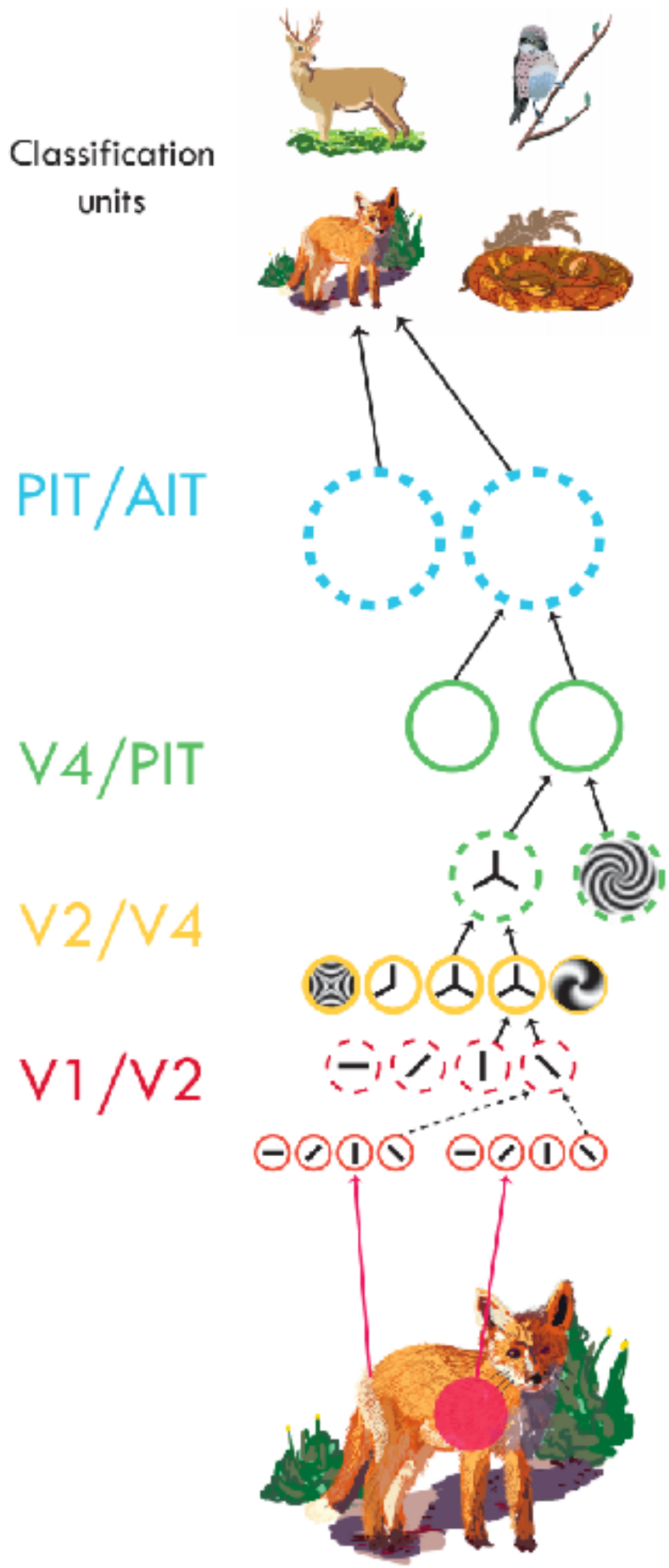
Carnegie Mellon University
Heinz College

Unstructured Data Analysis

Lecture 12: Intro to neural nets & deep learning

George Chen

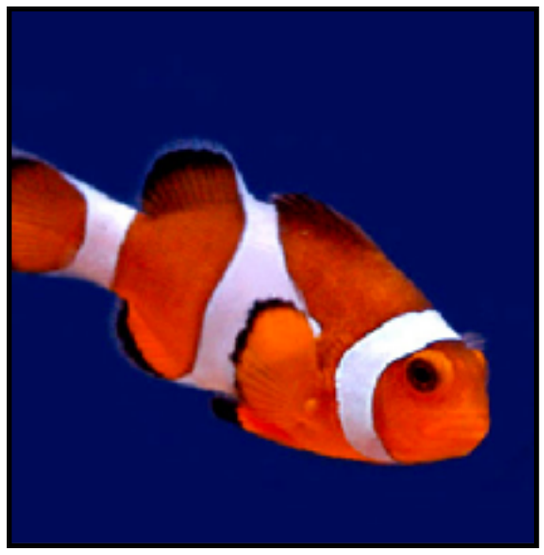
What is deep learning?



Slide by Phillip Isola

Serre, 2014

Basic Idea

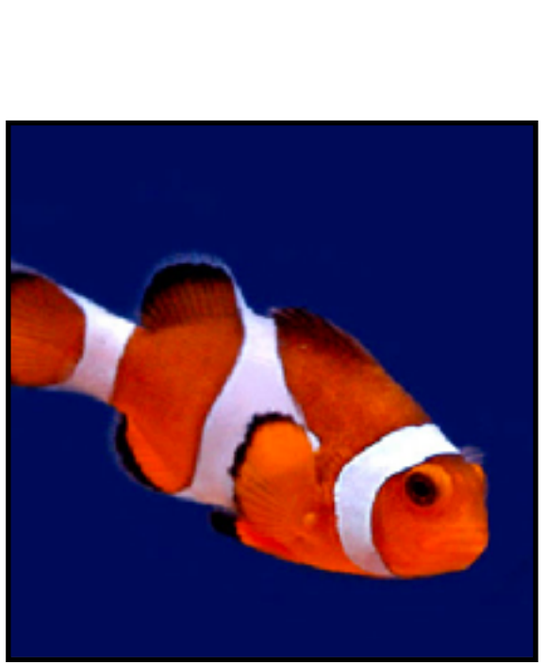


Brain/Machine

“clown fish”

Slide by Phillip Isola

Object Recognition



Slide by Phillip Isola

Object Recognition

Learned

Edges

Texture

Colors

Segments

Parts

“clown fish”

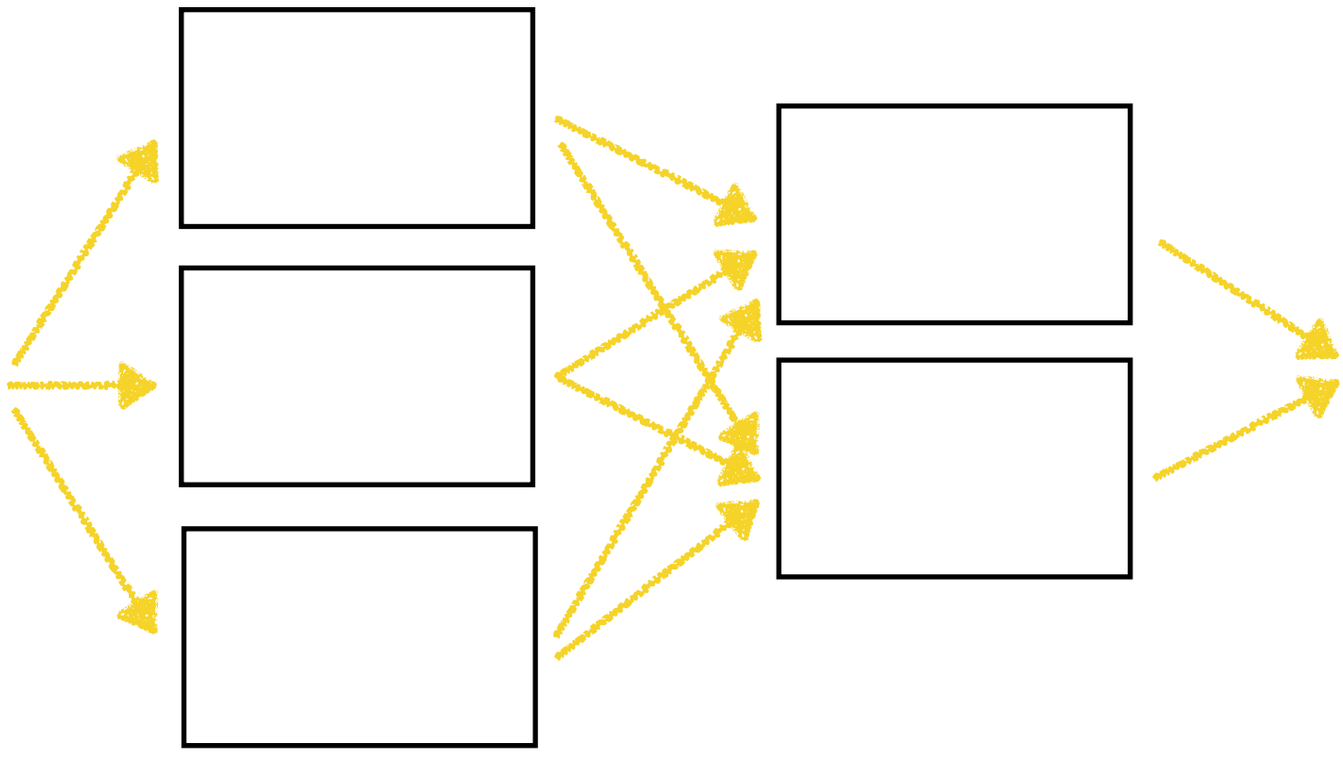
Feature extractors

Classifier

Slide by Phillip Isola

Neural Network

Learned

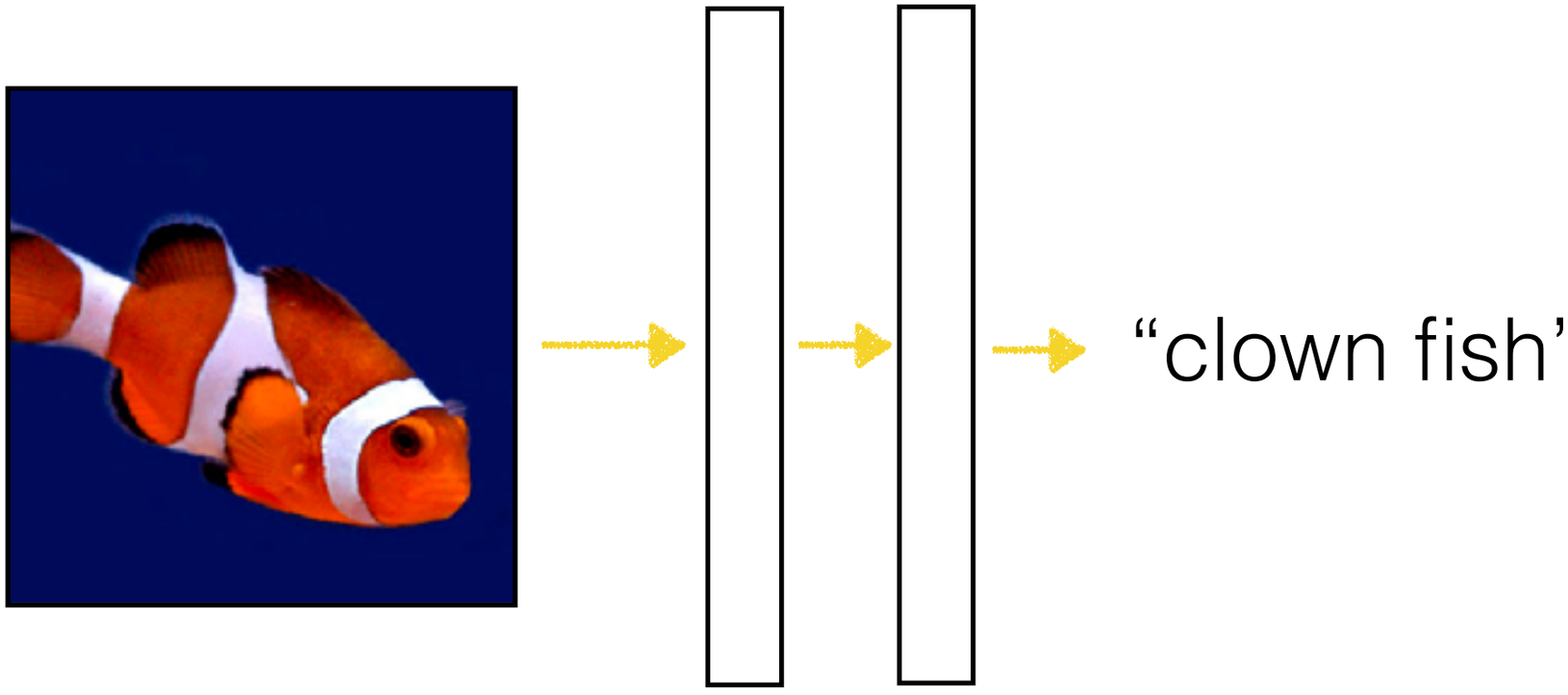


“clown fish”

Slide by Phillip Isola

Neural Network

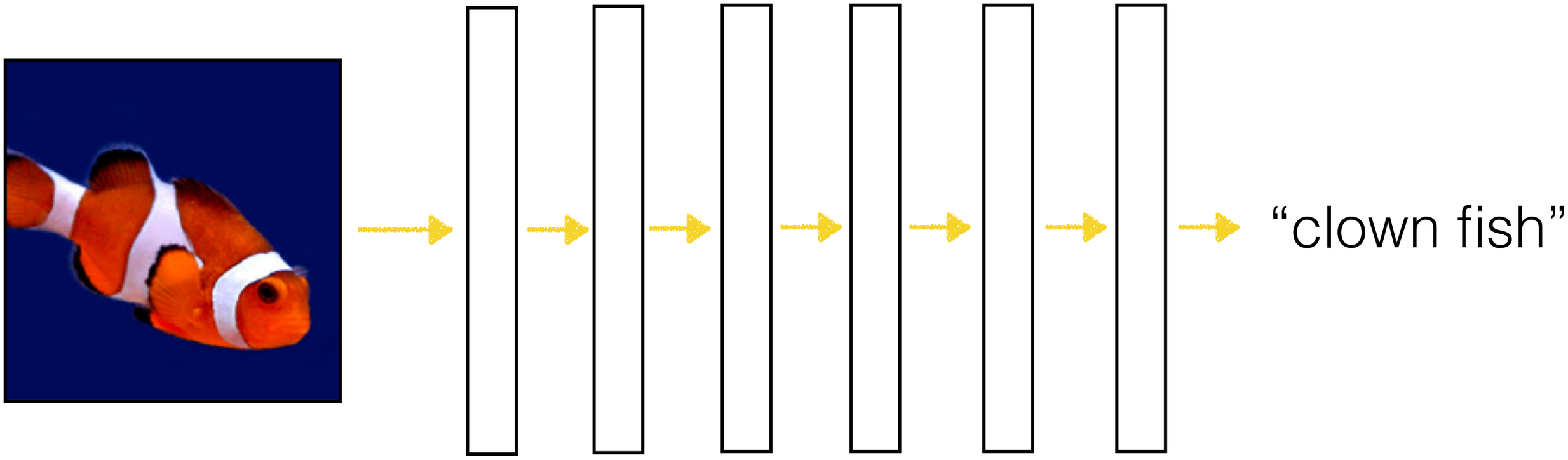
Learned



Slide by Phillip Isola

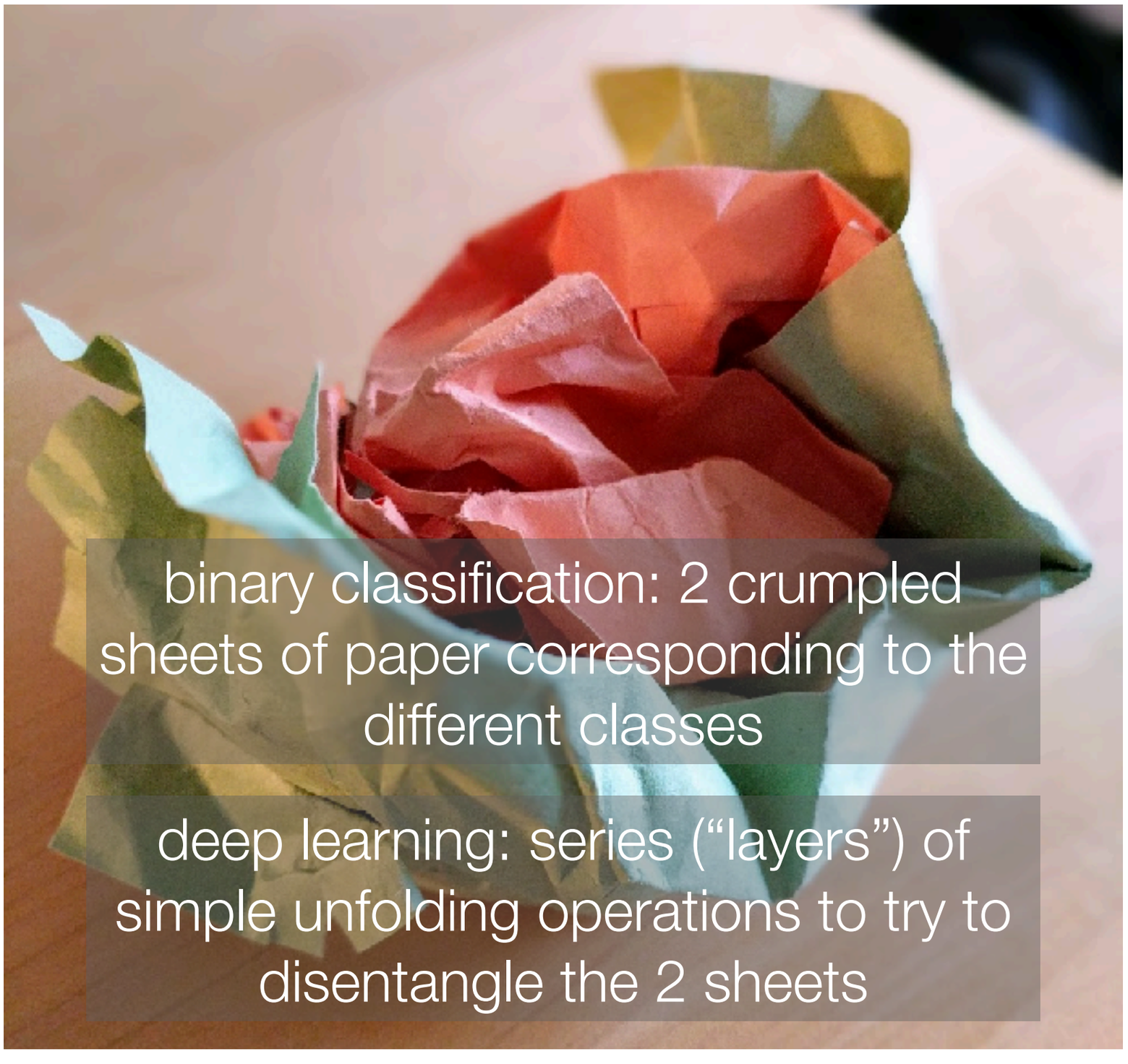
Deep Neural Network

Learned



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*

Learned

"clown fish"

Visualize (e.g., t-SNE)

Visualize

Visualize

Visualize

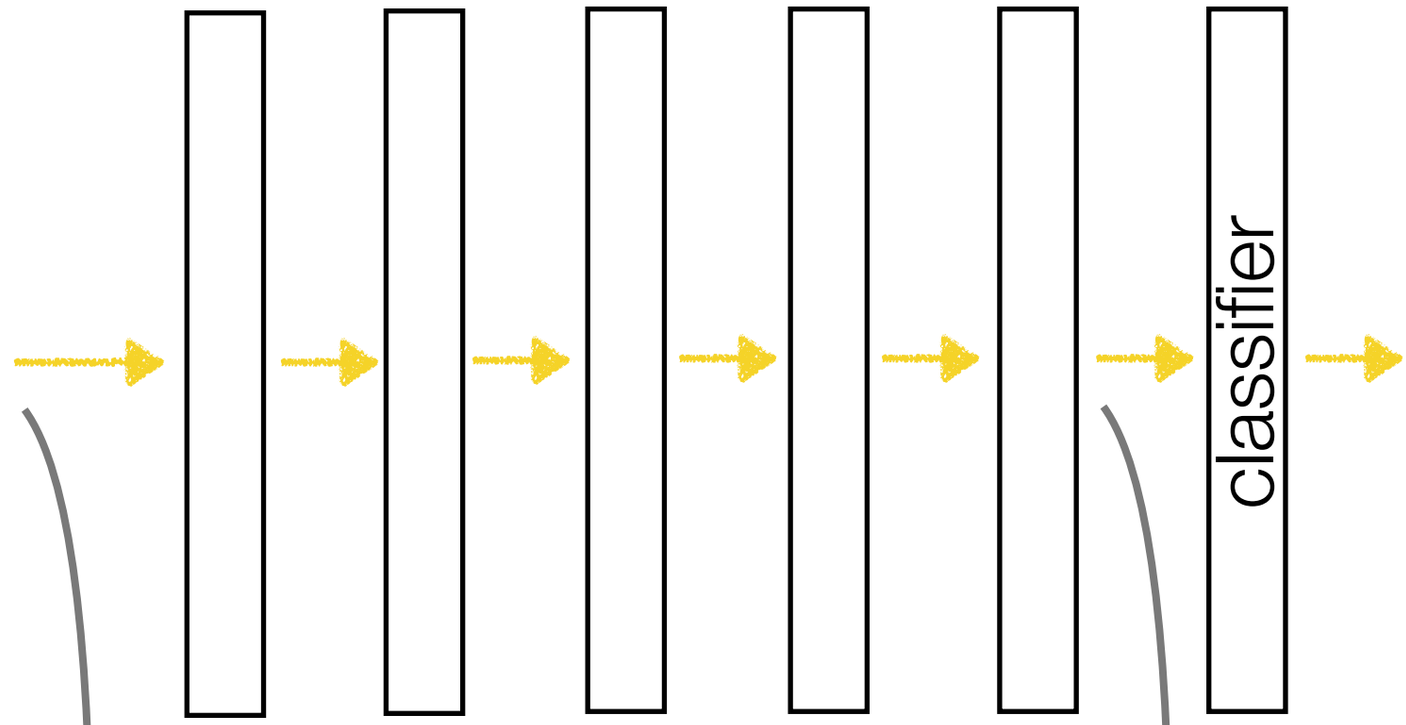
Visualize

Visualize

Visualize

Representation Learning

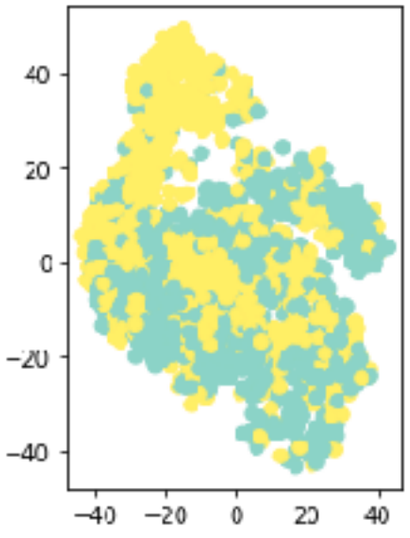
Each layer's output is *another way we could represent the input data*



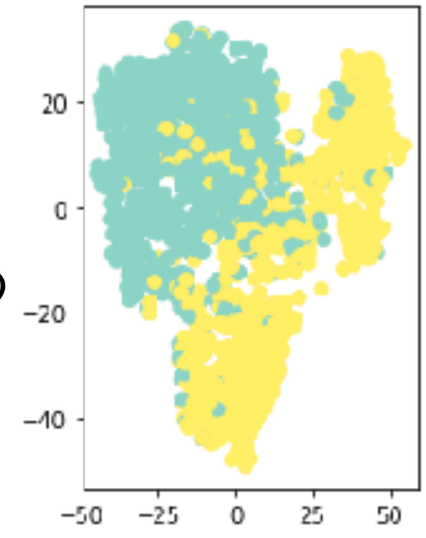
Learned

"clown fish"

Visualize (e.g., t-SNE)



Visualize



Why Does Deep Learning Work?

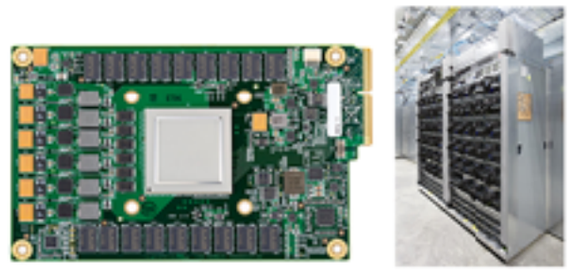
Actually the ideas behind deep learning are old (~1980's)

- 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 incorporates basic image processing structure

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

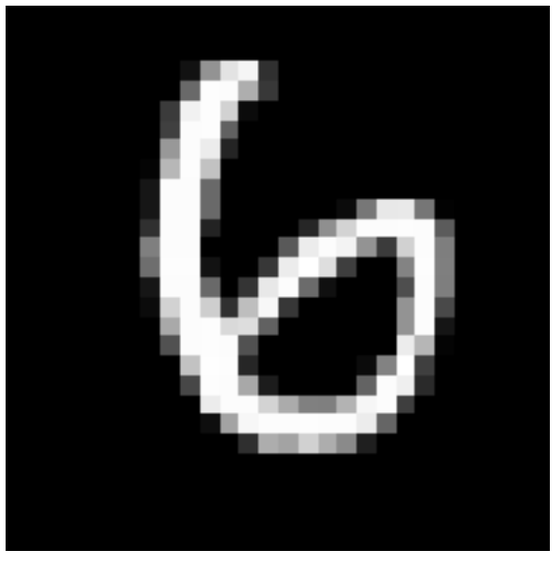
  - Note: text is a time series

  - Note: video is a time series

Handwritten Digit Recognition Example

Walkthrough of 2 extremely simple neural nets

Handwritten Digit Recognition



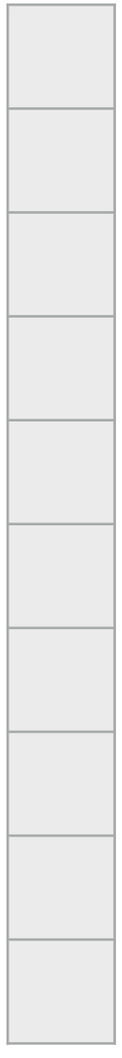
28x28 image

flatten

length 784 vector
(784 input nodes)

weighted sums

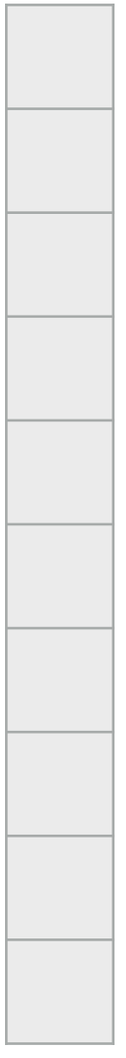
(parameterized by a weight matrix ***W*** and a bias ***b***)



linear layer
with 10 nodes

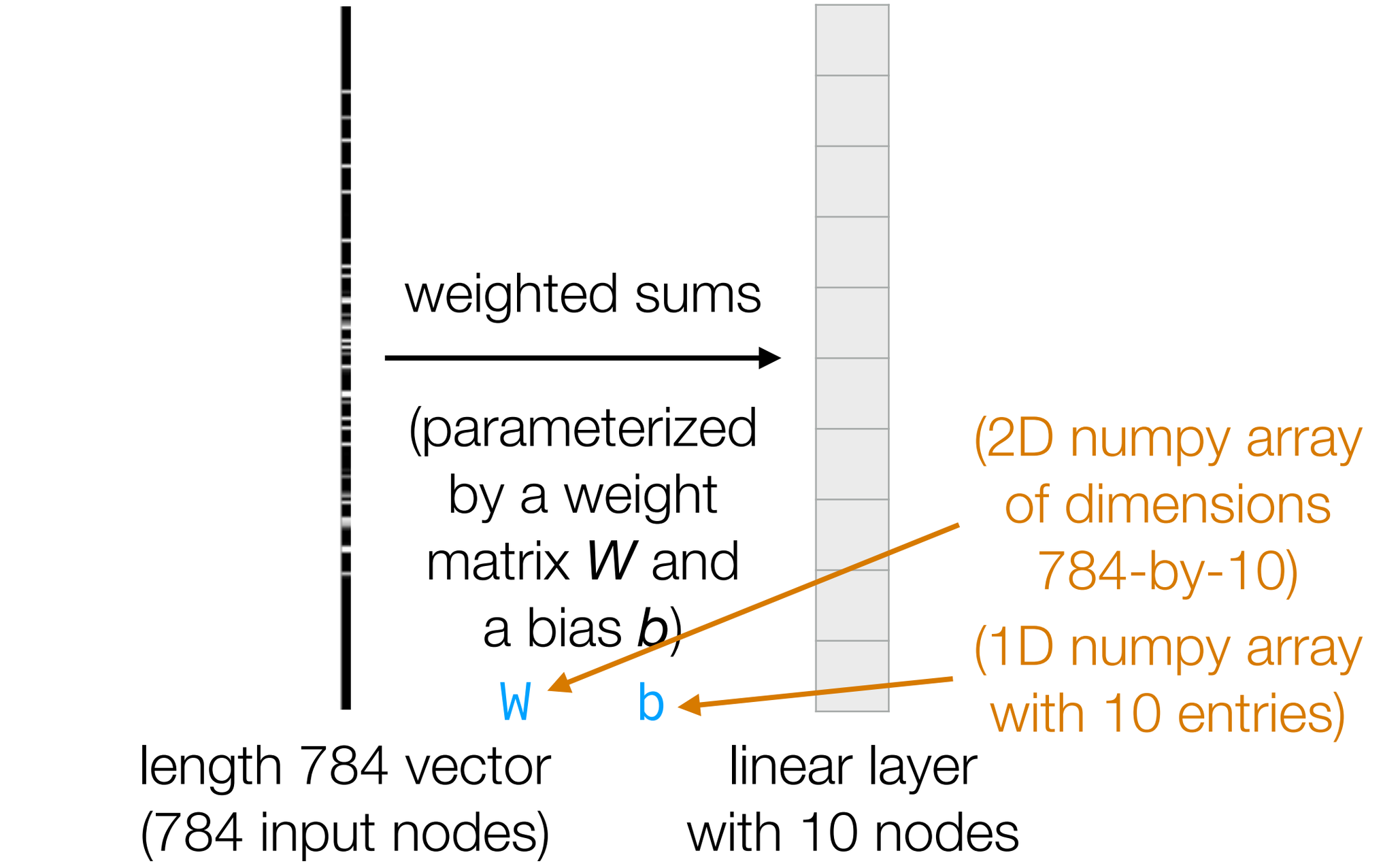
activation

(can be thought of as post-processing)

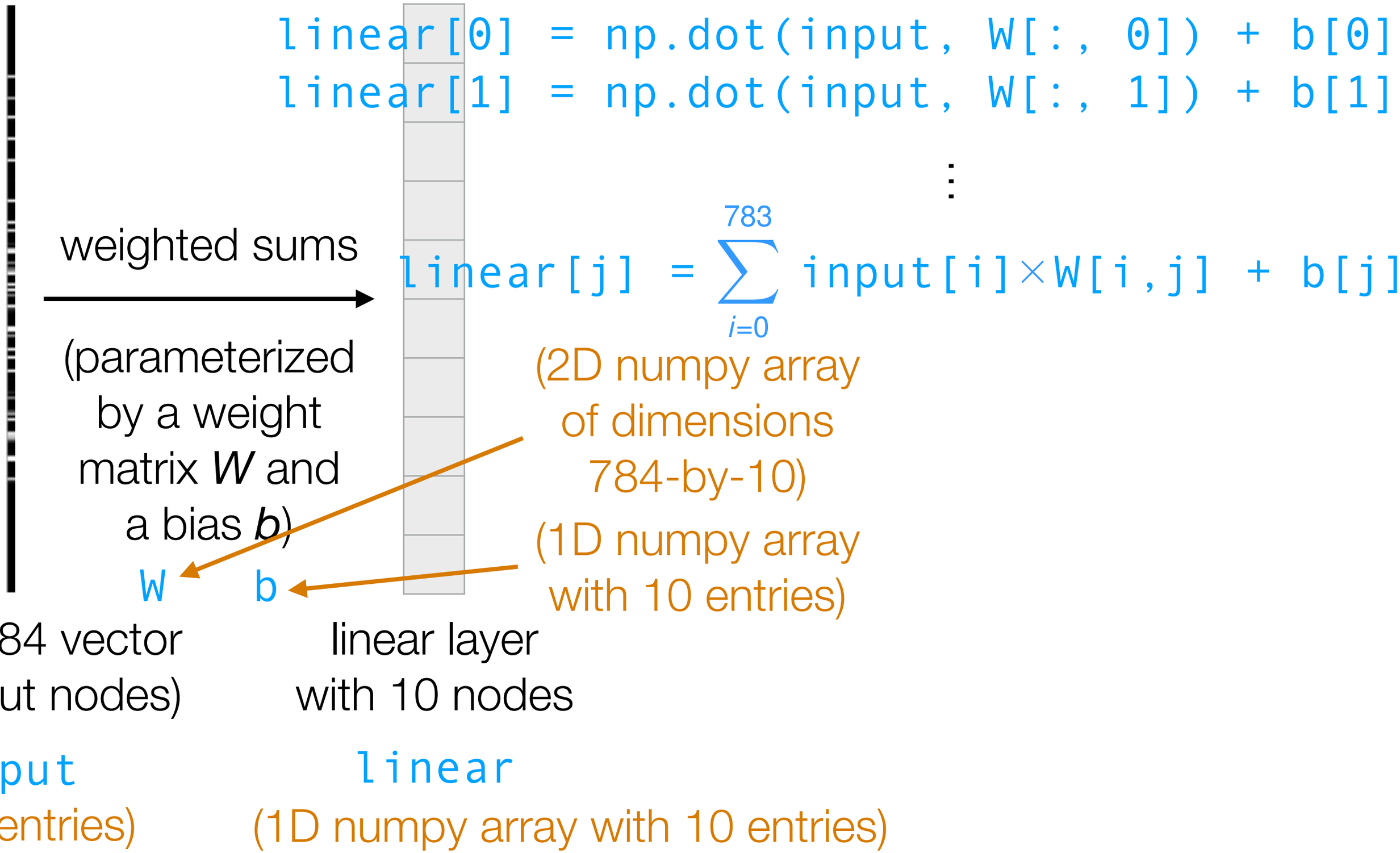


final
output

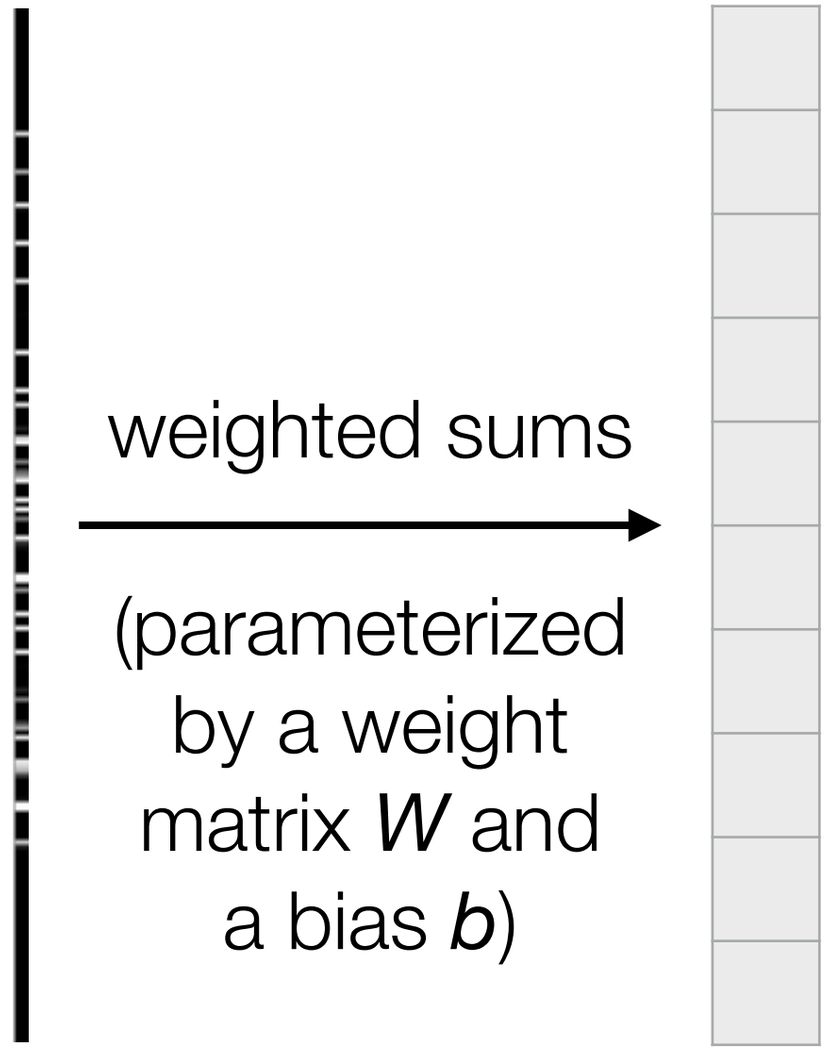
Handwritten Digit Recognition



Handwritten Digit Recognition



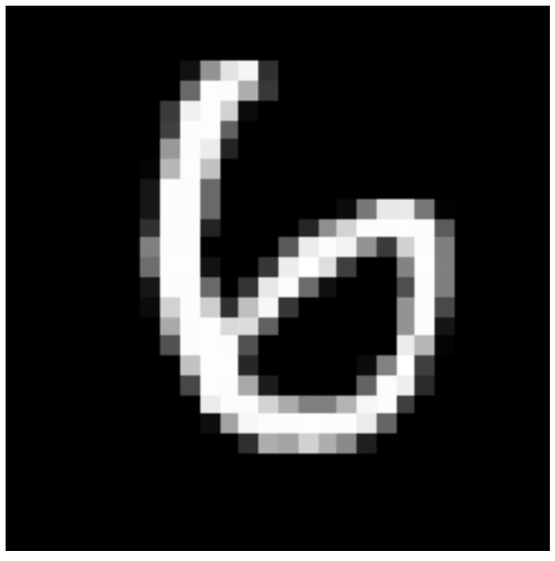
Handwritten Digit Recognition



length 784 vector
(784 input nodes)

linear layer
with 10 nodes

Handwritten Digit Recognition



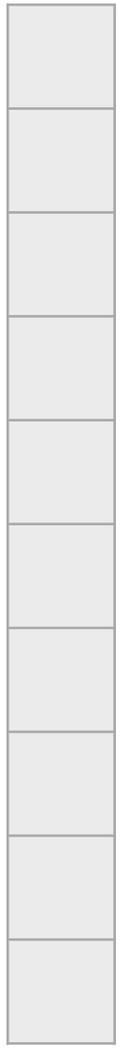
28x28 image

flatten

length 784 vector
(784 input nodes)

weighted sums

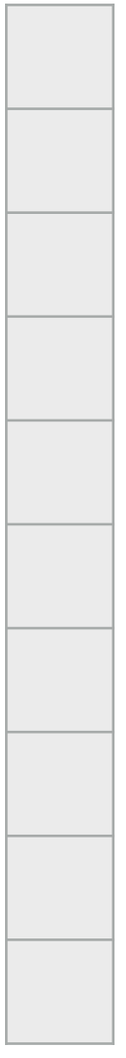
(parameterized by a weight matrix ***W*** and a bias ***b***)



linear layer
with 10 nodes

activation

(can be thought of as post-processing)



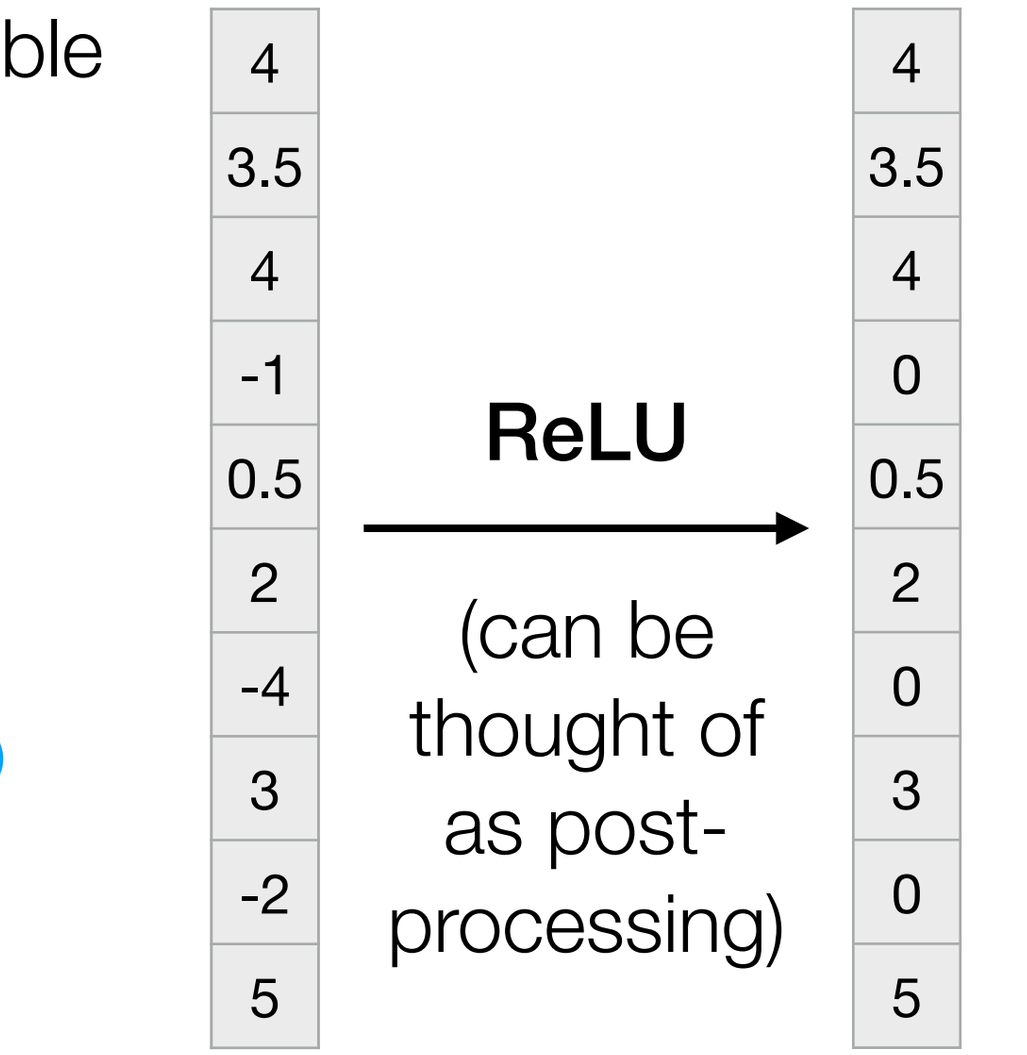
final
output

Handwritten Digit Recognition

Many different activation functions possible

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

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

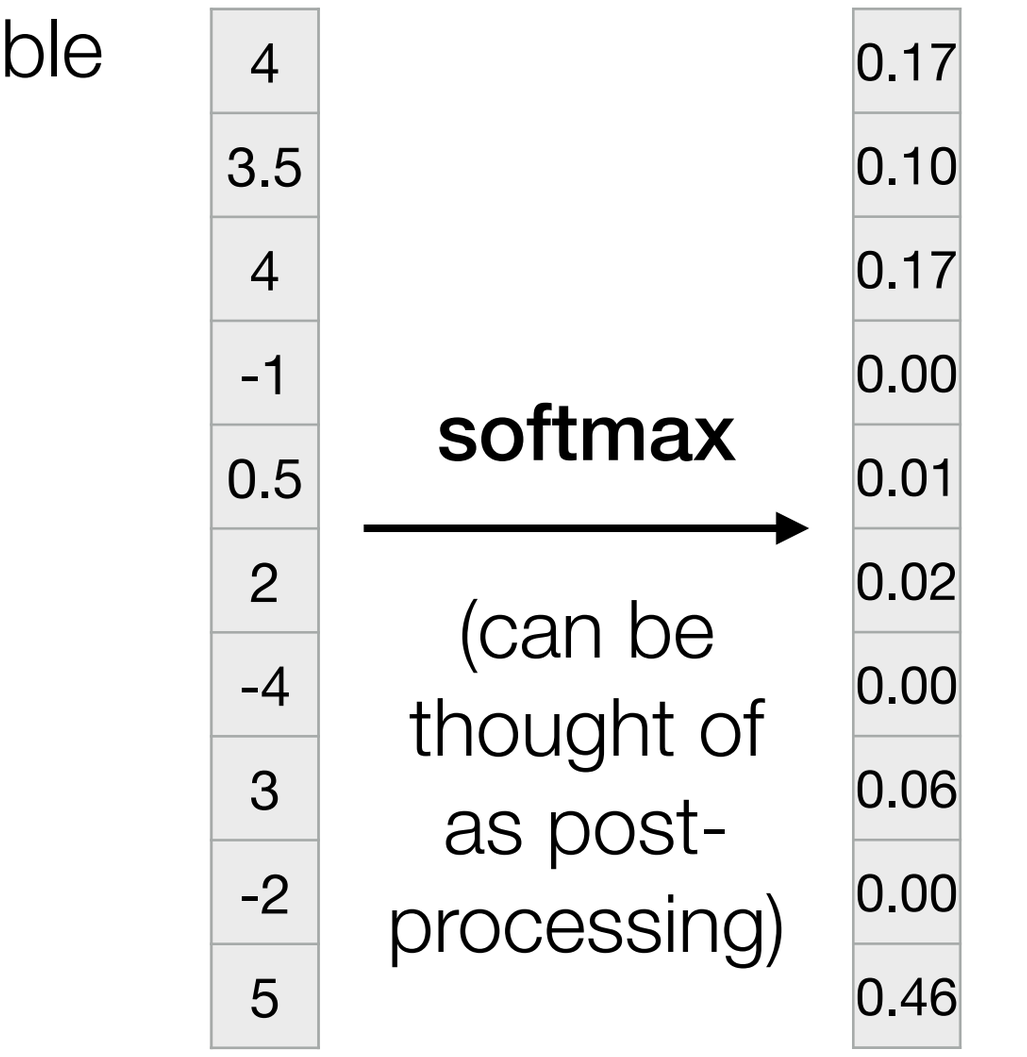


Handwritten Digit Recognition

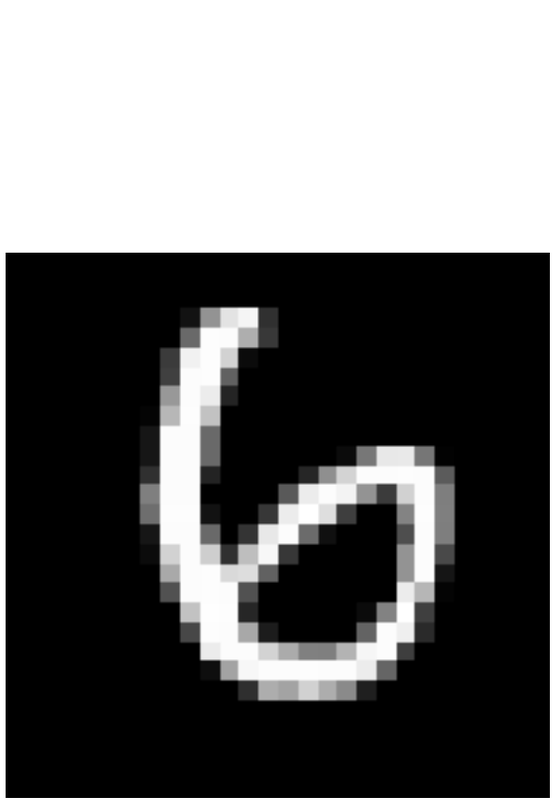
Many different activation functions possible

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

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



Handwritten Digit Recognition



28x28 image

flatten

length 784 vector
(784 input nodes)

weighted sums

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

linear layer
with 10 nodes

softmax

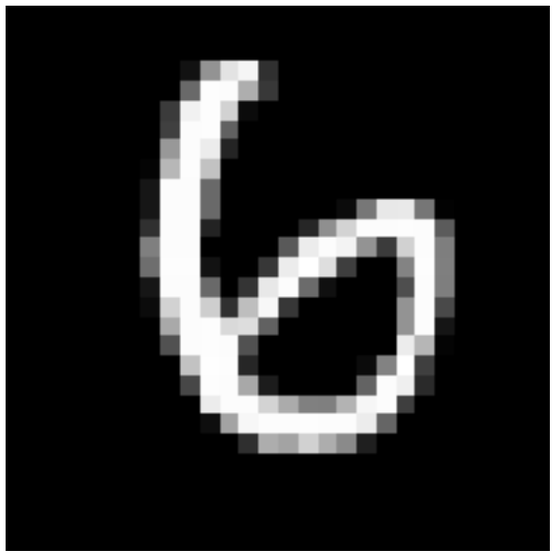
Pr(digit 0)
Pr(digit 1)
Pr(digit 2)
Pr(digit 3)
Pr(digit 4)
Pr(digit 5)
Pr(digit 6)
Pr(digit 7)
Pr(digit 8)
Pr(digit 9)

final output

Desired result

Handwritten Digit Recognition

Training label: 6



Input

Flatten → Linear (10 nodes) → Softmax → Loss/“error” → error

Error is averaged across training examples

Popular loss function for classification (> 2 classes): **categorical cross entropy**

$$\log \frac{1}{\text{estimated Pr(digit 6)}}$$

Learning this neural net means learning W and b

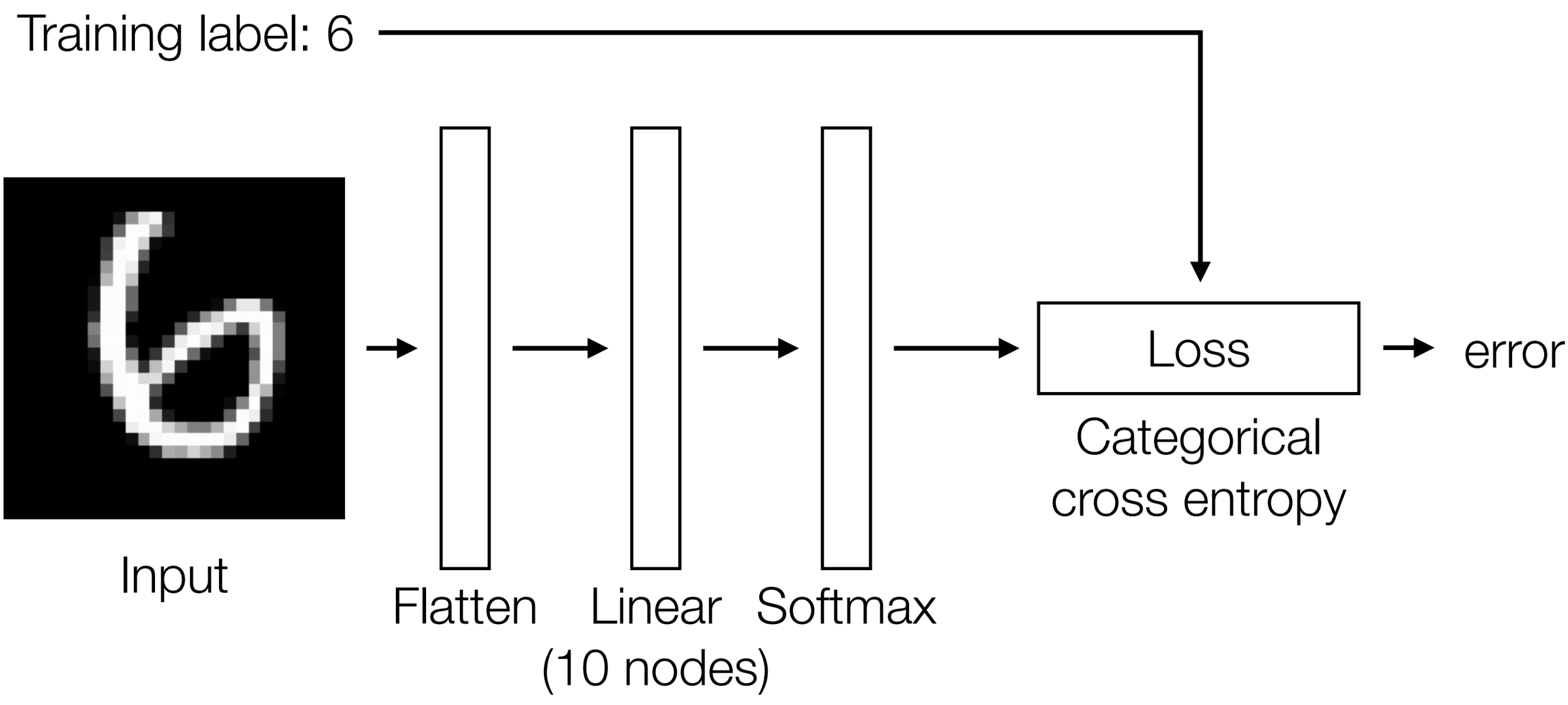
Also called *fully-connected* or *dense* layer

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

Handwritten Digit Recognition

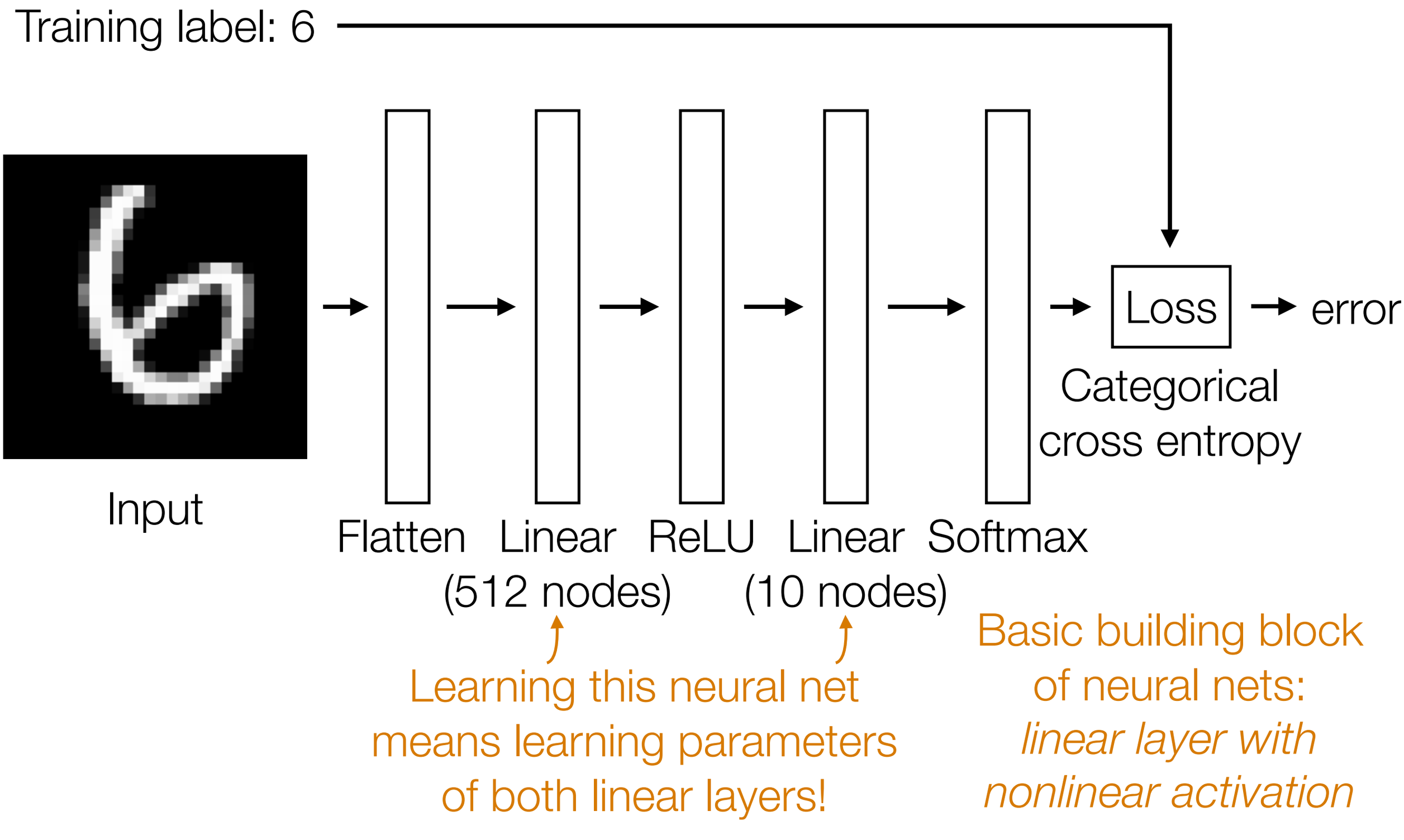
Demo part 1

Handwritten Digit Recognition

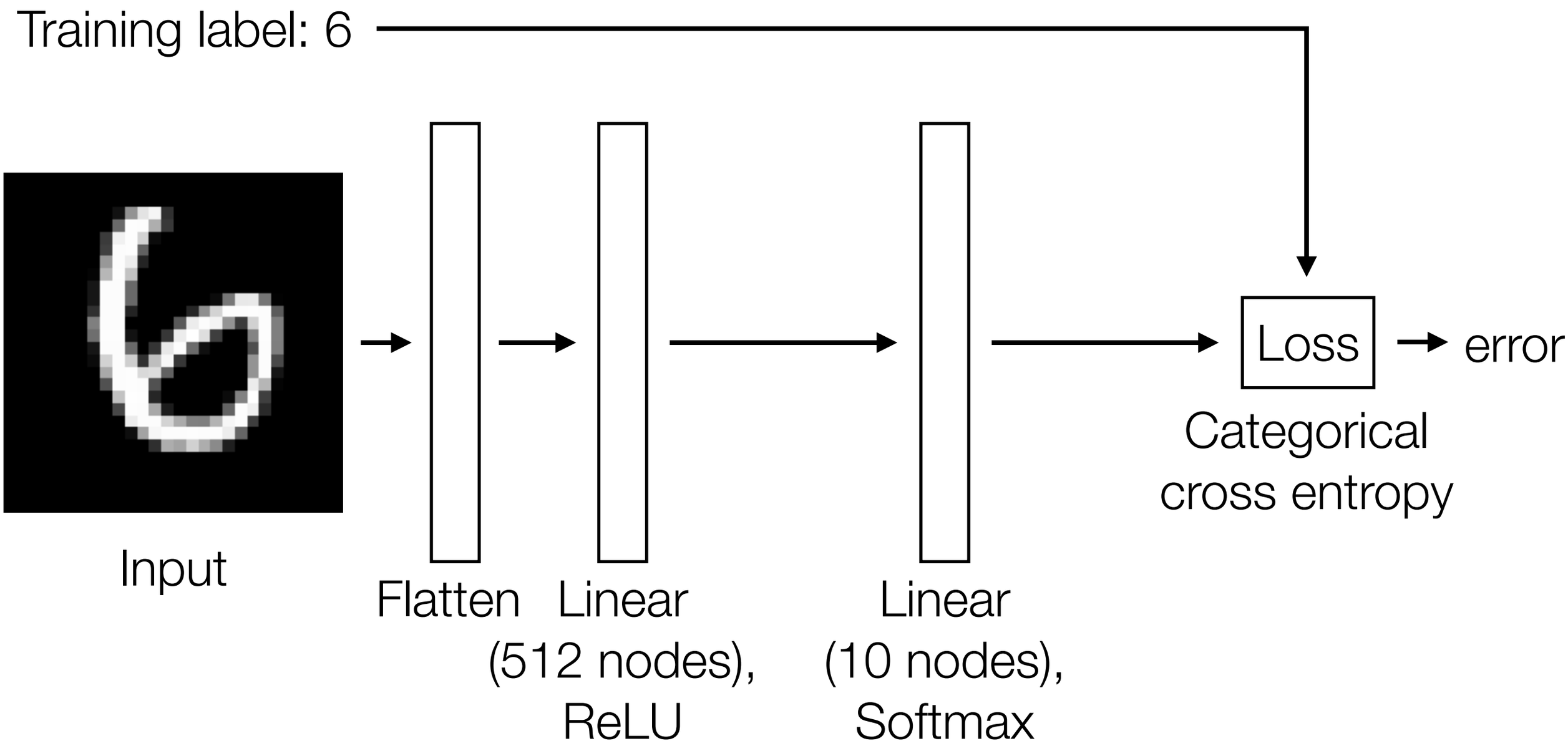


This neural net has a name: **multinomial logistic regression**
(when there are only 2 classes, it's called **logistic regression**)

Handwritten Digit Recognition



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 will some times use this notation instead

Handwritten Digit Recognition

Demo part 2

Architecting Neural Nets

- Basic building block that is often repeated:
  linear layer followed by nonlinear activation
  - Without nonlinear activation, two consecutive linear layers is mathematically equivalent to having a single linear layer!
- How to select # of nodes in a layer, or # of layers?
  - These are hyperparameters! Infinite possibilities!
  - Can choose between different options using hyperparameter selection strategy from earlier lectures
    - Very expensive in practice!
      (Active area of research: neural architecture search)
  - Much more common in practice: modify existing architectures that are known to work well
    (e.g., ResNet for image classification/object recognition)

PyTorch GitHub Has Lots of Examples

github.com/pytorch/examples

PyTorch Examples

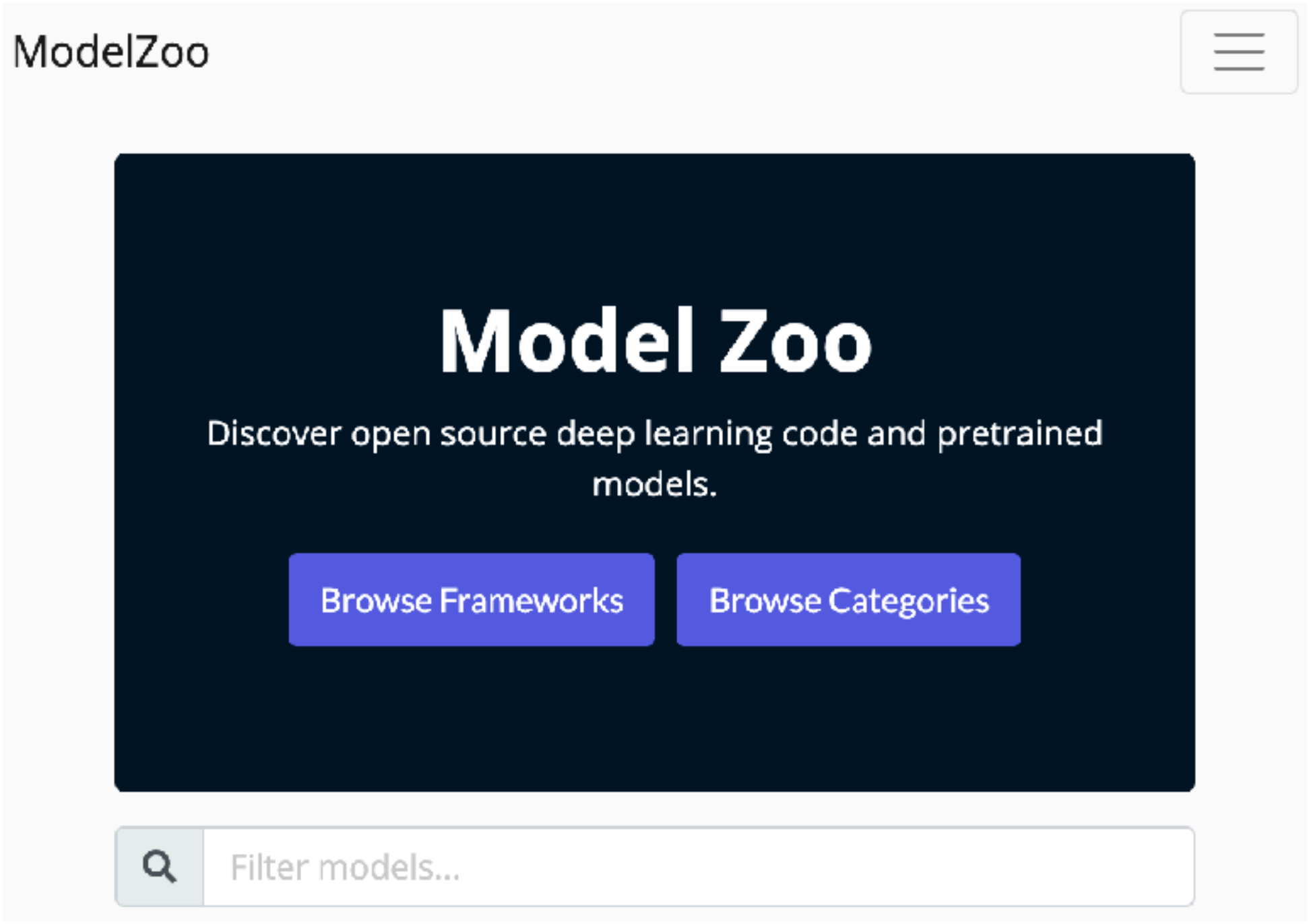
A repository showcasing examples of using PyTorch

- Image classification (MNIST) using Convnets
- Word level Language Modeling using LSTM RNNs
- Training Imagenet Classifiers with Residual Networks
- Generative Adversarial Networks (DCGAN)
- Variational Auto-Encoders
- Superresolution using an efficient sub-pixel convolutional neural network
- Hogwild training of shared ConvNets across multiple processes on MNIST
- Training a CartPole to balance in OpenAI Gym with actor-critic
- Natural Language Inference (SNLI) with GloVe vectors, LSTMs, and torchtext
- Time sequence prediction - use an LSTM to learn Sine waves
- Implement the Neural Style Transfer algorithm on images
- Several examples illustrating the C++ Frontend

Additionally, a list of good examples hosted in their own repositories:

- Neural Machine Translation using sequence-to-sequence RNN with attention (OpenNMT)

Find a Massive Collection of Models at the Model Zoo



Learning a neural net amounts to curve fitting

We’re just estimating a function

Neural Net as Function Approximation

Given `input`, learn a computer program that computes `output`

this is a **function**

Multinomial logistic regression:

```
def f(input):
    output = softmax(np.dot(input, W) + b)
    return output
```

the only things that we are learning
(we fix their dimensions in advance)

We are fixing what the function `f` looks like in code
and are only adjusting `W` and `b`!!!

Neural Net as Function Approximation

Given `input`, learn a computer program that computes `output`

Multinomial logistic regression:

```
output = softmax(np.dot(input, W) + b)
```

Multilayer perceptron:

```
intermediate = relu(np.dot(input, W1) + b1)

output = softmax(np.dot(intermediate, W2) + b2)
```

Learning a neural net: learning a simple computer program that maps inputs (raw feature vectors) to outputs (predictions)

Complexity of a Neural Net?

- Increasing number of layers (depth) makes neural net more "complex"
  - Learn computer program that has more lines of code
  - Some times, more parameters may be needed
  - If so, more training data may be needed

Earlier: multinomial logistic regression had fewer parameters than multilayer perceptron example

Upcoming: we'll see examples of deep nets with *fewer* parameters than "shallower" nets