95-865 Pittsburgh Lecture 9: Introduction to Neural Nets and Deep Learning

George Chen

Announcements

- No the quiz hasn't been graded yet
- **HW1 regrade requests due start of class next Tuesday**
- Please make sure you have AWS set up with AWS Educate credits for HW3 (bug your TA's if you need help)
- Python 3.7 currently has some compatibility issues with Keras and Tensorflow — please downgrade your Python to version 3.6 if you're using 3.7!!!

```
conda install python=3.6
conda install keras
```

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

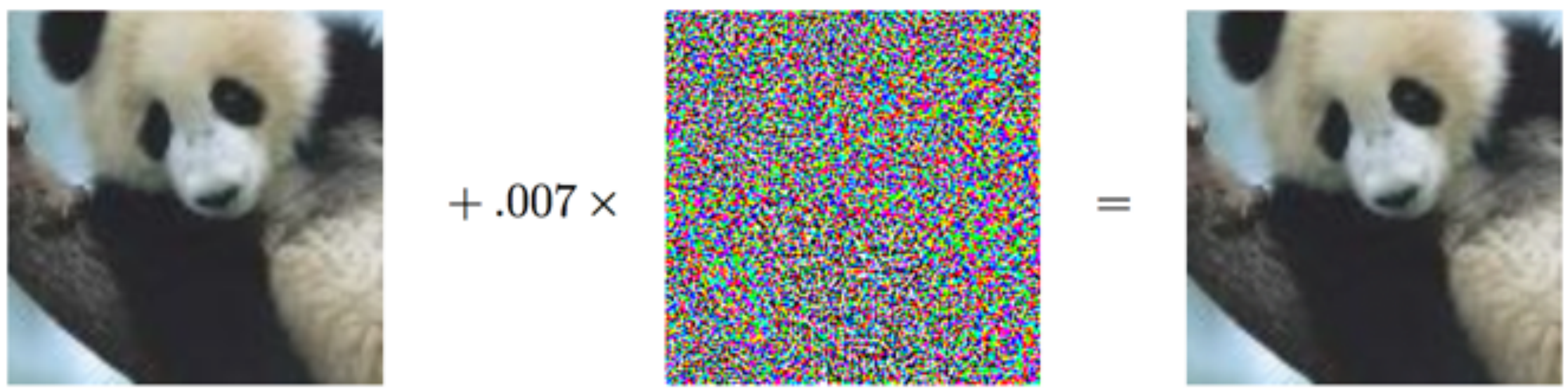
Heavily dominated by industry now!

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

Is it all hype?

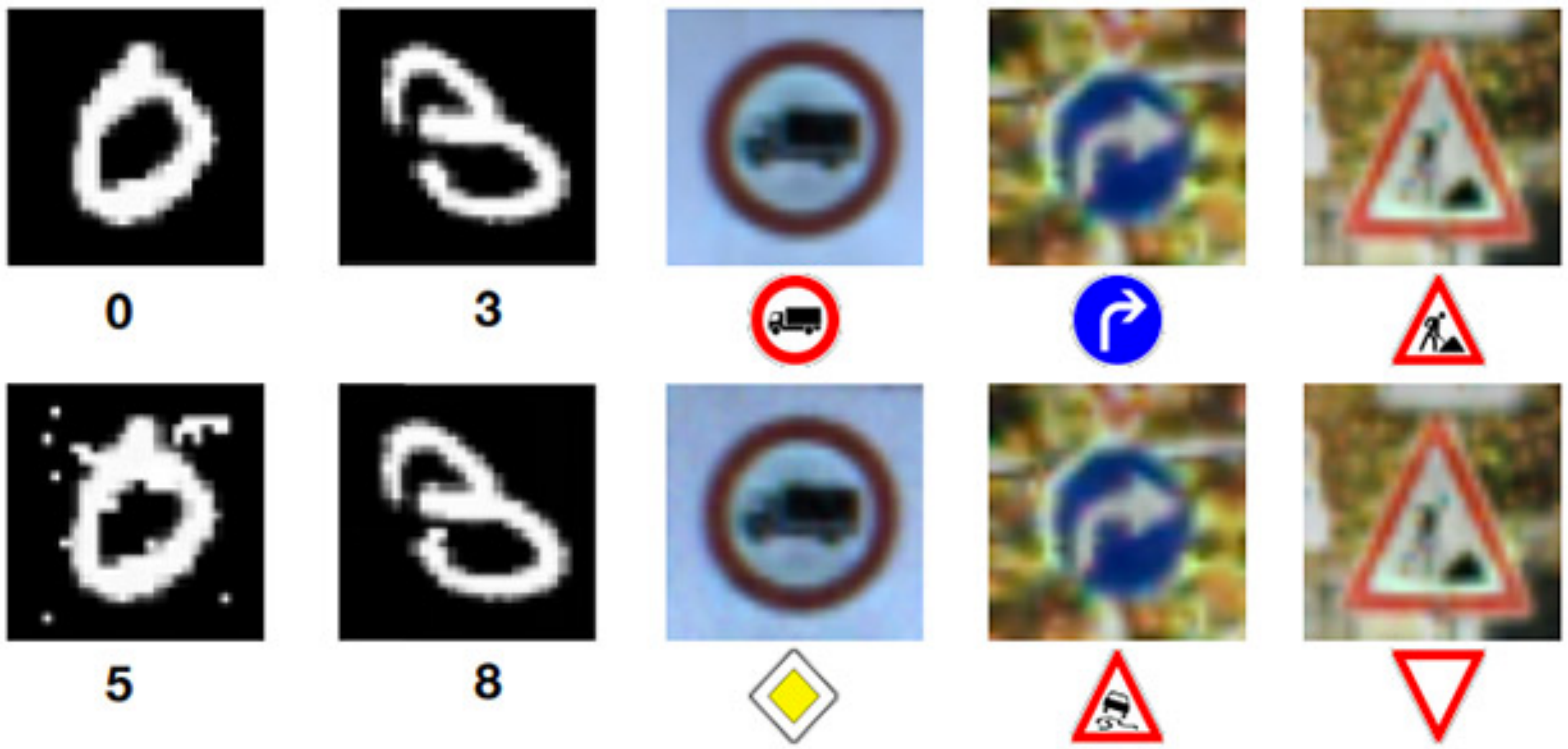


panda
~58% confidence

adversarial
noise

gibbon
~99% confidence

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

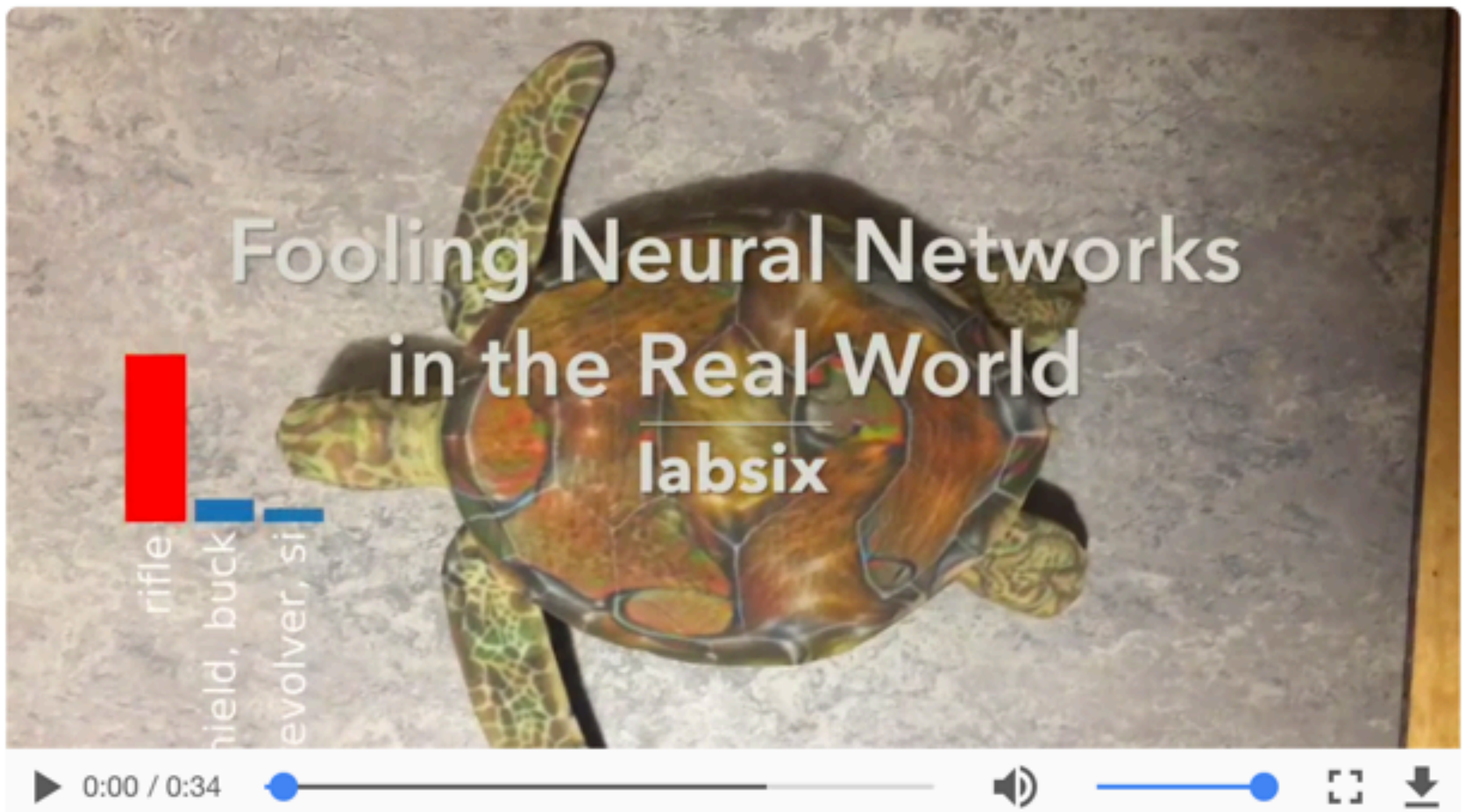


Source: Papernot et al. Practical Black-Box Attacks against Machine Learning. Asia Conference on Computer and Communications Security 2017.

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.

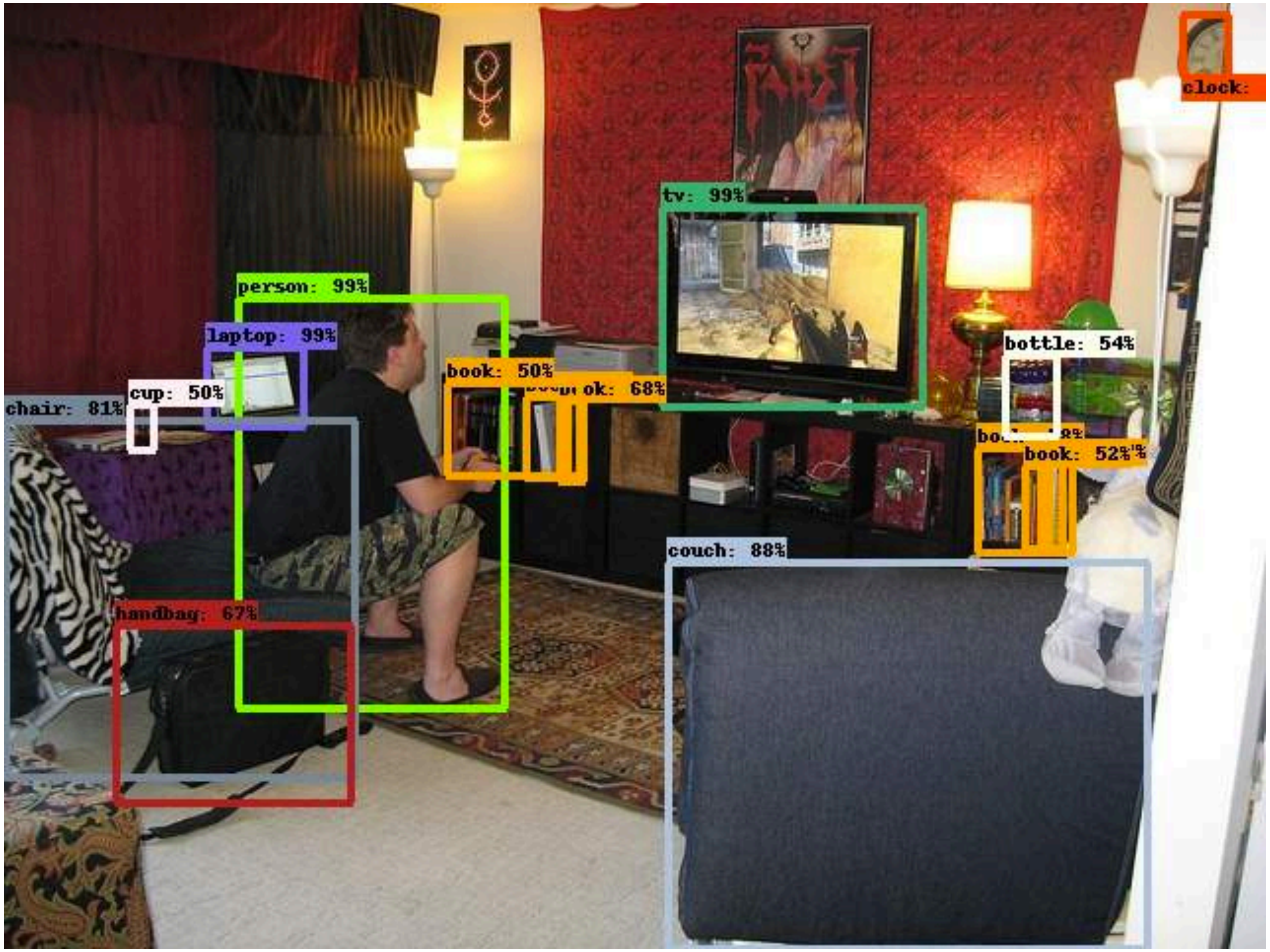


Neural network based classifiers reach near-human performance in many tasks, and they're used in high risk, real world systems. Yet, these same neural networks are particularly vulnerable to *adversarial examples*, carefully perturbed inputs that cause

Source: labsix

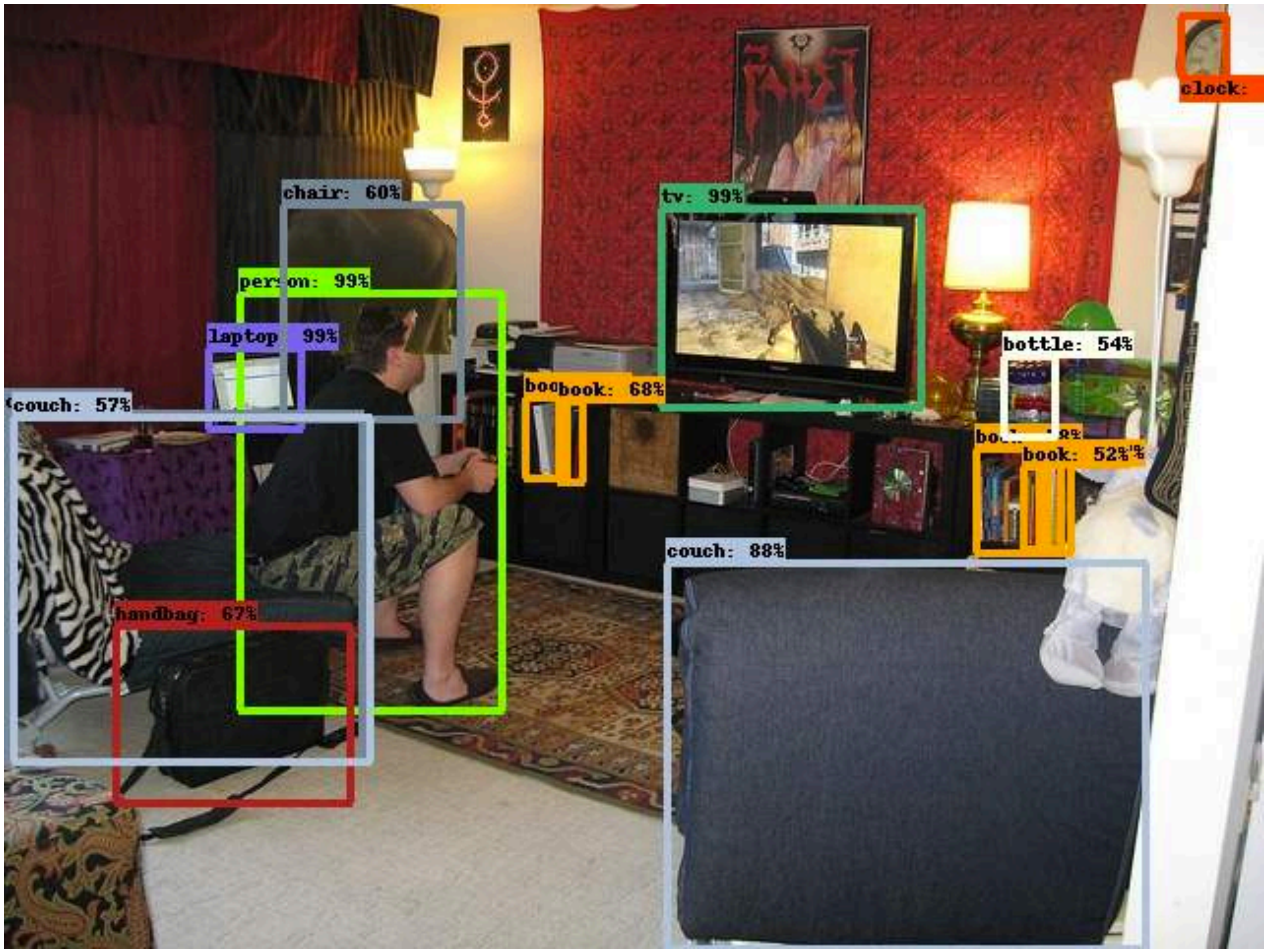
Source: Gizmodo article "This Neural Network's Hilariously Bad Image Descriptions Are Still Advanced AI". September 16, 2015. (They're using the NeuralTalk image-to-caption software.)

Slightly modifying an image results in different prediction results



Source: Quanta Magazine article “Machine Learning Confronts the Elephant in the Room”. September 20, 2018.

Slightly modifying an image results in different prediction results



Source: Quanta Magazine article “Machine Learning Confronts the Elephant in the Room”. September 20, 2018.

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

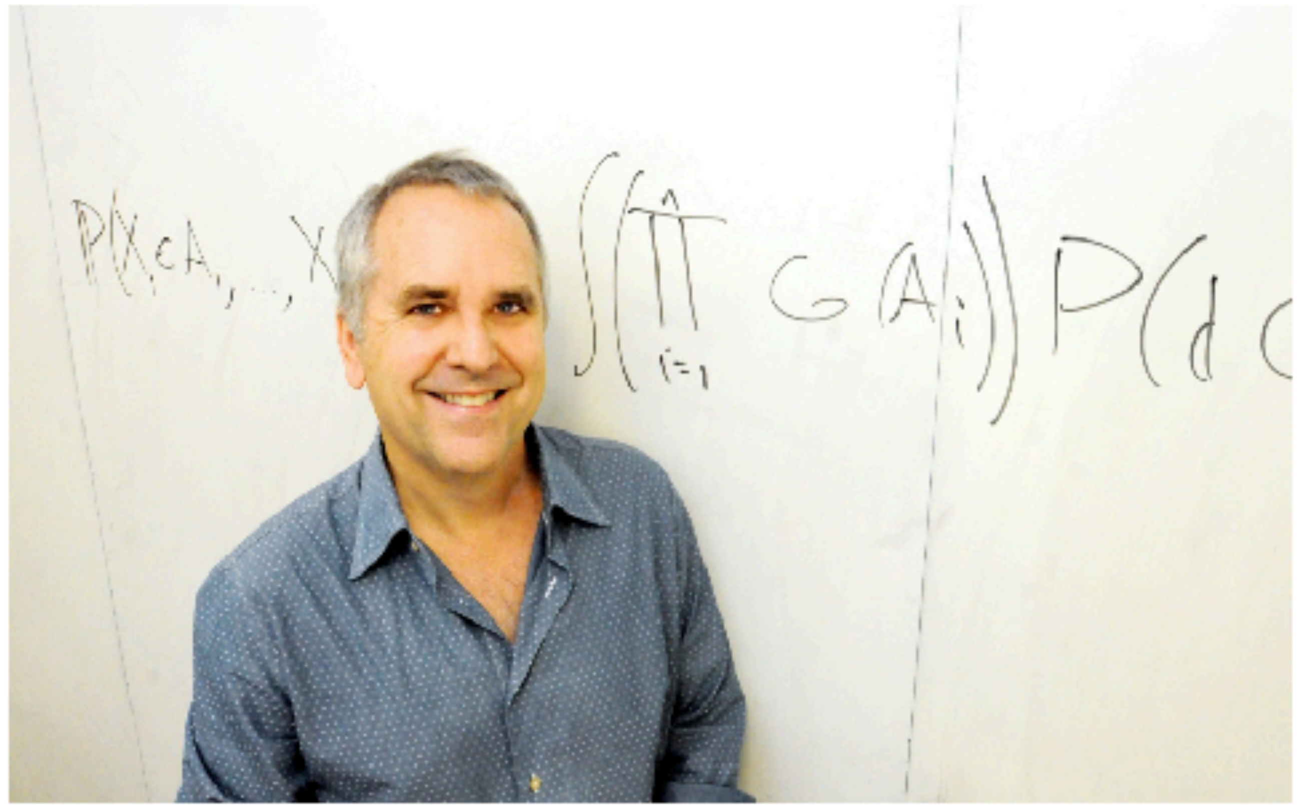


Photo credit: Peg Skorpinski

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

The Atlantic Popular Latest Sections Magazine More

TECHNOLOGY

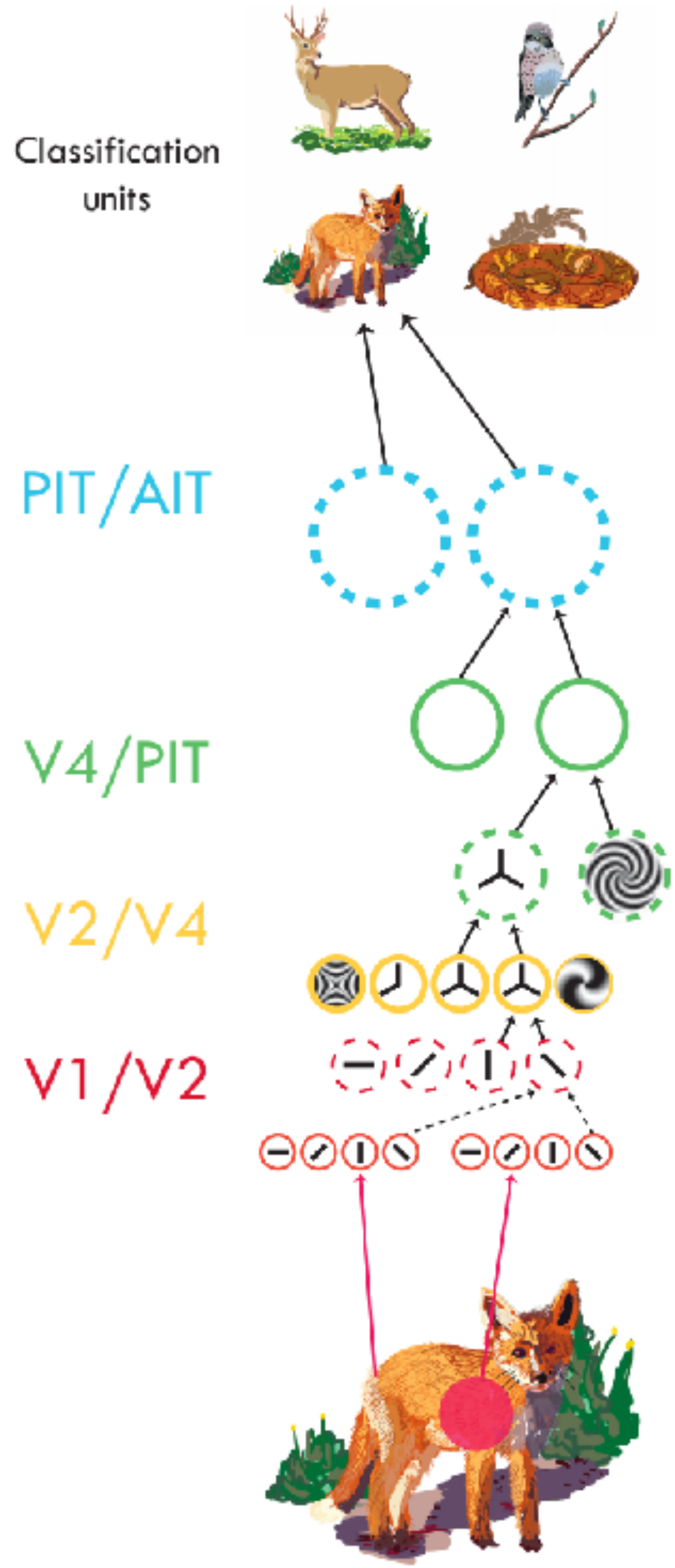
How a Pioneer of Machine Learning Became One of Its Sharpest Critics

Judea Pearl helped artificial intelligence gain a strong grasp on probability, but laments that it still can't compute cause and effect.

KEVIN HARTNETT AND QUANTA MAY 19, 2018

https://www.theatlantic.com/technology/archive/2018/05/machine-learning-is-stuck-on-asking-why/560675/?single_page=true

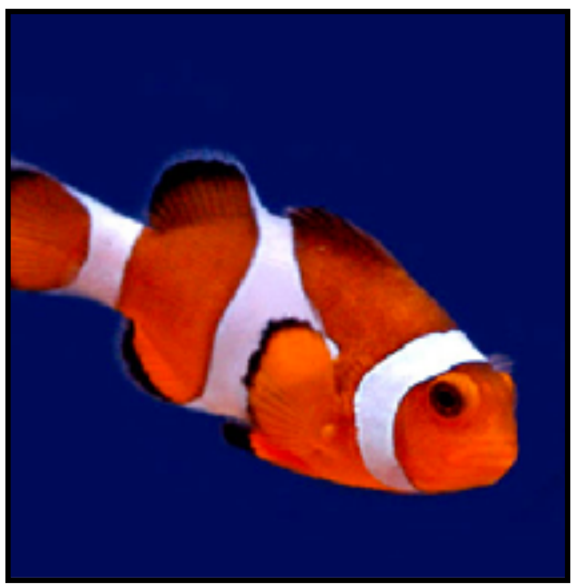
What is deep learning?



Slide by Phillip Isola

Serre, 2014

Basic Idea

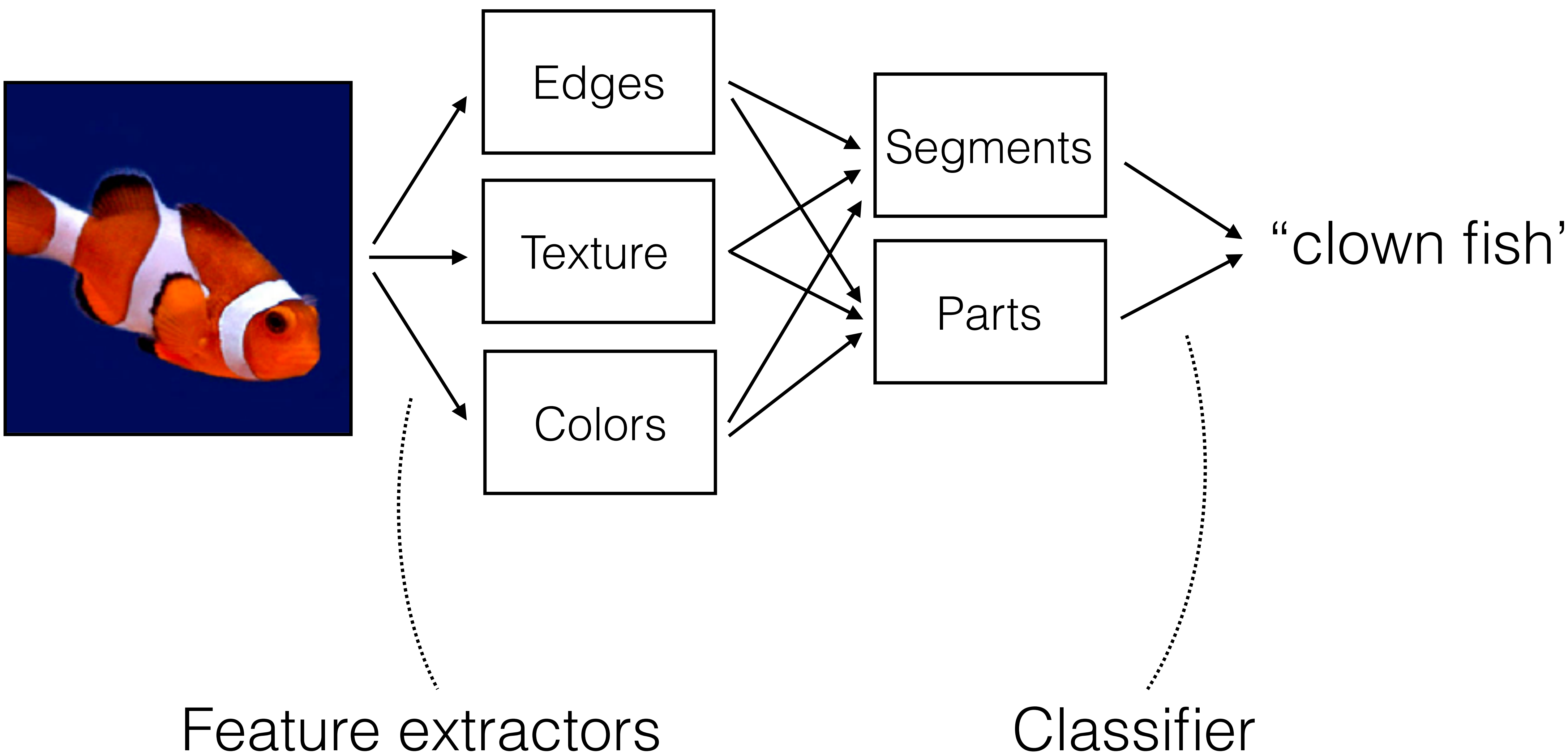


Brain/Machine

"clown fish"

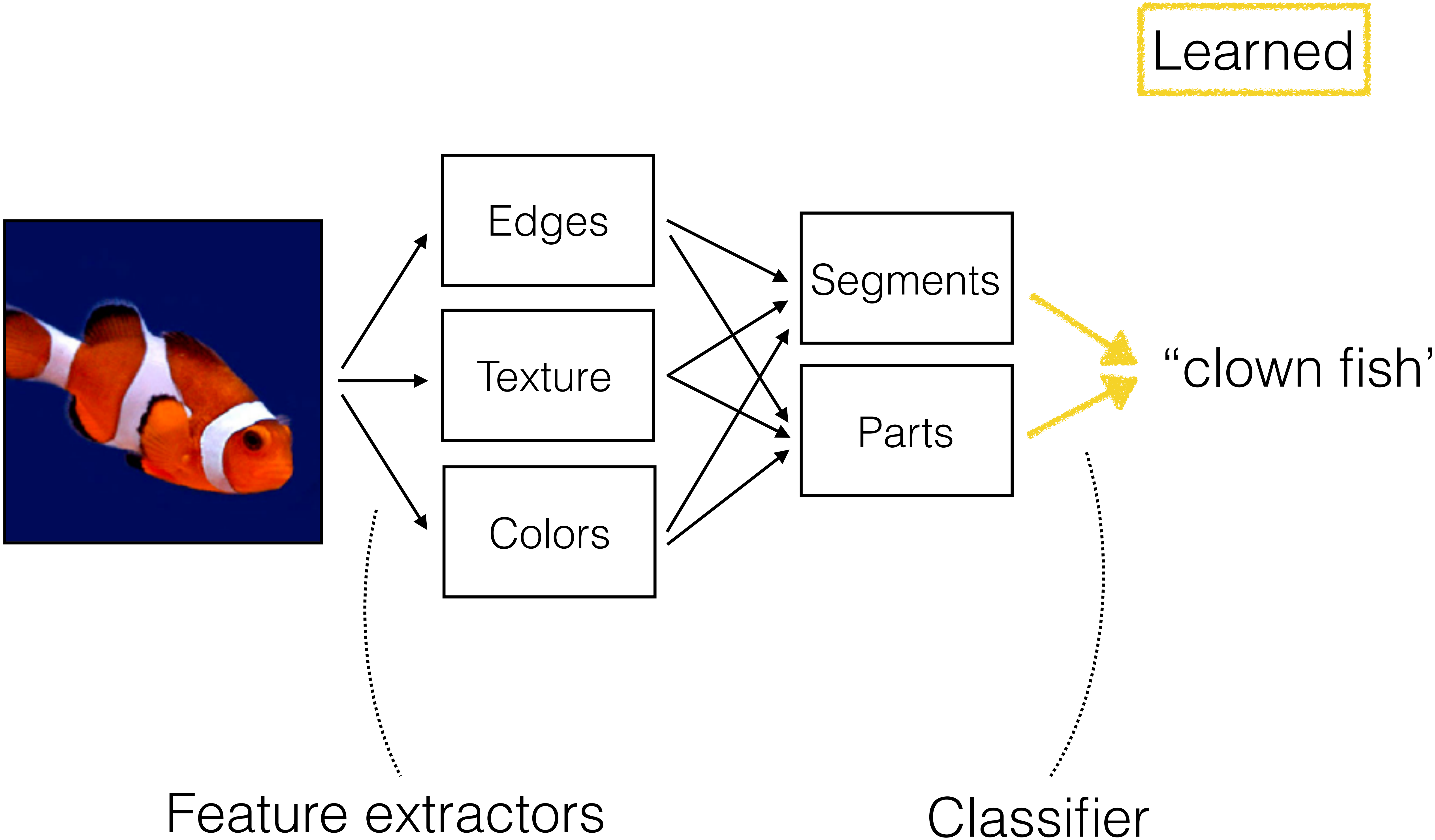
Slide by Phillip Isola

Object Recognition



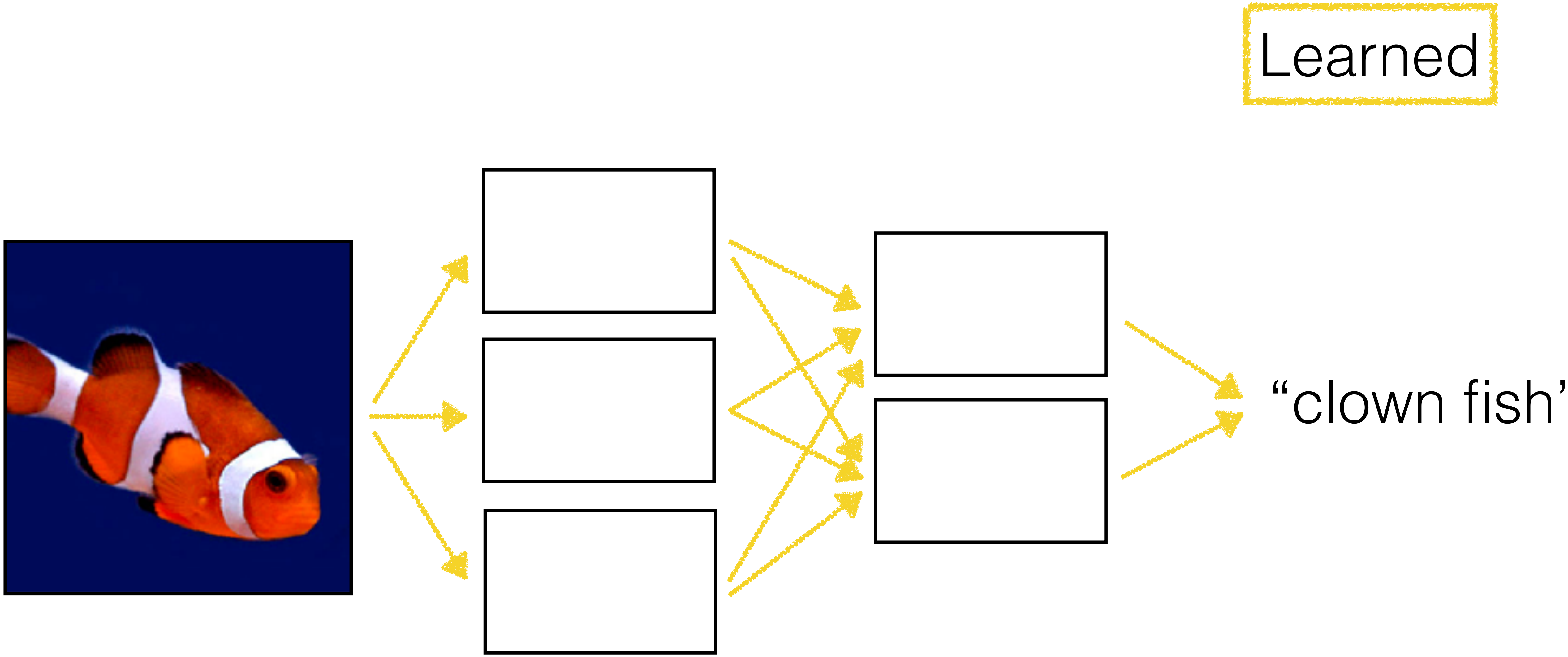
Slide by Phillip Isola

Object Recognition



Slide by Phillip Isola

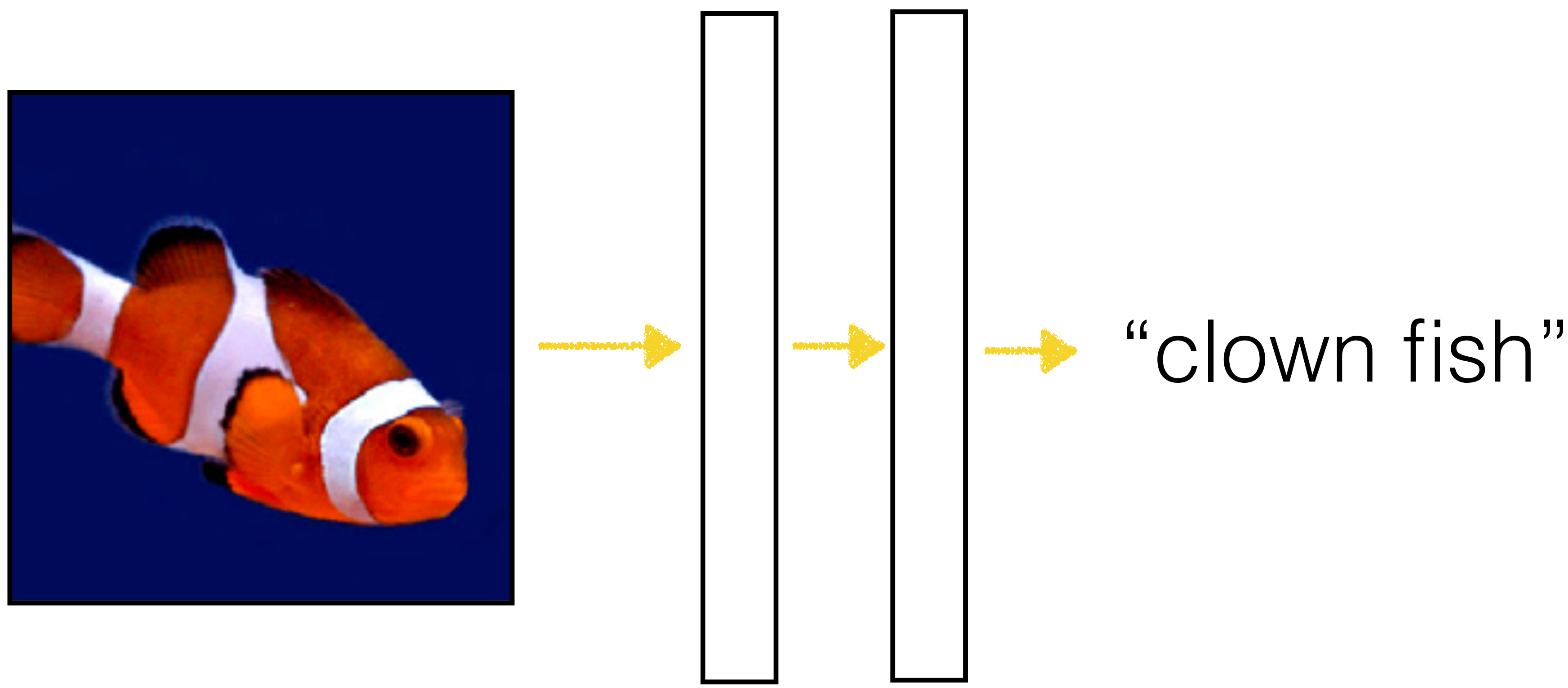
Neural Network



Slide by Phillip Isola

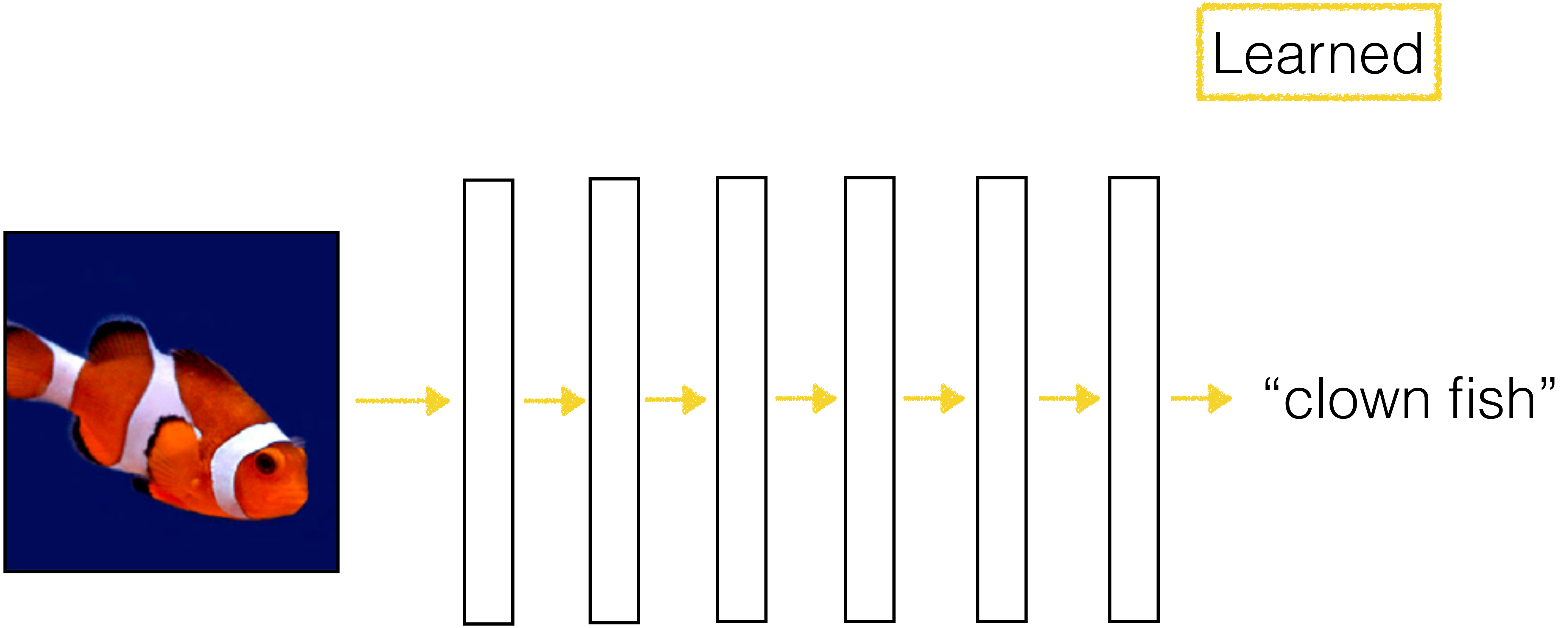
Neural Network

Learned



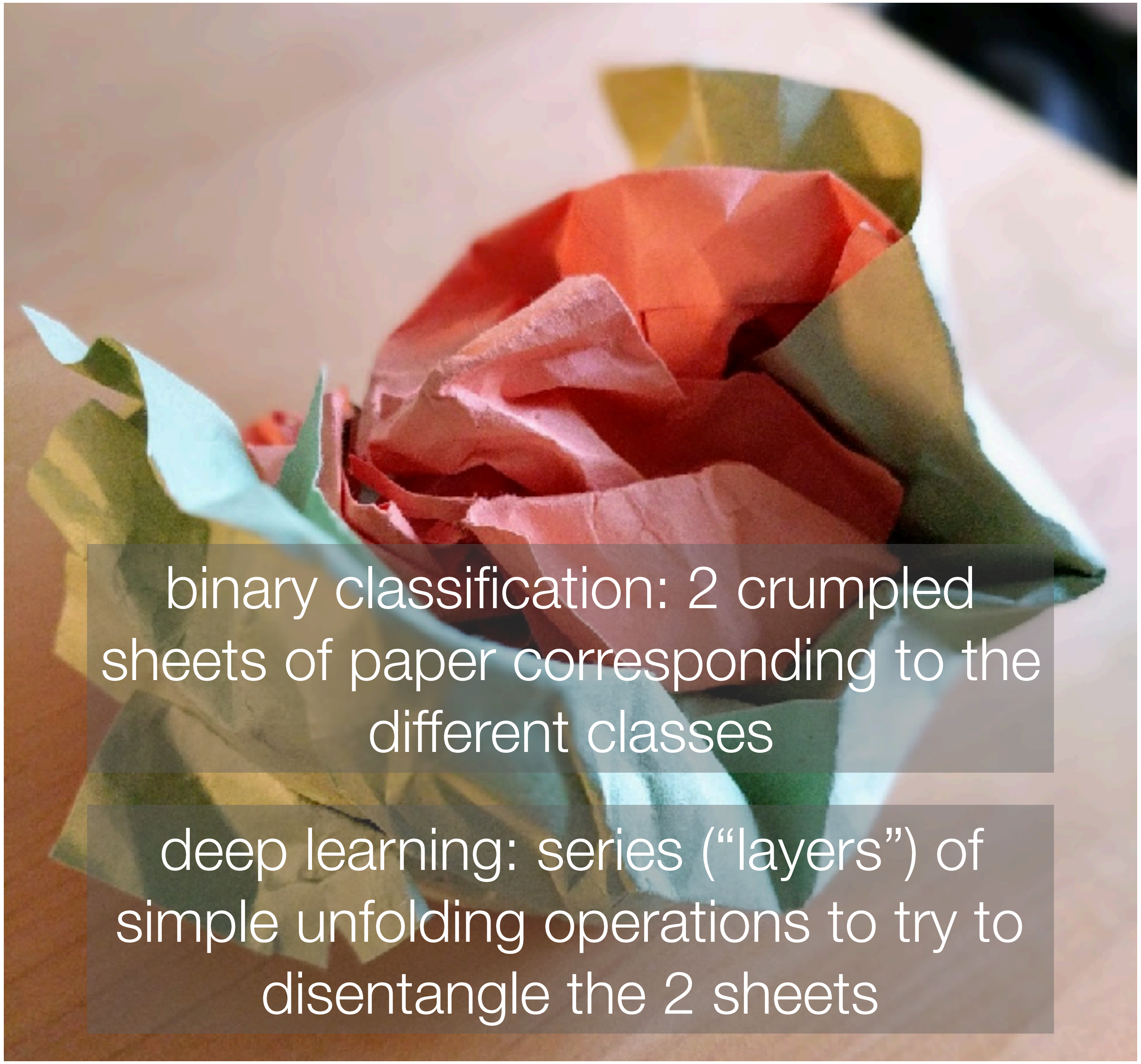
Slide by Phillip Isola

Deep Neural Network



Slide by Phillip Isola

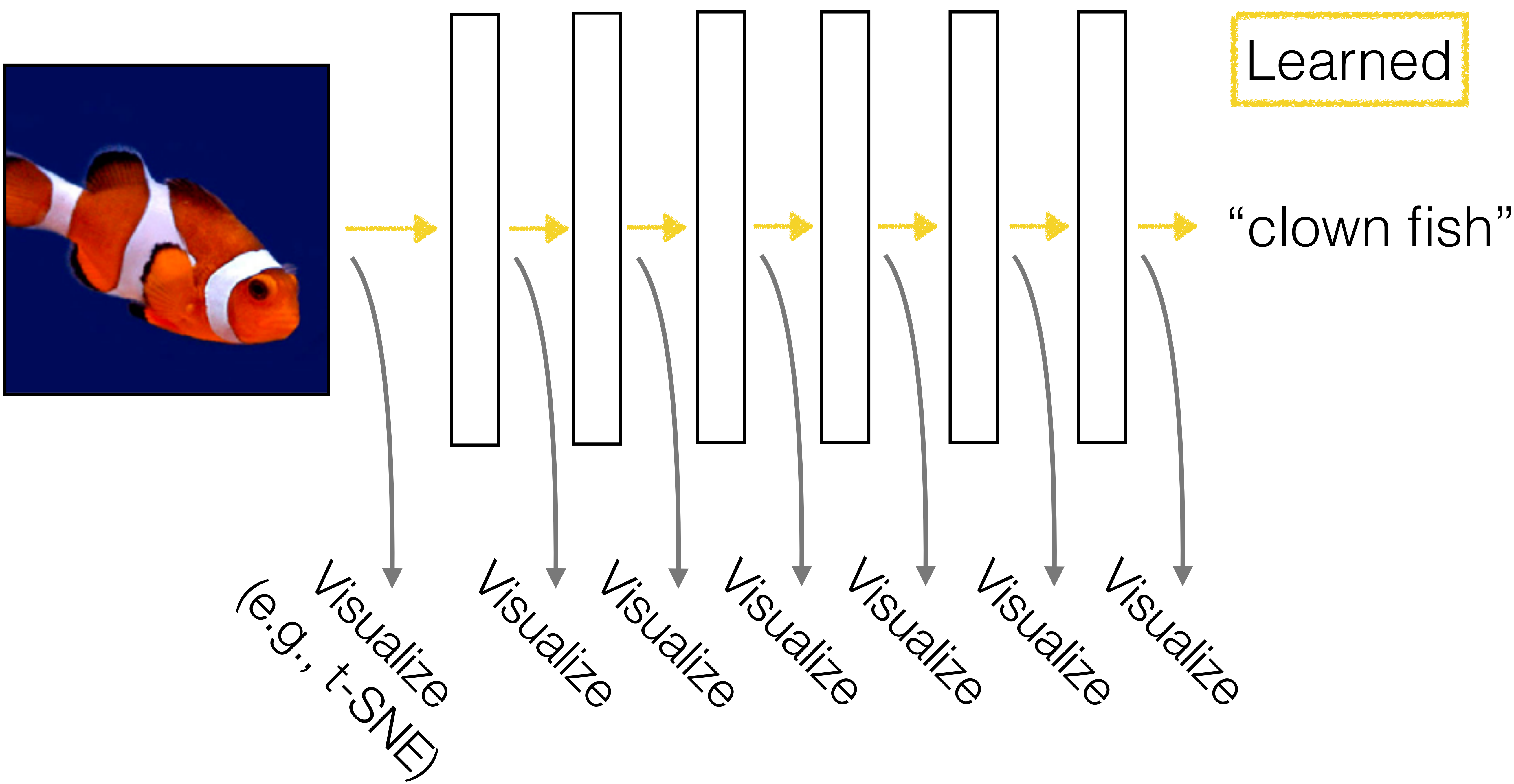
Crumpled Paper Analogy



Analogy: Francois Chollet, photo: George Chen

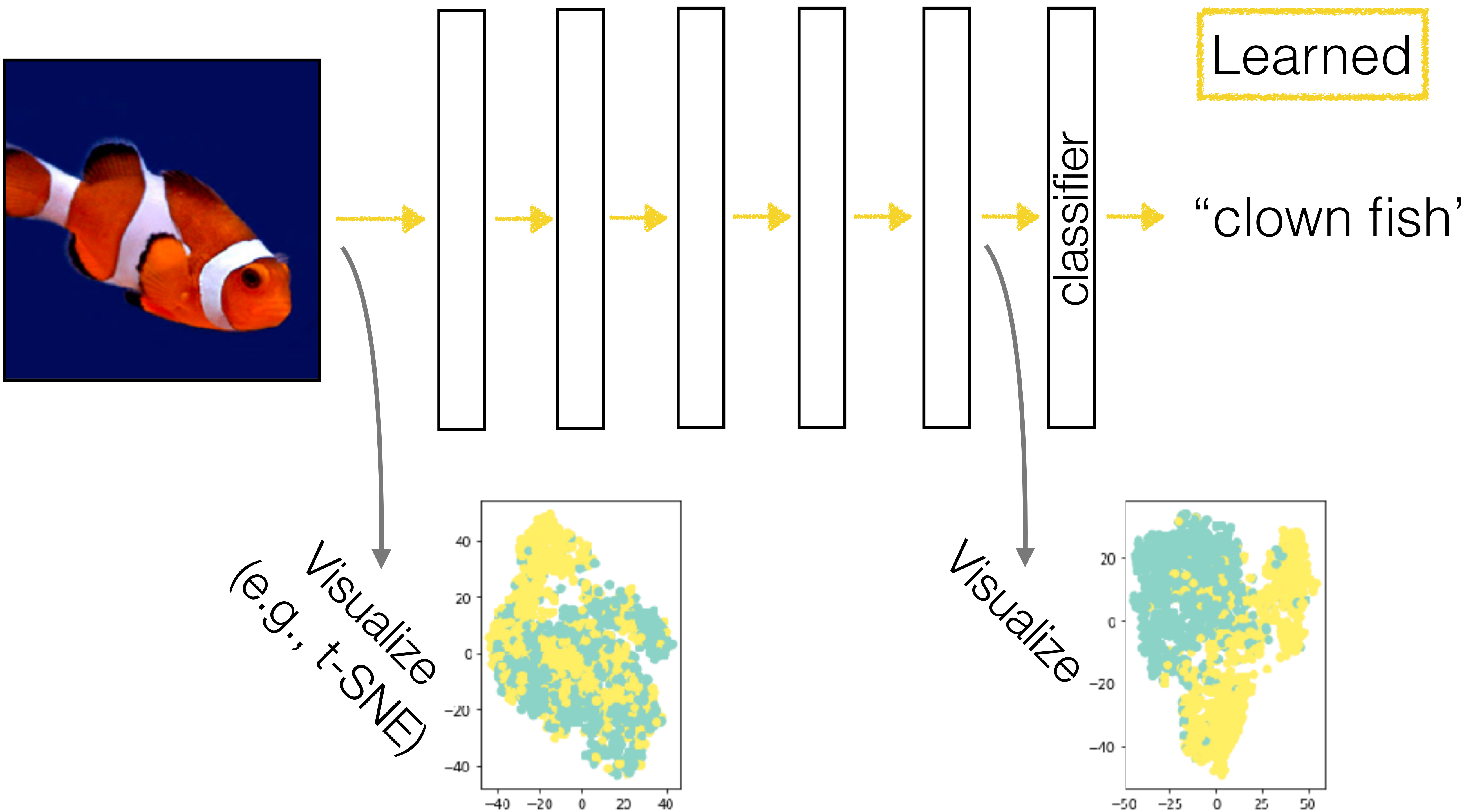
Representation Learning

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?

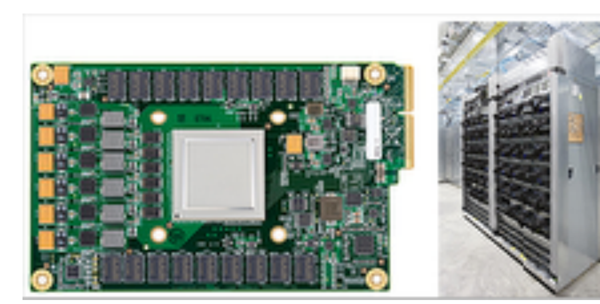
Actually the ideas behind deep learning are old (~1980's)

- Big data

- Better hardware

CPU's
& Moore's law

GPU's



TPU's

- Better algorithms

Structure Present in Data Matters

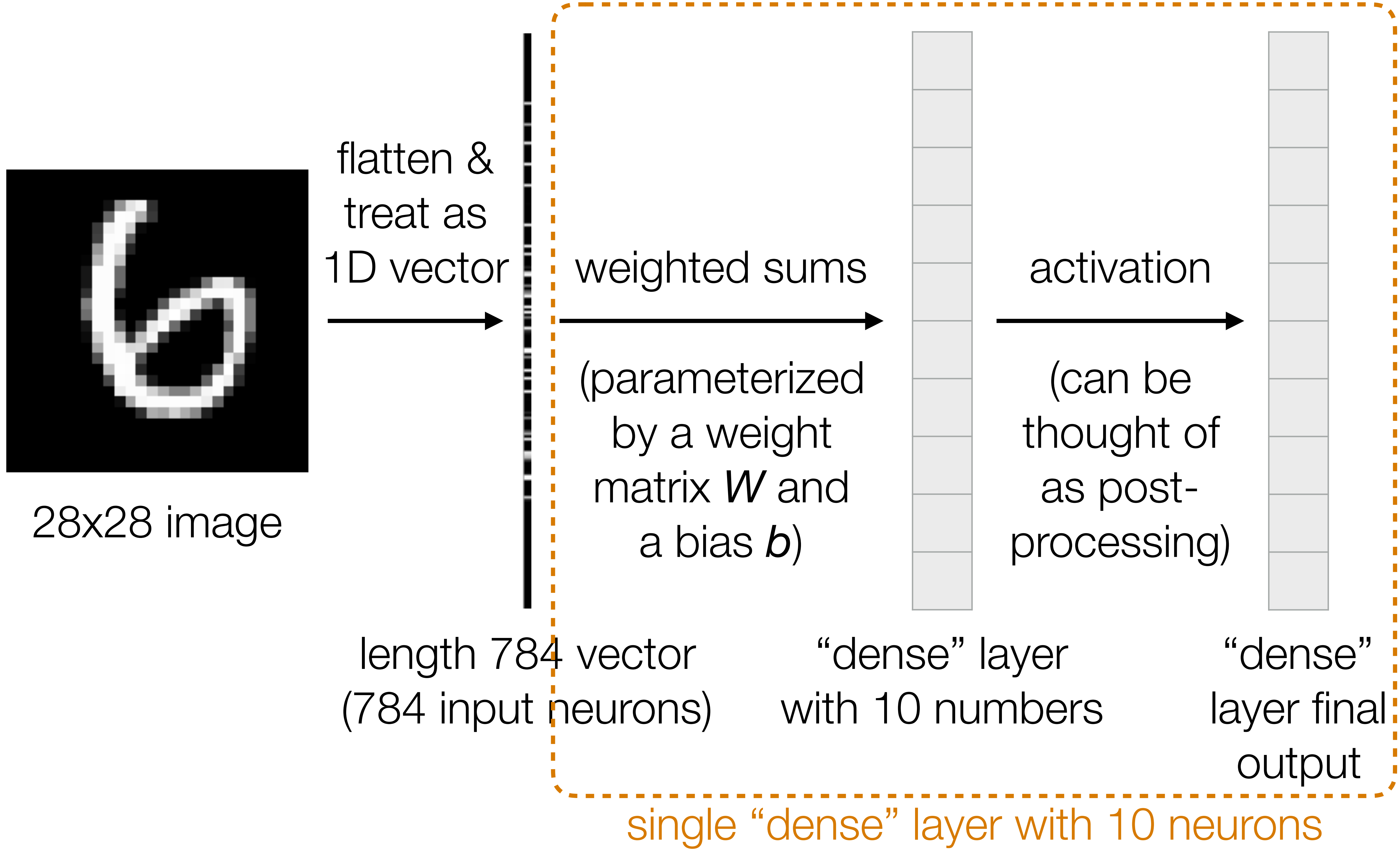
Neural nets aren’t doing black magic

- **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

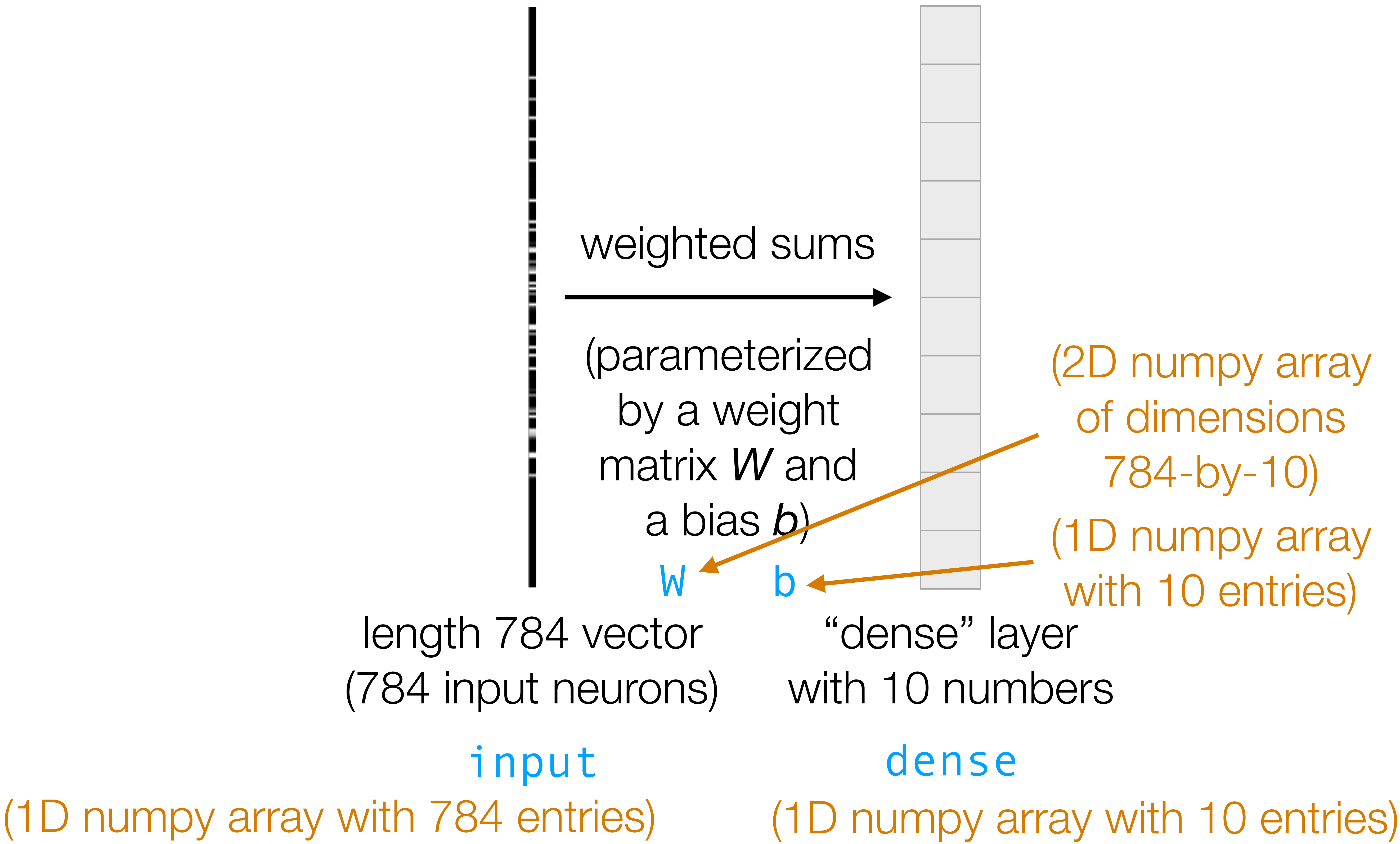
Handwritten Digit Recognition Example

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` `b`

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

(1D numpy array with 10 entries)

n 784 vector
nput neurons)

`input`

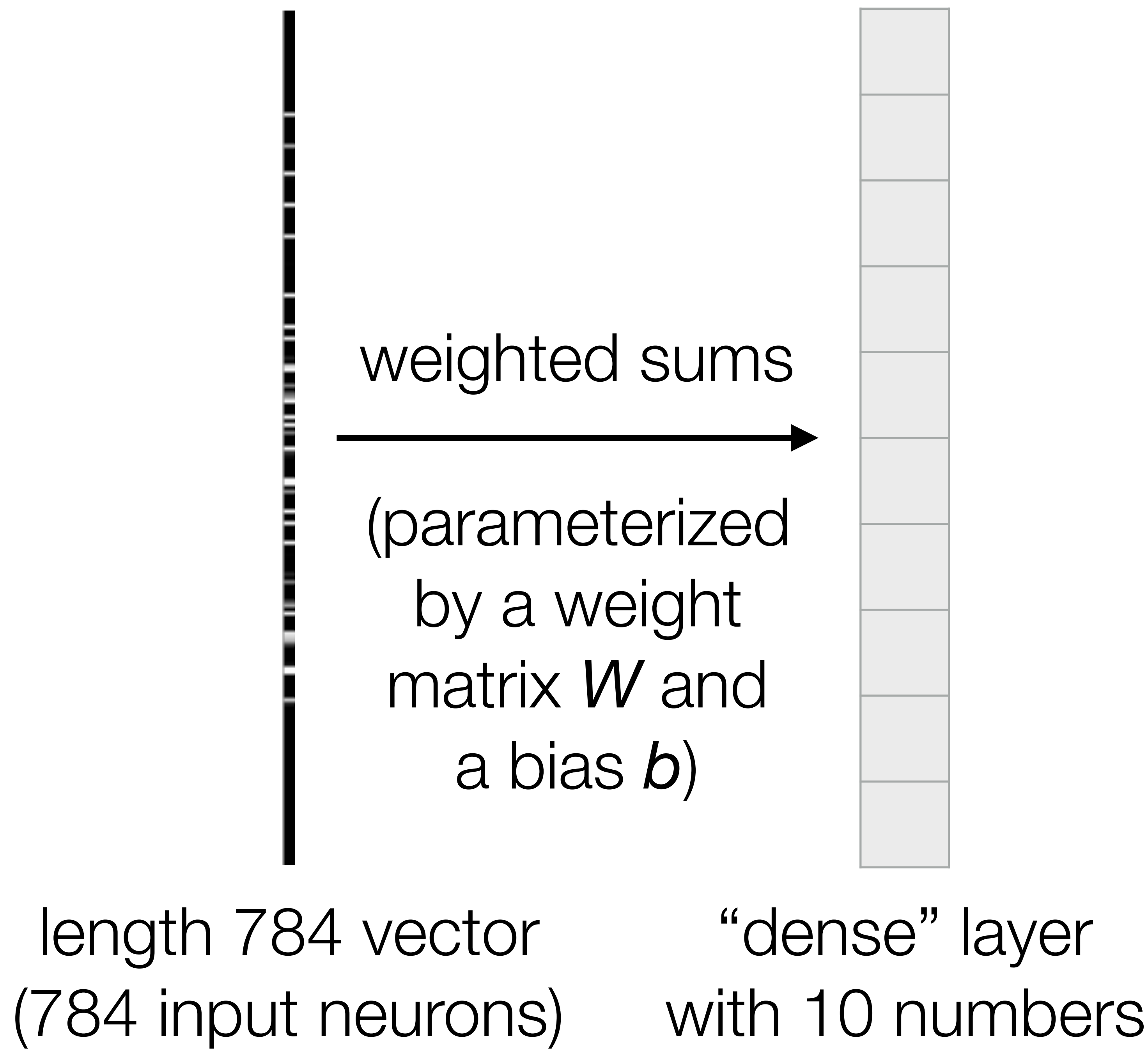
34 entries)

“dense” layer
with 10 numbers

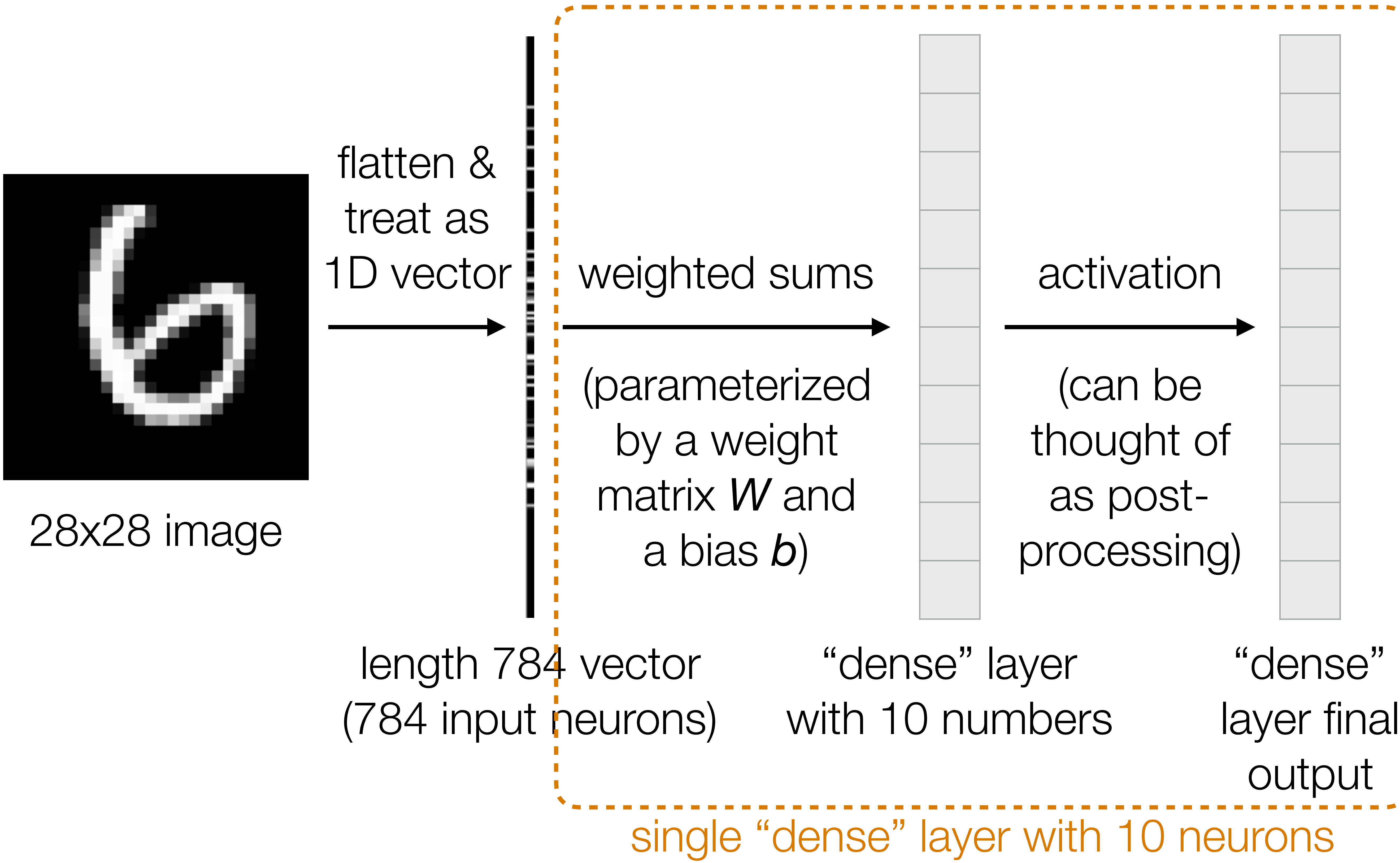
`dense`

(1D numpy array with 10 entries)

Handwritten Digit Recognition



Handwritten Digit Recognition

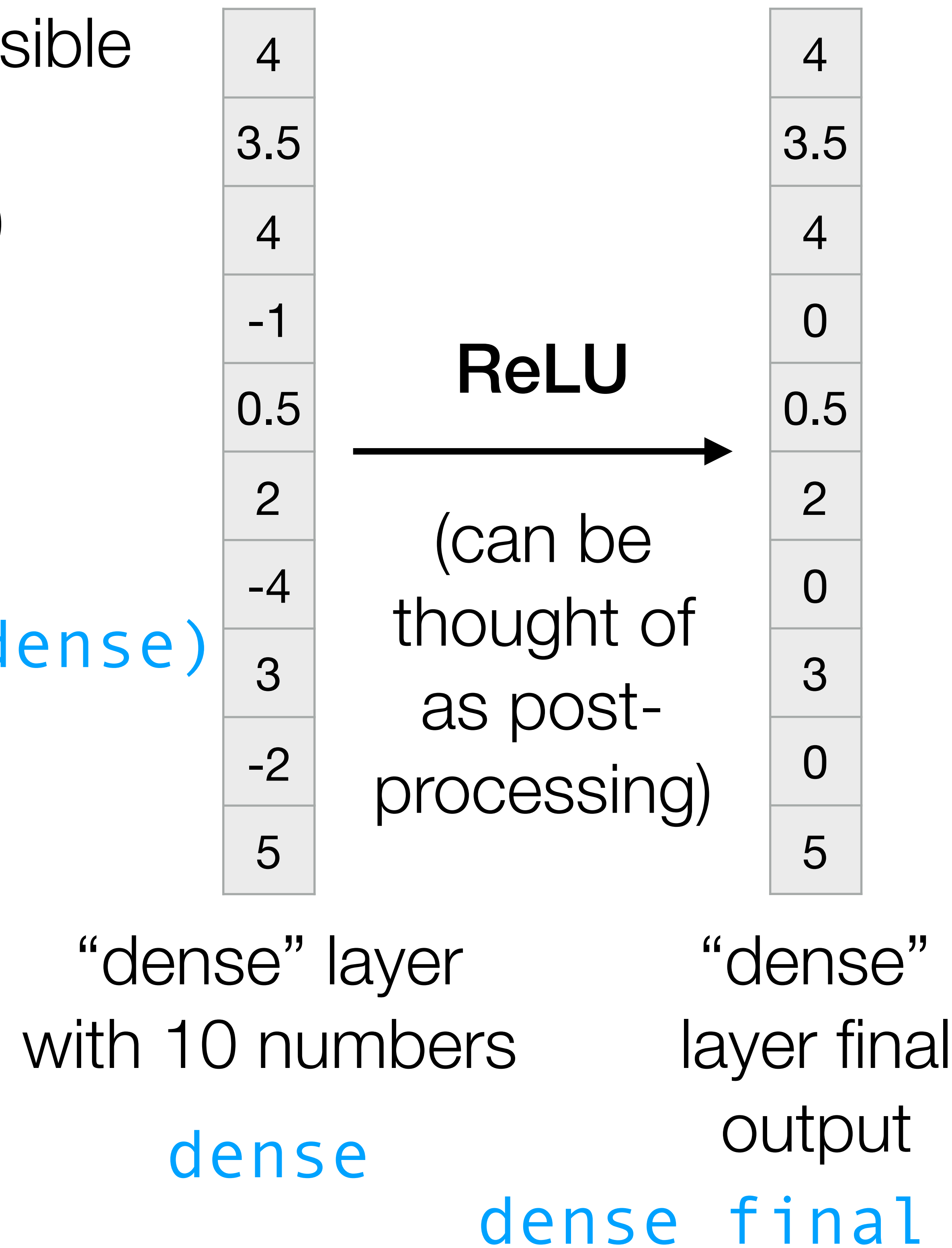


Handwritten Digit Recognition

Many different activation functions possible

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

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



Handwritten Digit Recognition

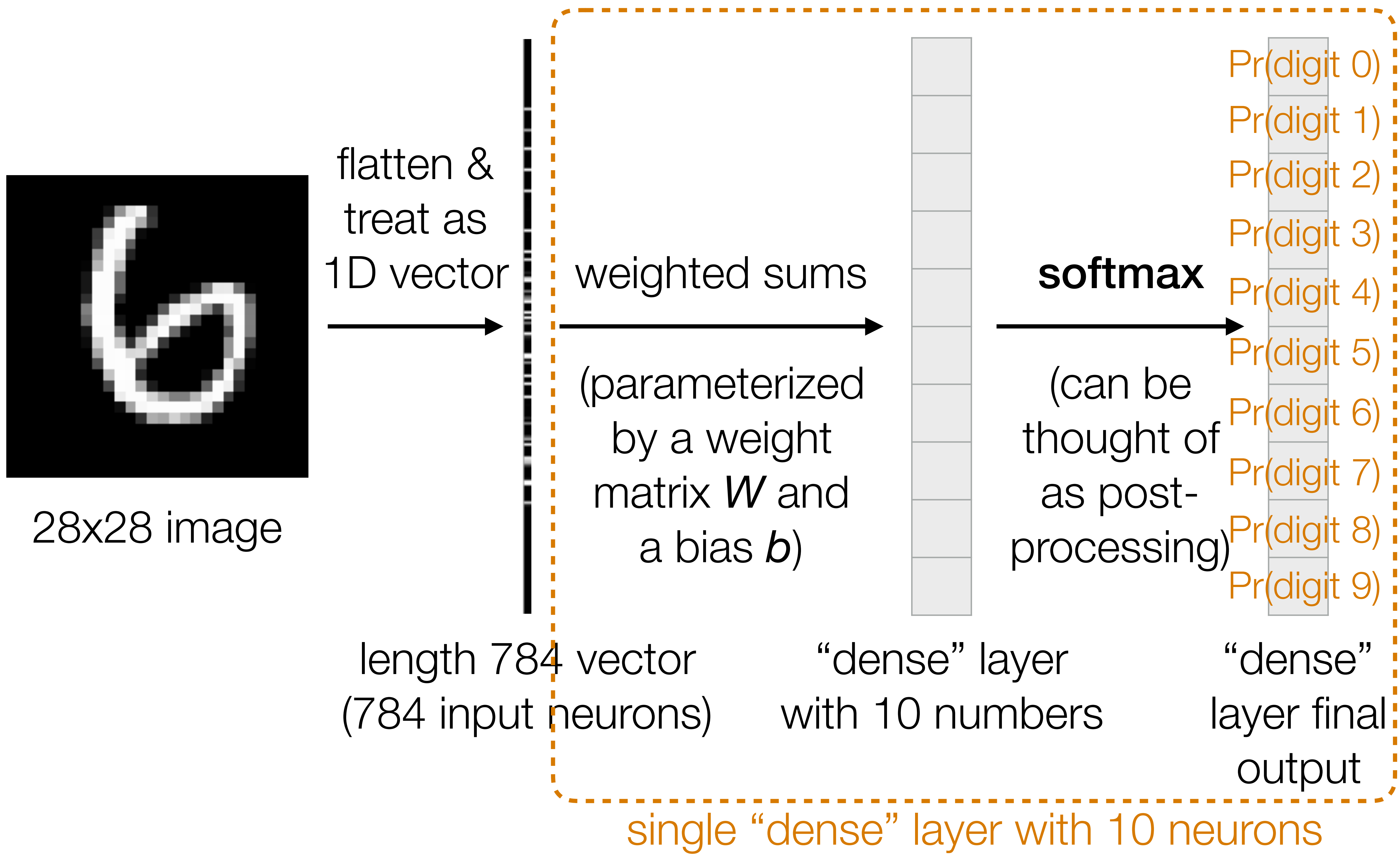
Many different activation functions possible

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

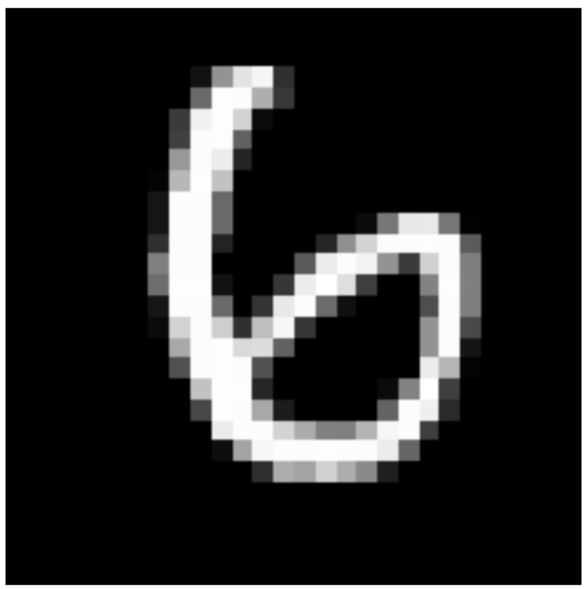
```
dense_exp = np.exp(dense)
dense_exp /= np.sum(dense_exp)
dense_final = dense_exp
```

"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)

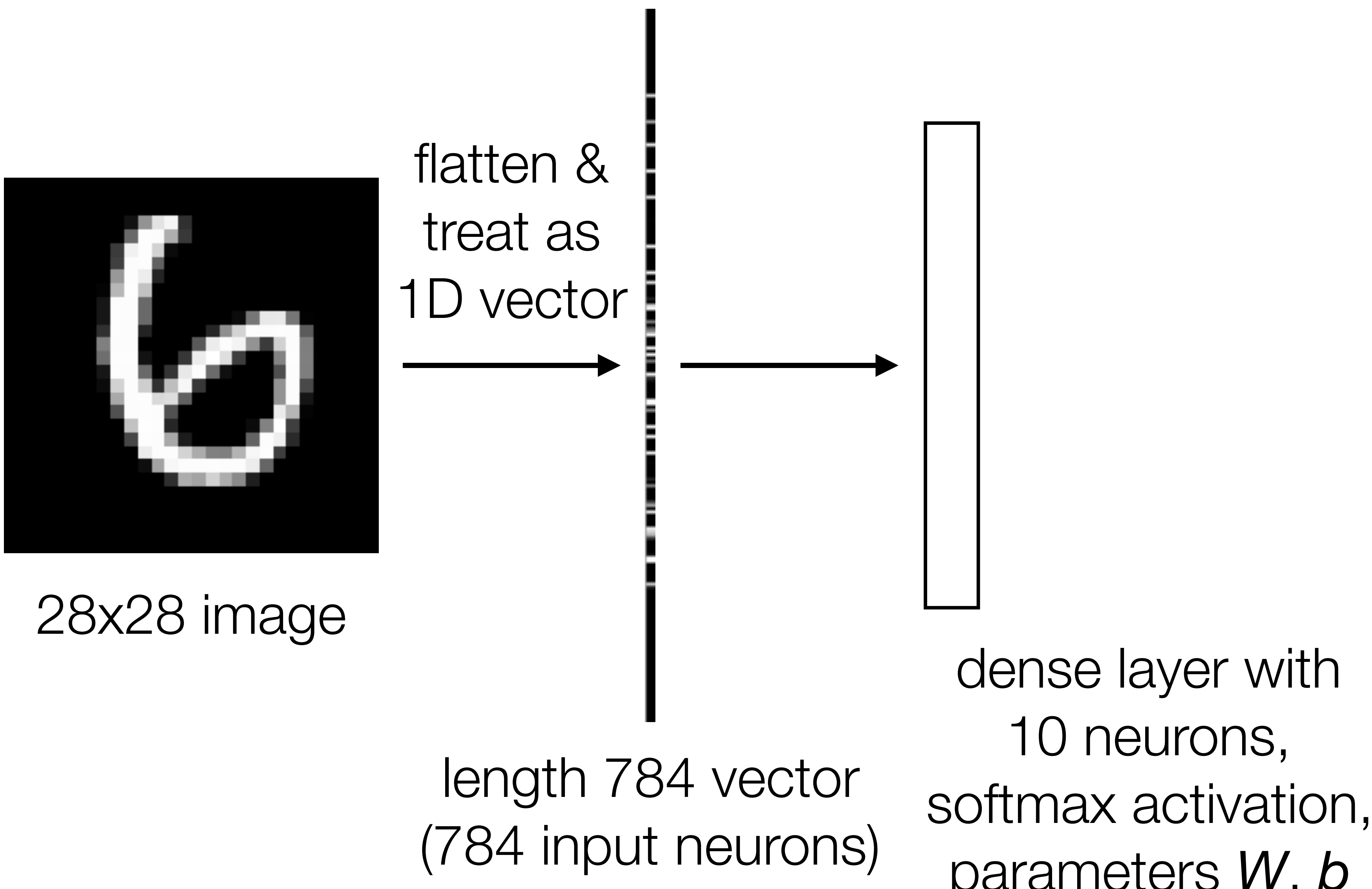
dense layer with 10 neurons, softmax activation, parameters *W*, *b*

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*

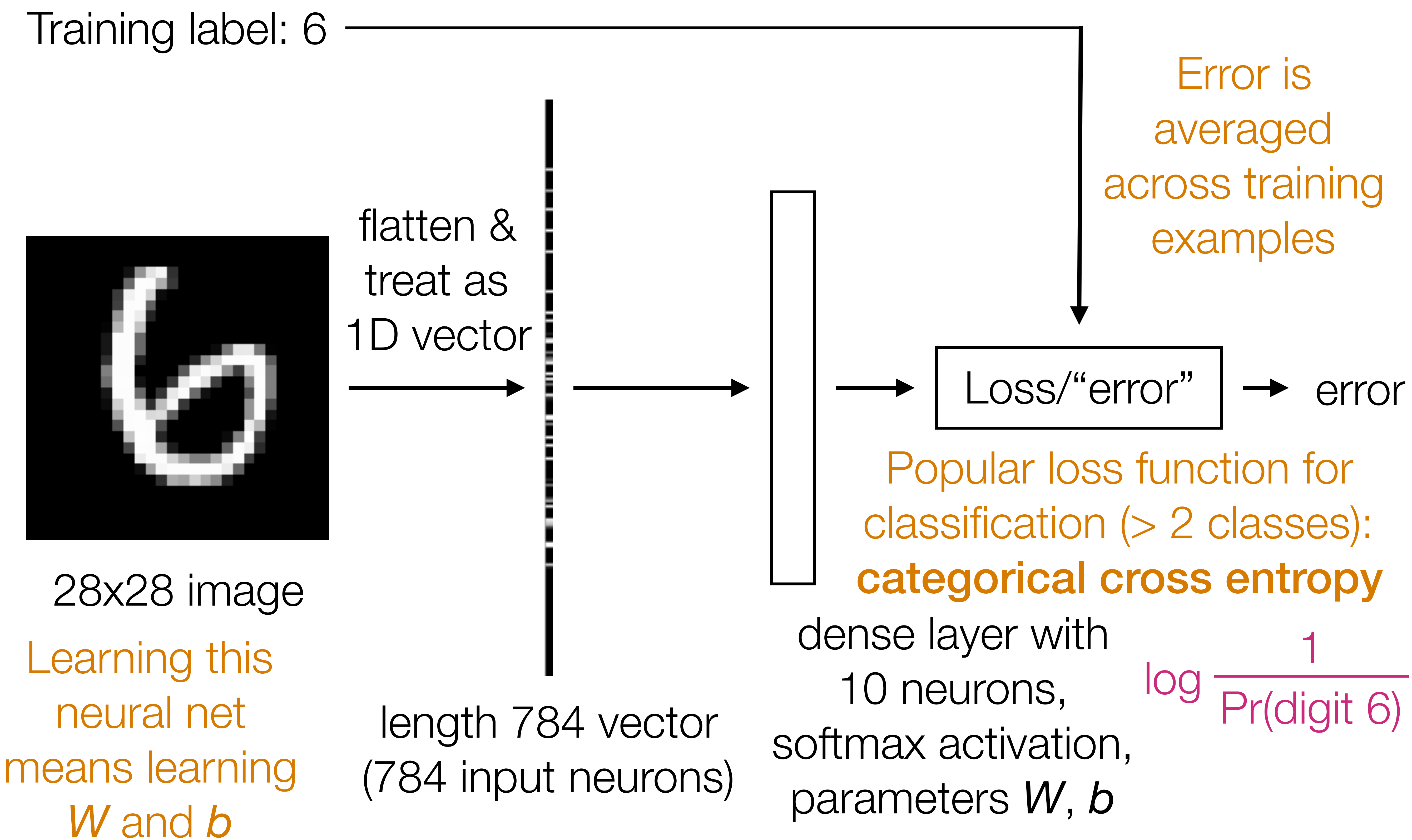
Handwritten Digit Recognition

Demo part 1

Handwritten Digit Recognition



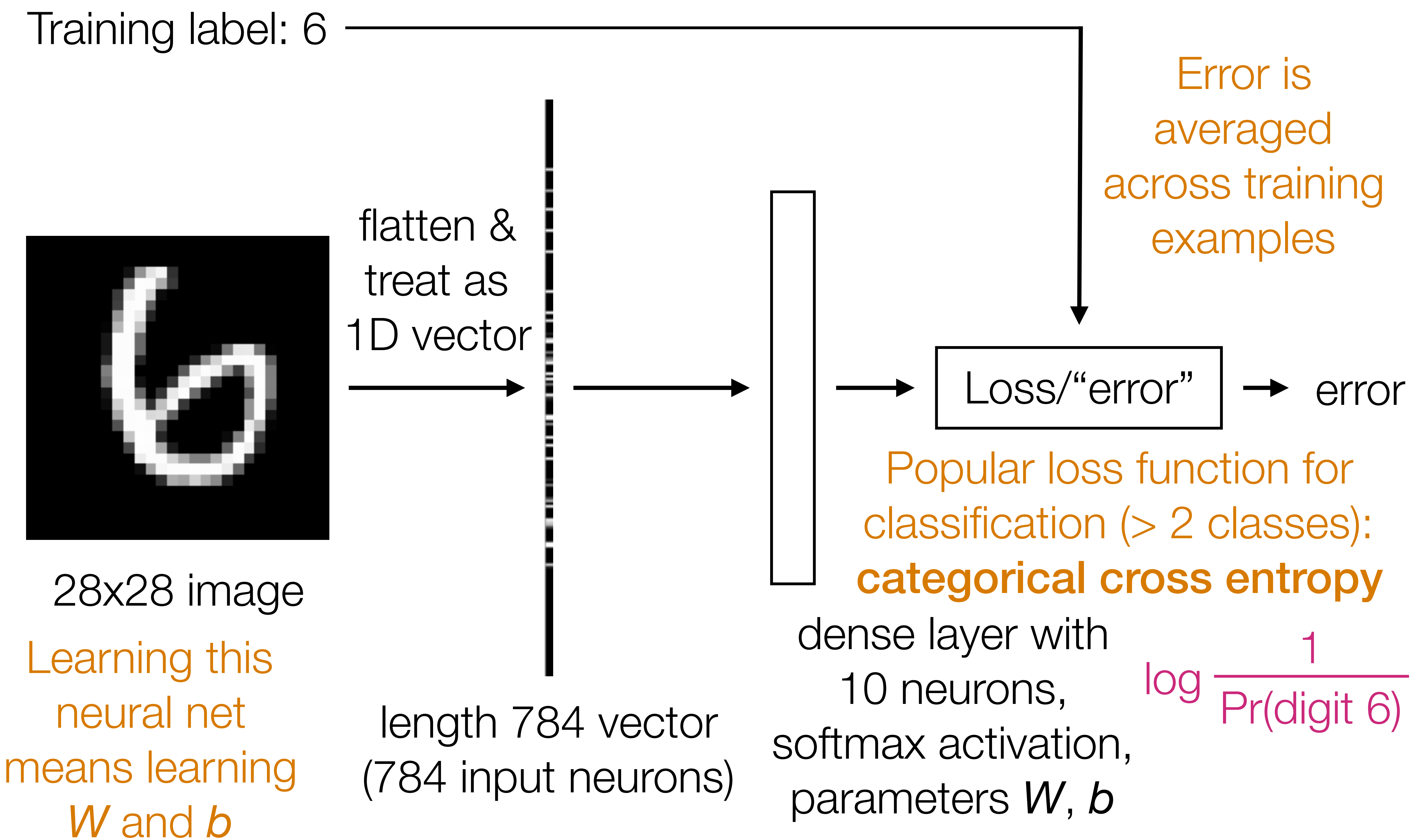
Handwritten Digit Recognition



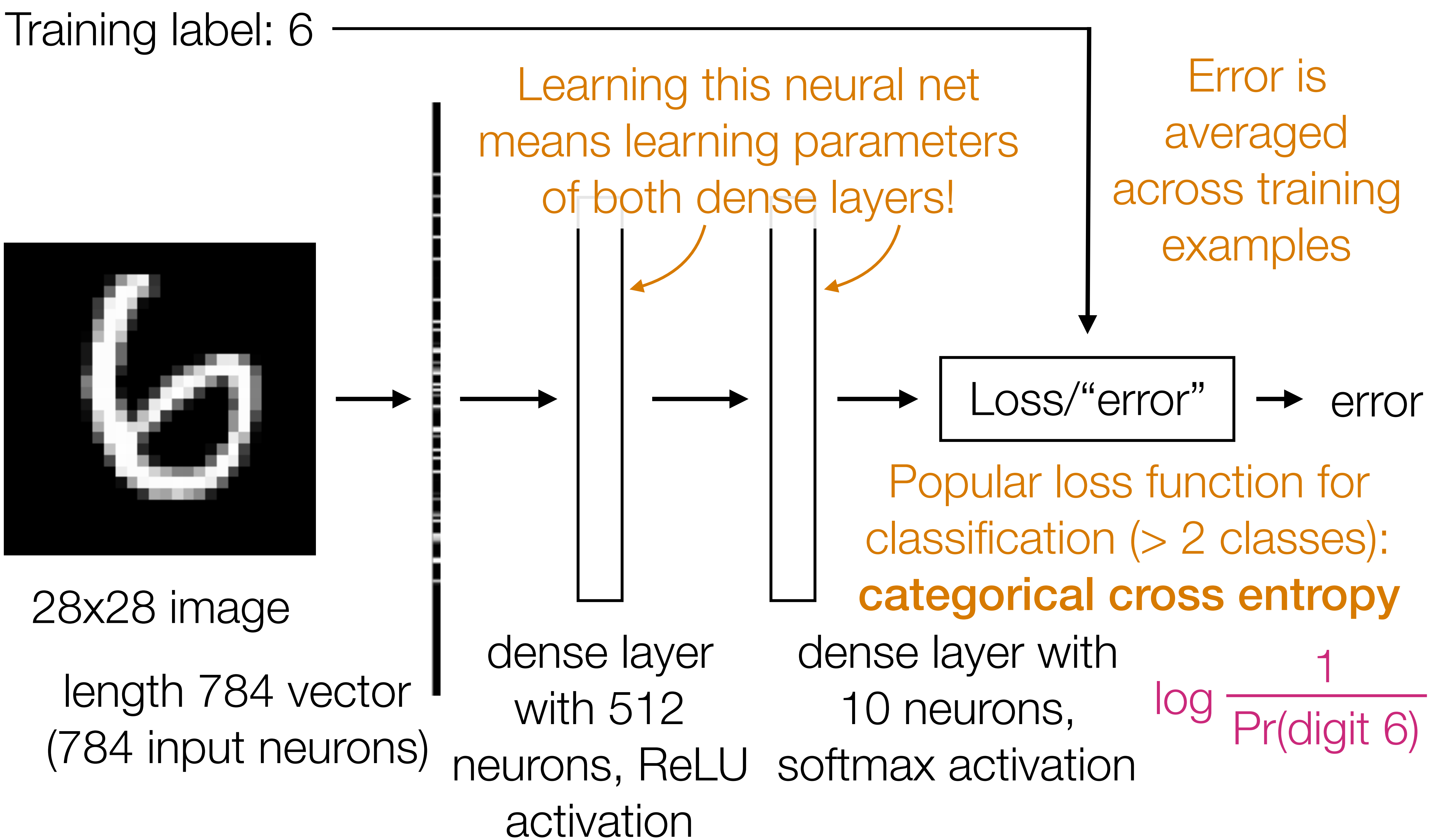
Handwritten Digit Recognition

Demo part 2

Handwritten Digit Recognition



Handwritten Digit Recognition



Handwritten Digit Recognition

Demo part 3