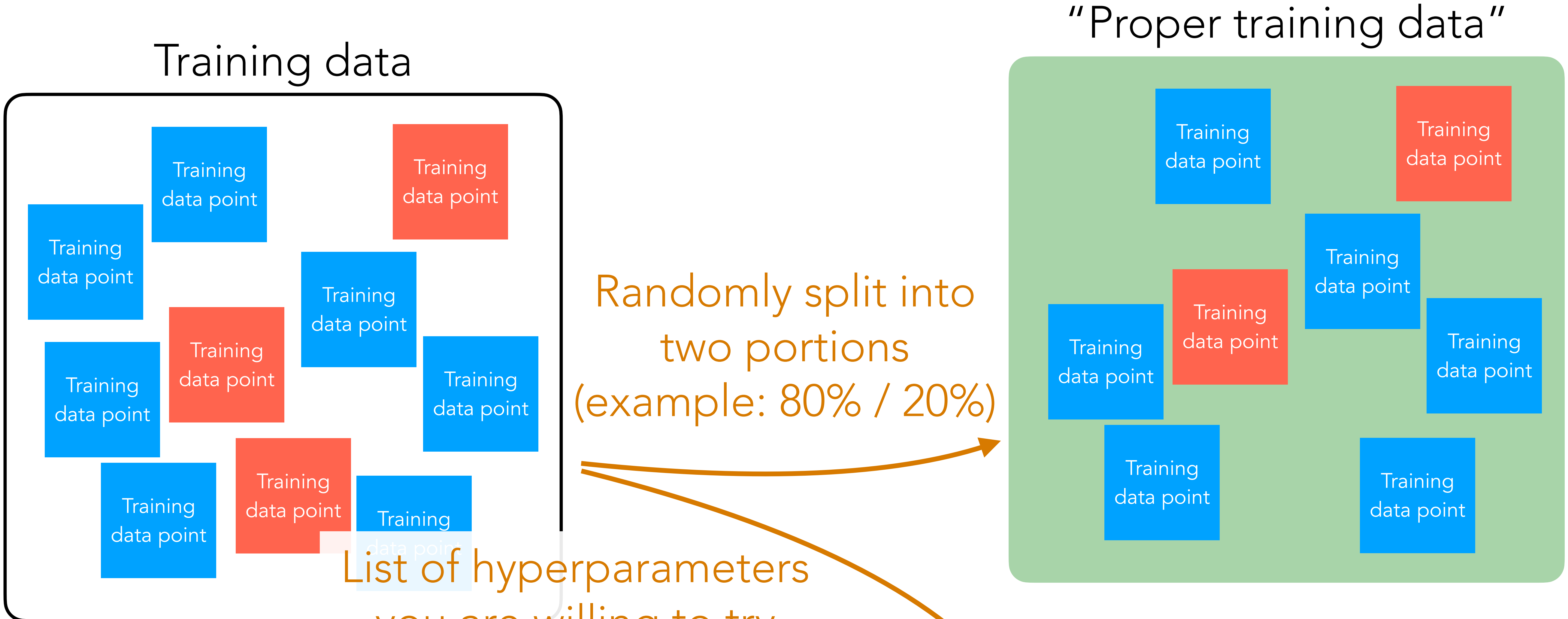
For *k* = 1, 2, …, some user-specified max:

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

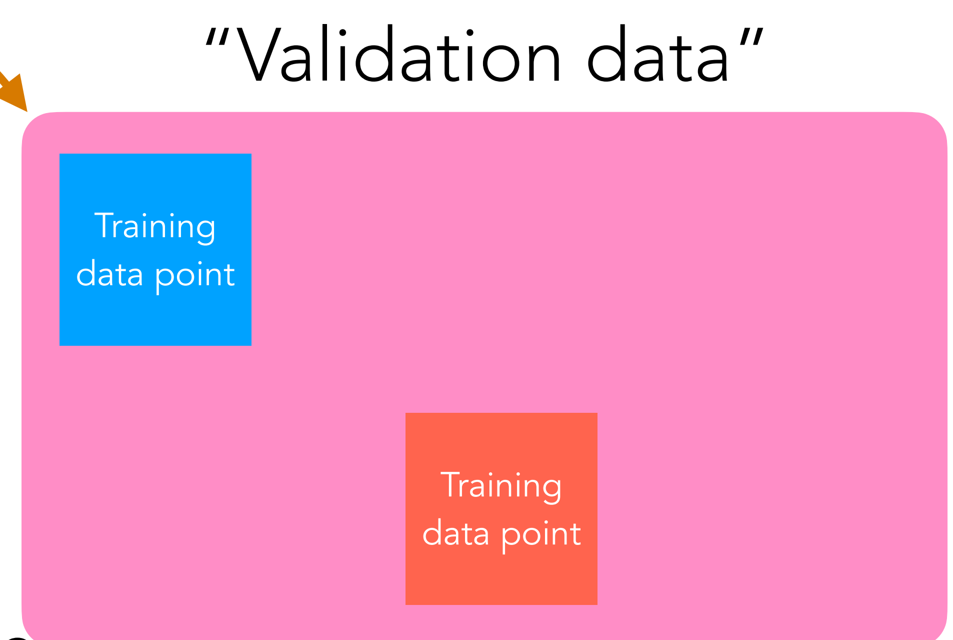
Use whichever value of *k* achieves the best score

There are many score functions possible

Examples: raw accuracy, true positive rate/recall, false positive rate, precision



List of hyperparameters you are willing to try



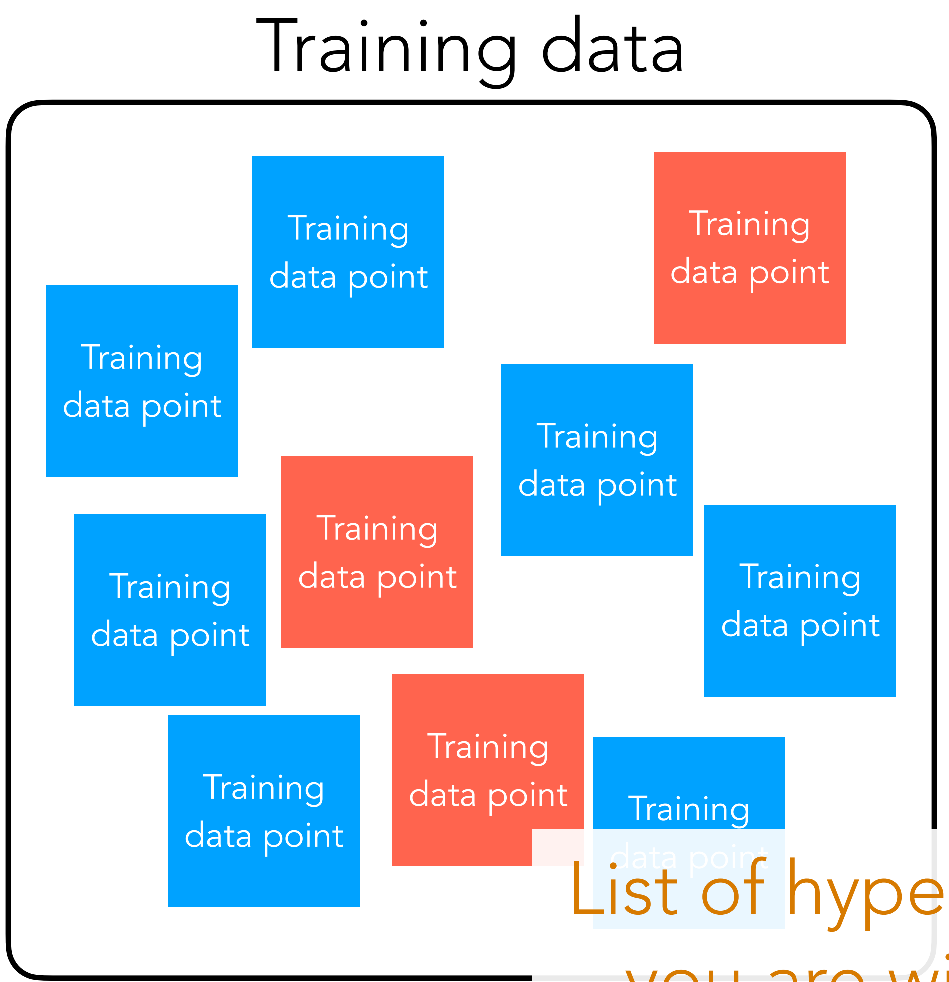
For *k*, ρ = (1, "Euclidean"), (1, "Cosine"), …:

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

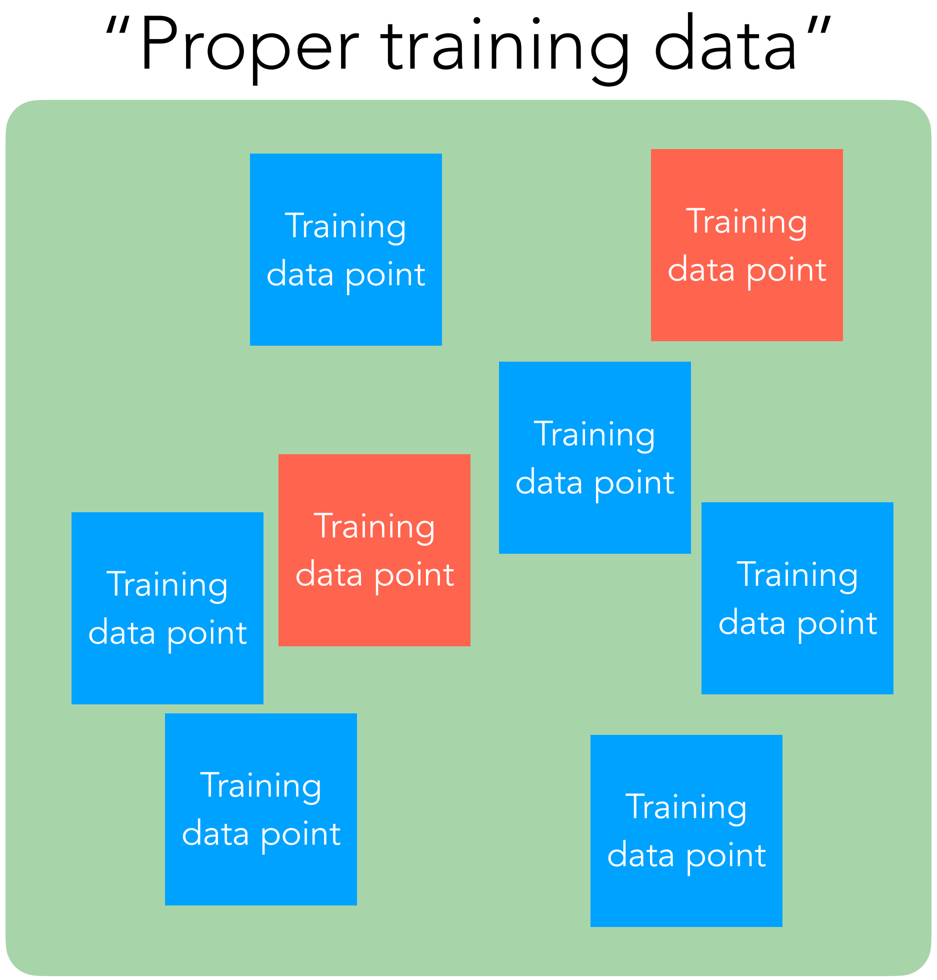
Use whichever value of *k*, ρ achieves the best score

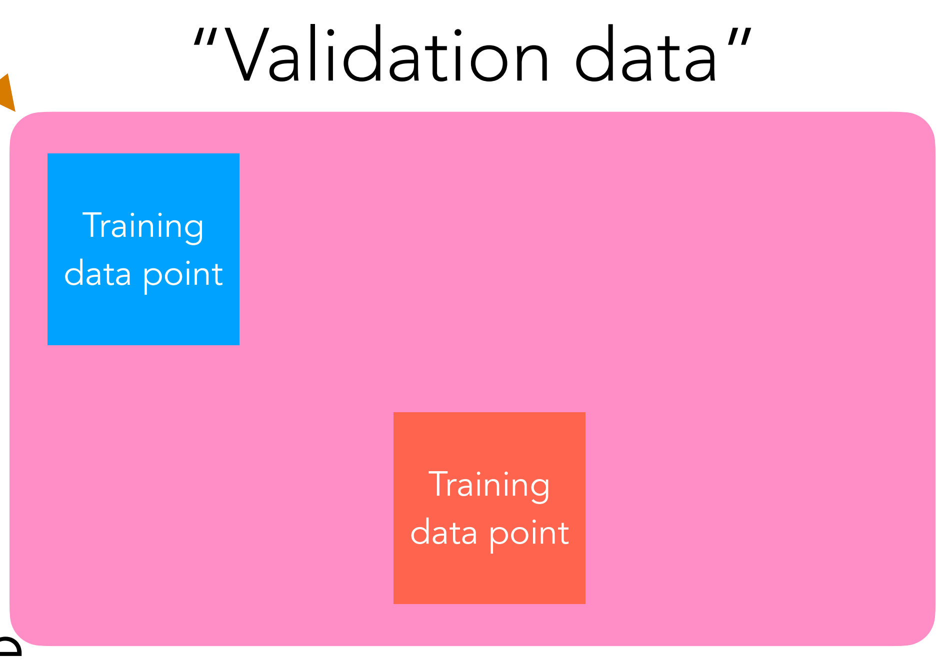
There are many score functions possible

Examples: raw accuracy, true positive rate/recall, false positive rate, precision



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





List of hyperparameters you are willing to try

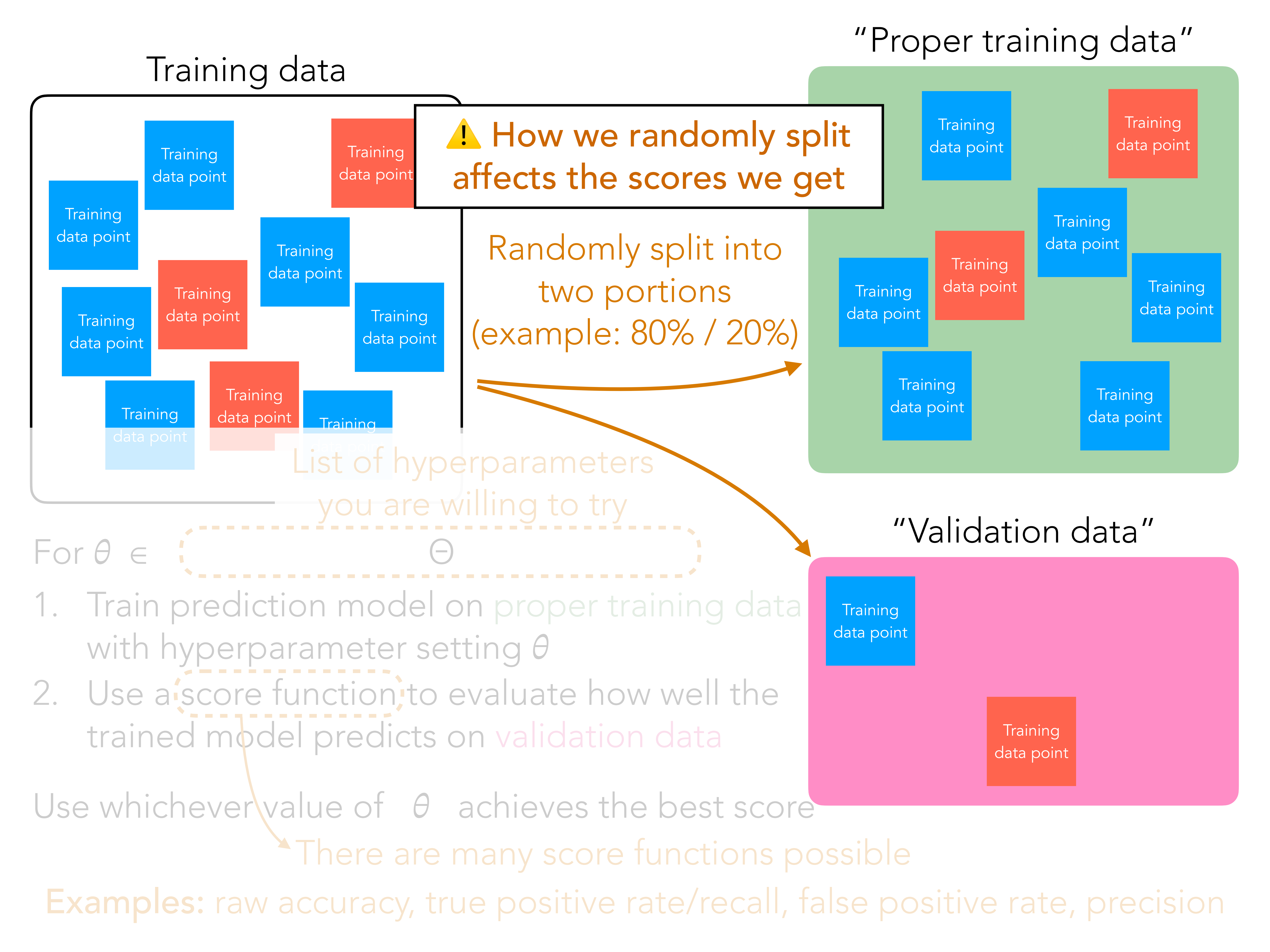
For $\theta \in \Theta$

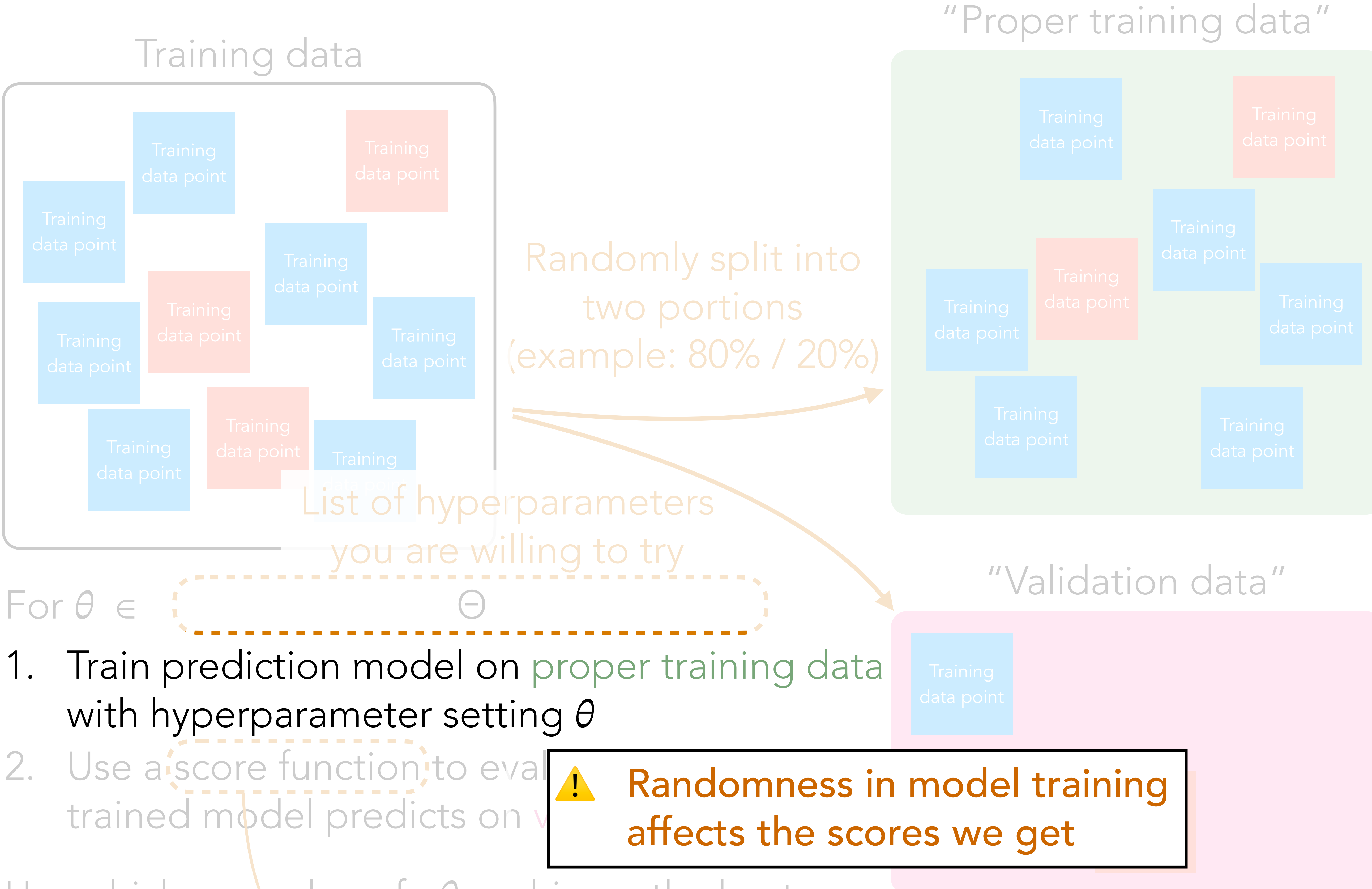
1. Train prediction model on proper training data with hyperparameter setting θ
2. Use a score function to evaluate how well the trained model predicts on validation data

Use whichever value of θ achieves the best score

There are many score functions possible

Examples: raw accuracy, true positive rate/recall, false positive rate, precision


Training data
"Proper training data"
⚠️ How we randomly split affects the scores we get
Randomly split into two portions (example: 80% / 20%)
"Validation data"
Training data point
List of hyperparameters you are willing to try
For θ ∈ 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
Examples: raw accuracy, true positive rate/recall, false positive rate, precision



List of hyperparameters you are willing to try

For $\theta \in \Theta$

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

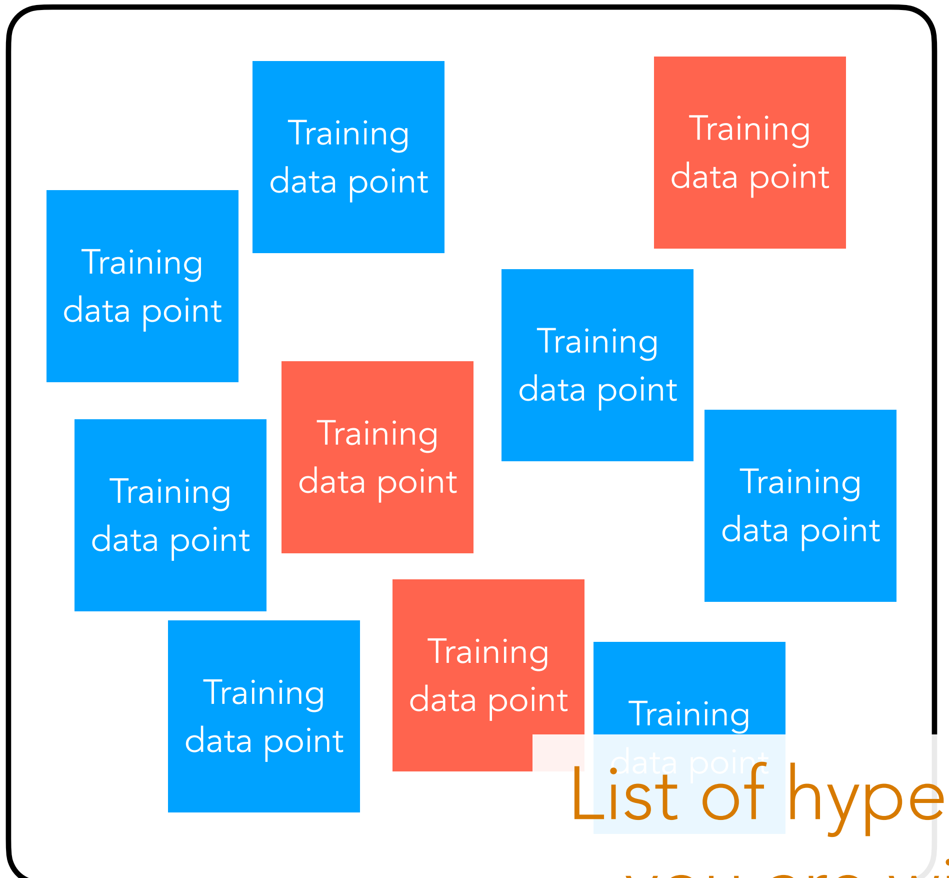
⚠ Randomness in model training affects the scores we get

Use whichever value of θ achieves the best score

There are many score functions possible

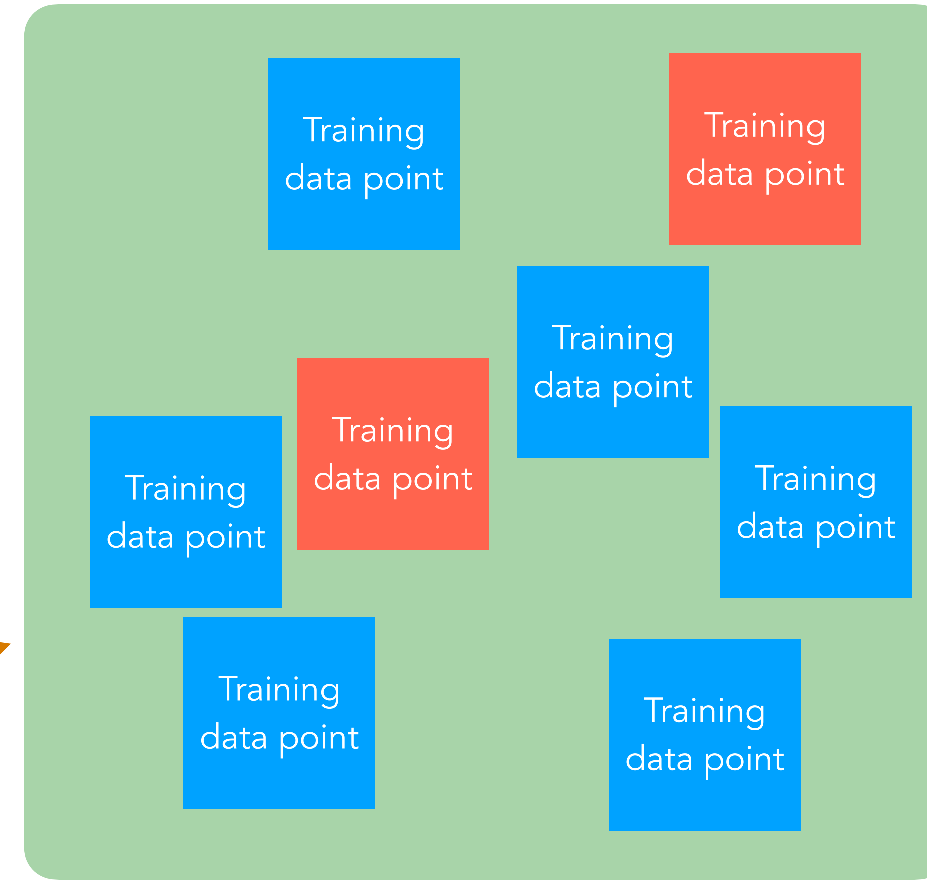
Examples: raw accuracy, true positive rate/recall, false positive rate, precision

Training data



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

“Proper training data”



“Validation data”

List of hyperparameters you are willing to try

For $\theta \in$ 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

Examples: raw accuracy, true positive rate/recall, false positive rate, precision

The rest of the prediction models we consider 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 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

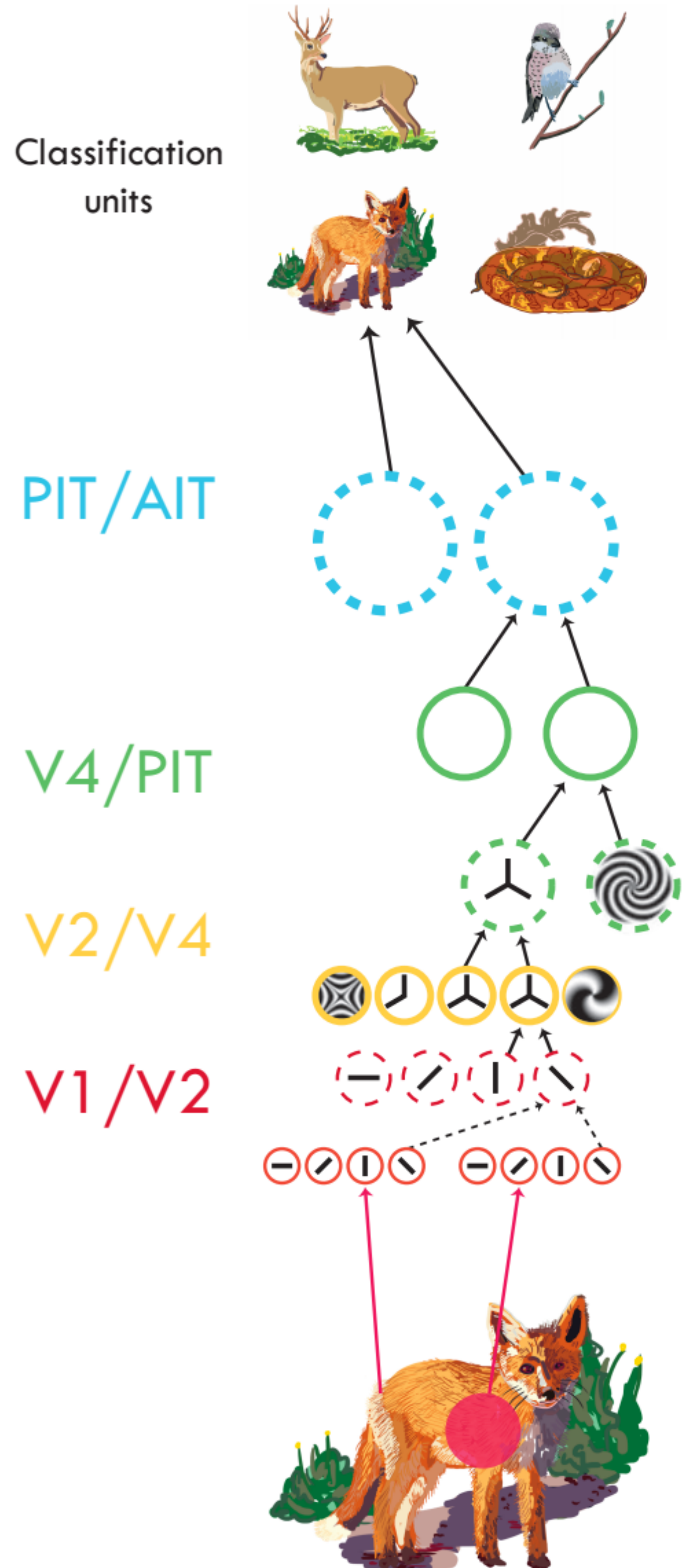
Extremely useful in practice:

- 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 are neural nets & what does "deep learning" refer to?



Slide by Phillip Isola

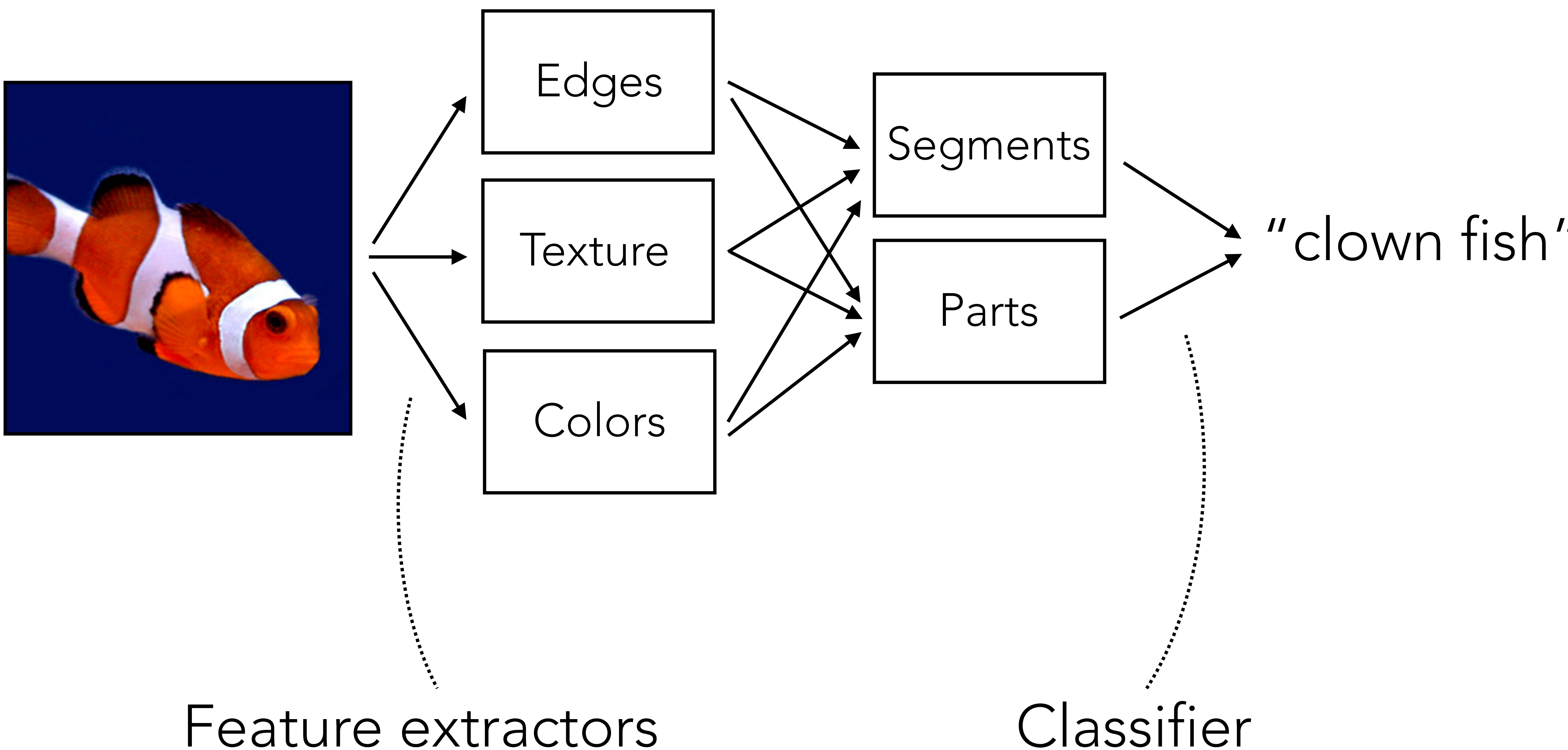
Serre, 2014

Basic Idea

→ Brain/Machine → “clown fish”

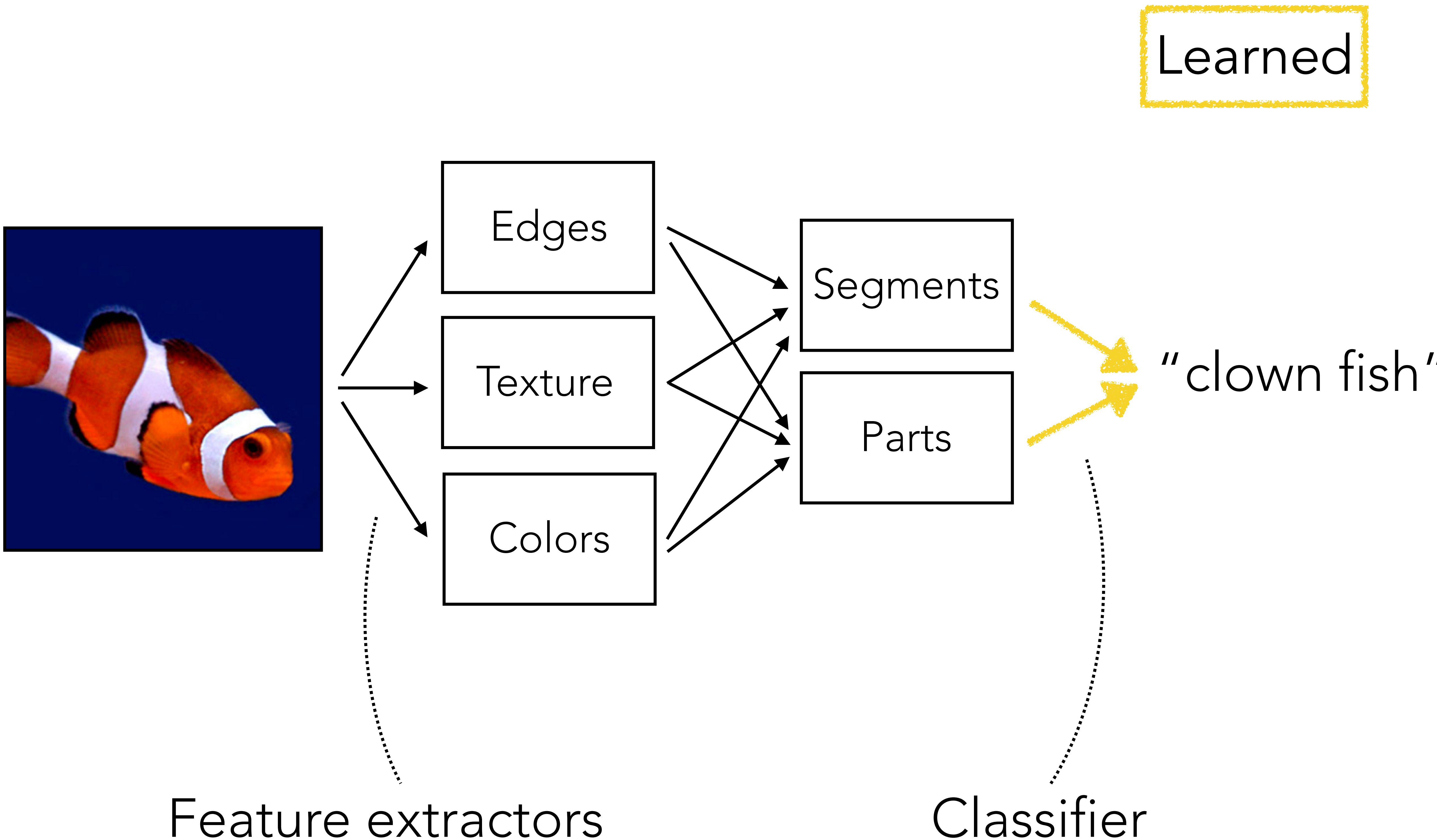
Slide by Phillip Isola

Classical Approach



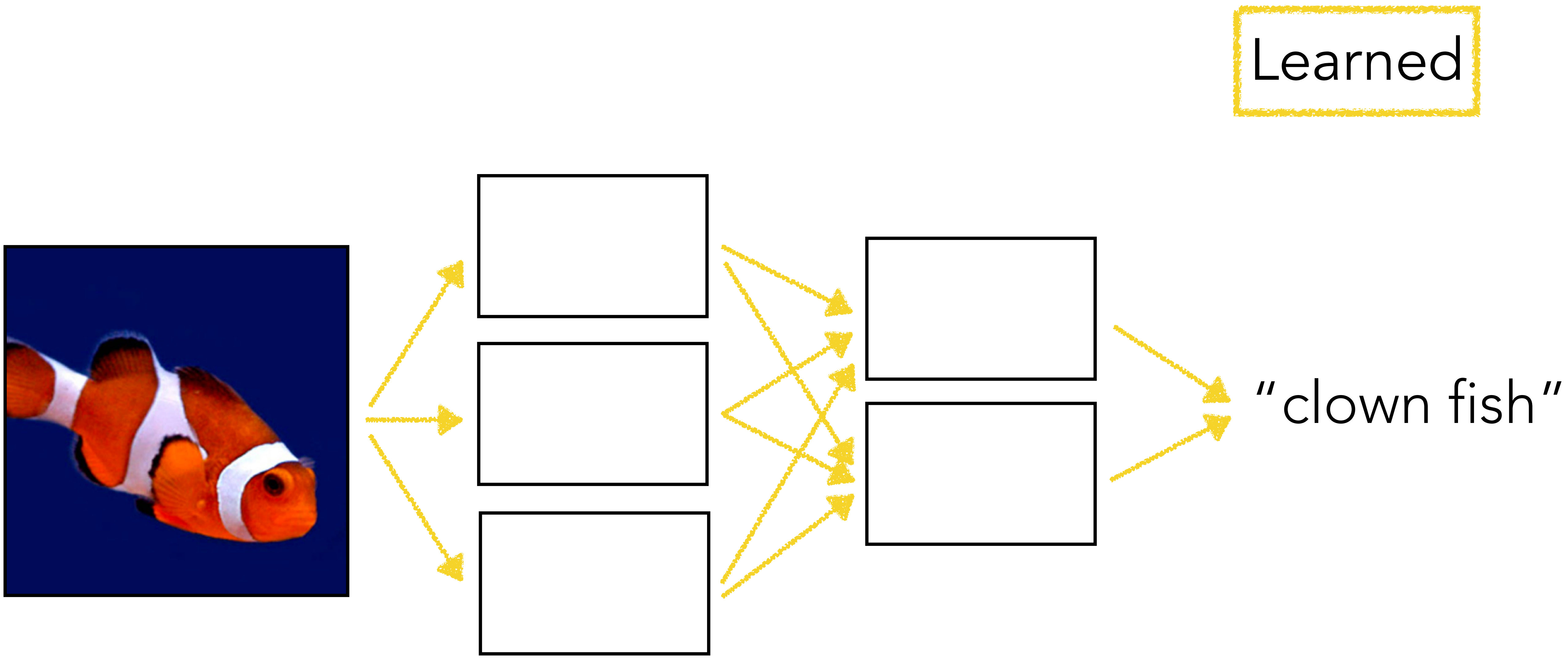
Slide by Phillip Isola

Classical Approach



Slide by Phillip Isola

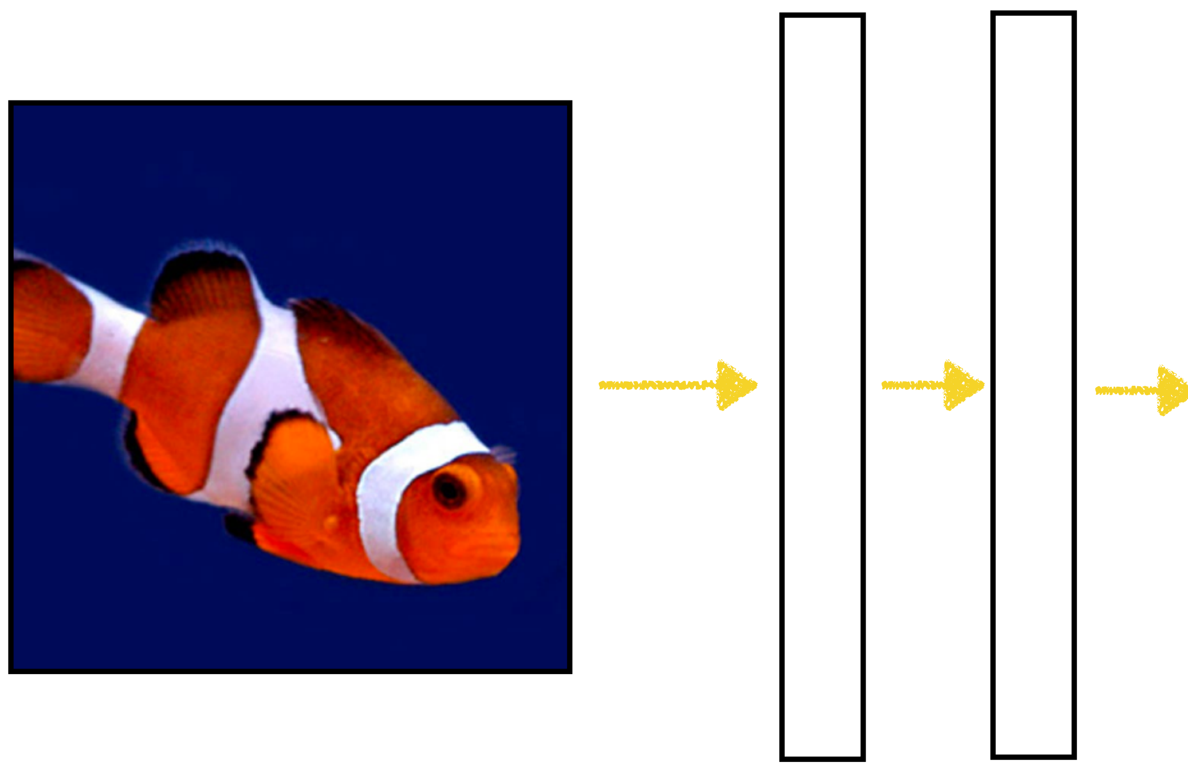
Neural Network



Slide by Phillip Isola

Neural Network

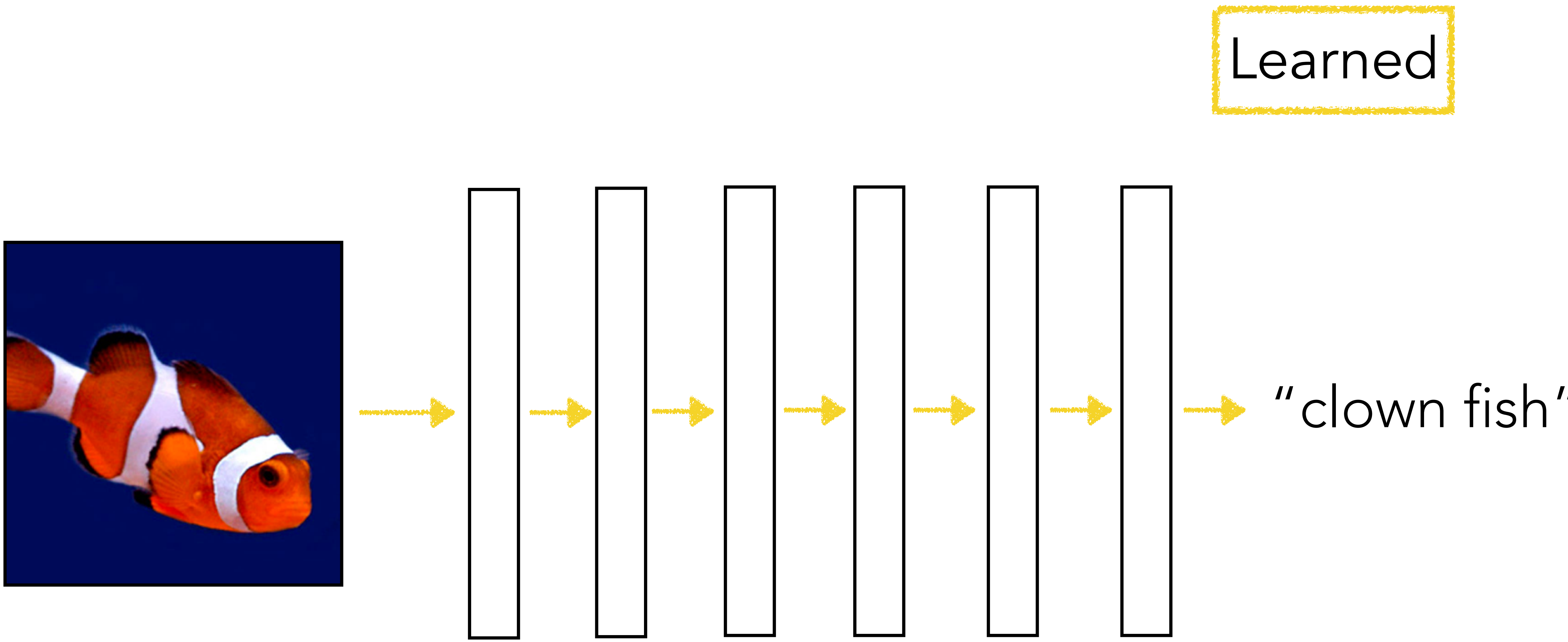
Learned



“clown fish”

Slide by Phillip Isola

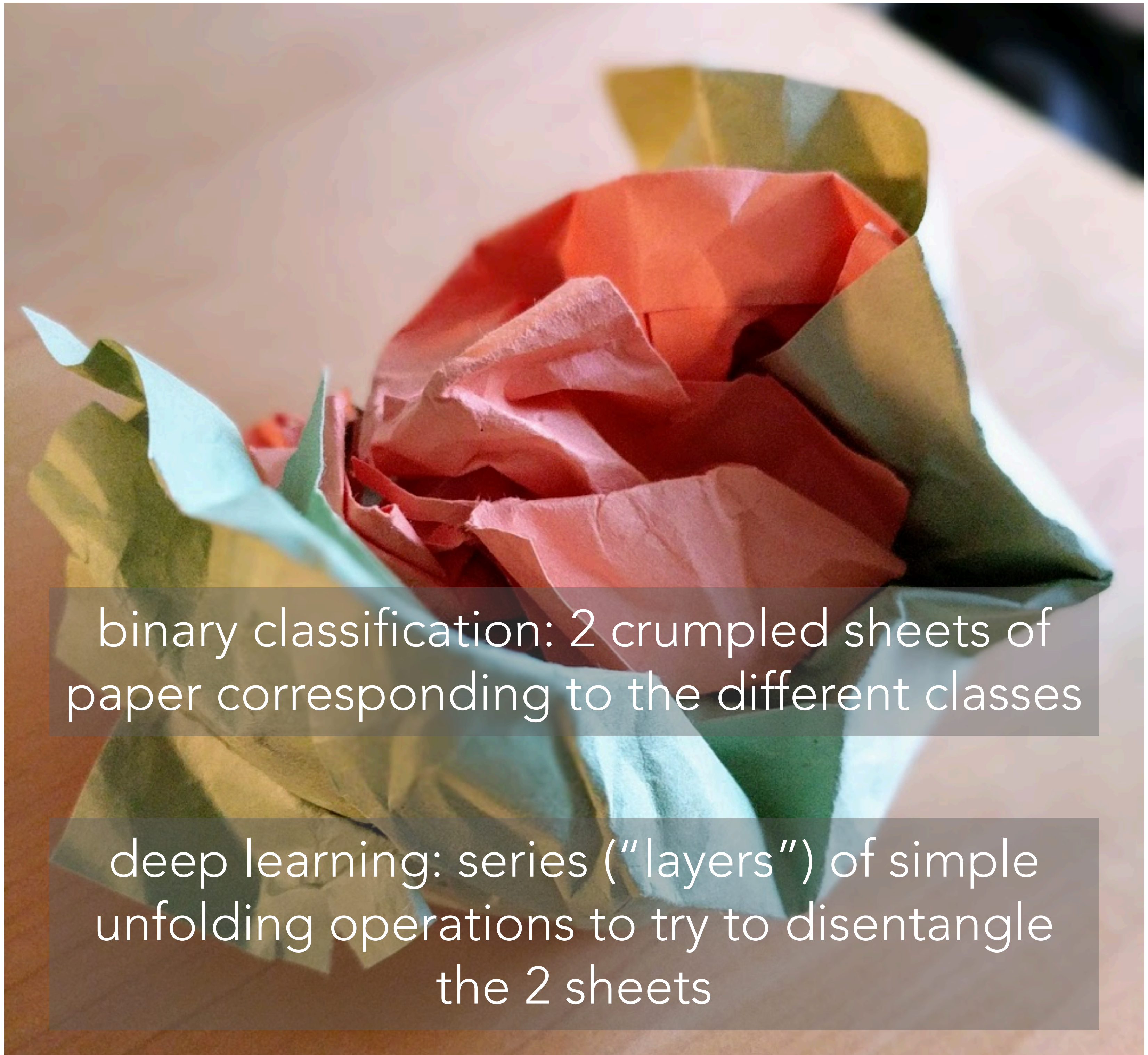
Deep Neural Network



Deep learning just refers to learning deep neural nets

Slide by Phillip Isola

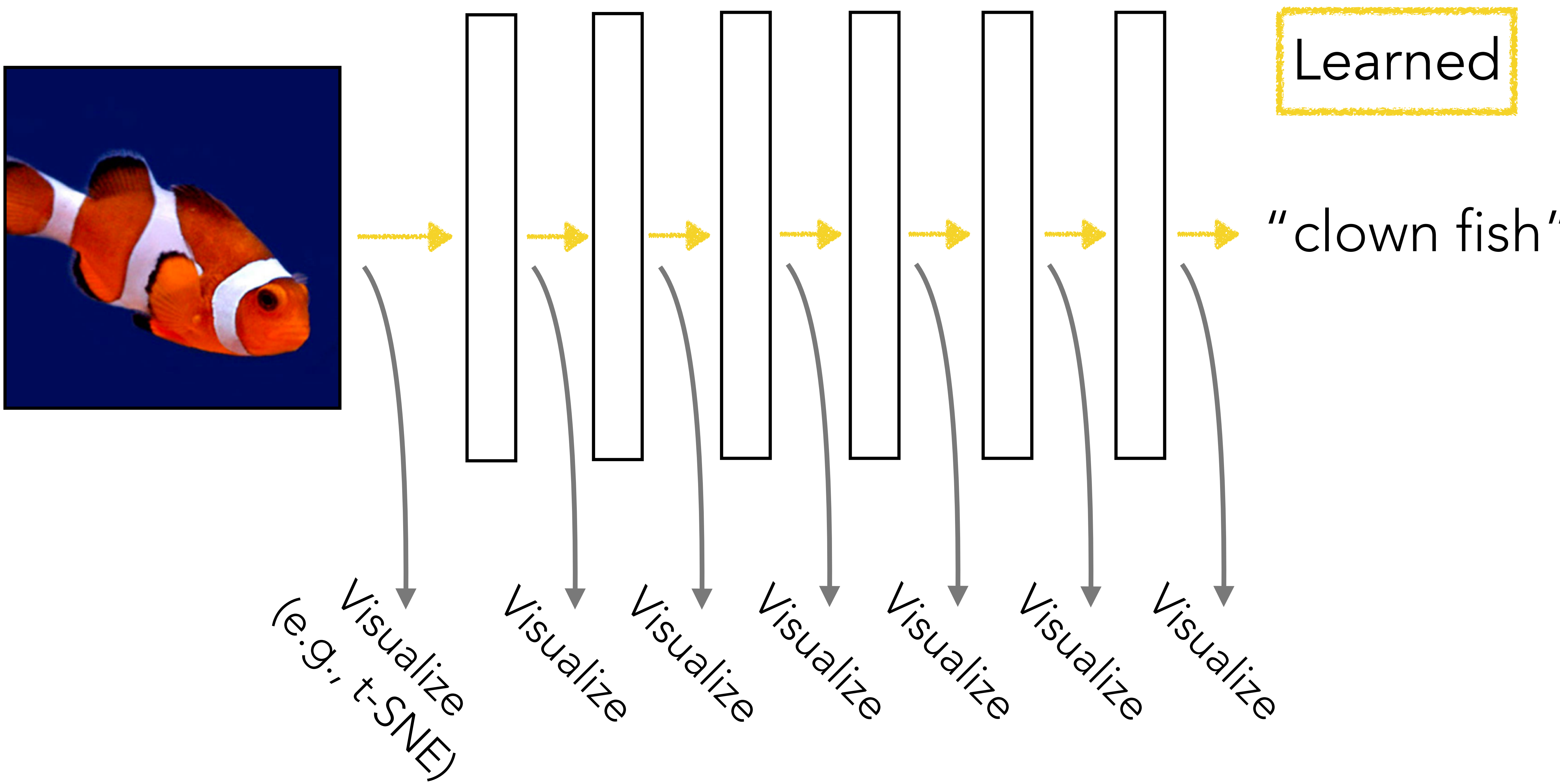
Crumpled Paper Analogy



Analogy: Francois Chollet, photo: George Chen

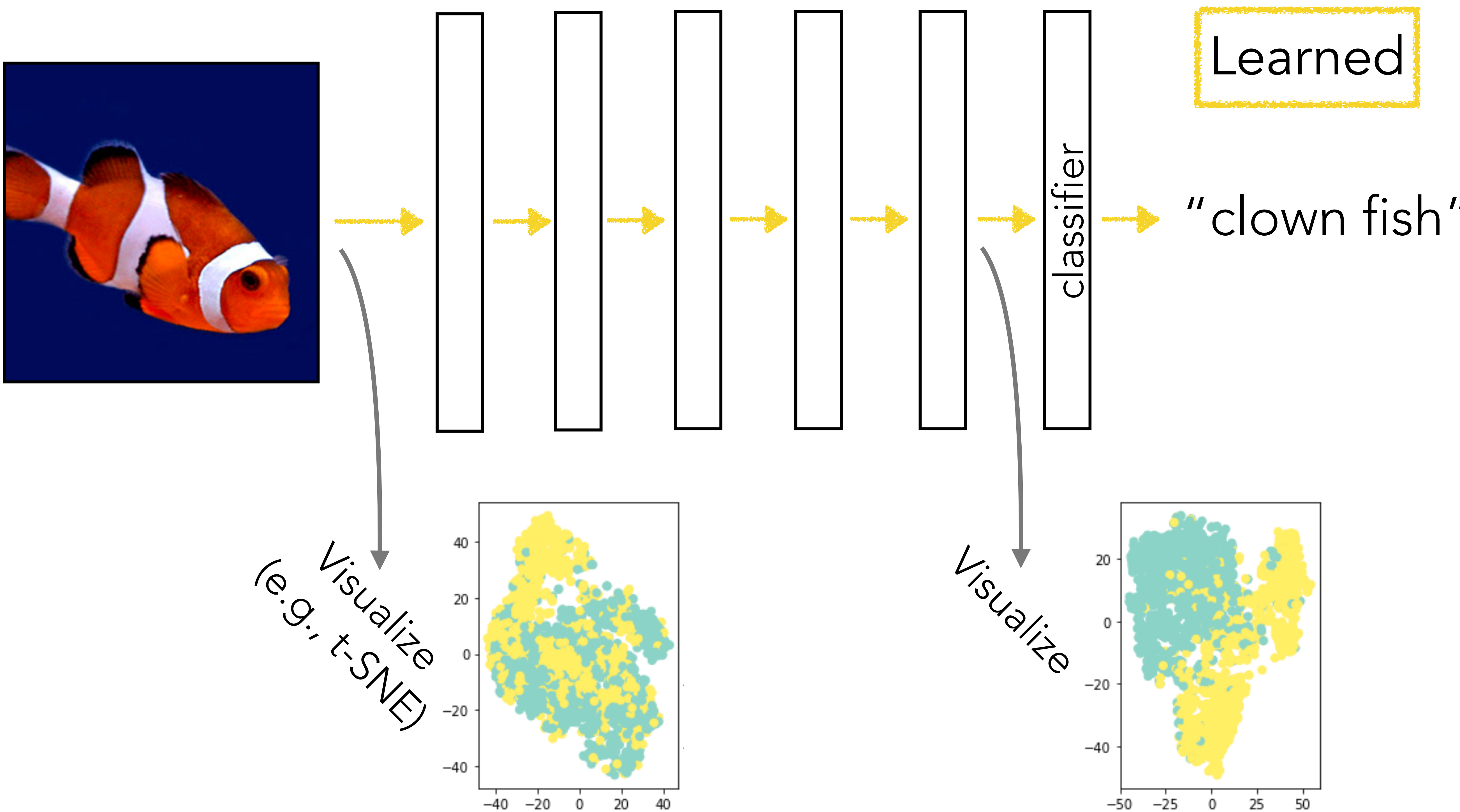
Representation Learning

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)

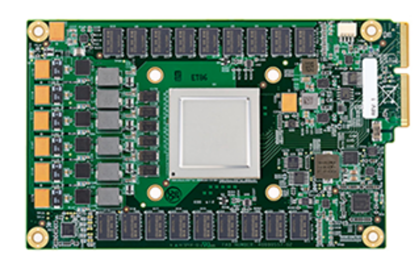
There's even a patent from 1961 that basically amounts to a convolutional neural net for OCR

- Big data

- Better hardware

CPU's
& Moore's law

GPU's



TPU's

Many companies now make dedicated hardware for deep nets (e.g., Google, Apple, Tesla)

- Better algorithms

Structure Present in Data Matters

Neural nets aren't doing black magic

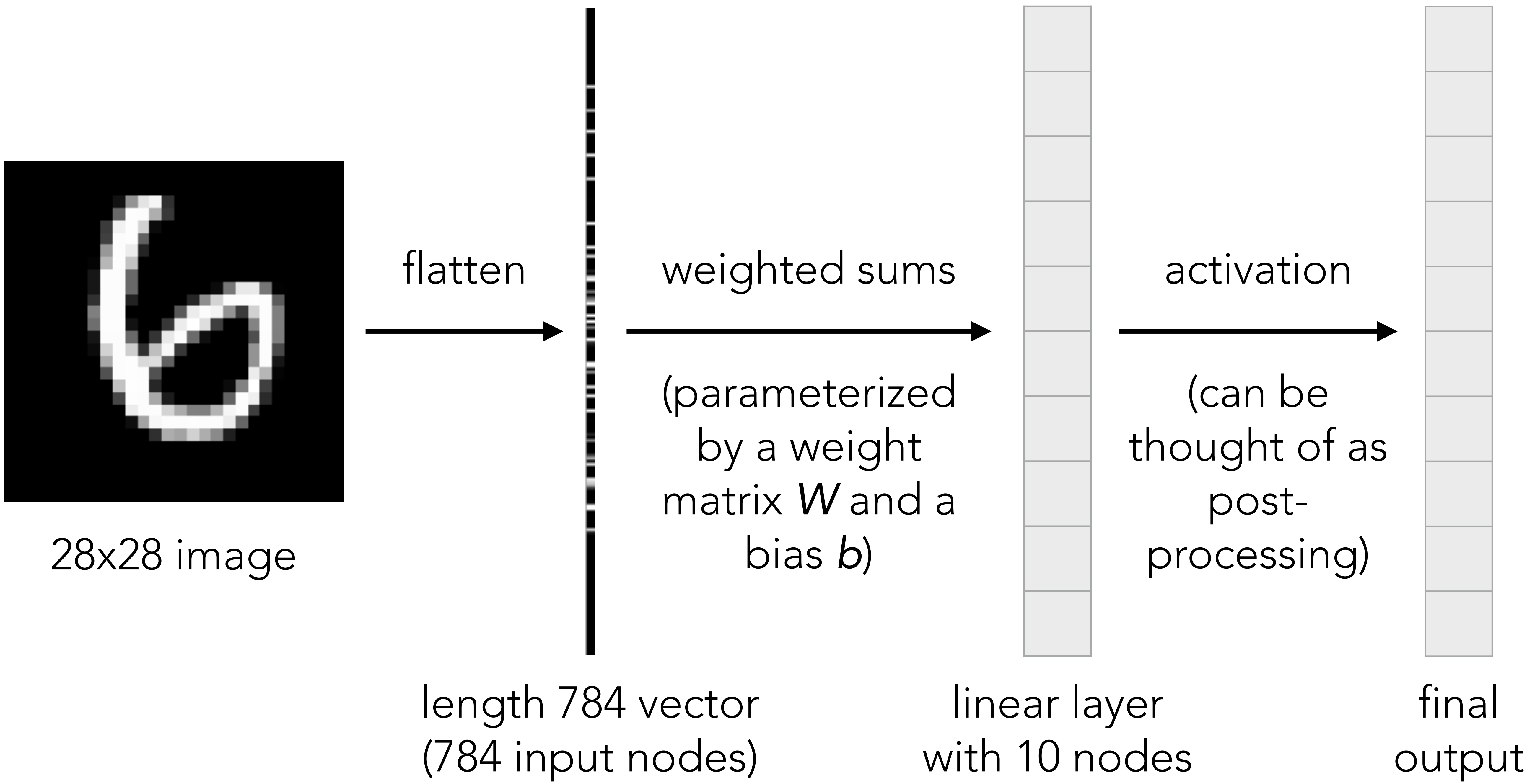
- **Image analysis:** convolutional neural networks (convnets) neatly <u>incorporates basic image processing structure</u>

- **Time series analysis:** transformers <u>learn how to weight previous time steps' contributions to a prediction at the current time step</u>
  - Note: text is a time series of tokens
  - Note: video is a time series of images

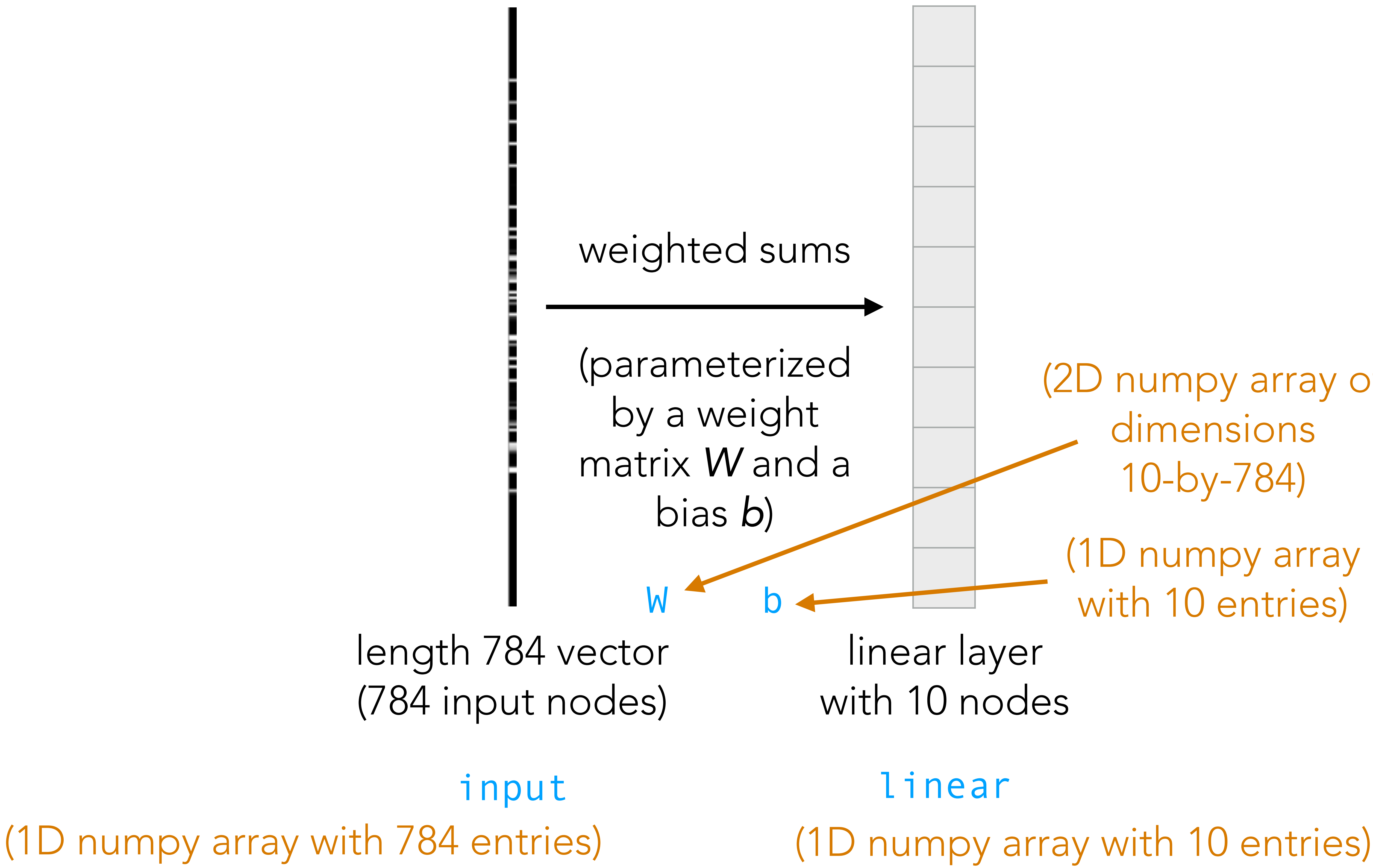
Handwritten Digit Recognition Example

Walkthrough of 2 extremely simple neural nets

Handwritten Digit Recognition



Handwritten Digit Recognition

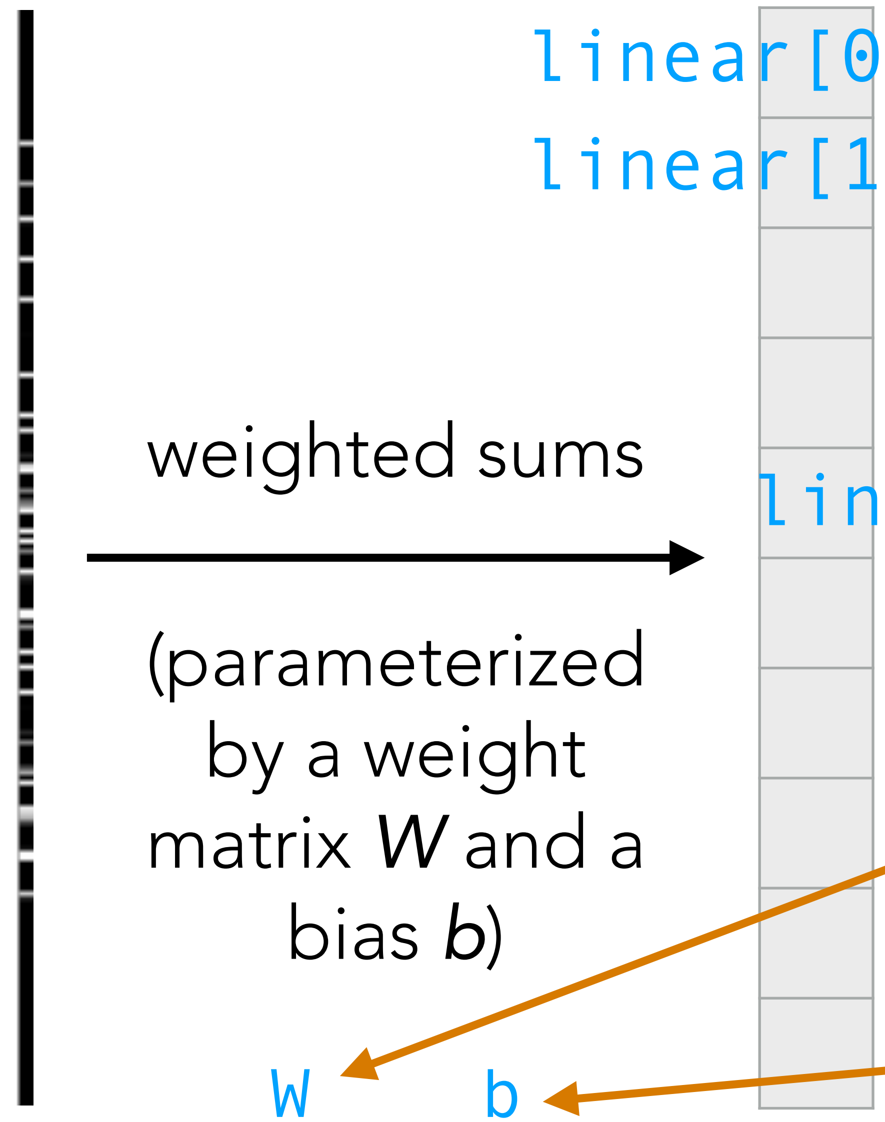


Handwritten Digit Recognition

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

$$\vdots$$

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



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

(1D numpy array with 10 entries)

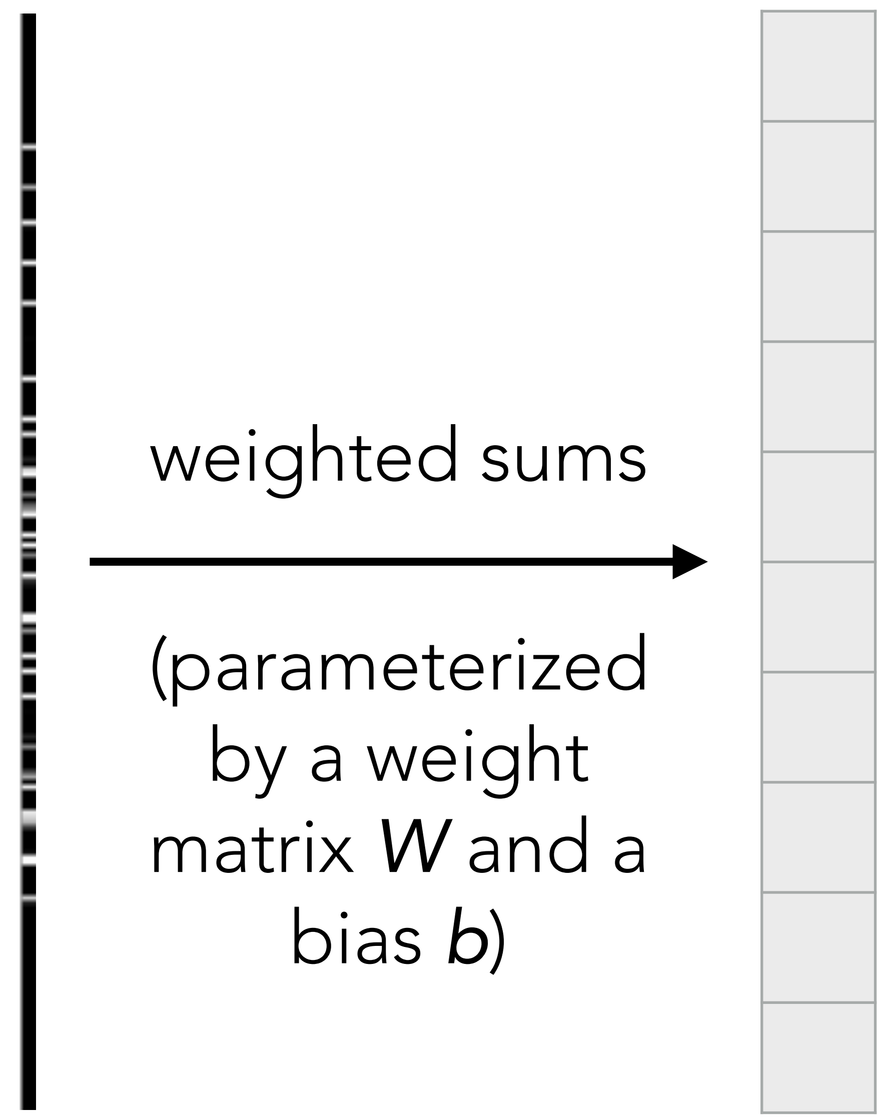
784 vector
put nodes)

linear layer
with 10 nodes

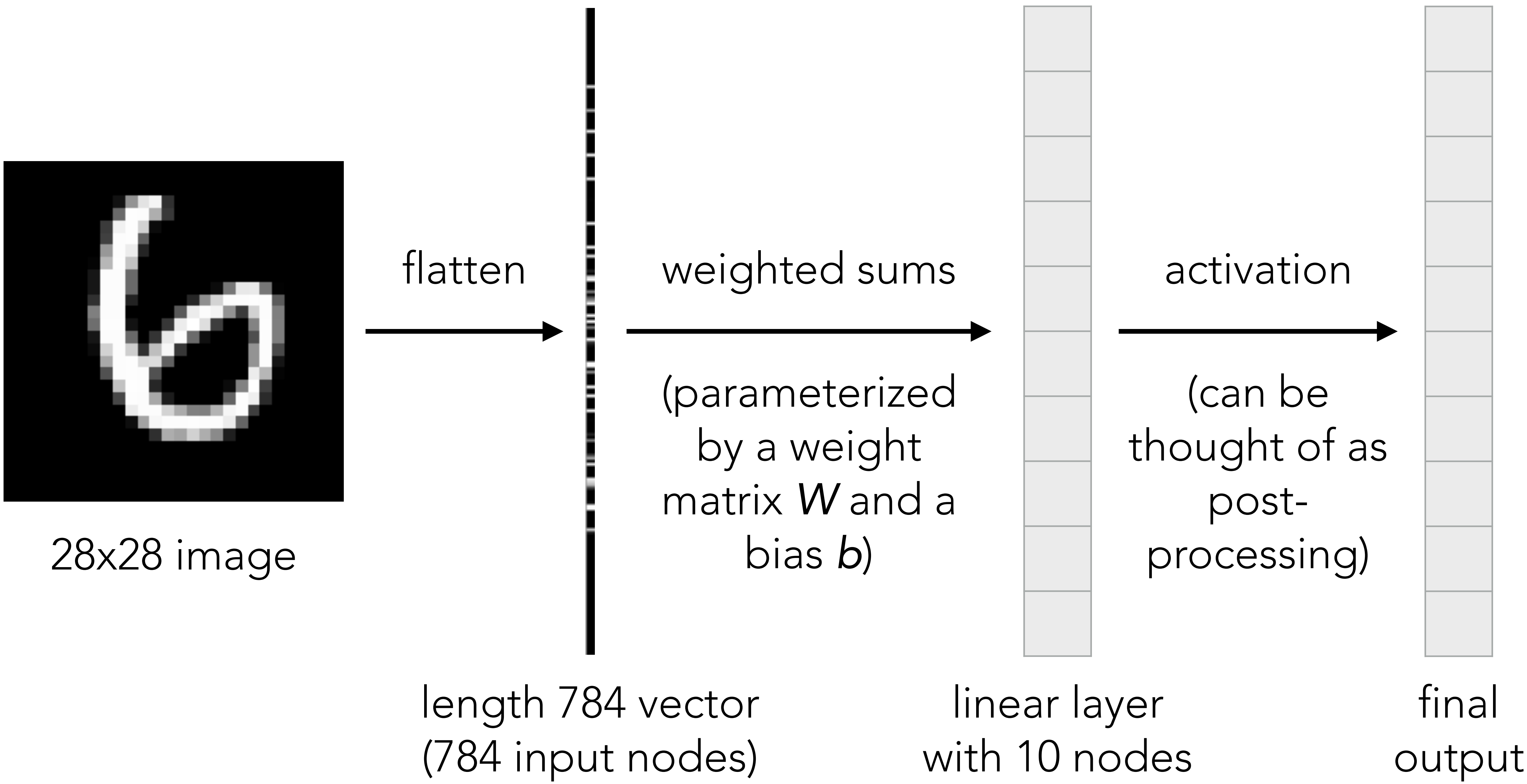
`nput`
entries)

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

Handwritten Digit Recognition



Handwritten Digit Recognition

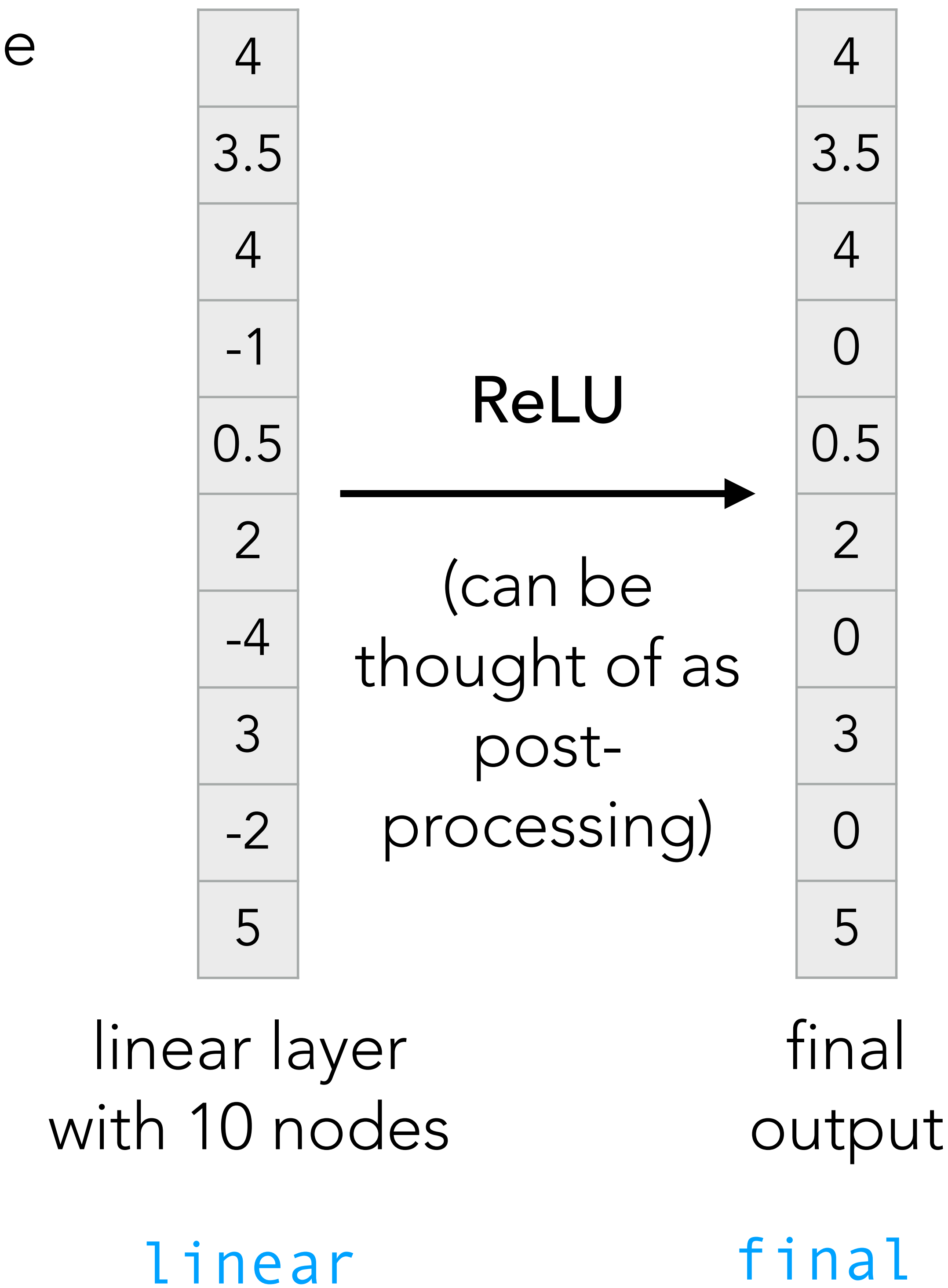


Handwritten Digit Recognition

Many different activation functions possible

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

```
final = np.maximum(0, linear)
```



Handwritten Digit Recognition

Many different activation functions possible

Example: **softmax** converts a table of numbers into a probability distribution

```
exp = np.exp(linear)
final = exp / exp.sum()
```

linear layer with 10 nodes	final output
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

softmax →

(can be thought of as post-processing)

`linear` `final`

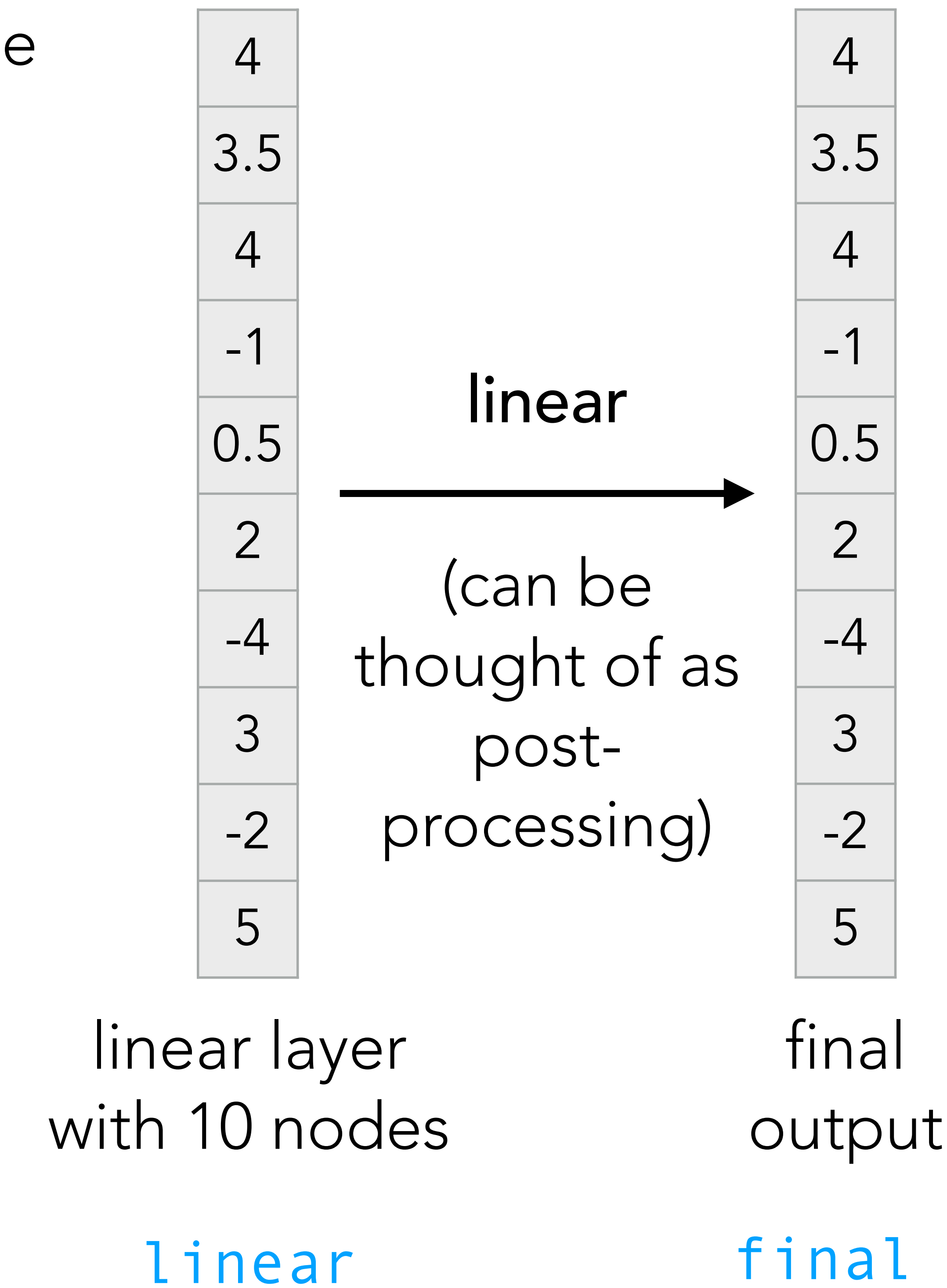
Handwritten Digit Recognition

Many different activation functions possible

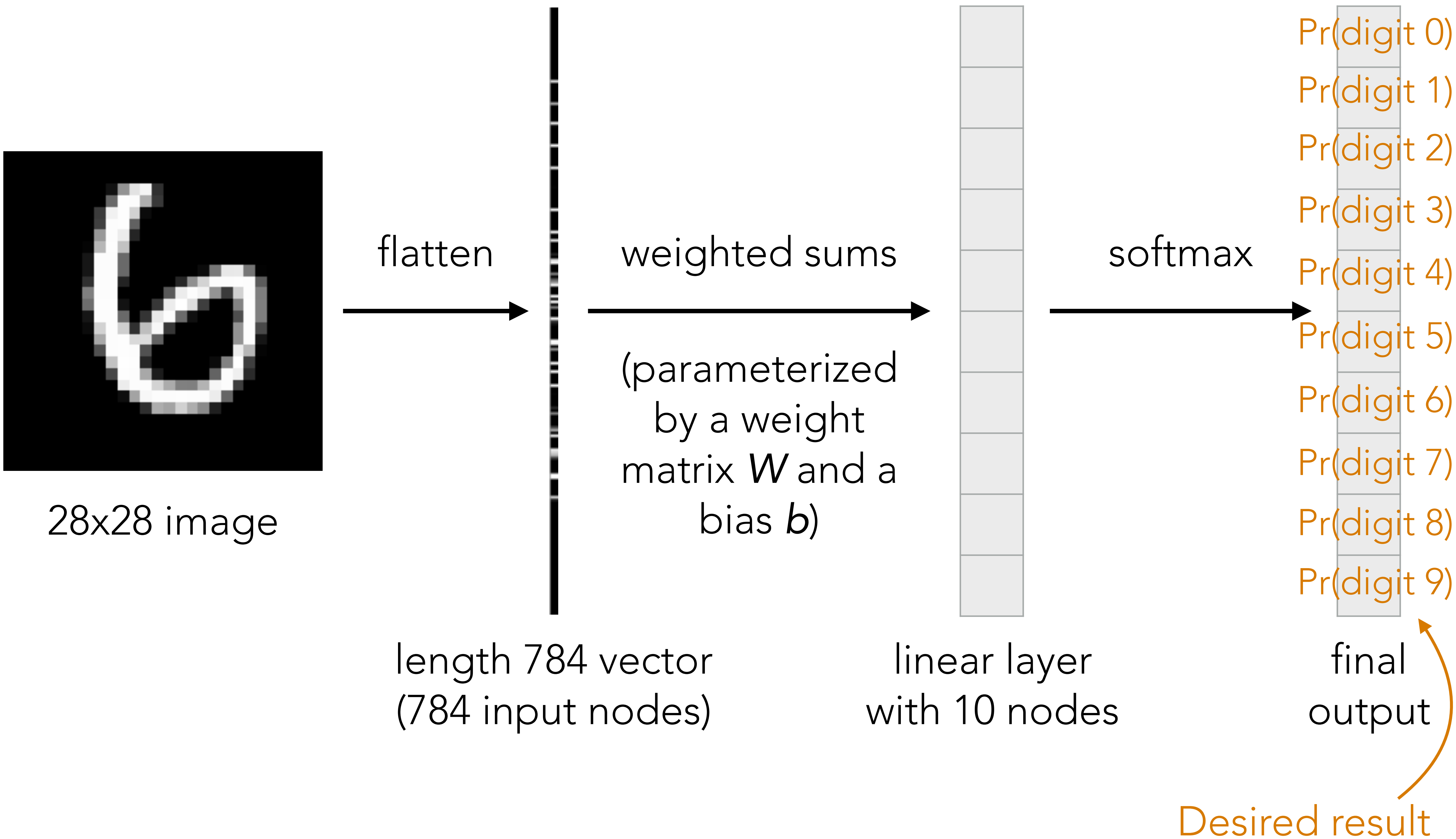
Example: **linear** activation does nothing

This is equivalent to there being
no activation function

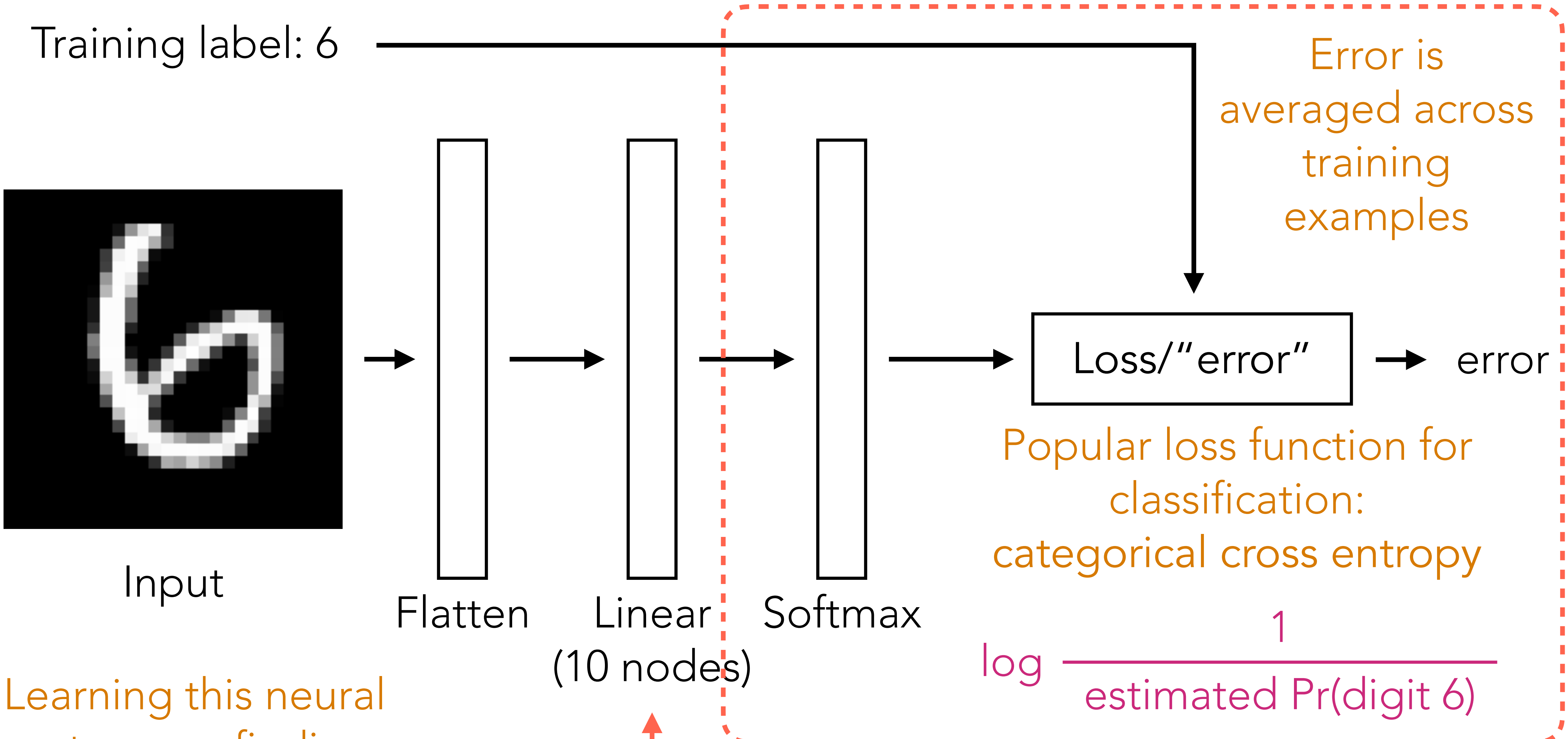
```
final = linear
```



Handwritten Digit Recognition



Handwritten Digit Recognition

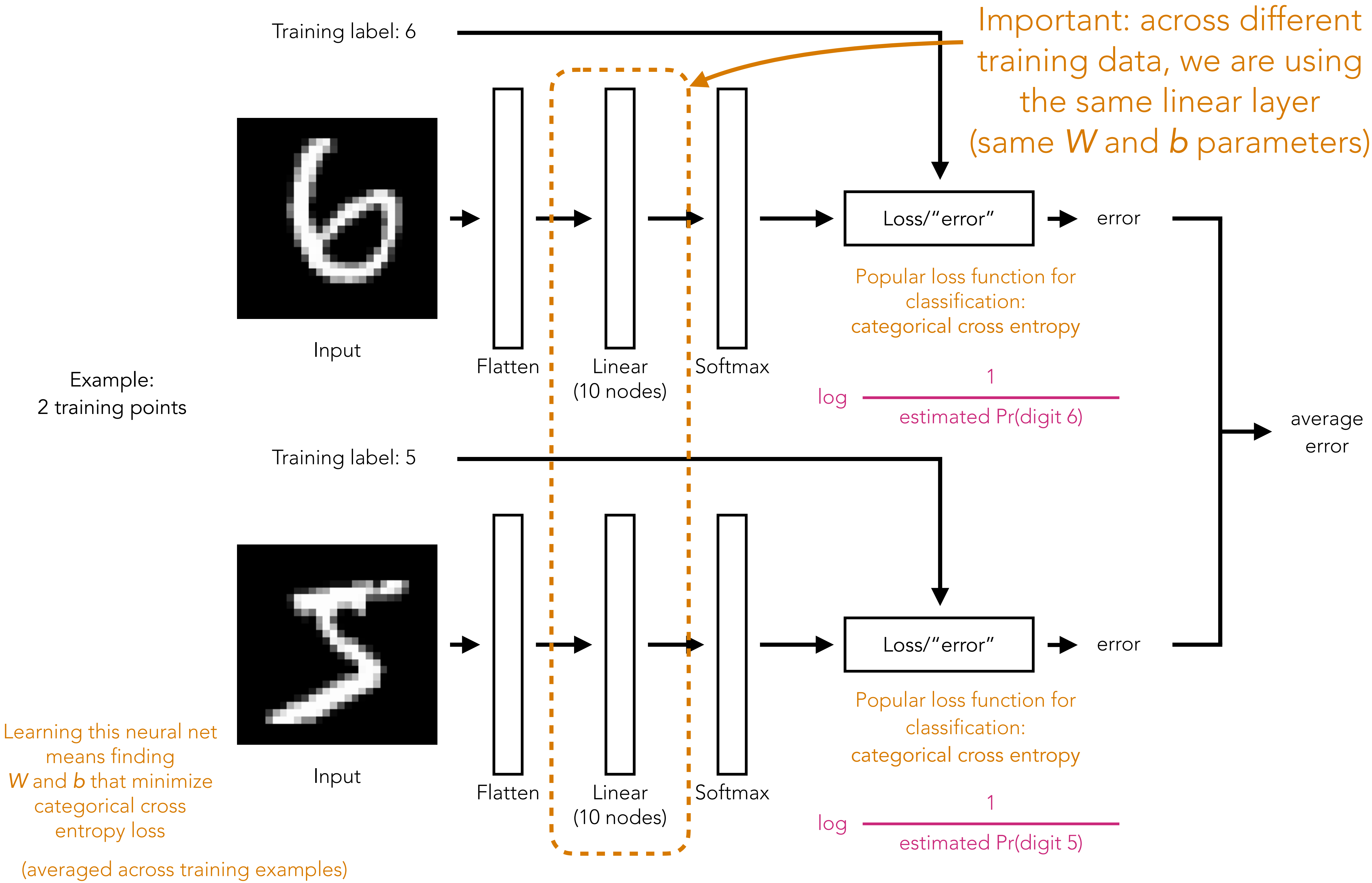


Learning this neural net means finding *W* and *b* that minimize categorical cross entropy loss

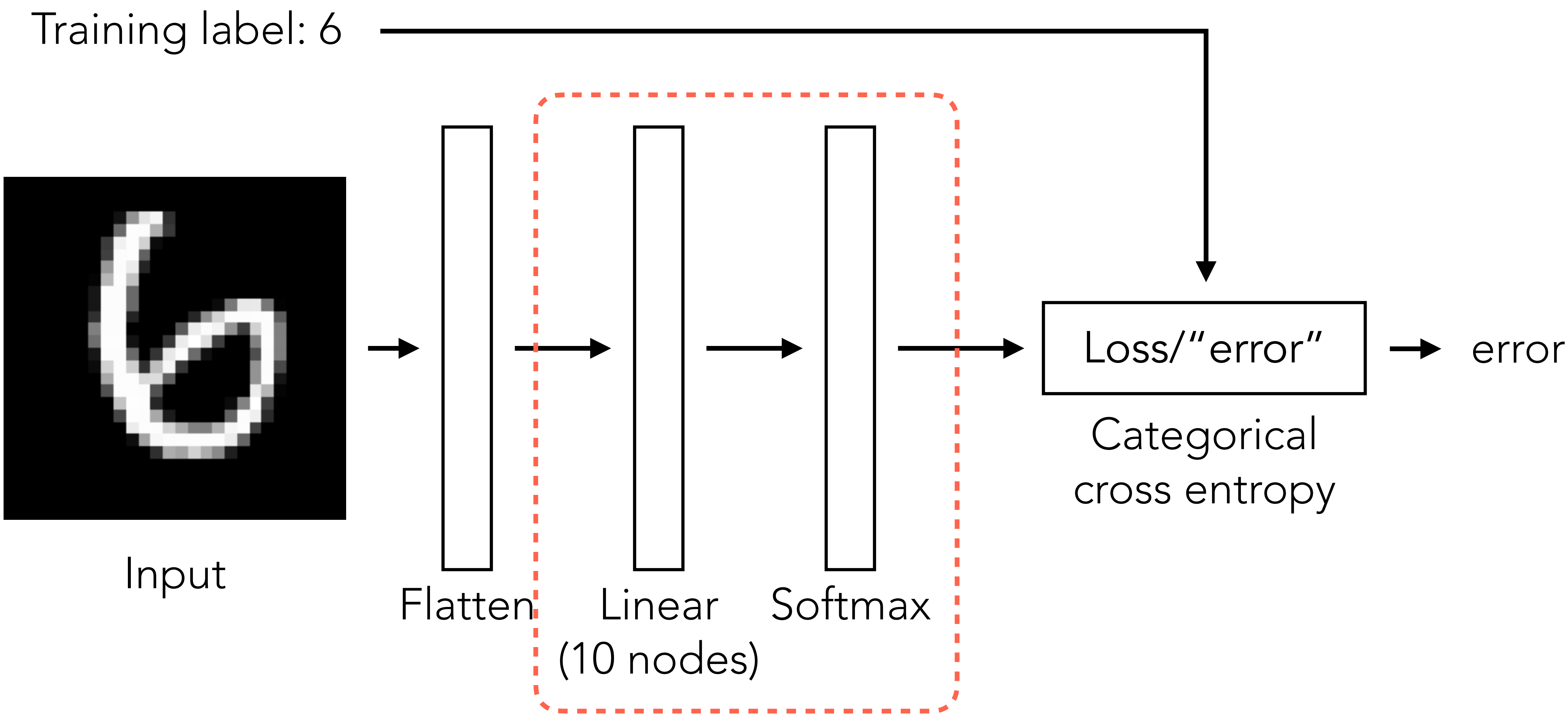
Also called *fully-connected* or *dense* layer

⚠️ In PyTorch, softmax is included as part of the cross entropy loss

Handwritten Digit Recognition

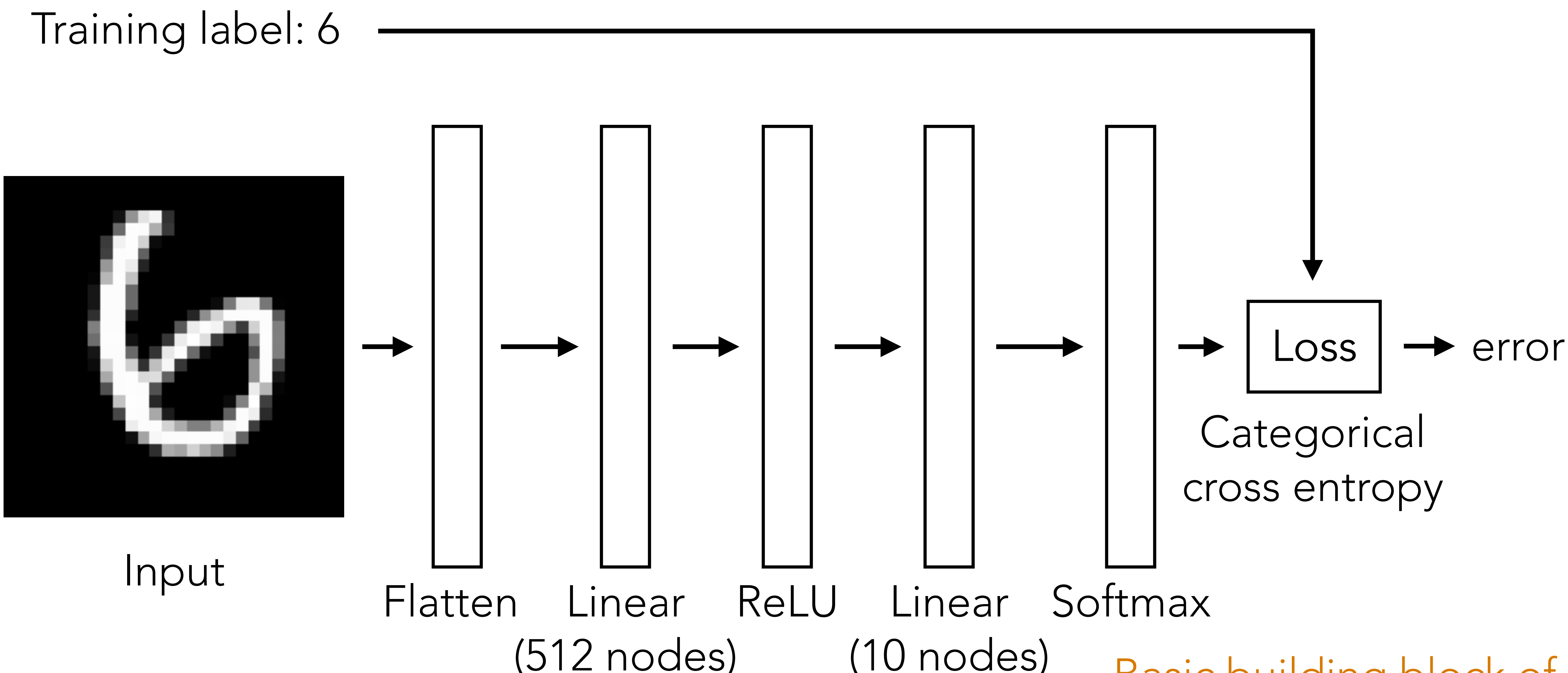


Handwritten Digit Recognition



This neural net has a name: **multinomial logistic regression**
(when there are only 2 classes, it's called **logistic regression**)

Handwritten Digit Recognition

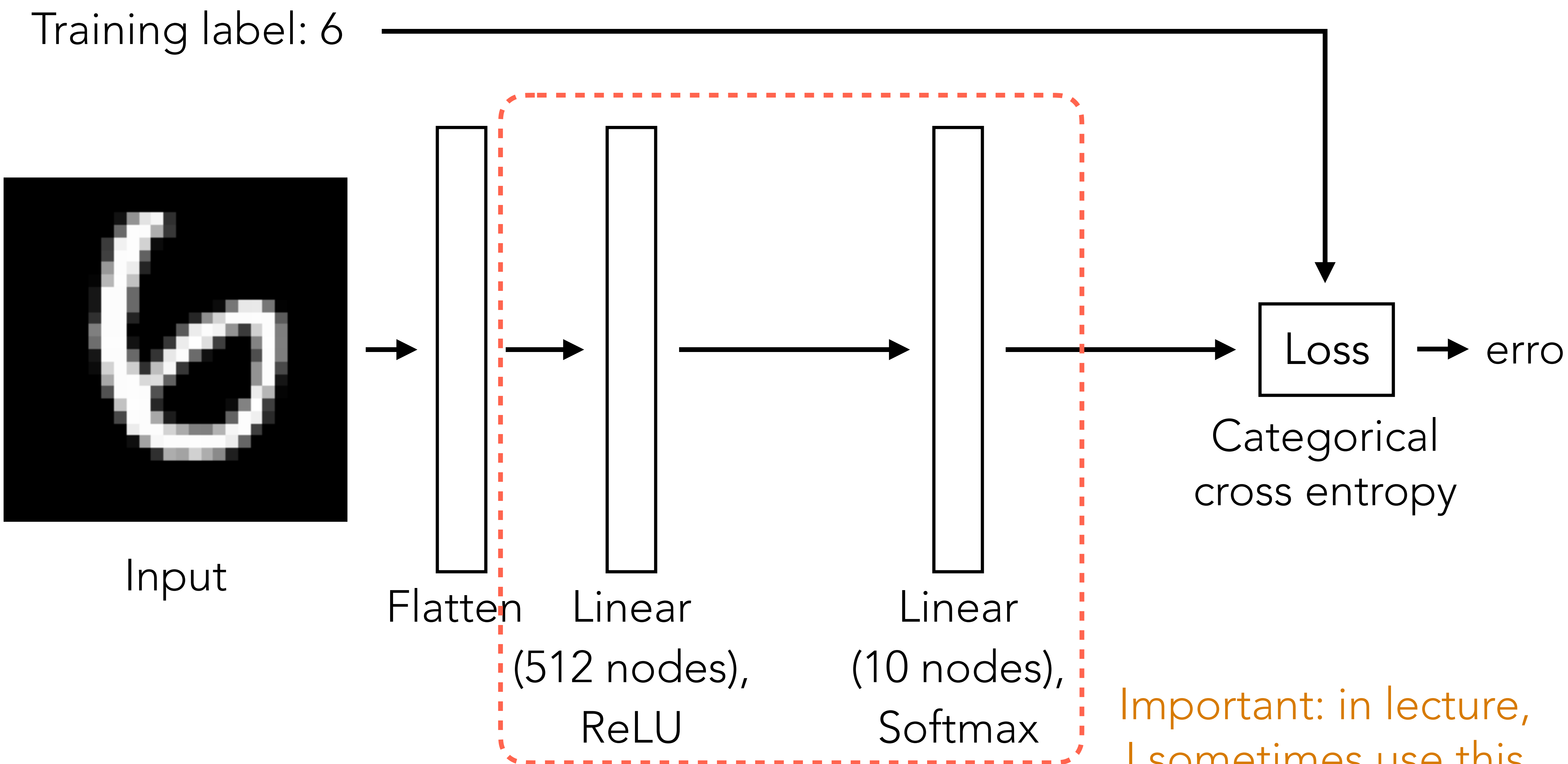


Different linear layers; each has its own weight matrix and bias vector

Basic building block of neural nets:
linear layer with nonlinear activation

Learning this neural net ⇒ learn parameters of both linear layers

Handwritten Digit Recognition



This neural net is called a multilayer perceptron
(# nodes need not be 512 & 10;
activations need not be ReLU and softmax)

Important: in lecture,
I sometimes use this
shorthand notation
(specifying activation to
go with each linear layer)

PyTorch

- Designed to be like NumPy
  - A lot of (but not all) function names are the same as numpy (e.g., instead of calling `np.sum`, you would call `torch.sum`, etc)
  - ⚠️ PyTorch does not use NumPy arrays and instead uses tensors (so instead of `np.array`, you use `torch.tensor`)
- What's the big difference then? Why not just use NumPy?
  - **PyTorch tensors keep track of what device they reside on**
    - ⚠️ For example, trying to add a tensor stored on the CPU and a tensor stored on a GPU will result in an error!
  - **PyTorch tensors can automatically store "gradient" information** (important for learning model parameters; details in later lecture)

PyTorch code is often harder to debug than NumPy code

There's a PyTorch tutorial posted in supplemental reading

Handwritten Digit Recognition

Demo