



Deep Learning Workshop

Pepe Bonet Giner

17th October 2022

Index

- Topic 1: What is Deep Learning?
- Topic 2: The Building blocks of Neural Networks?
- Topic 3: How do Neural Networks learn?
- Topic 4: Introduction to Keras. The MNIST dataset
- Topic 5: Types of Neural Networks and their applications.
- Topic 6: How Deep Learning is changing /will change the world?

Who am I?



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Maastricht
University



B.Sc. Biotechnology

M.Sc.
Computational Biology



Universitat
Pompeu Fabra
Barcelona

ETH zürich



P.hD. @ BbgLab
Computational Biology
EMIBA



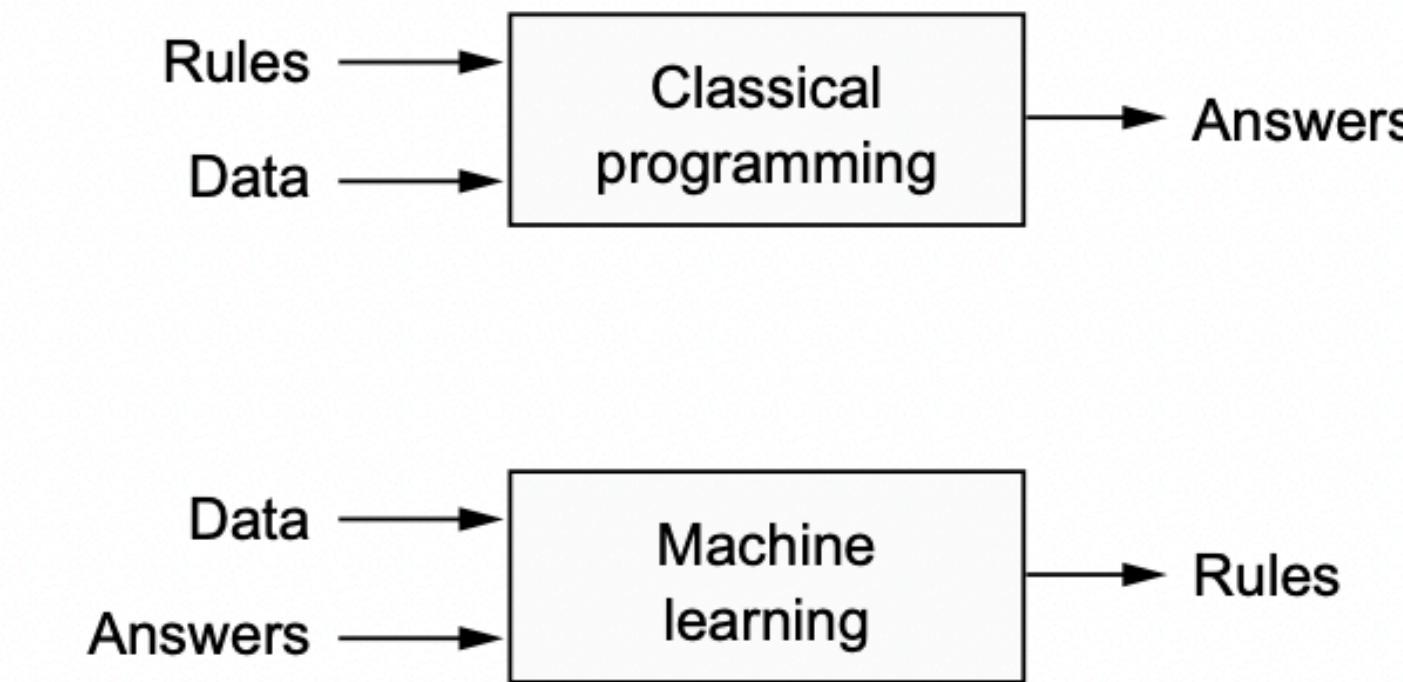
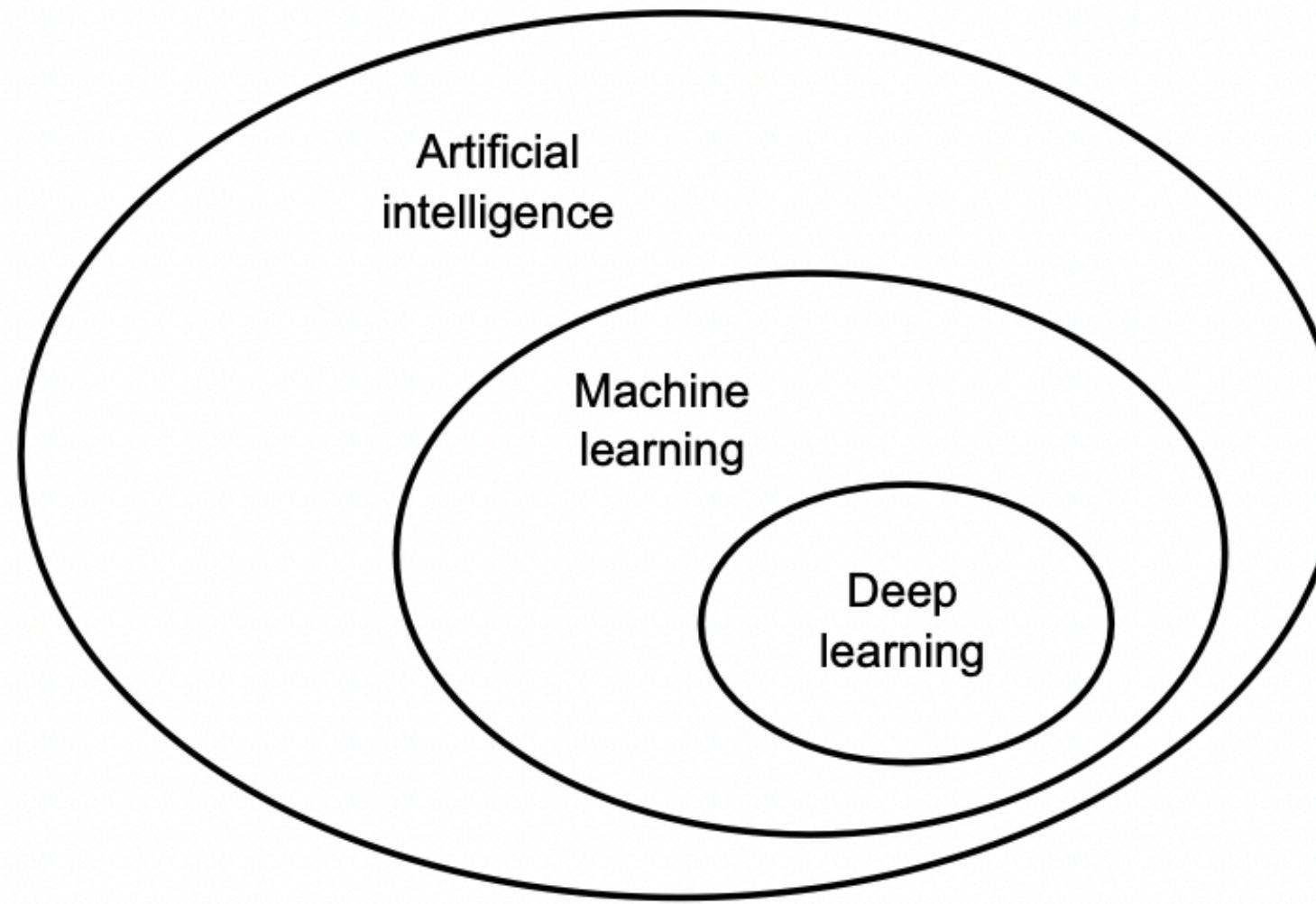
hynts

Hynts Analytics



What is Deep Learning?

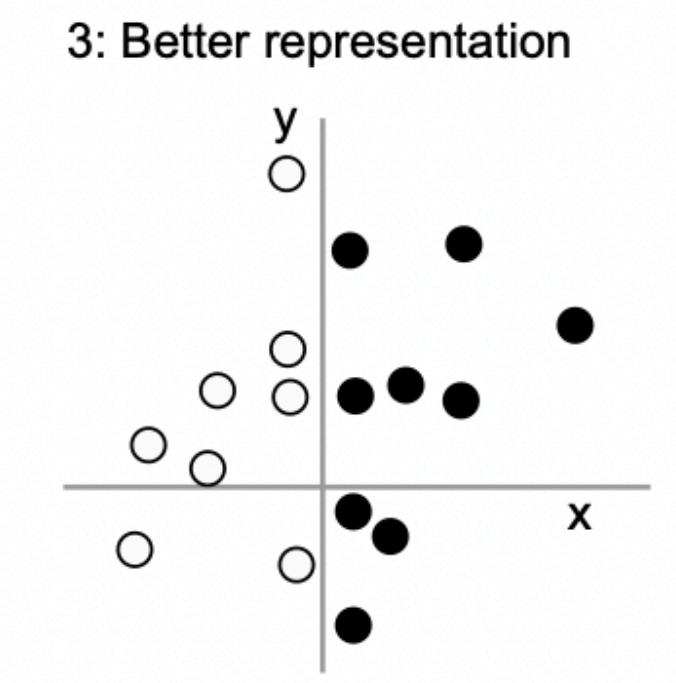
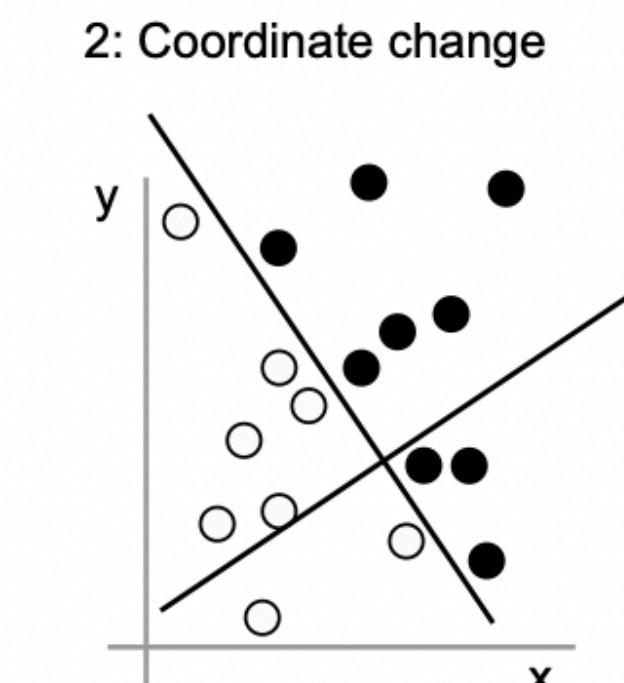
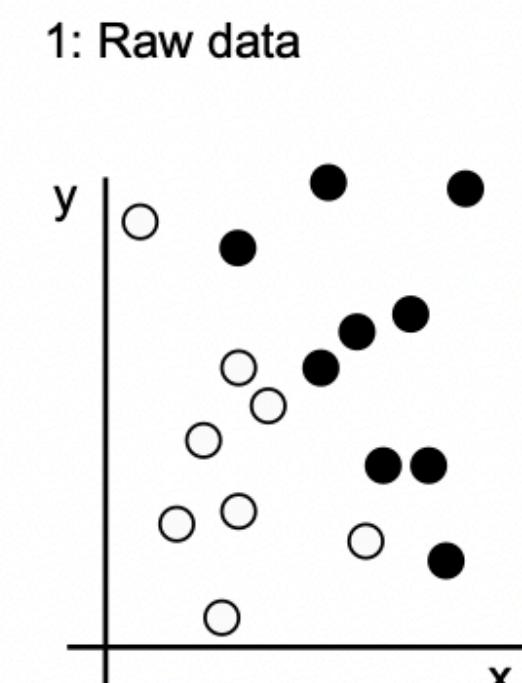
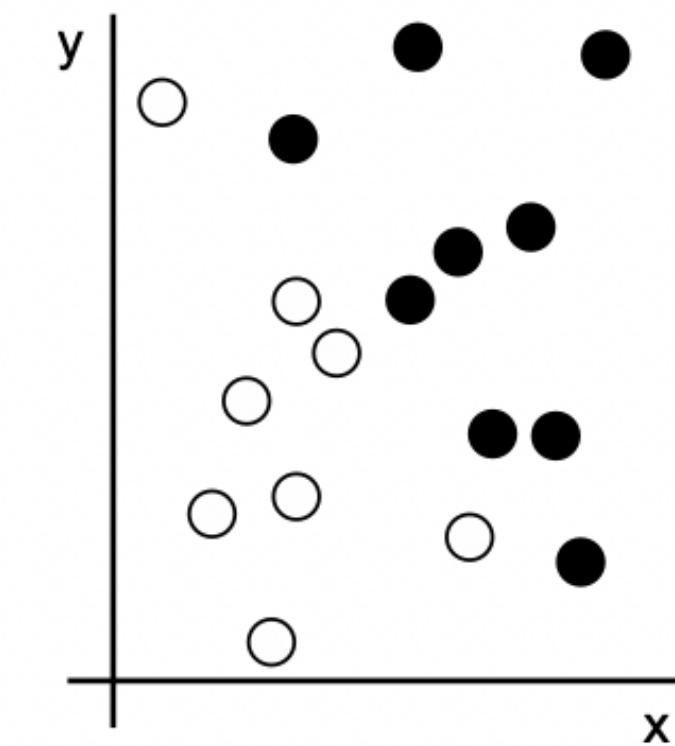
Differences between symbolic AI and Machine Learning



- AI: Effort to automate intellectual tasks normally performed by humans
- ~ 1950-1980 Symbolic AI. Rules + Data → Answers
- ML: Data + Answers → Rules

ML Recap

- ML: Data + Answers → Rules
- What do we need:
 1. Input data points
 2. Examples of the expected output
 3. A way to measure whether the algorithm is doing a good job
- Transform data into meaningful outputs
(Learning from exposure to known examples)
- Meaningfully transform data: learn representations



Limitations of ML. What makes DL different?

Easier Problem-solving

Automation of feature engineering

3	4	2	1	9	5	6	2	1	8
8	9	1	2	5	0	0	6	6	4
6	7	0	1	6	3	6	3	7	0
3	7	7	9	4	6	6	1	8	2
2	9	3	4	3	9	8	7	2	5
1	5	9	8	3	6	5	7	2	3
9	3	1	9	1	5	8	0	8	4
5	6	2	6	8	5	8	8	9	9
3	7	7	0	9	4	8	5	4	3
7	9	6	4	7	0	6	9	2	3

SVM,
Decision Trees...

Neural Nets

Limitations of ML. What makes DL different?

Easier Problem-solving

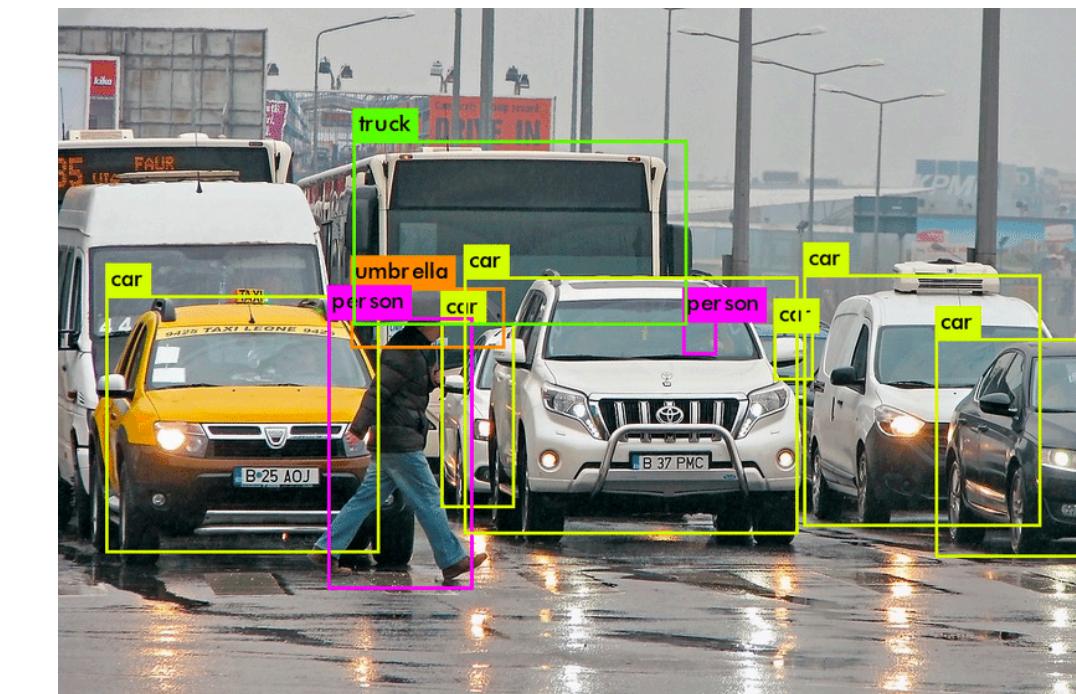
Automation of feature engineering



SVM,
Decision Trees...

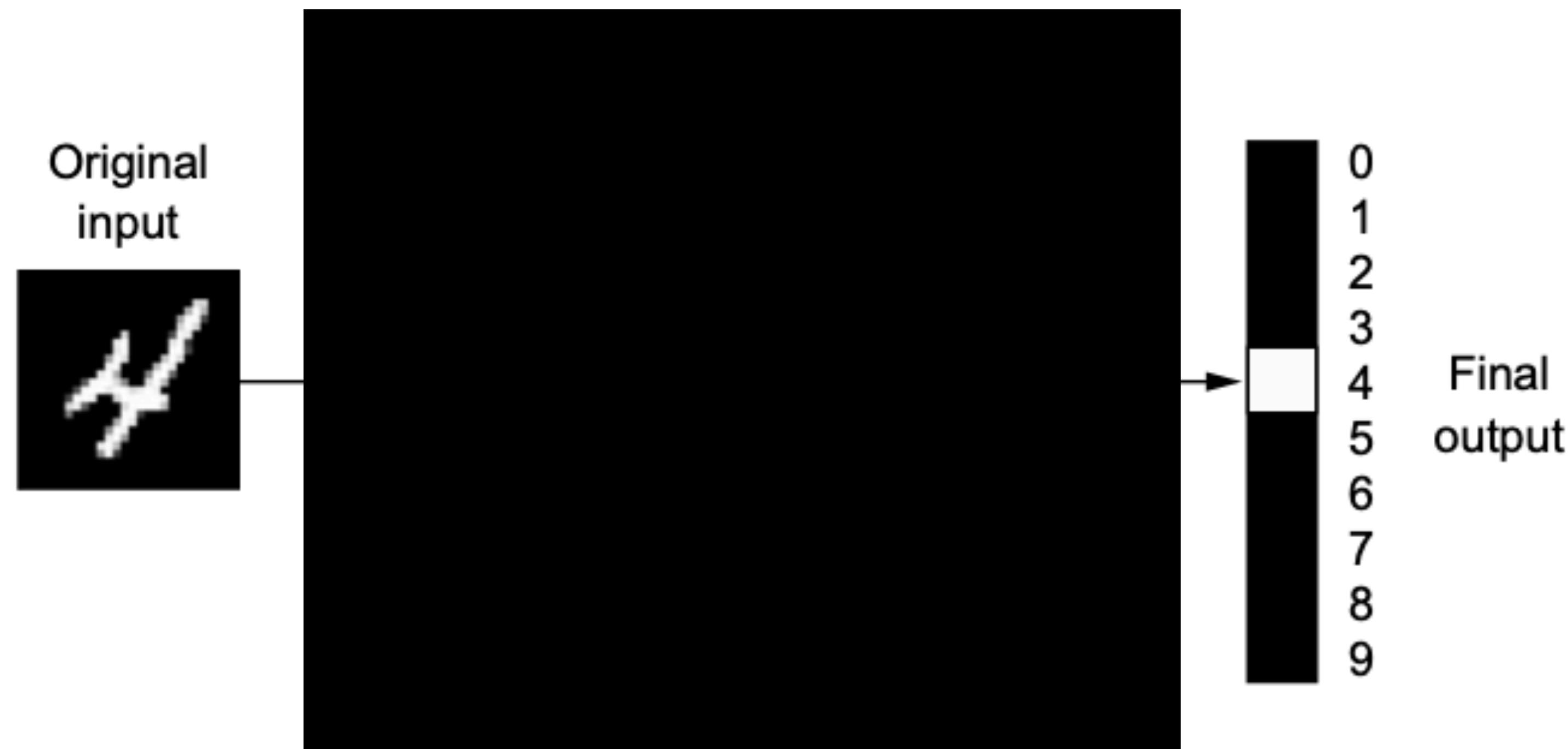
Neural Nets

Better performance for many problems



A first glance at Deep Learning

Neural Networks

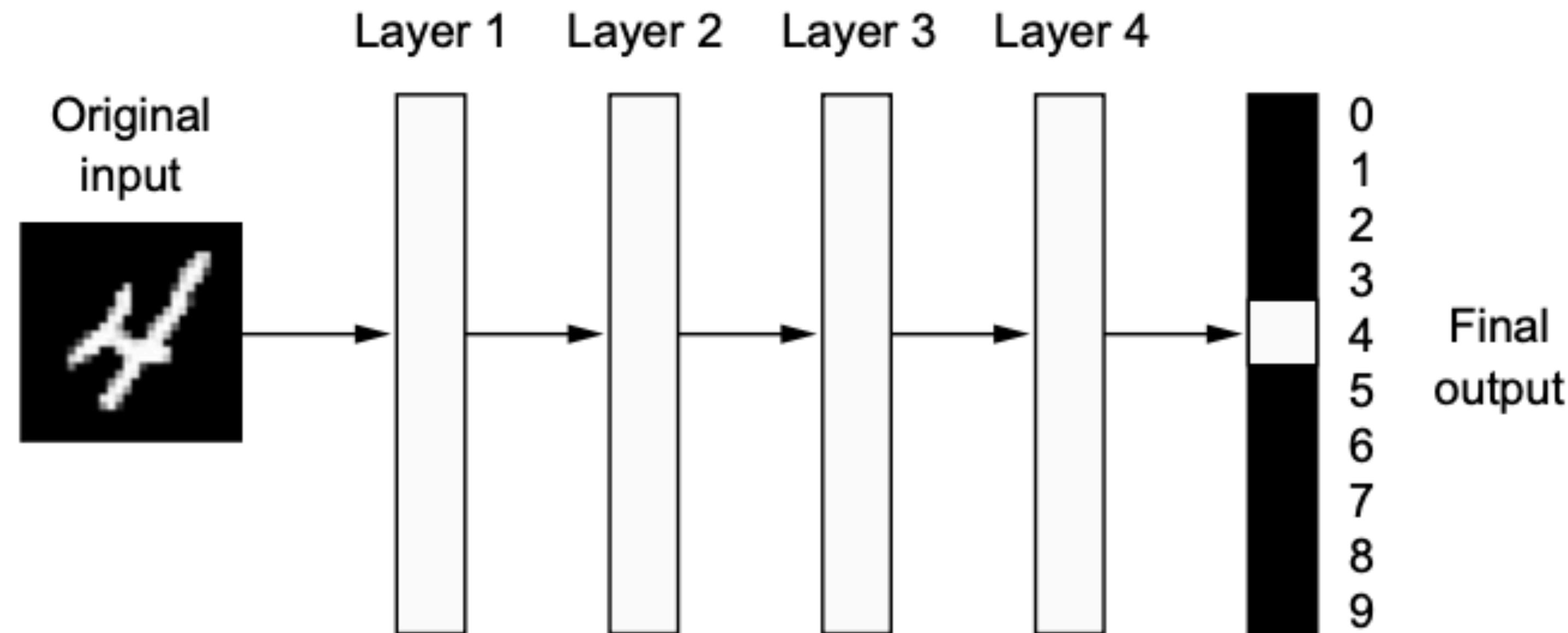


A first glance at Deep Learning

Neural Networks

Hidden Layers

Deep (Depth):
Successive Layers

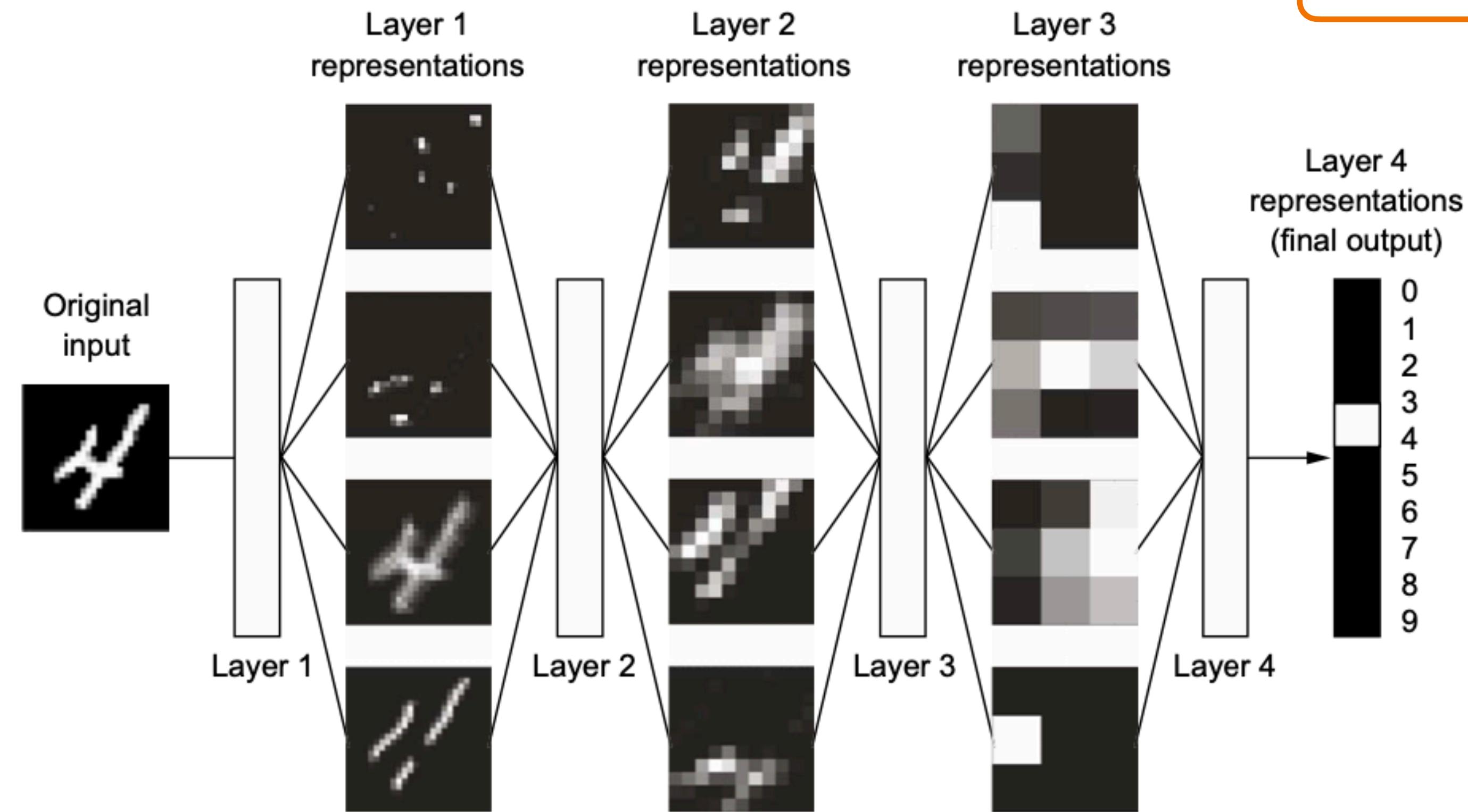


A first glance at Deep Learning

Neural Networks

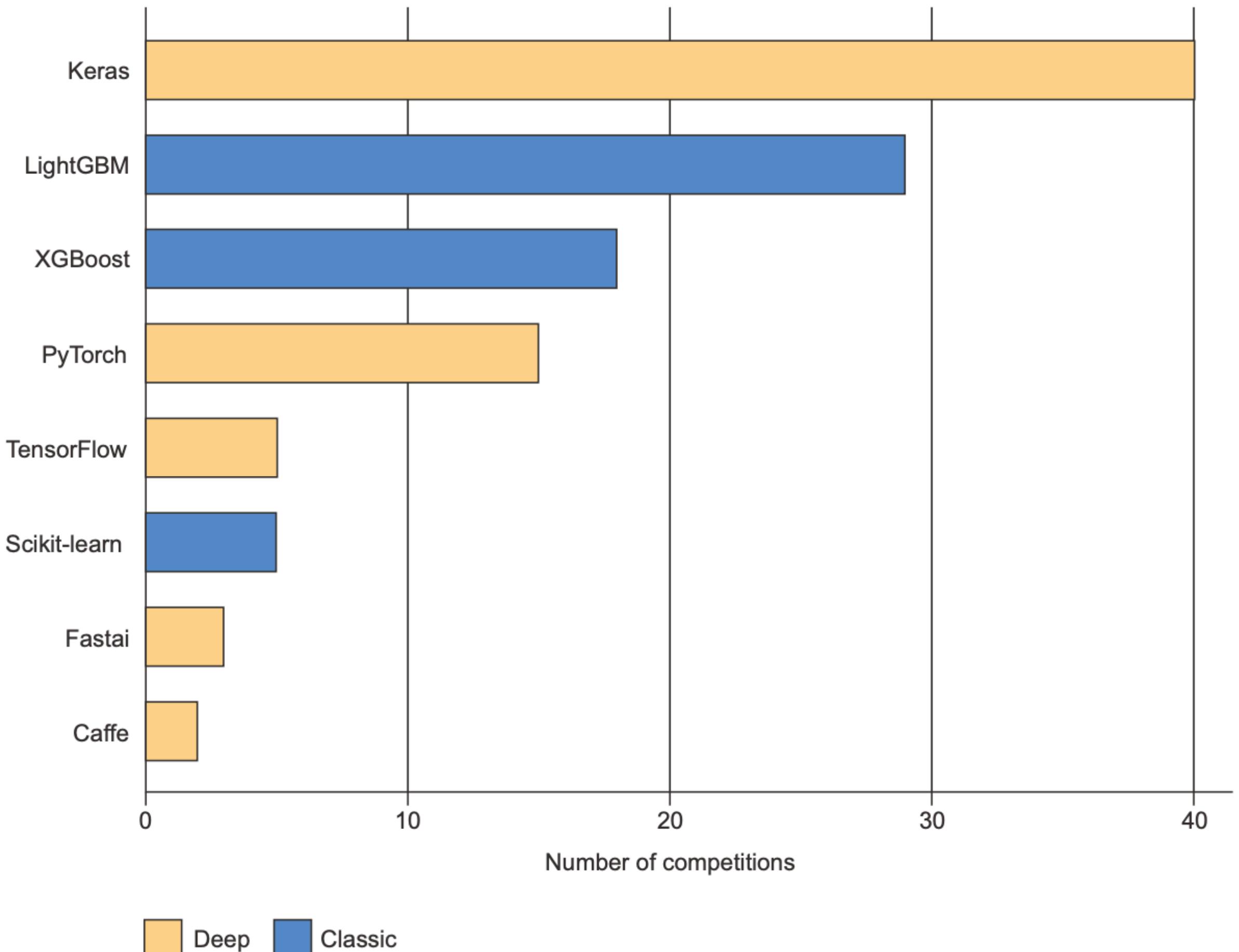
Hidden Layers

*Multistage (several layers)
way to learn data
representations*



What deep learning has achieved so far?

- Near-human-level image classification
- Near-human-level speech transcription
- Near-human-level handwriting transcription
- Dramatically improved machine translation
- Dramatically improved text-to-speech conversion
- Google assistant - Alexa
- Near-human-level autonomous driving
- Superhuman game-playing (AlphaGo, Alphazero, AlphaStar, etc...)



Why Deep Learning and why now?

- CNNs were well understood already in 1990.
Why did this not happen earlier?

1. Hardware:

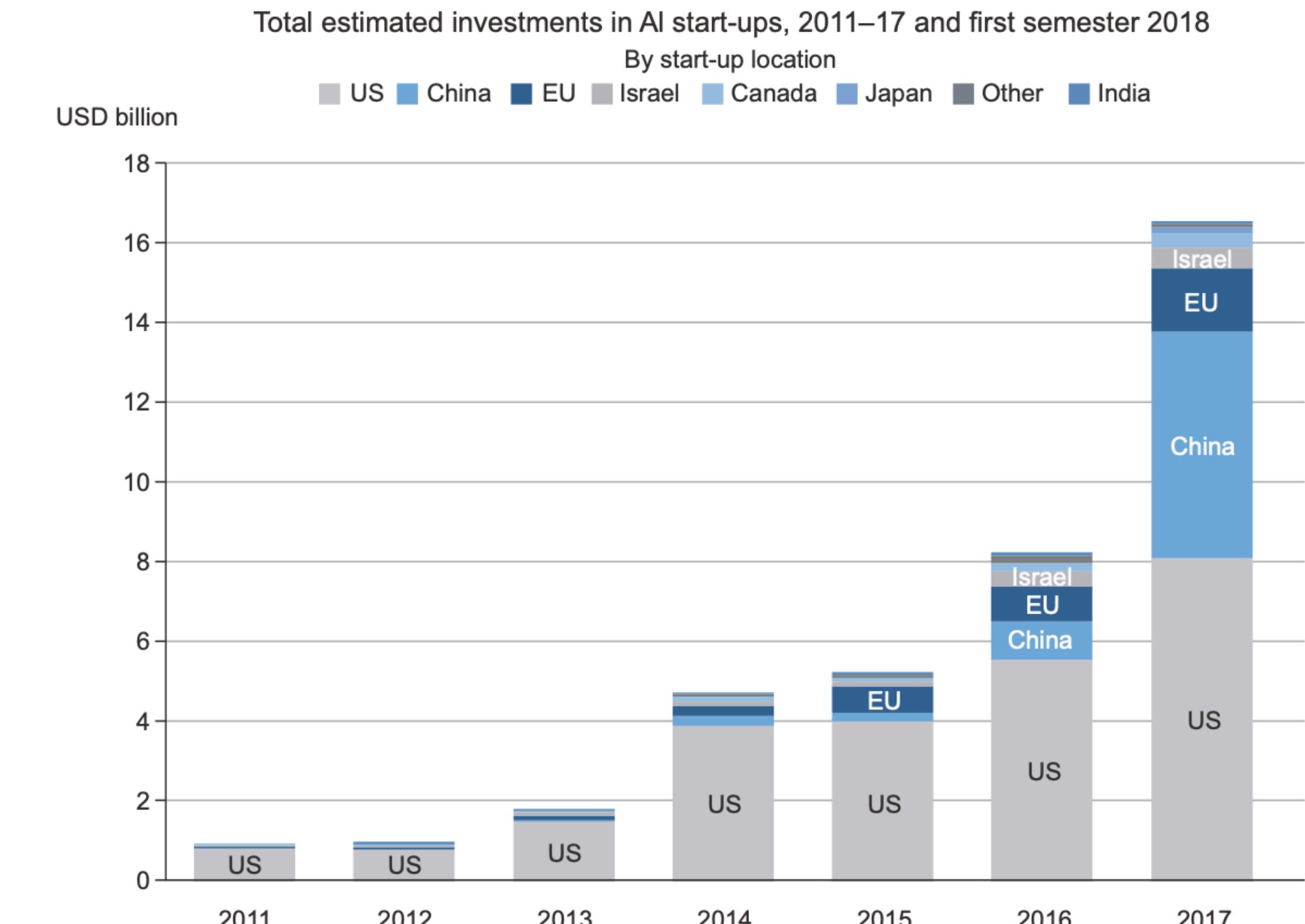
- Development of GPUs, TPUs

2. Data (ImageNet)

- Kaggle competitions

3. Algorithmic advances

- Extensive research





The building blocks of Neural Networks

Back to the hand-written digits



Label: 3



Label: 3



Label: 3

Could you write a
program that takes the pixels
as input and tells you the
number?

Back to the hand-written digits



Label: 3



Label: 3



Label: 3

Could you write a
program that takes the pixels
as input and tells you the
number?



Back to the hand-written digits



Label: 3



Label: 3

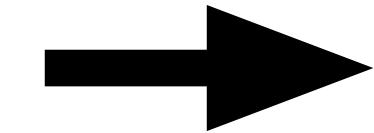


Label: 3

Could you write a
program that takes the pixels
as input and tells you the
number?



Label: 3



0
1
2
3
4
5
6
7
8
9

Neural Network
to recognize hand-
written digits

What is a neuron?

Neural Networks

What are neurons?

How are they linked
together?

What is a neuron?

Neural Networks

What are neurons?

How are they linked
together?

0.8

A thing that holds a number
between 0 and 1

What is a neuron?

Neural Networks

What are neurons?

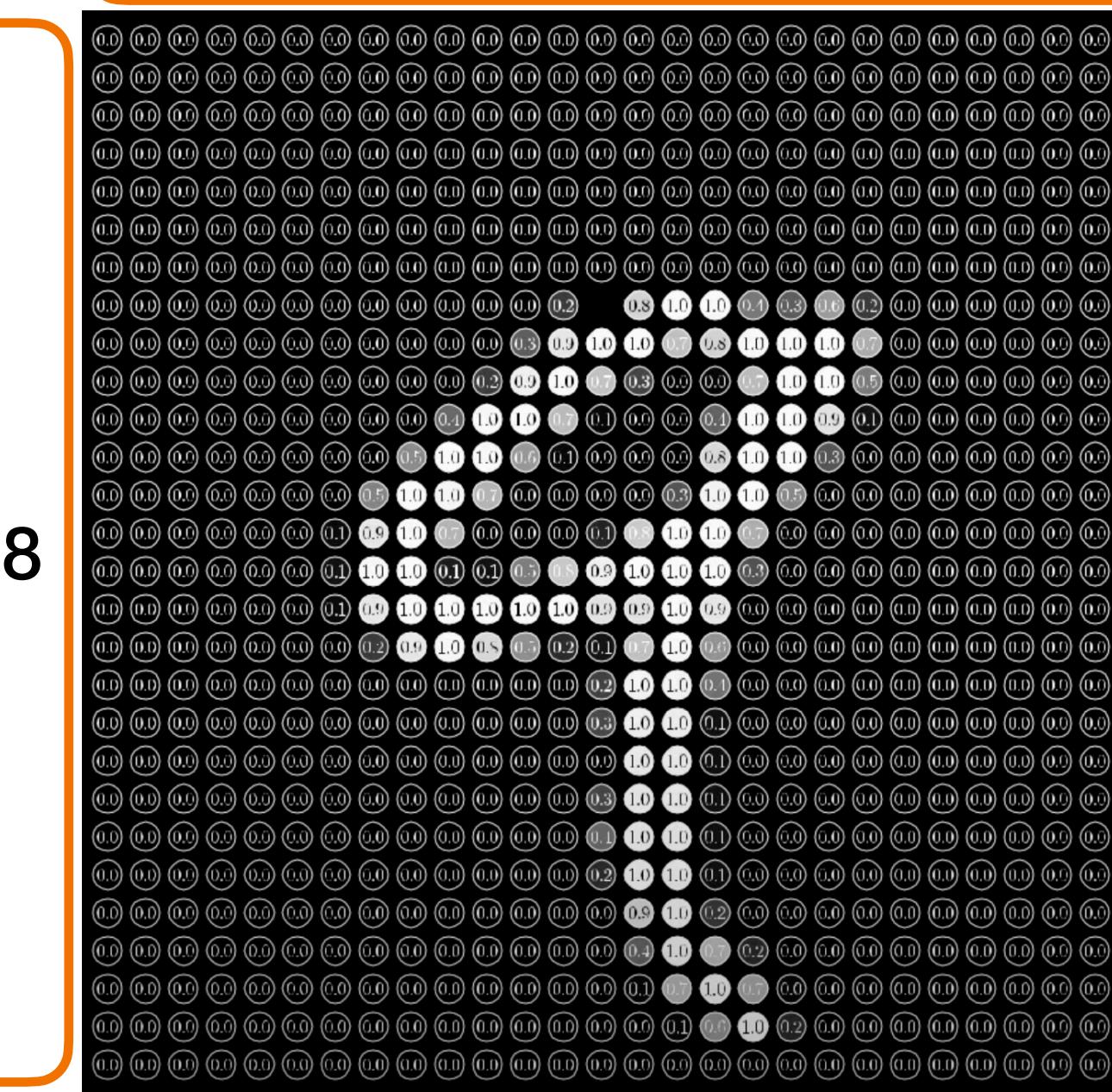
How are they linked together?

0.8

A thing that holds a number
between 0 and 1

23

28



What is a neuron?

Neural Networks

What are neurons?

How are they linked together?

28

A thing that holds a number between 0 and 1

0.8

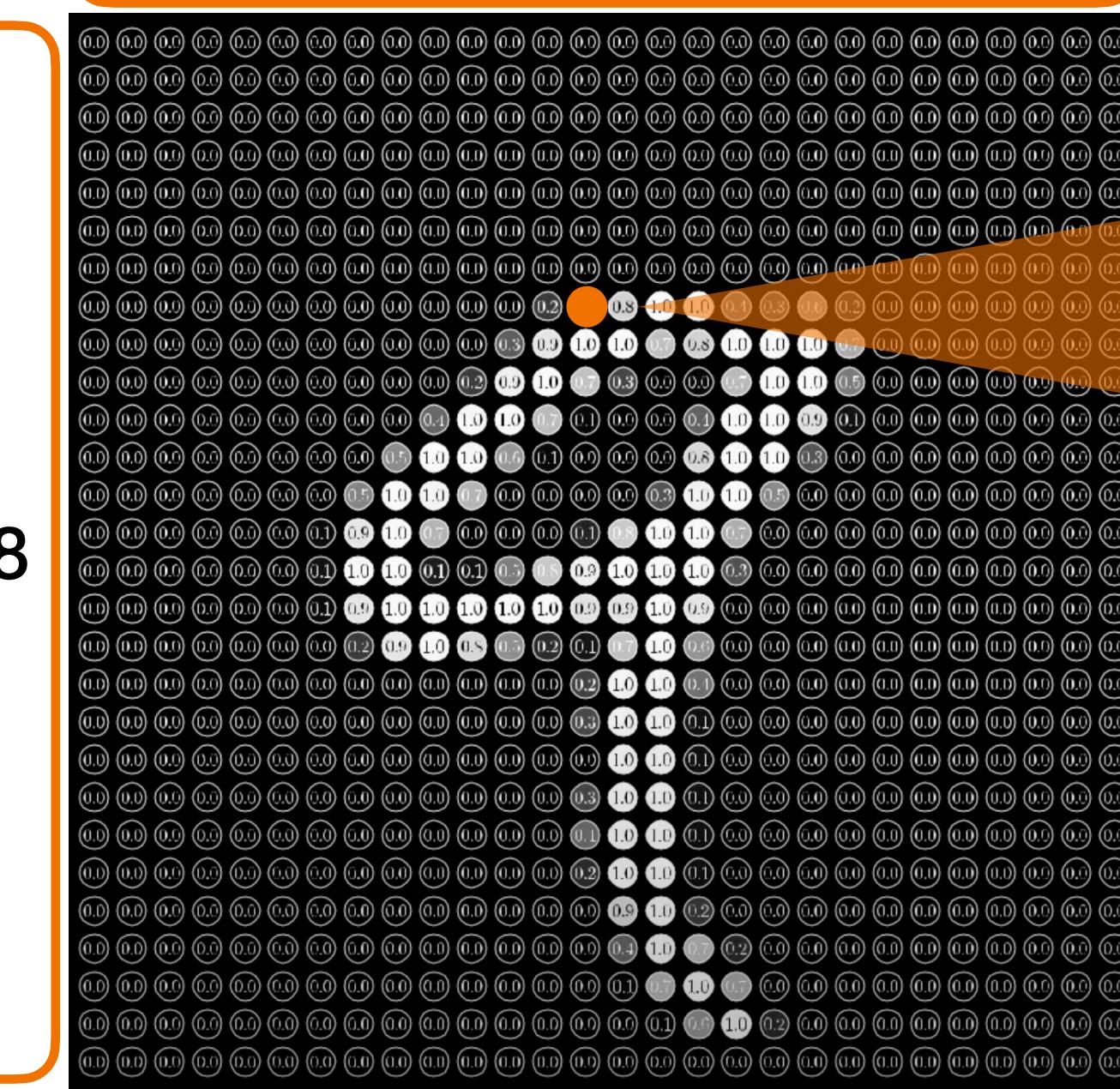
28

$28 \times 28 = 784$

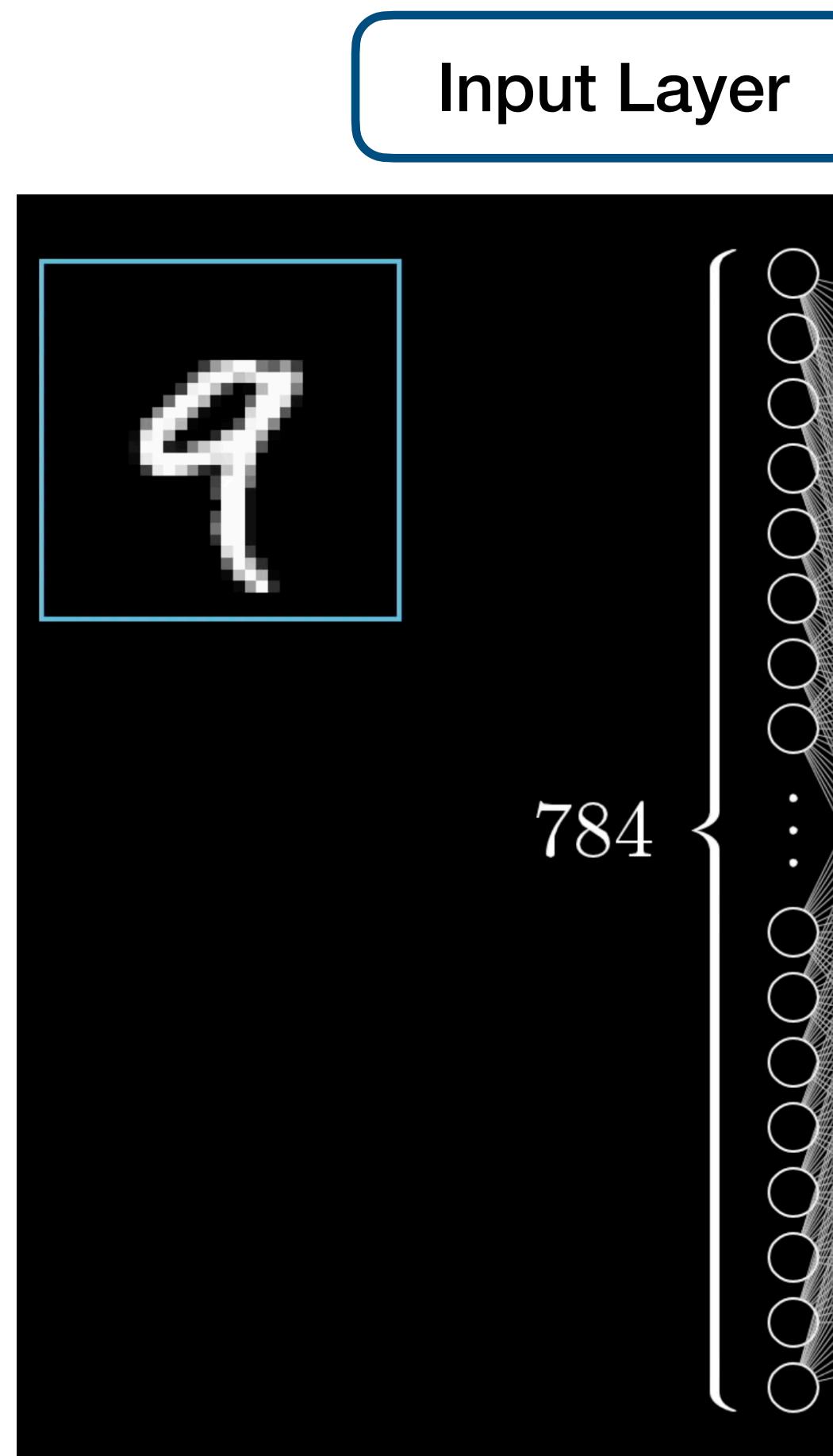
0.9

Activation

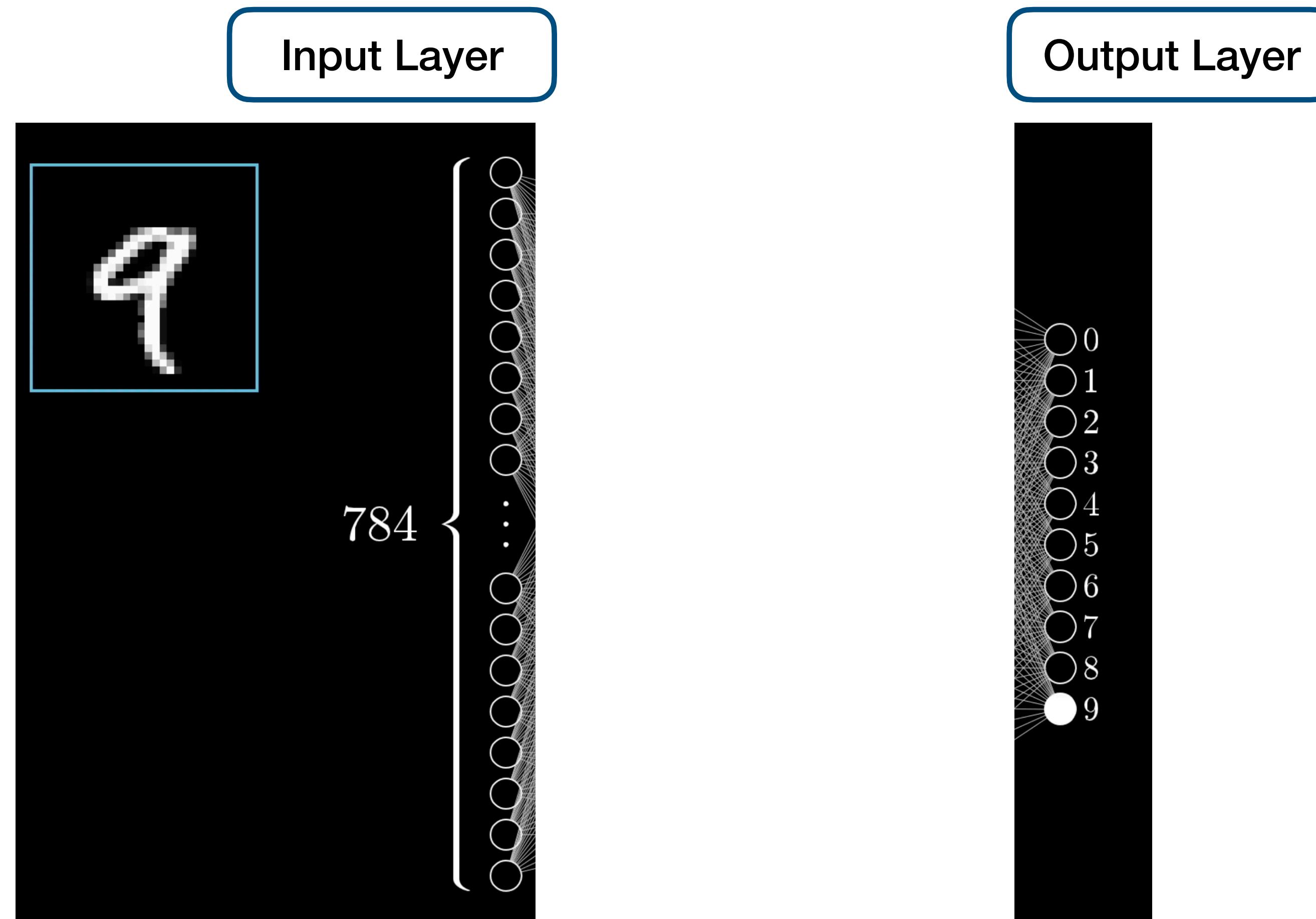
0.1



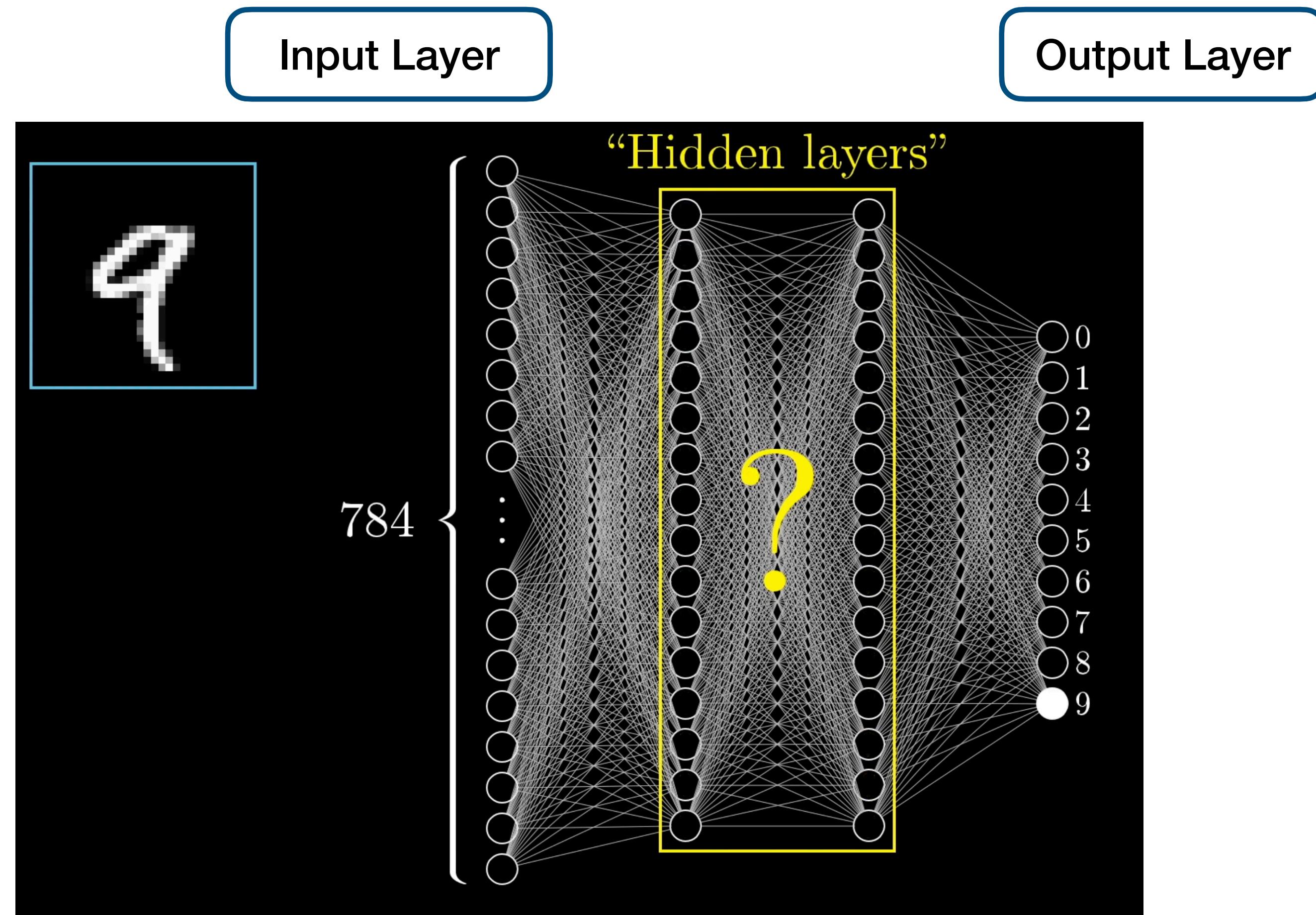
Layers of Neurons lead to the Neural network structure



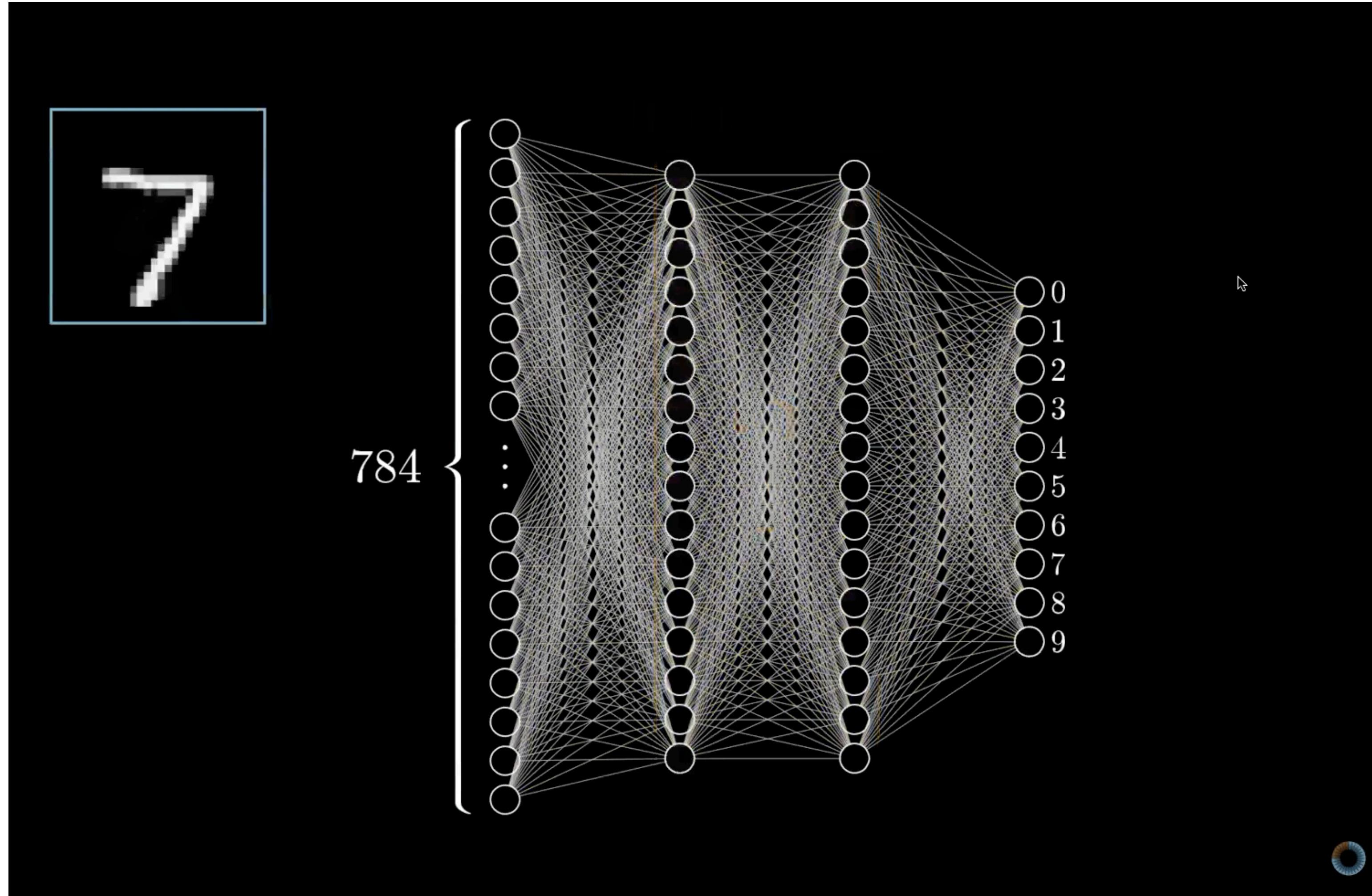
Layers of Neurons lead to the Neural network structure



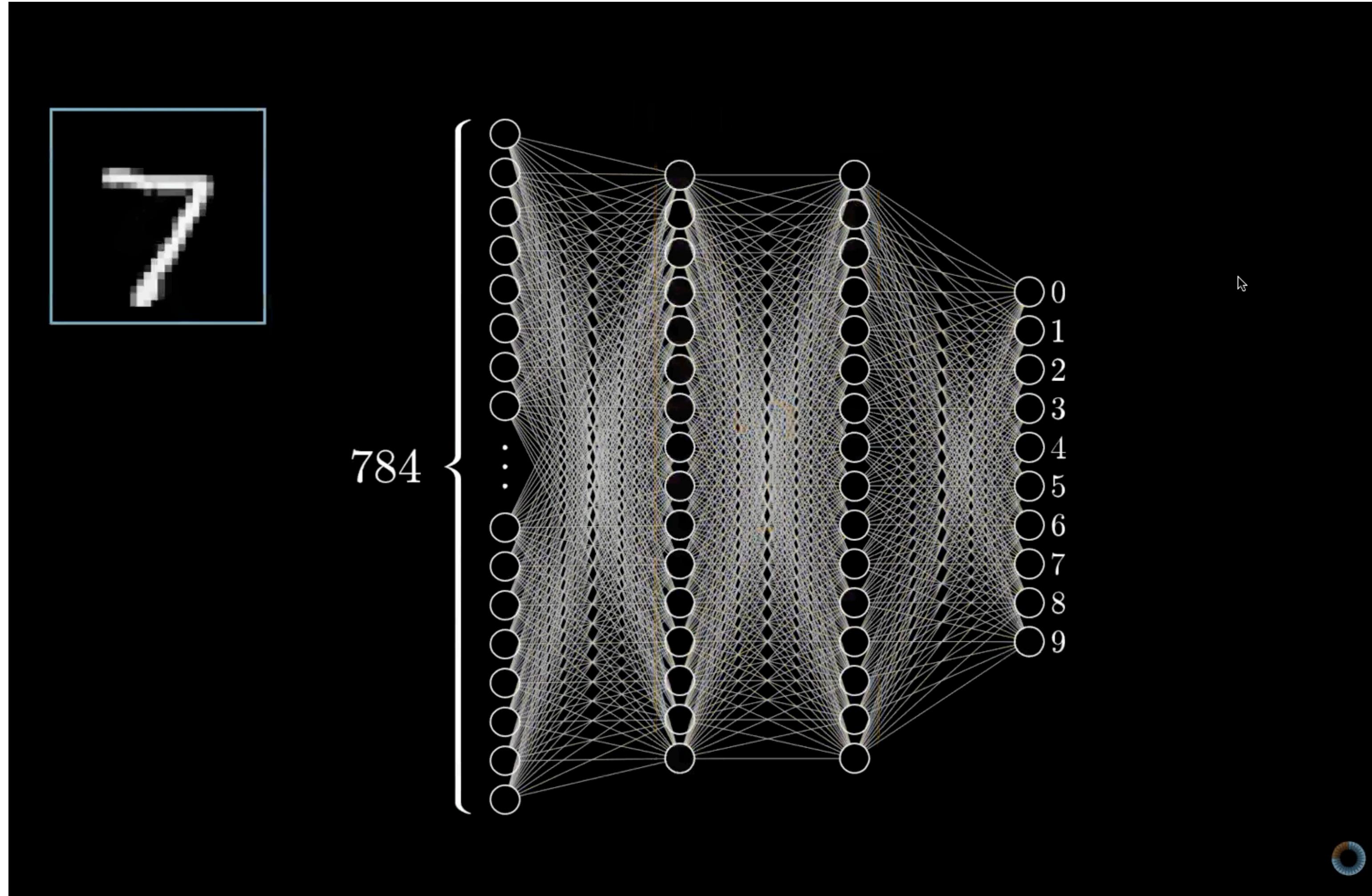
Layers of Neurons lead to the Neural network structure



Layers of Neurons lead to the Neural network structure

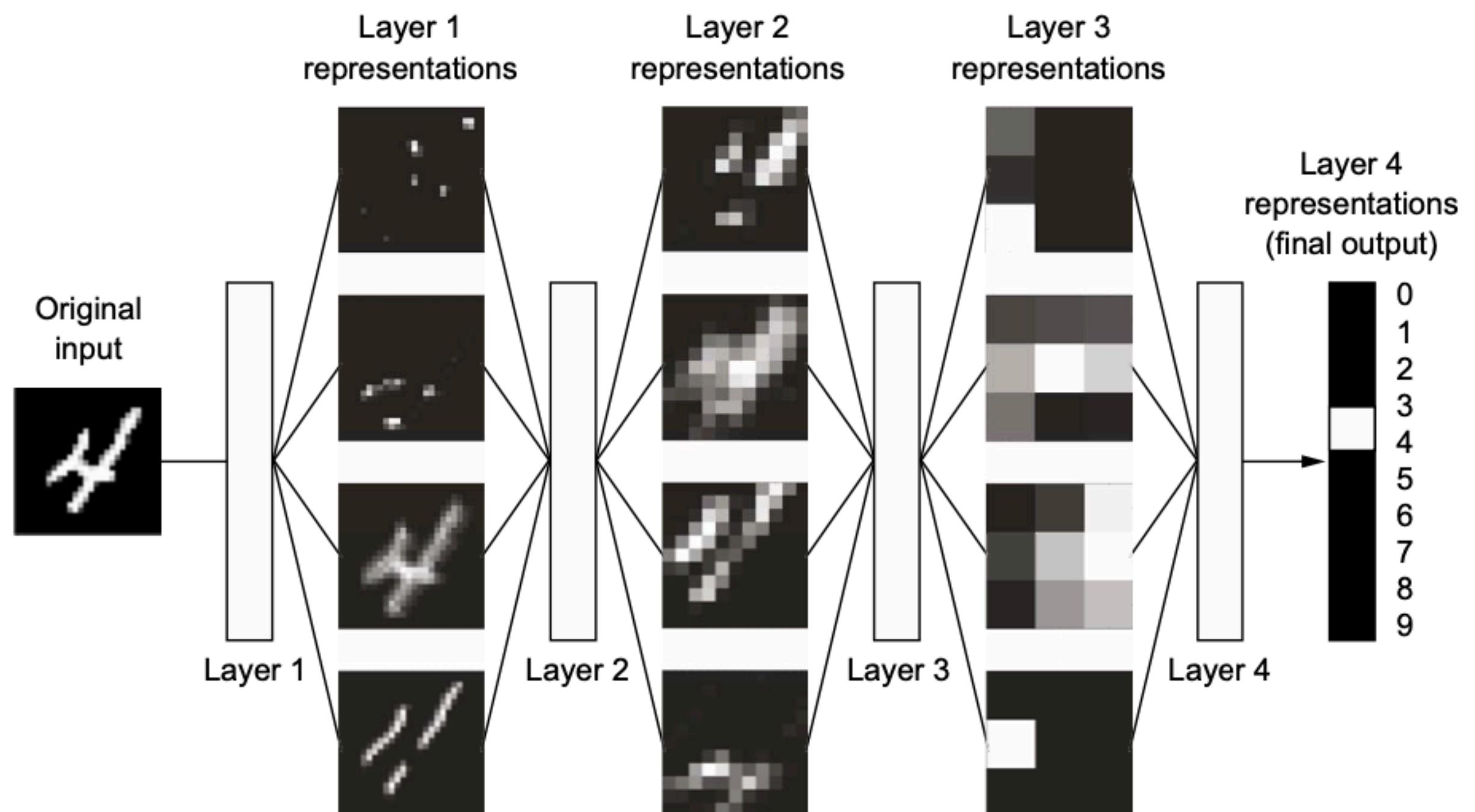


Layers of Neurons lead to the Neural network structure



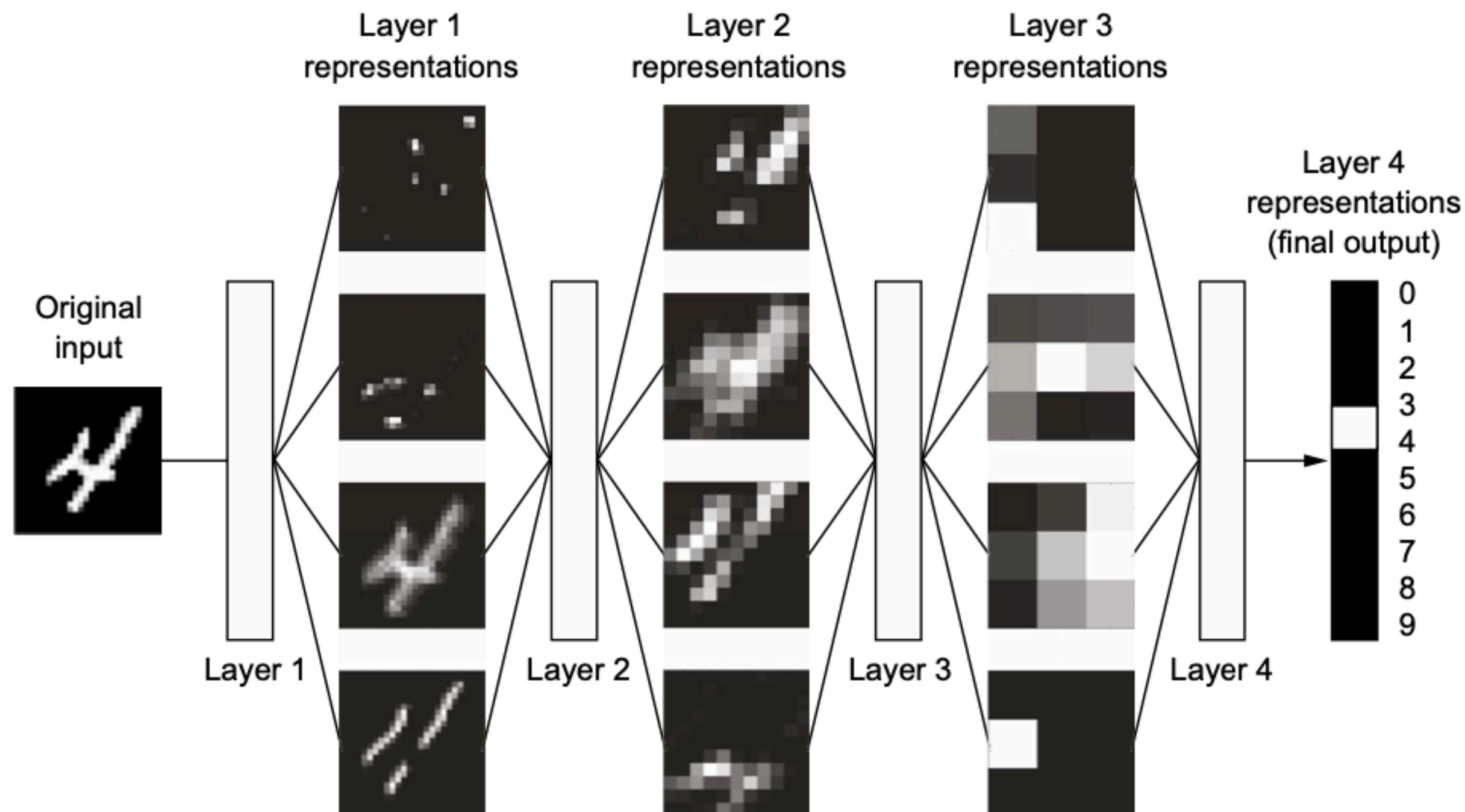
Why layers?

Find data representations



Why layers?

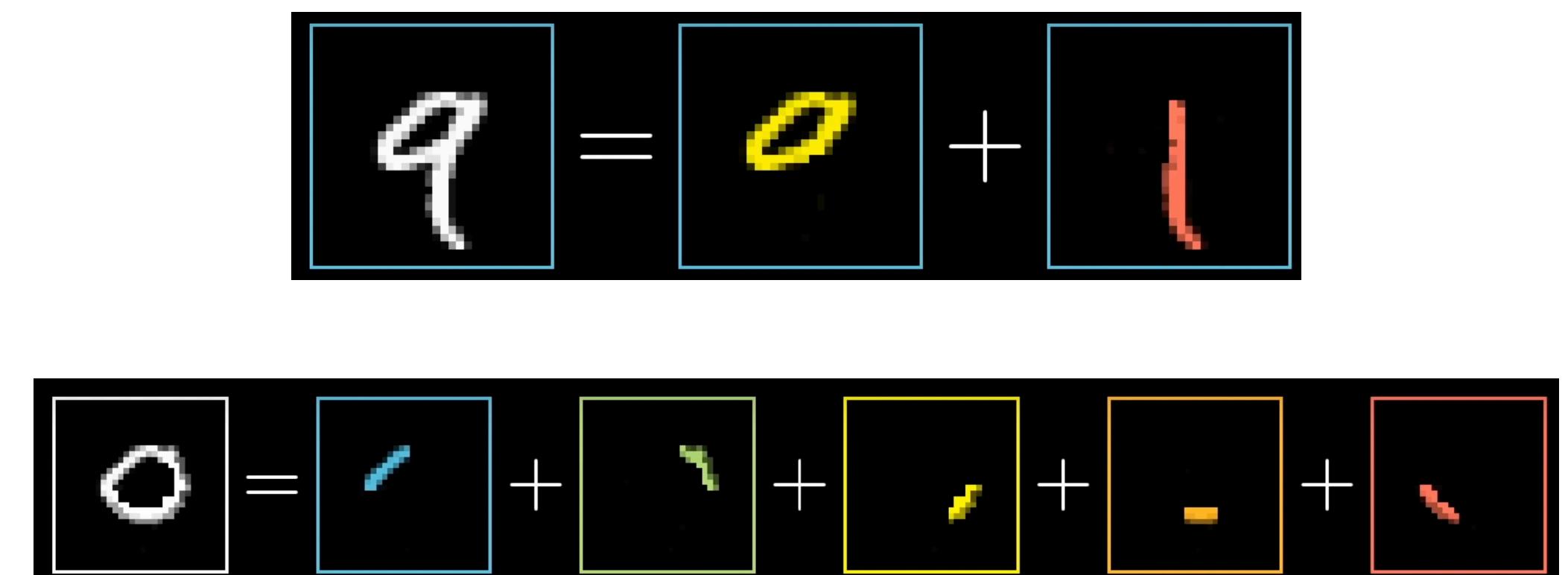
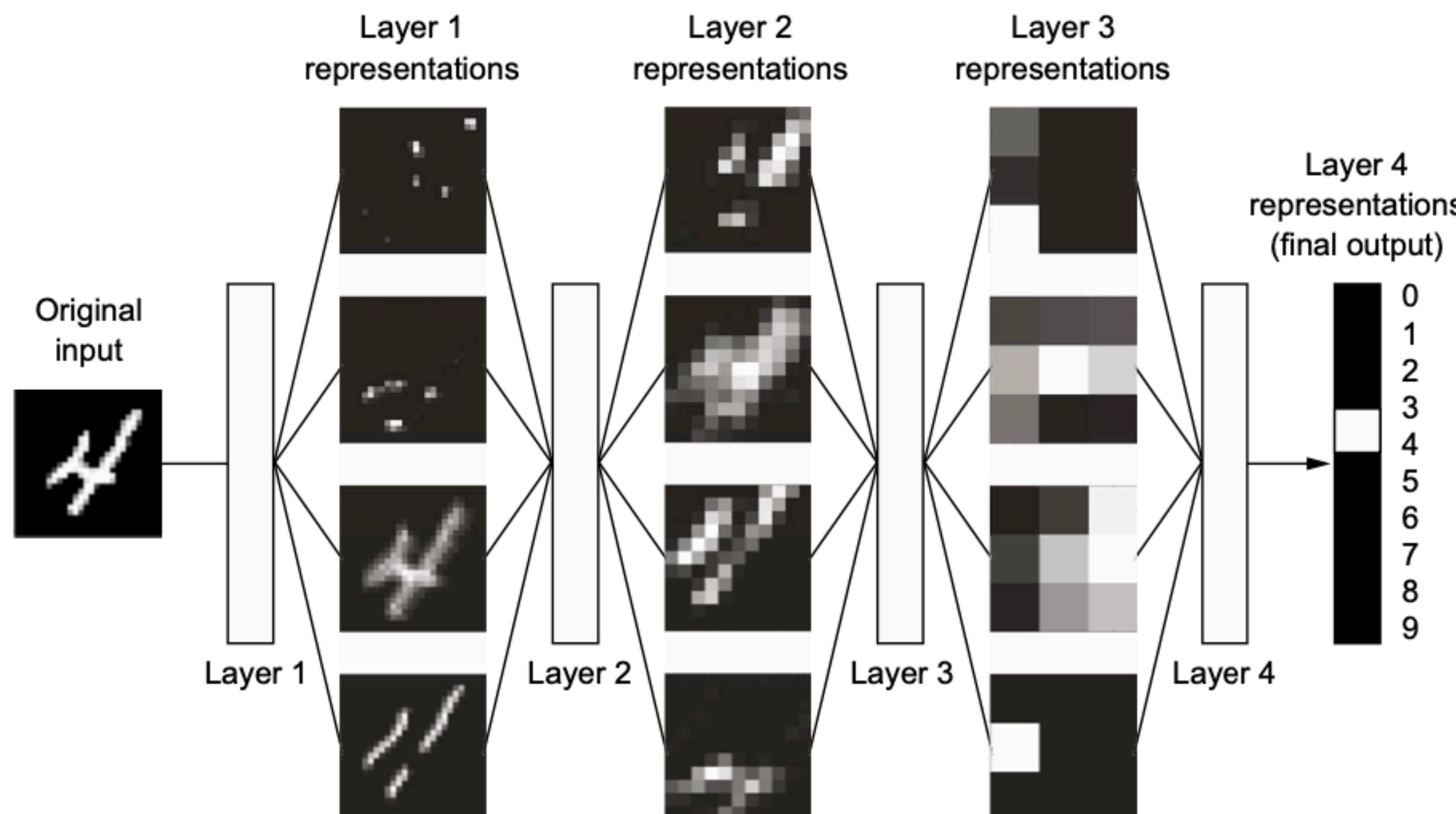
Find data representations



$$\begin{aligned} \text{Digit } 4 &= \text{Feature } 1 + \text{Feature } 2 \\ \text{Digit } 0 &= \text{Feature } 1 + \text{Feature } 2 + \text{Feature } 3 + \text{Feature } 4 + \text{Feature } 5 \end{aligned}$$

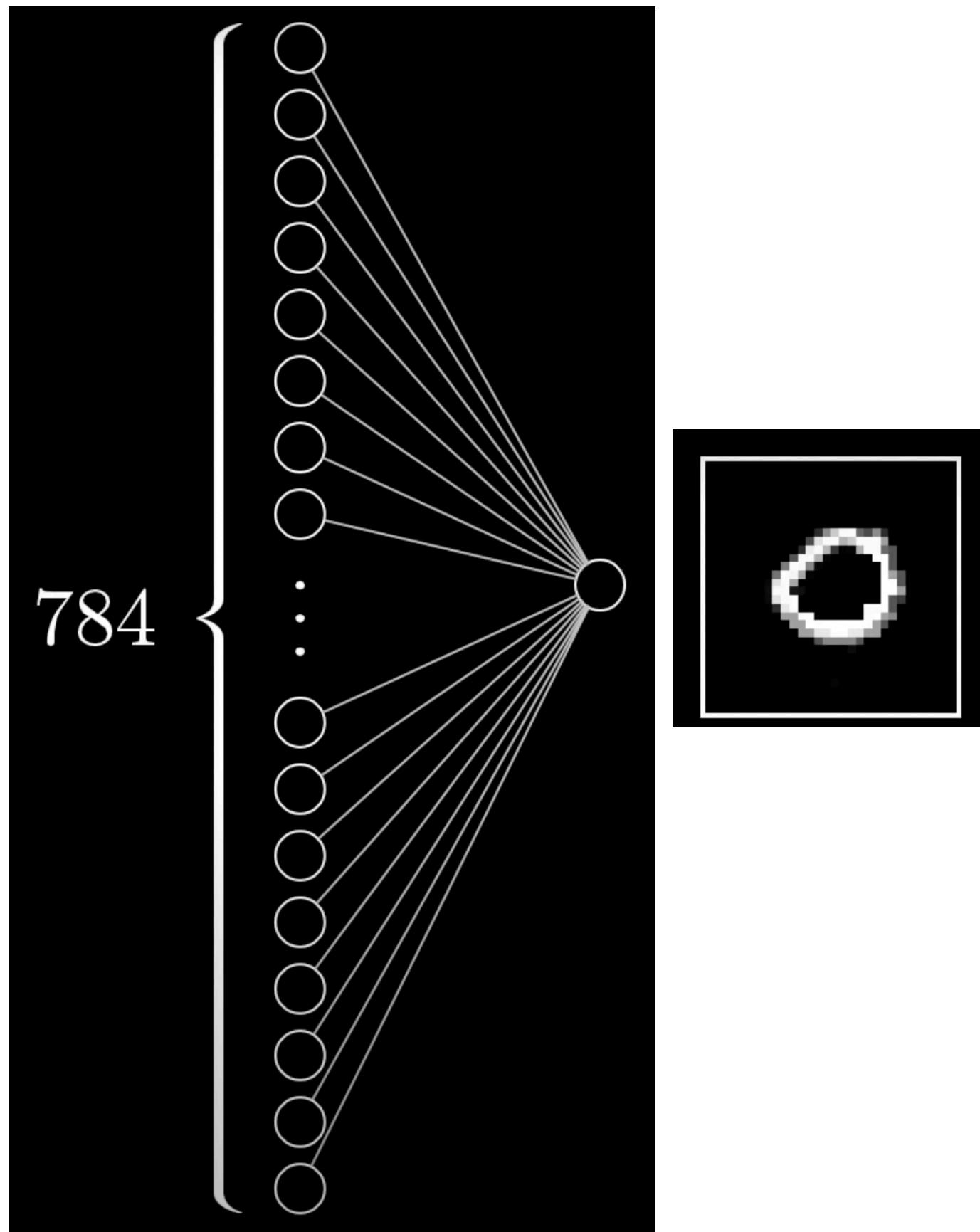
Why layers?

Find data representations

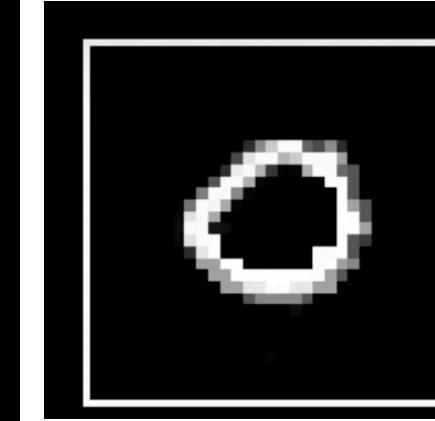
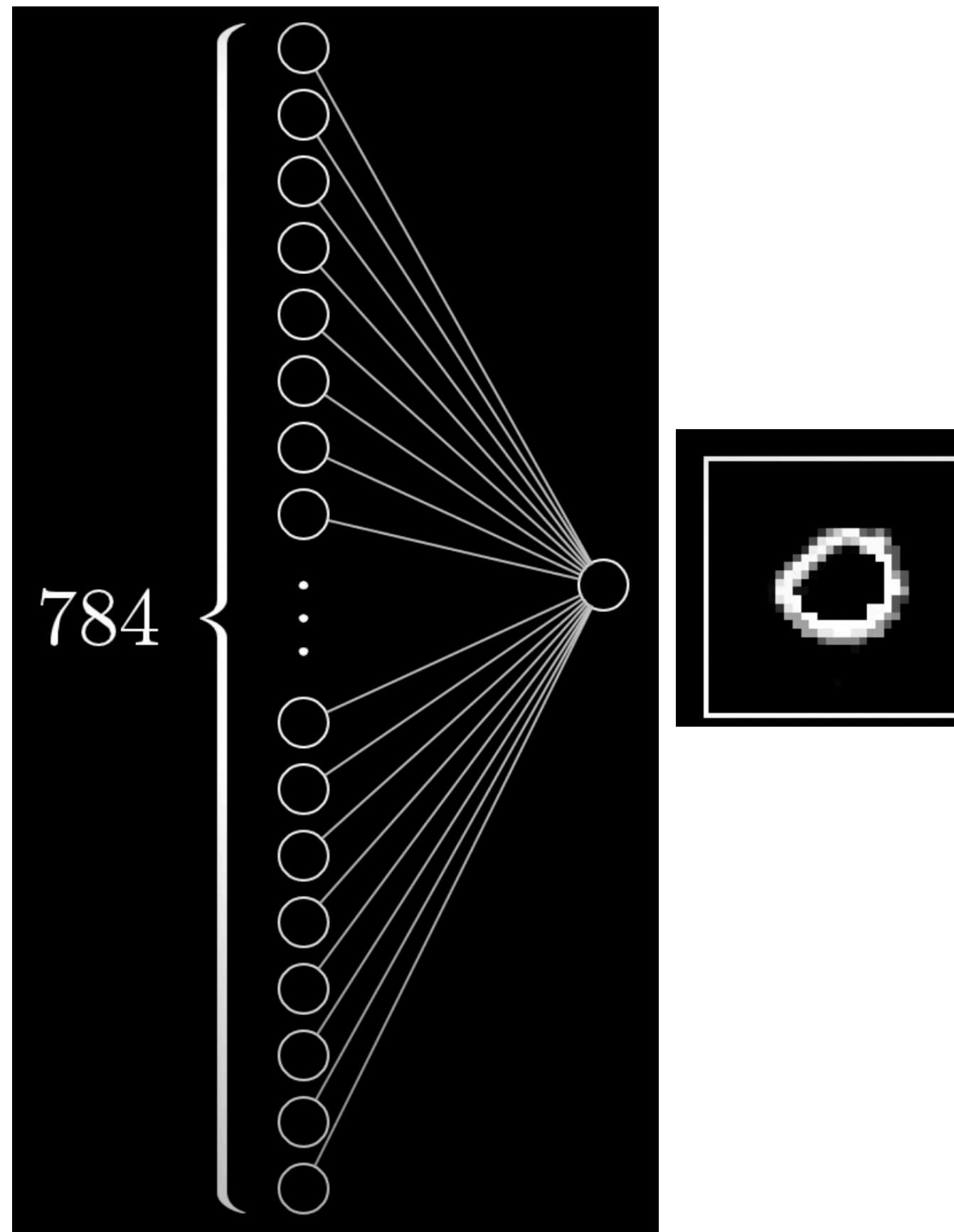


How connections in one layer may determine the activations in the next layer?

The parameters of the Network (Weights and Biases)



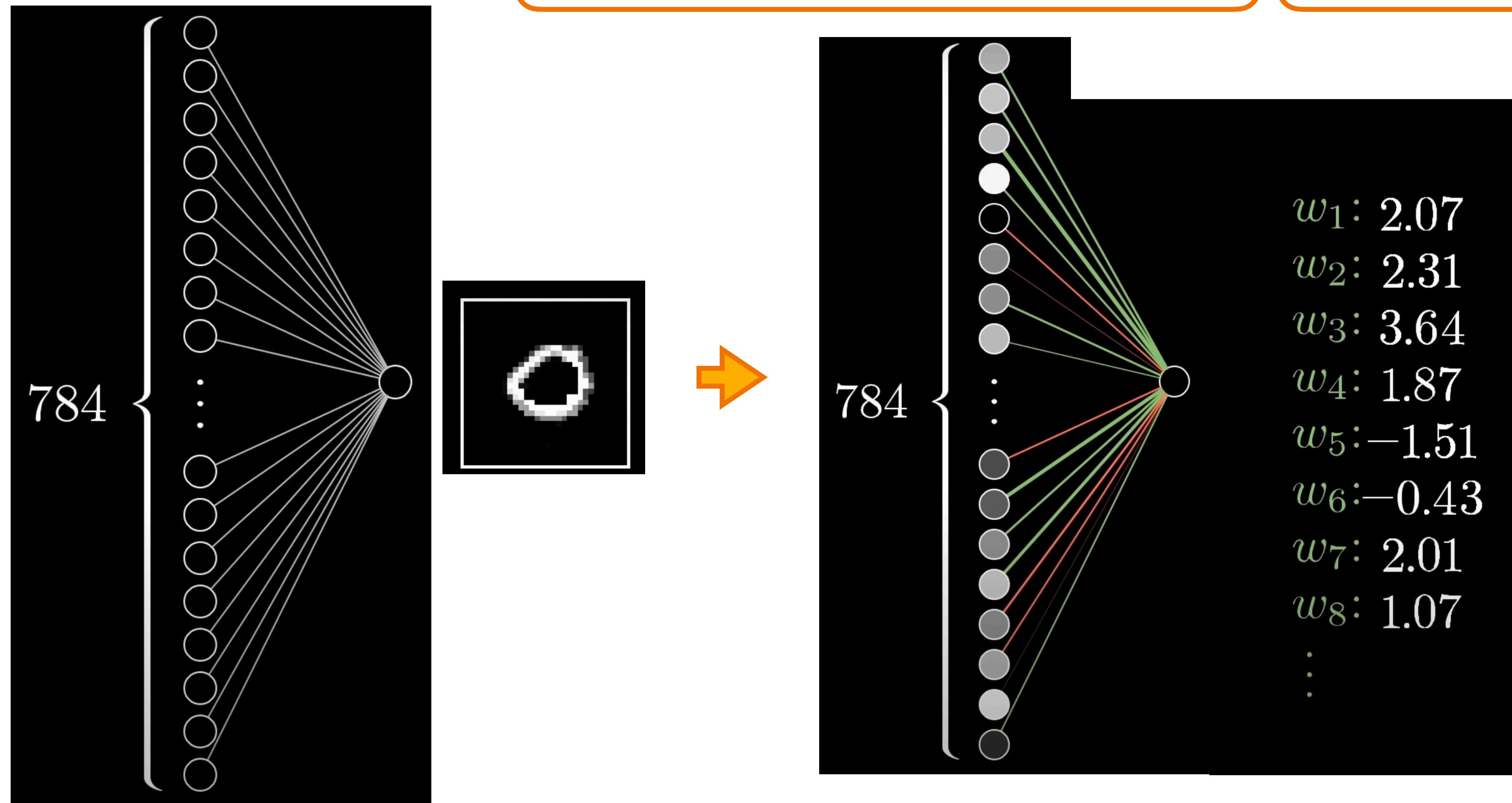
The parameters of the Network (Weights and Biases)



Which pixels are more important to recognize the circle?

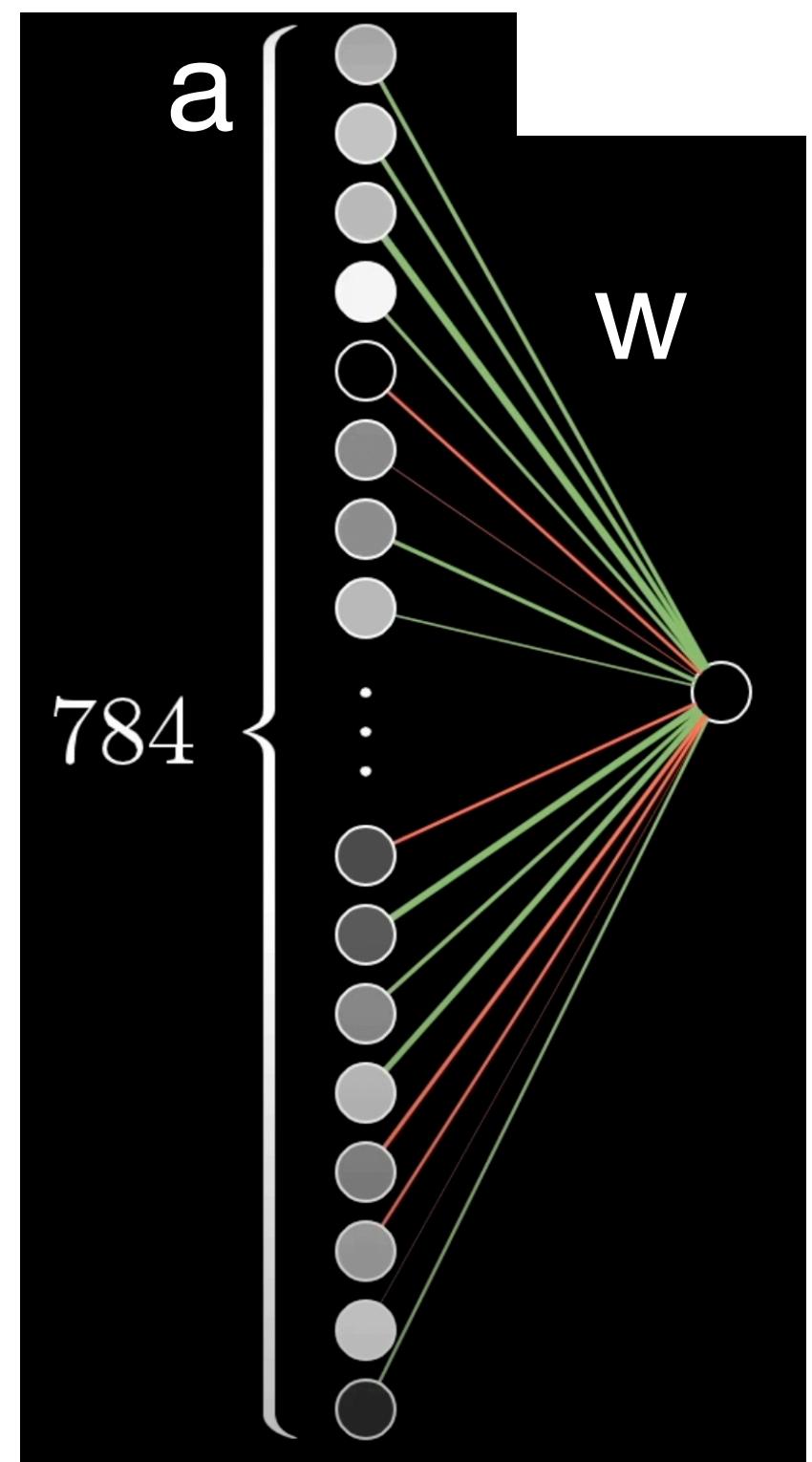
What parameters does the network need to detect that circle?

The parameters of the Network (Weights and Biases)

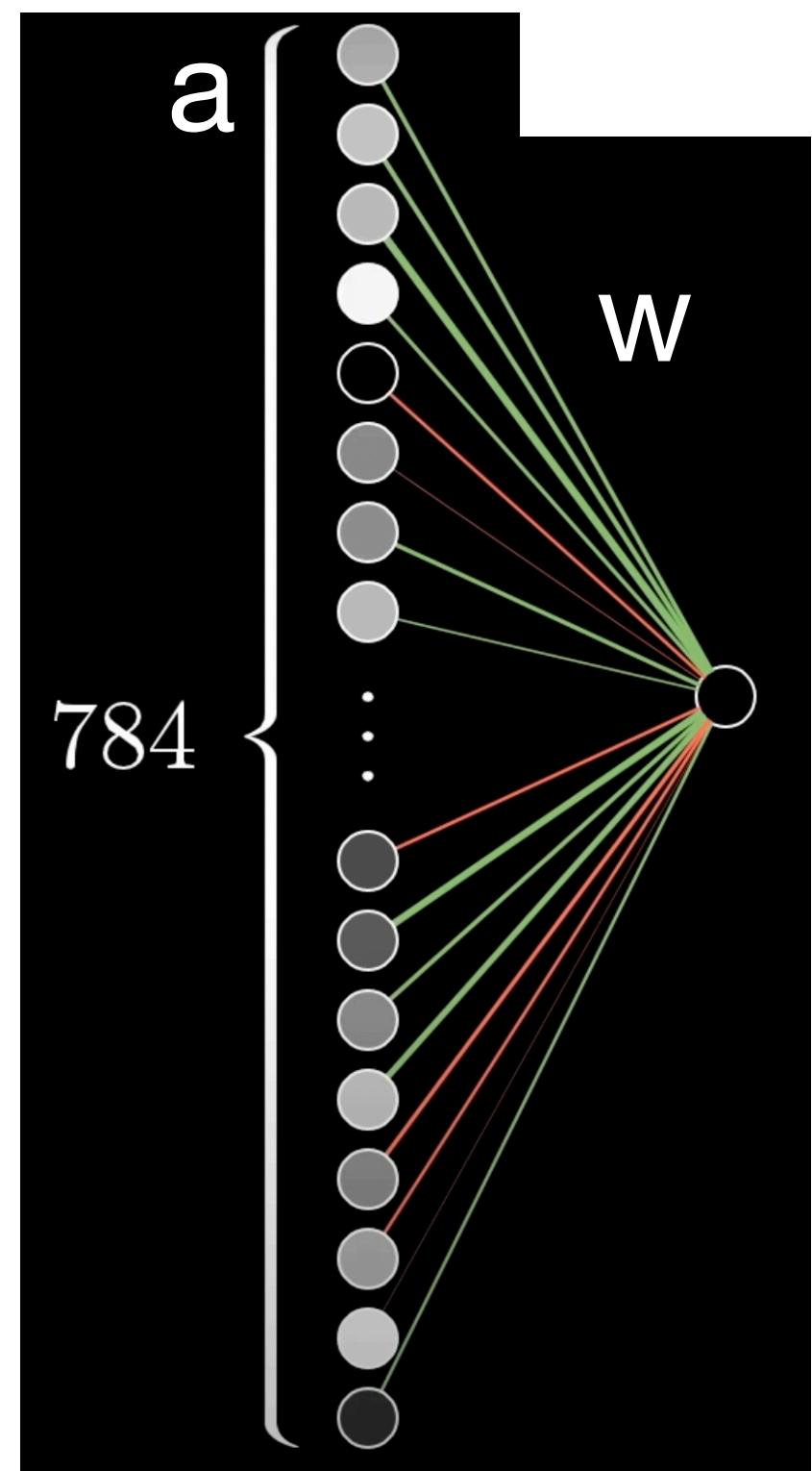


For every connection we have what we call weights

The parameters of the Network (Weights and Biases)



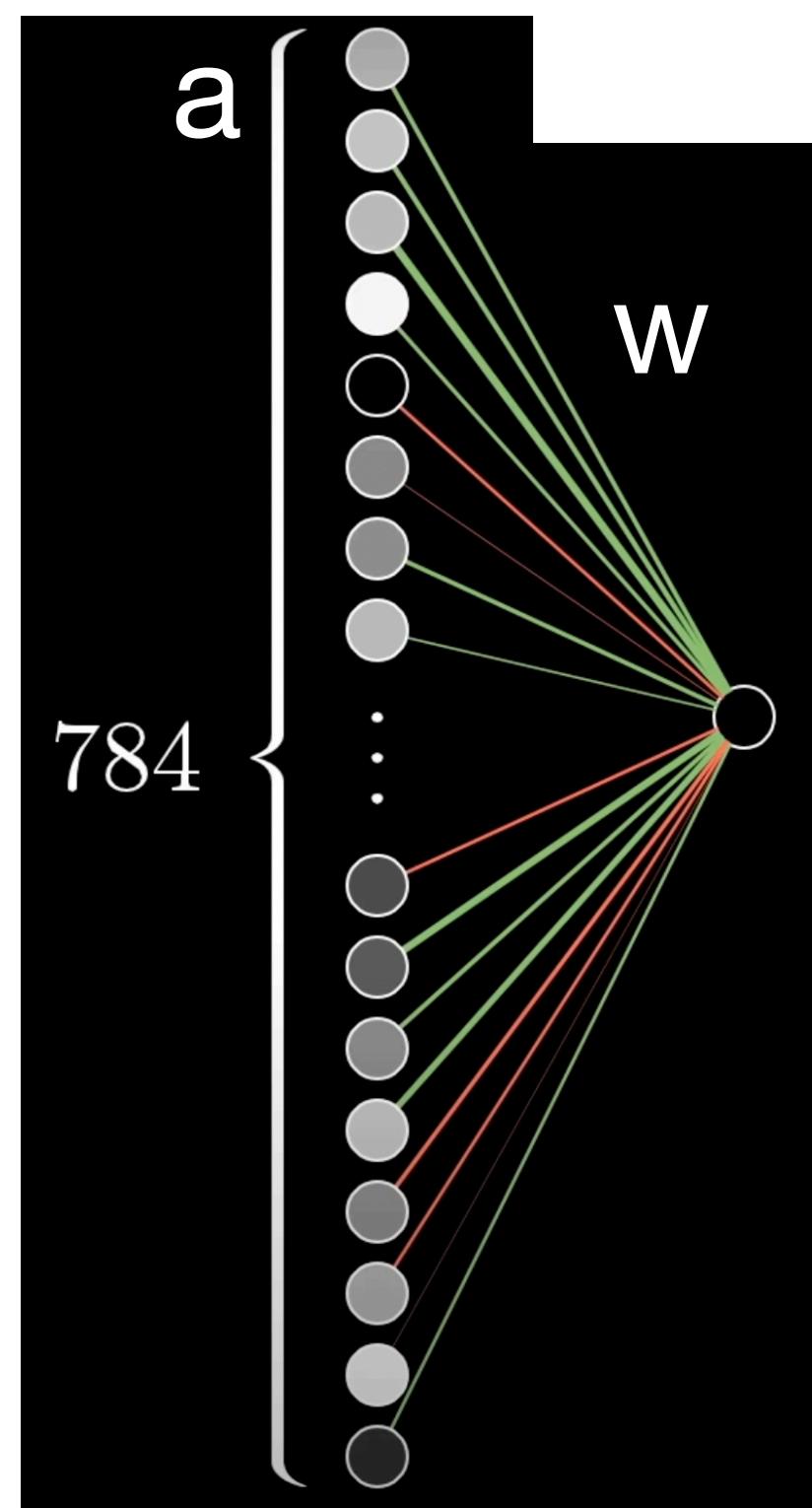
The parameters of the Network (Weights and Biases)



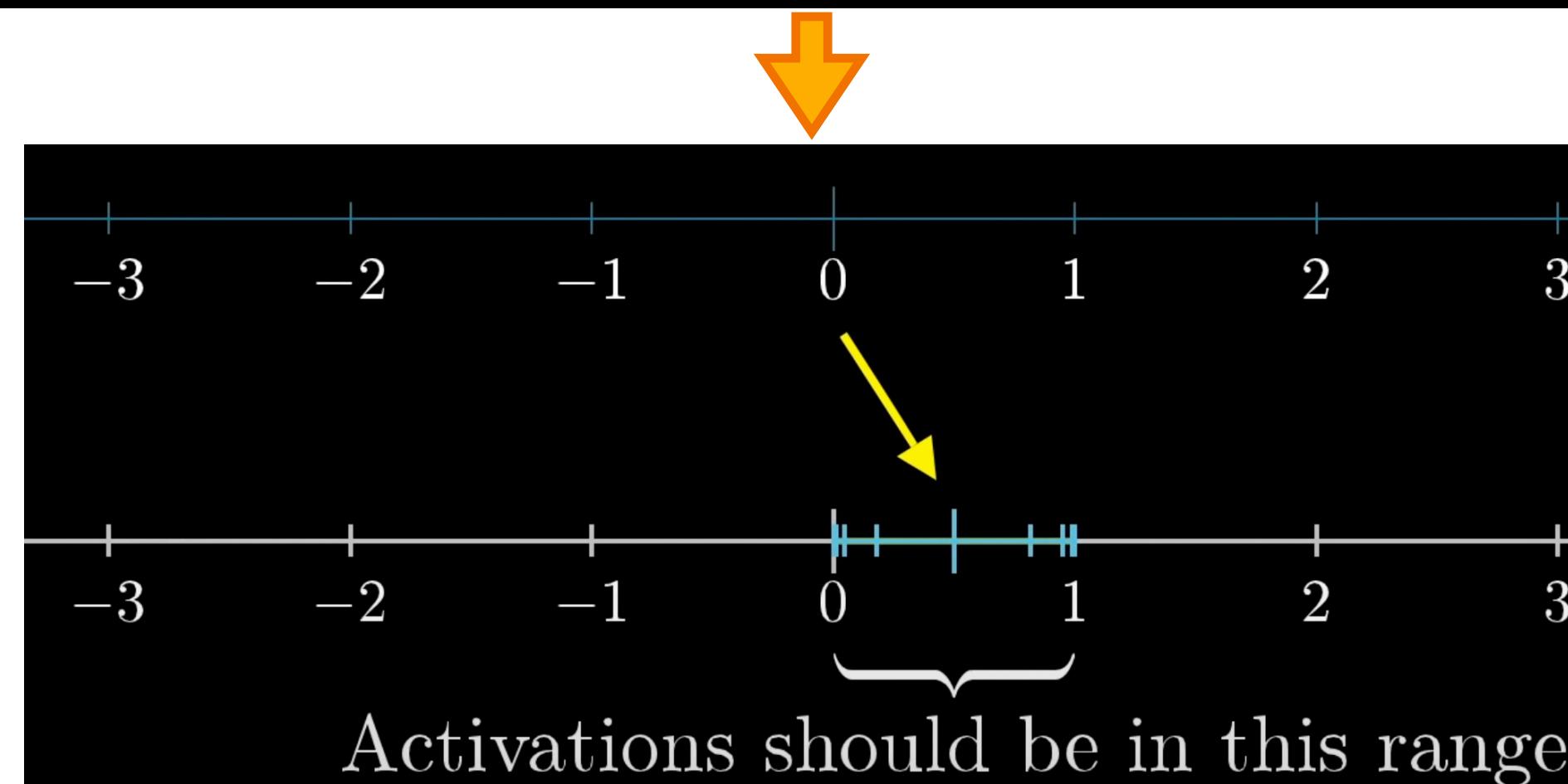
$$w_1 a_1 + w_2 a_2 + w_3 a_3 + w_4 a_4 + \dots + w_n a_n$$

But we need to combine
the activations and the
weights

The parameters of the Network (Weights and Biases)



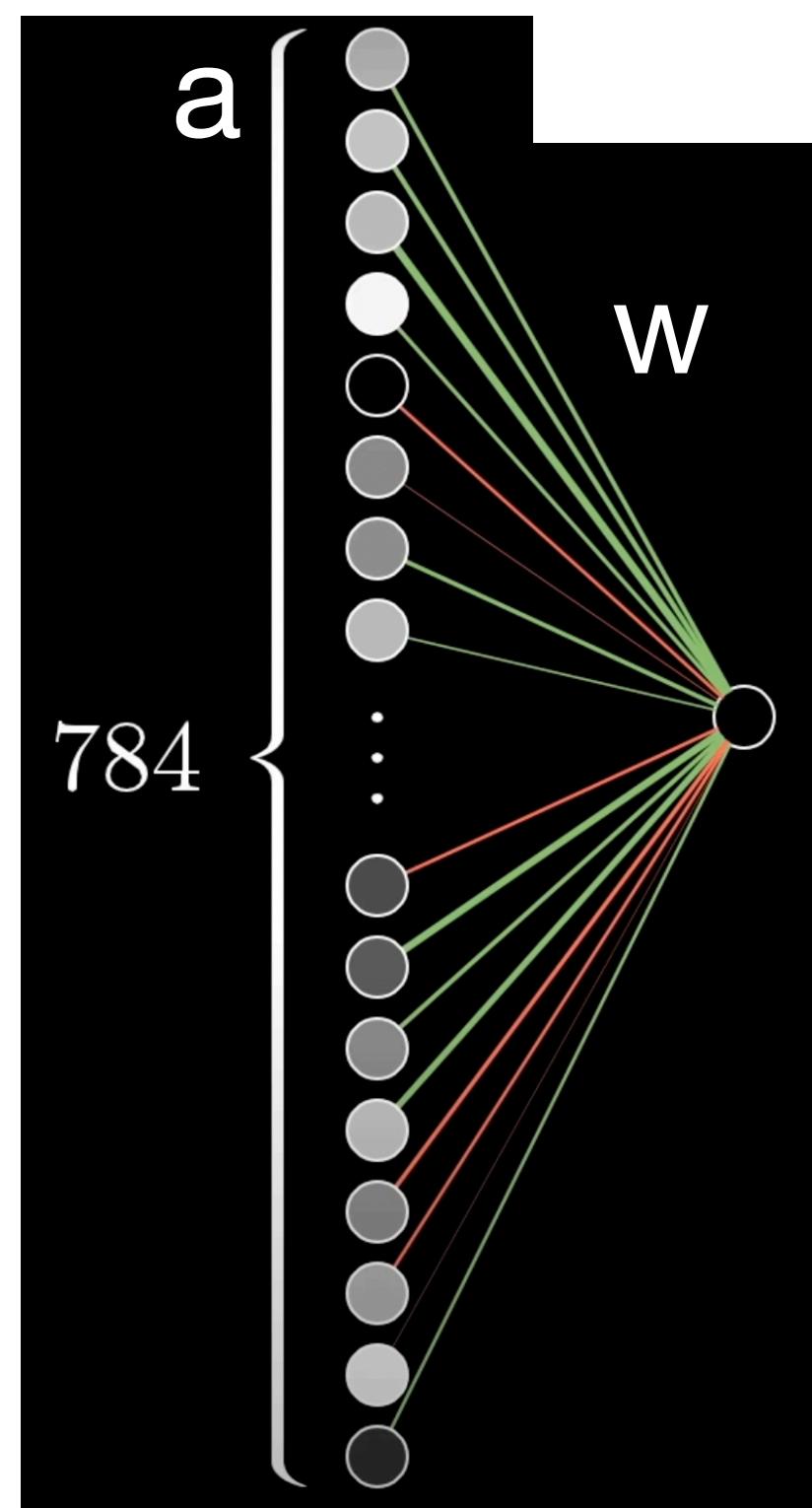
$$w_1 a_1 + w_2 a_2 + w_3 a_3 + w_4 a_4 + \dots + w_n a_n$$



But we need to combine the activations and the weights

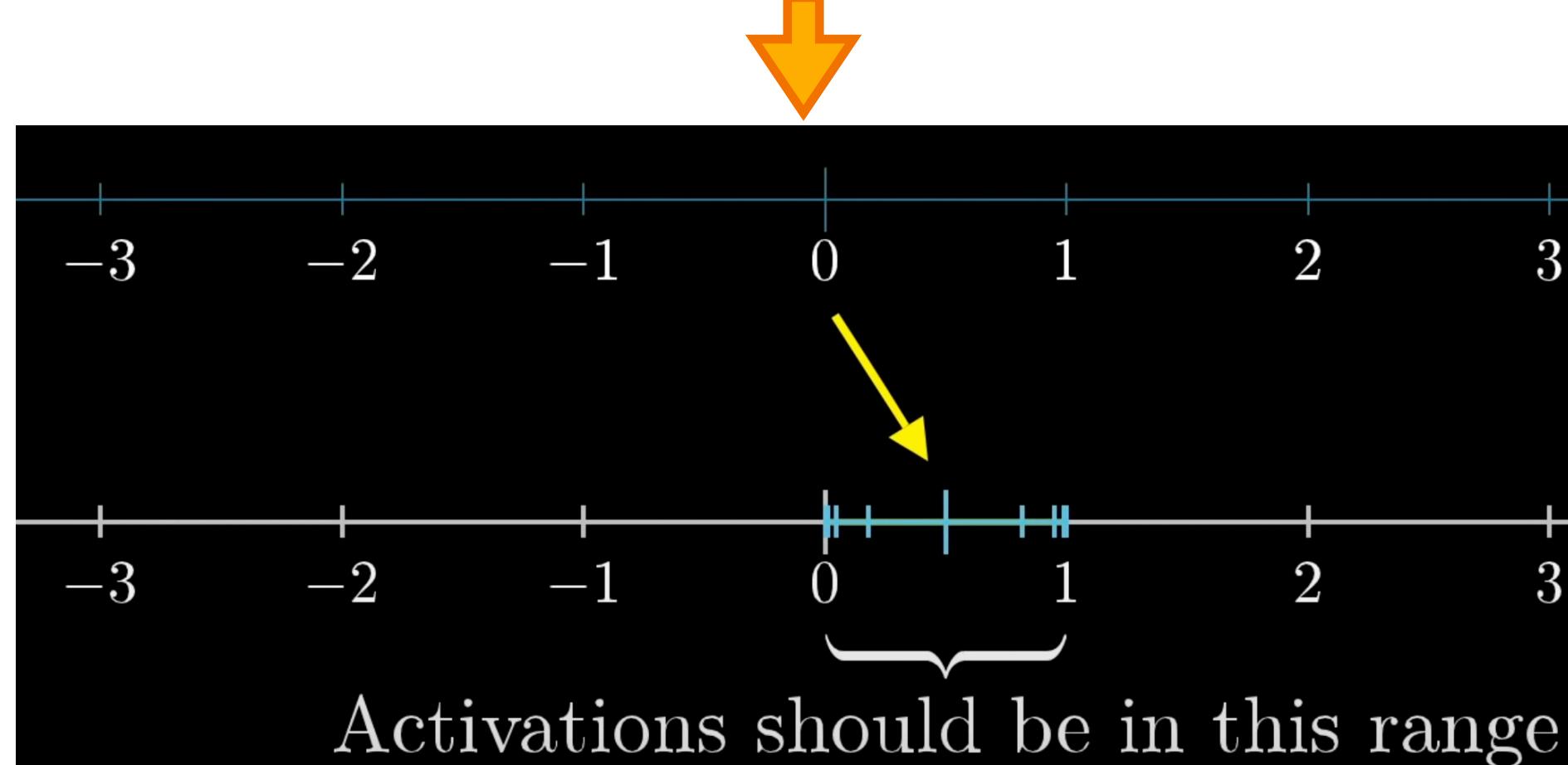
We also need values between 0 and 1 for the activation of our new neuron therefore we need an additional step

The parameters of the Network (Weights and Biases)



$$w_1 a_1 + w_2 a_2 + w_3 a_3 + w_4 a_4 + \dots + w_n a_n$$

But we need to combine the activations and the weights

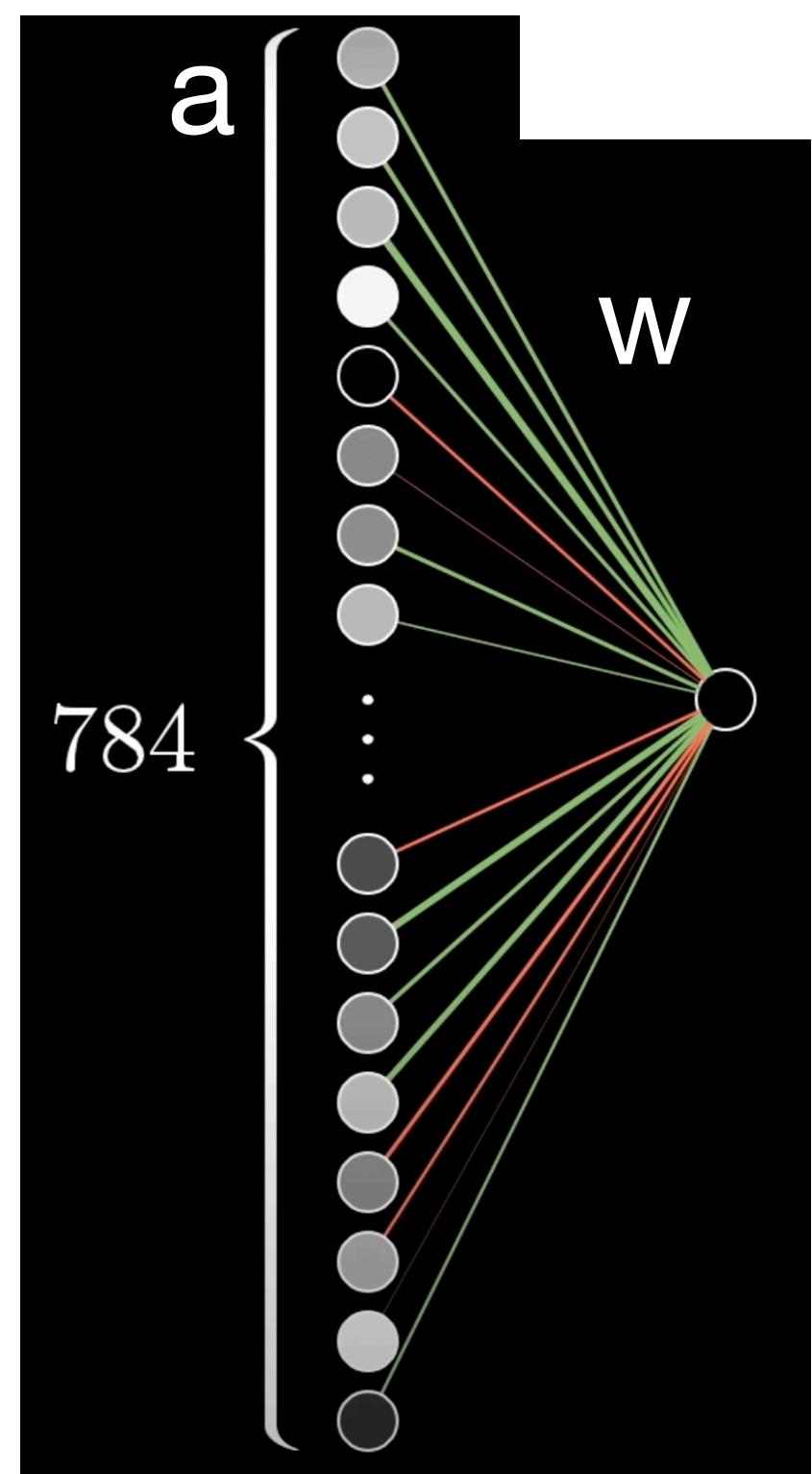


Sigmoid
↓
 $\sigma(w_1 a_1 + w_2 a_2 + w_3 a_3 + \dots + w_n a_n)$

How positive is this?

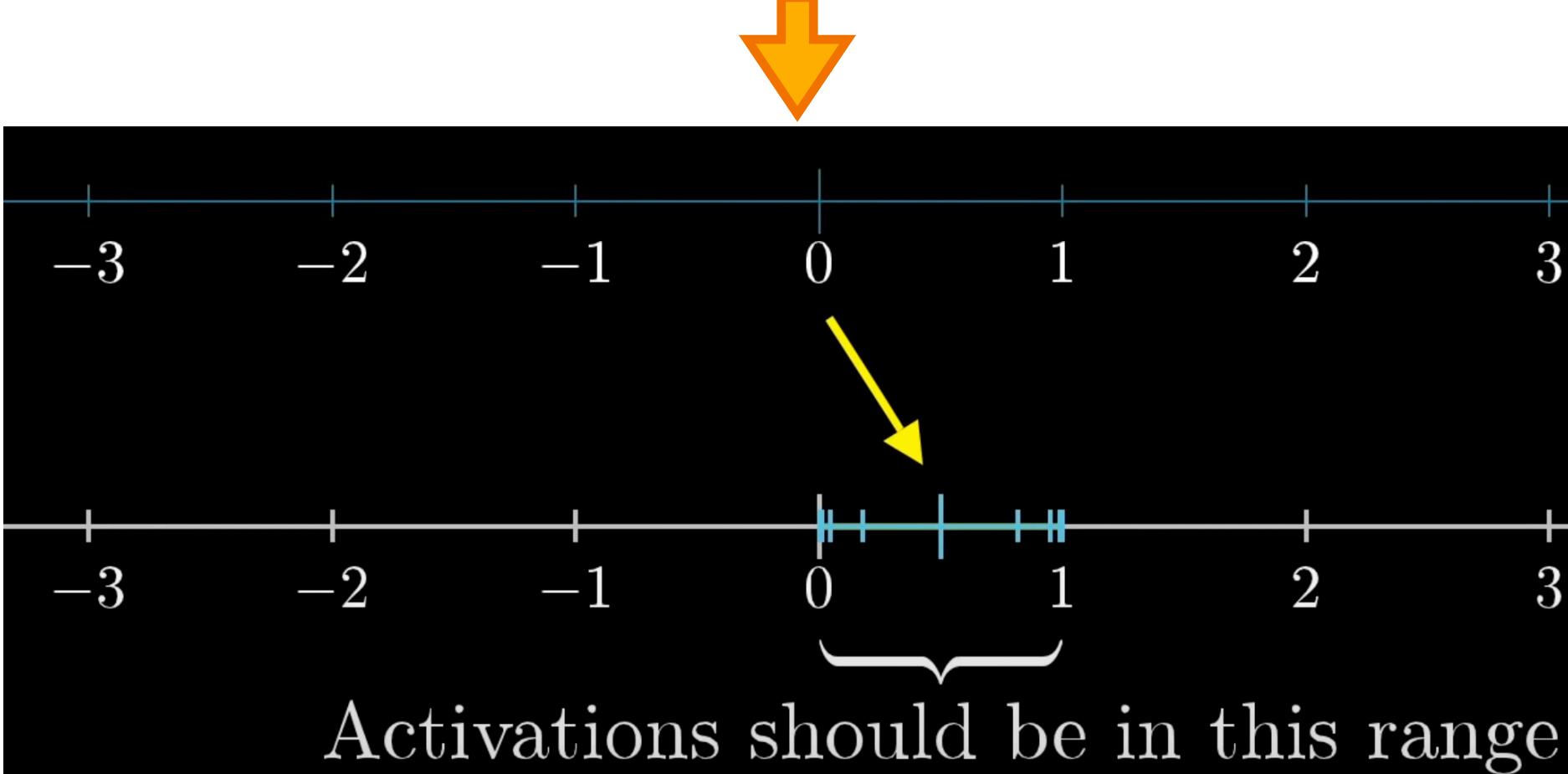
We also need values between 0 and 1 for the activation of our new neuron therefore we need an additional step

The parameters of the Network (Weights and Biases)



$$w_1 a_1 + w_2 a_2 + w_3 a_3 + w_4 a_4 + \dots + w_n a_n$$

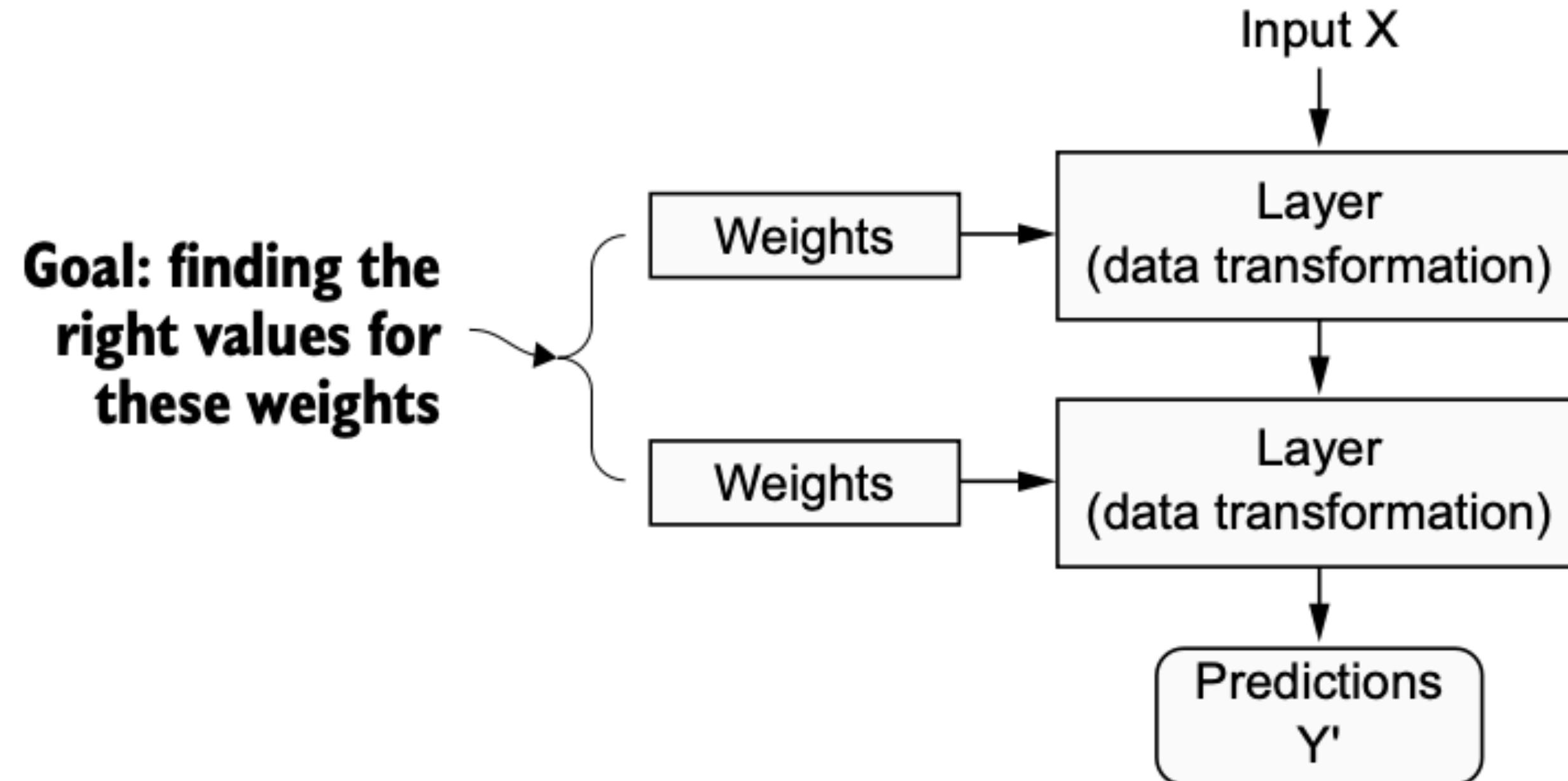
But we need to combine the activations and the weights



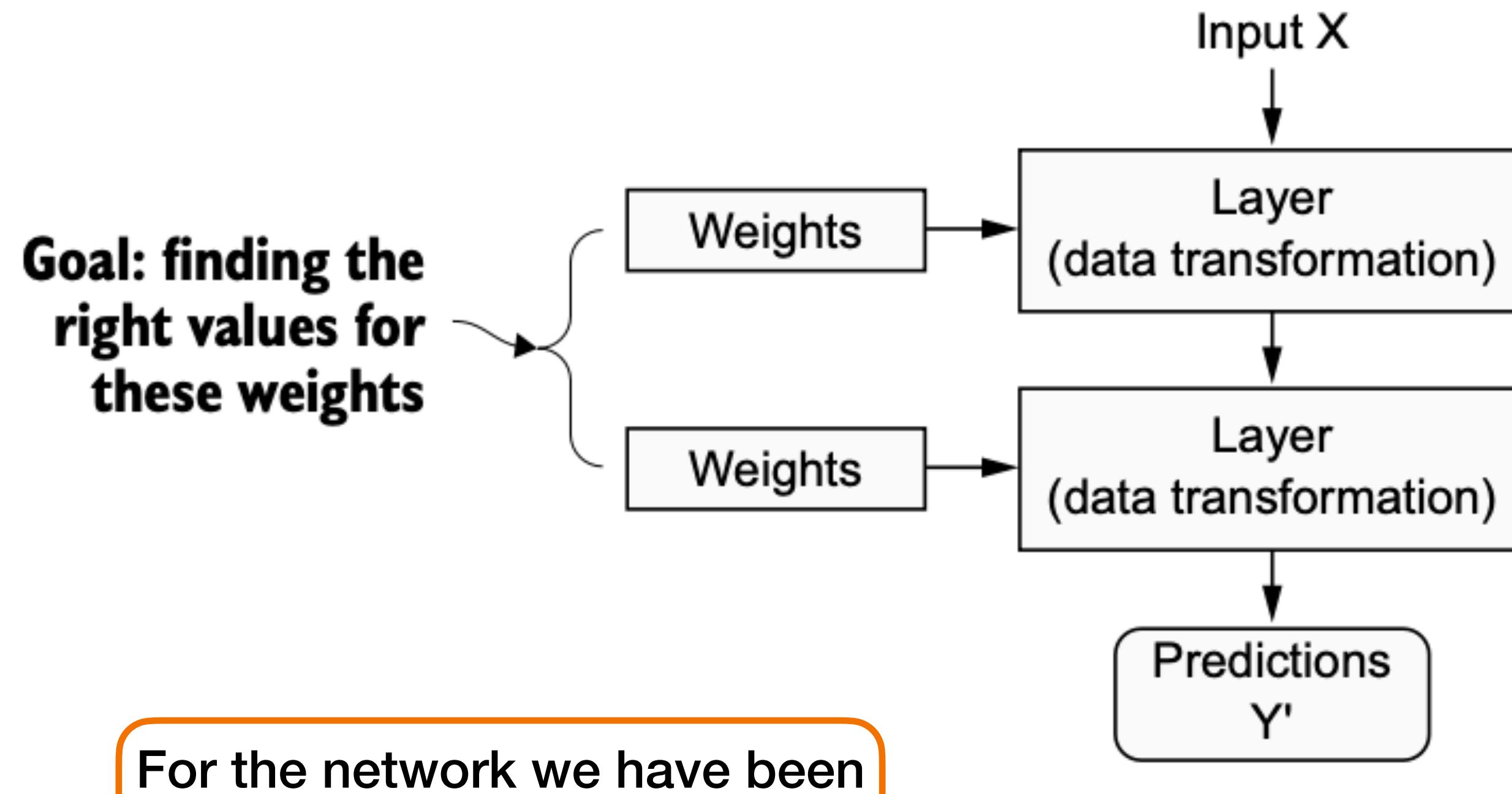
We also need values between 0 and 1 for the activation of our new neuron therefore we need an additional step

Neural Networks also present a bias parameter (How high the Weighted sum needs to be before the neuron becomes active)

Recap



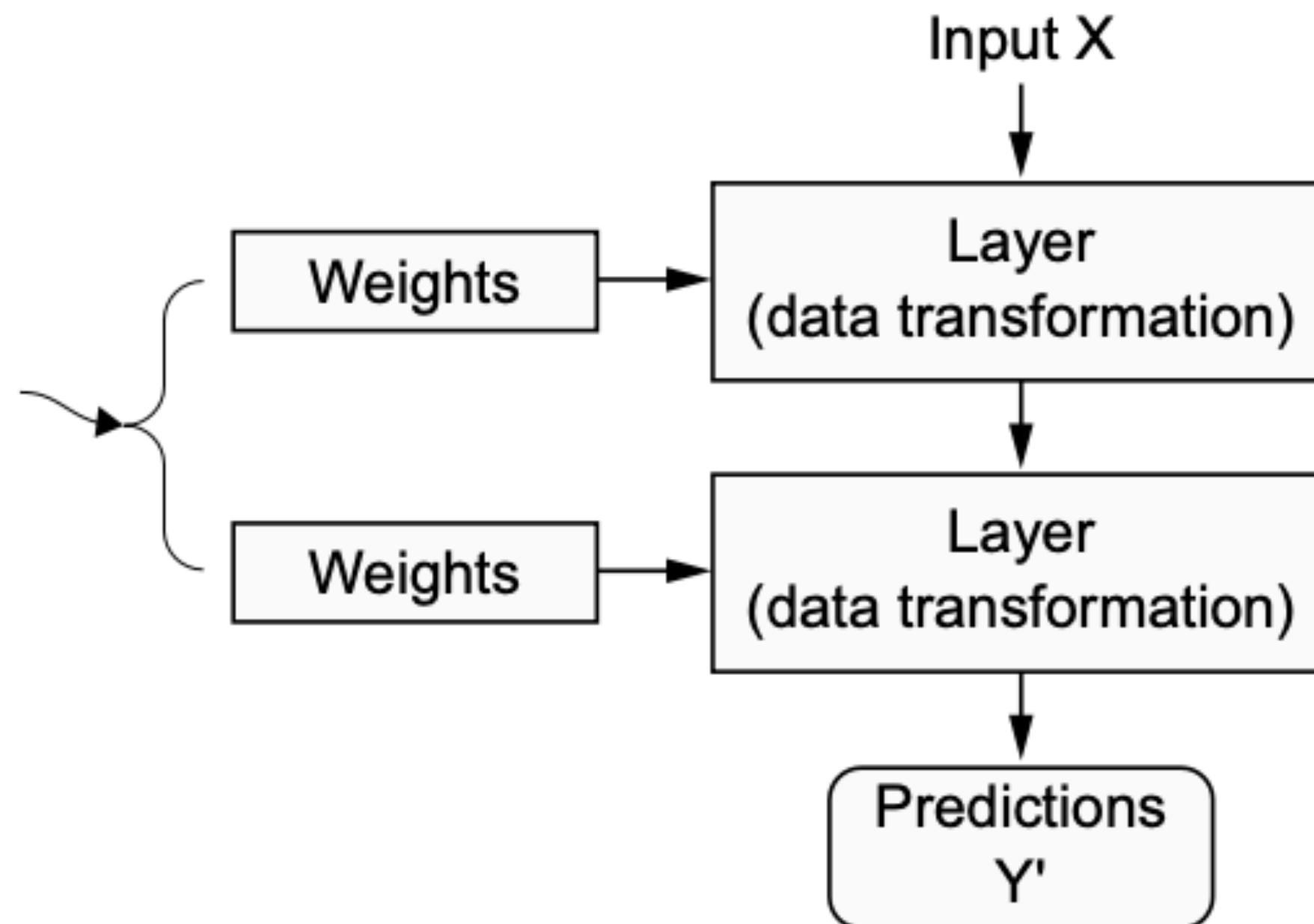
Recap



For the network we have been using we have $> 13\text{ K}$ weights and biases

Recap

Goal: finding the right values for these weights



New Neuron Definition

Is a function that squeezes all the activations from the neurons in the previous layers and their corresponding weights to a value between 0 and 1

$$\mathbf{a}^{(1)} = \sigma(\mathbf{W}\mathbf{a}^{(0)} + \mathbf{b})$$



How do Neural Networks learn?

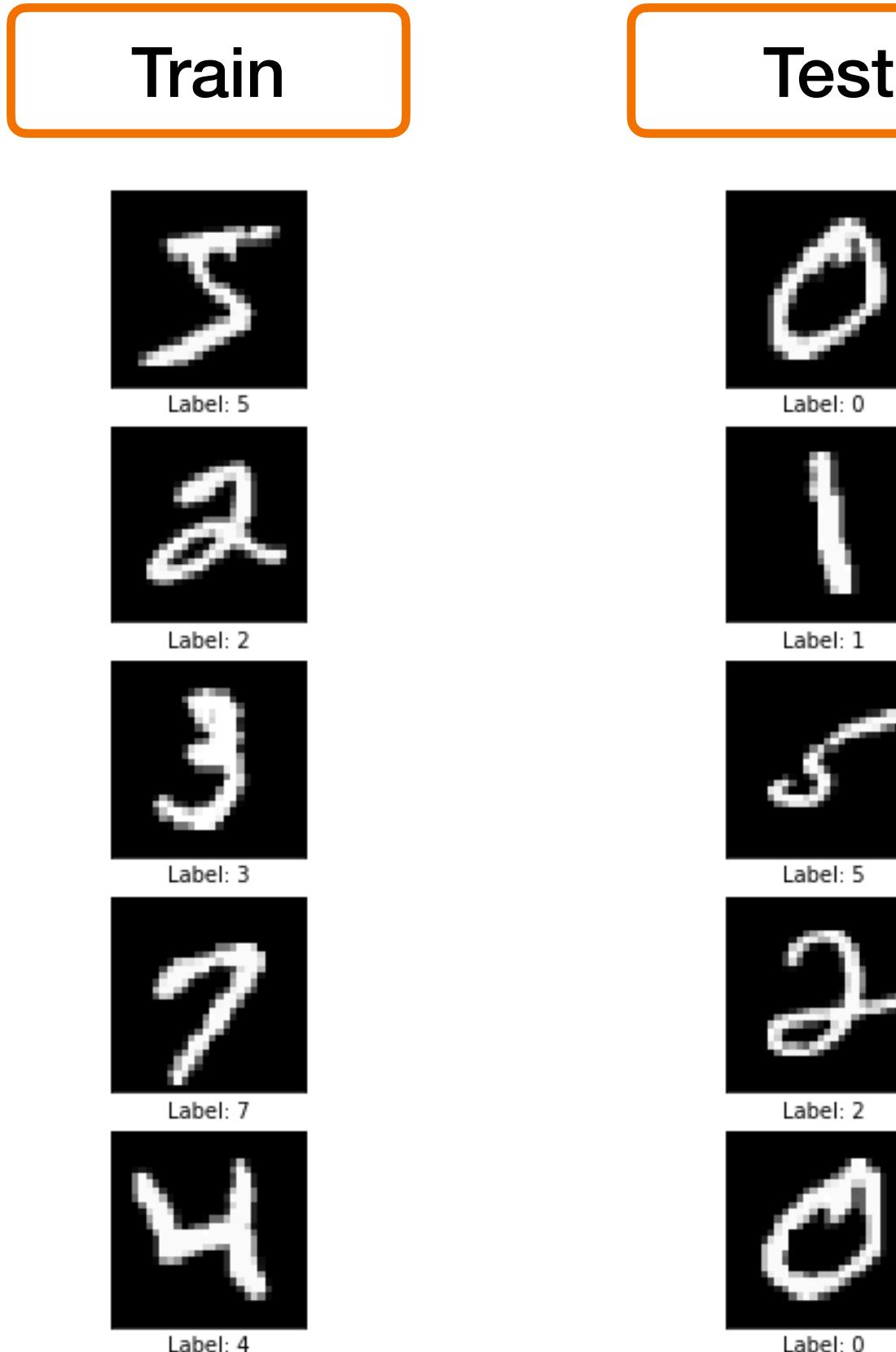
What are we gonna learn?

- 1. Forward Pass**
- 2. Backward Pass**
- 3. Update Weights**

What are we gonna learn?

- 1. Forward Pass**
- 2. Backward Pass**
- 3. Update Weights**

- The objective is two-fold:
- To train the network to recognize images of numbers
 - To have a model able to generalize for images beyond that training data



How accurately does the network classify those new images?

The first step: Randomly initialize the parameters of the network

Weights

Biases

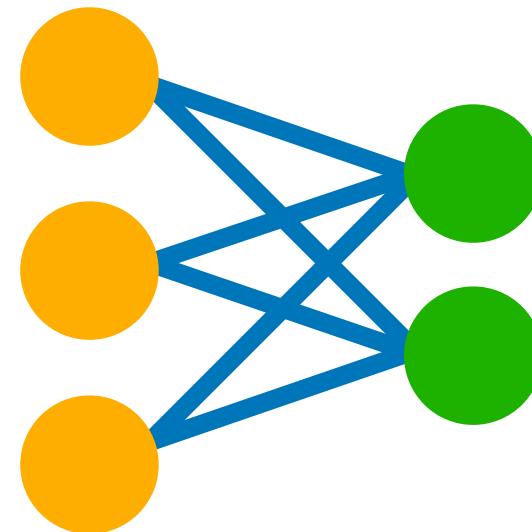
$$a^{(1)} = \sigma(Wa^{(0)} + b)$$

The first step: Randomly initialize the parameters of the network

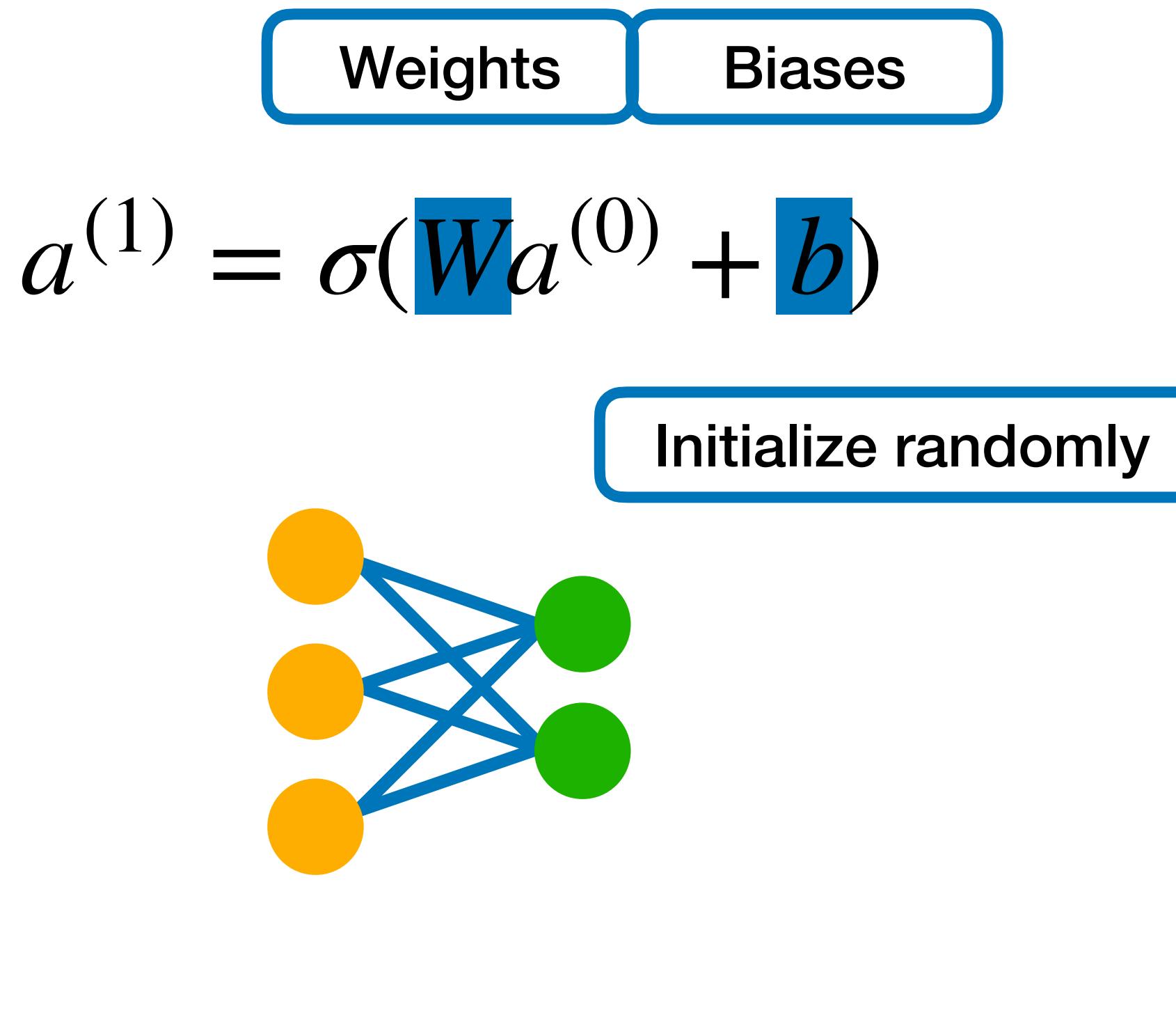
Weights Biases

$$a^{(1)} = \sigma(Wa^{(0)} + b)$$

Initialize randomly

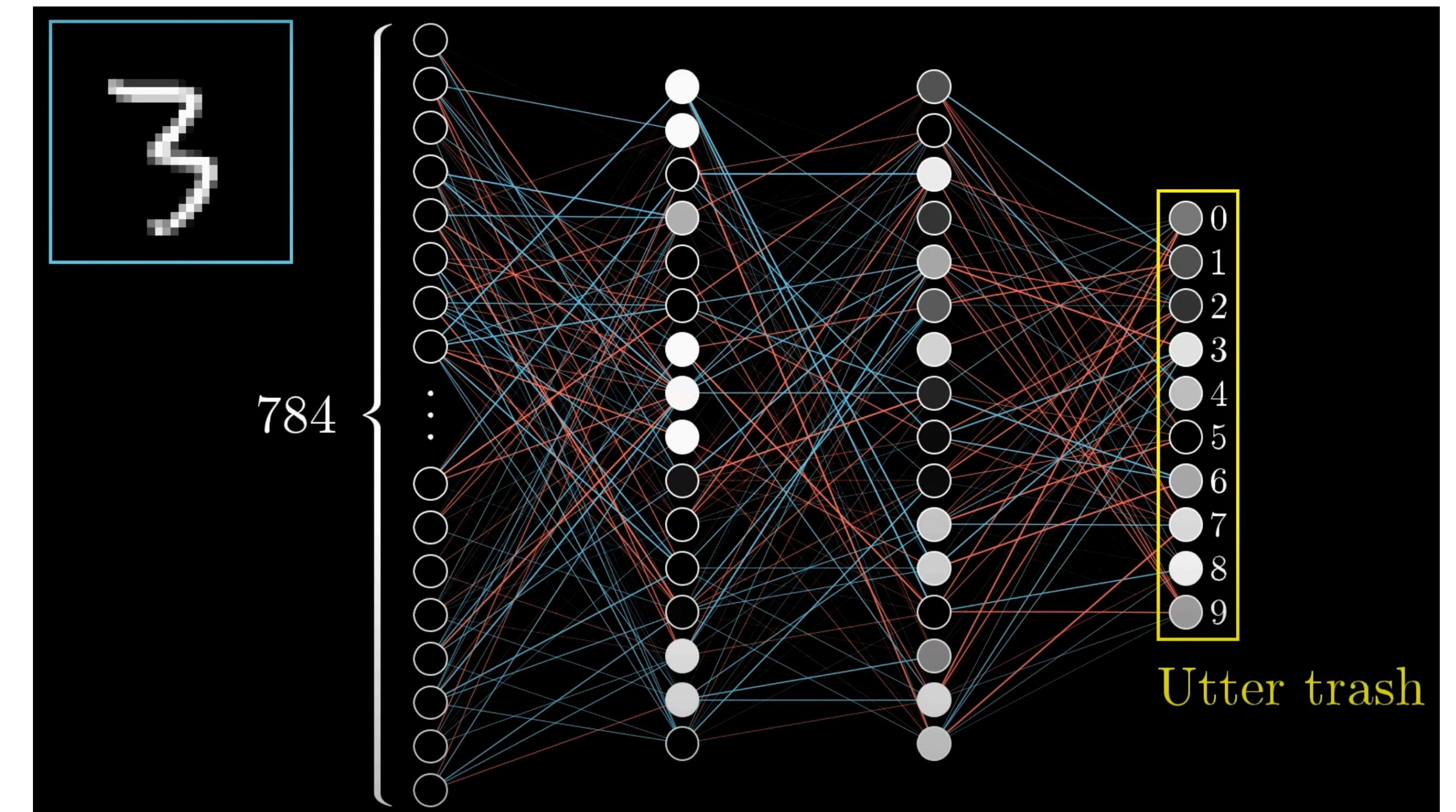
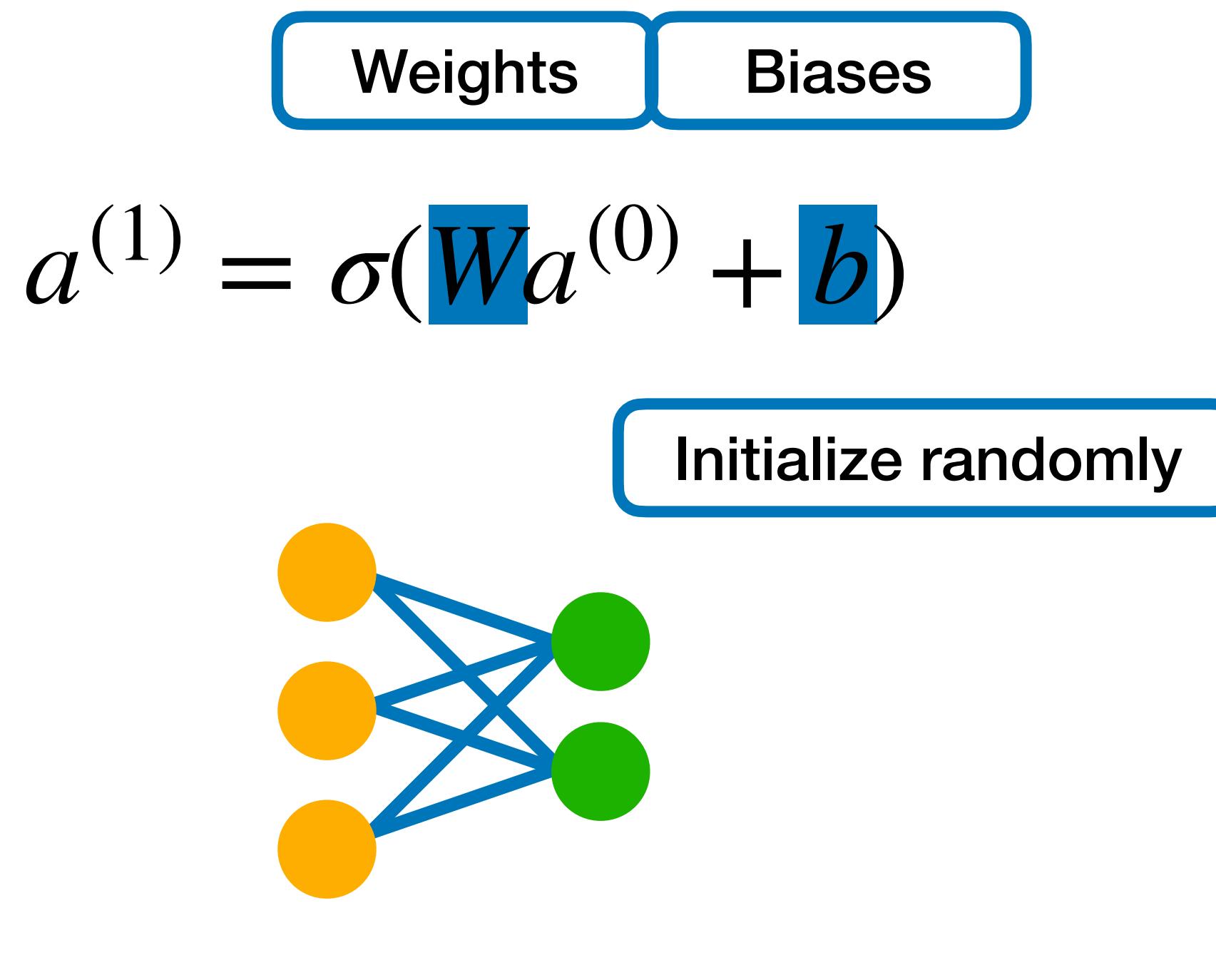


The first step: Randomly initialize the parameters of the network

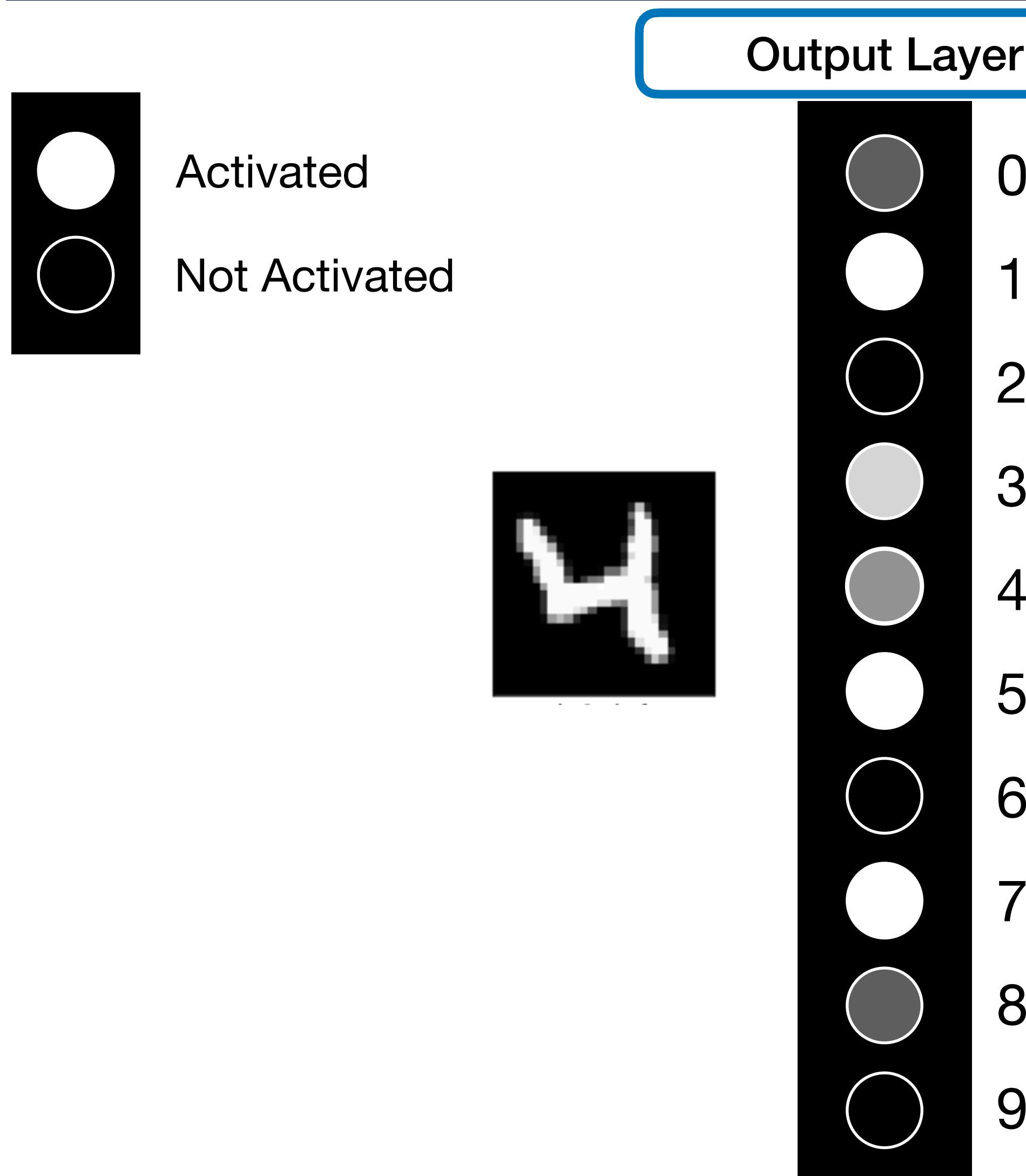


What do you think
will happen?

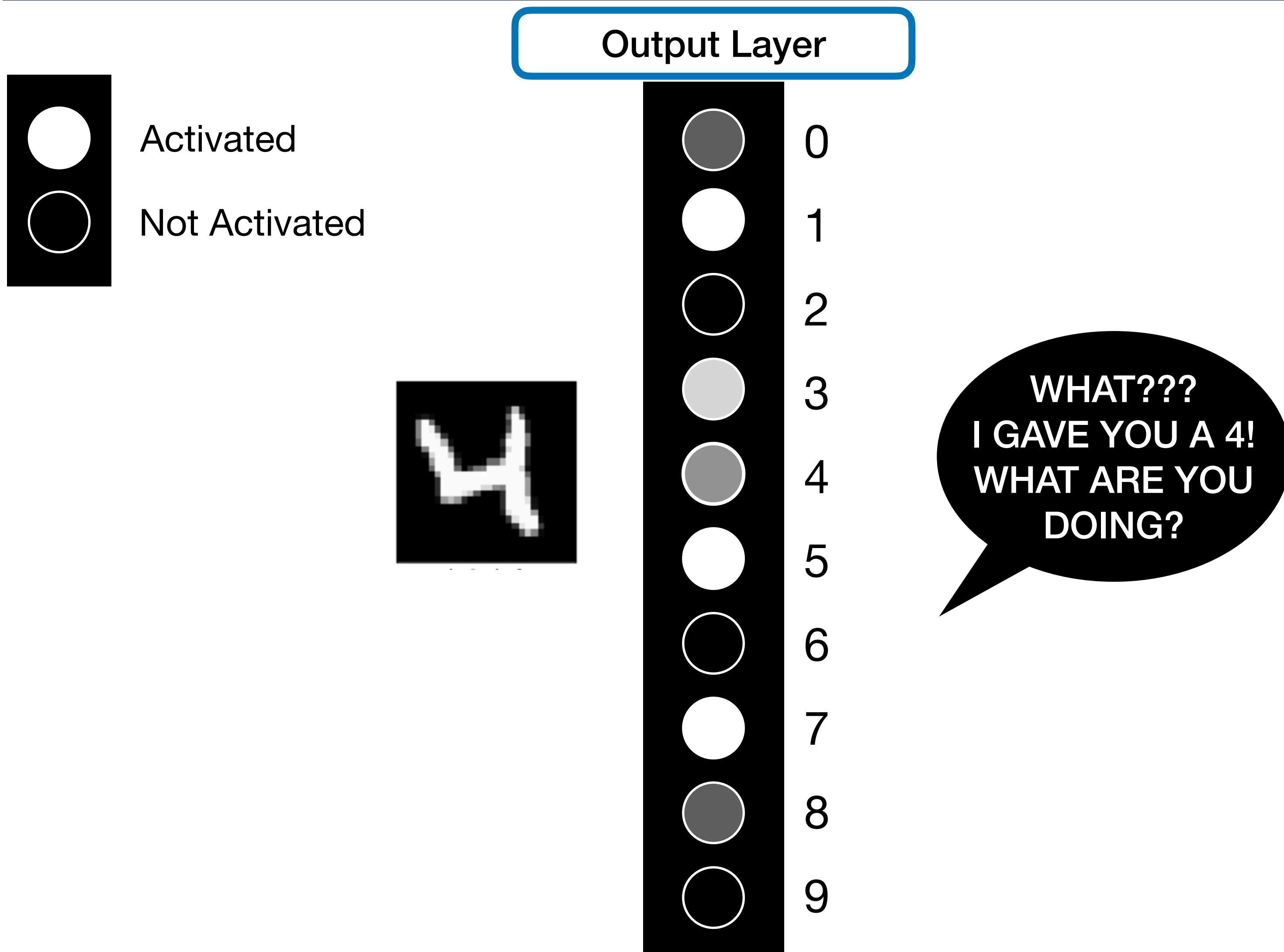
First step: Randomly initialize the parameters of the network



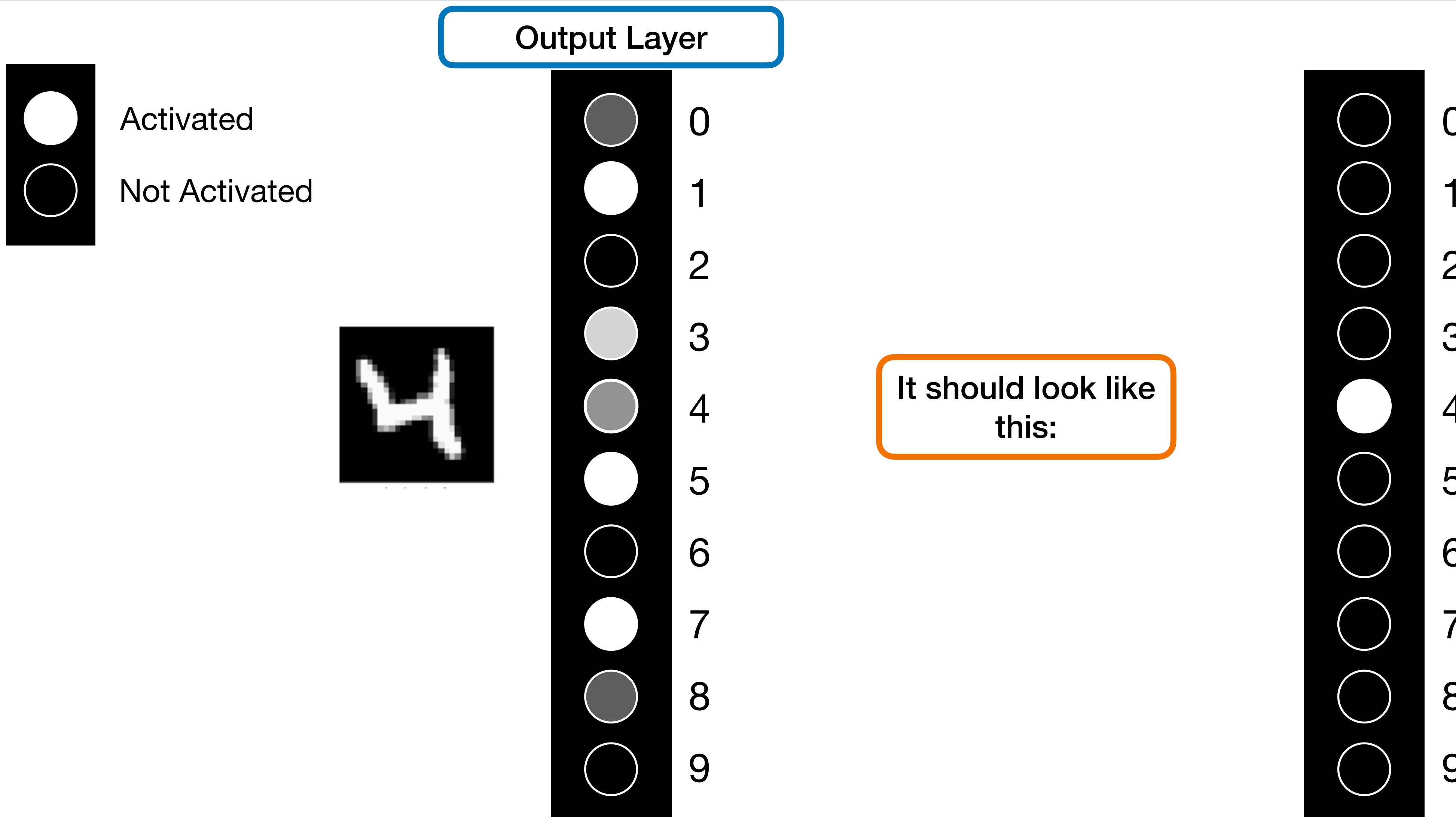
Second step: Tell the computer that is doing it wrong



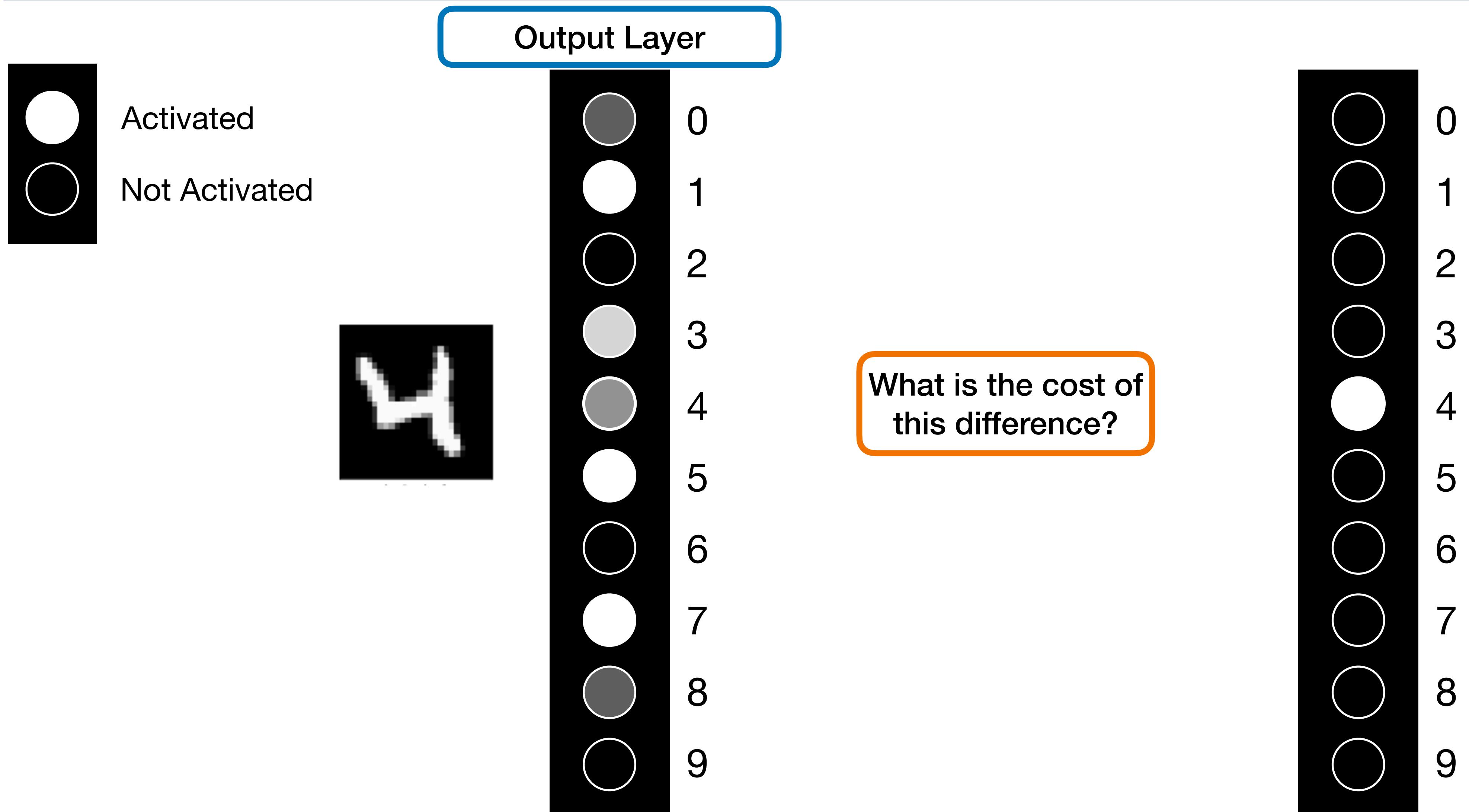
Second step: Tell the computer that is doing it wrong



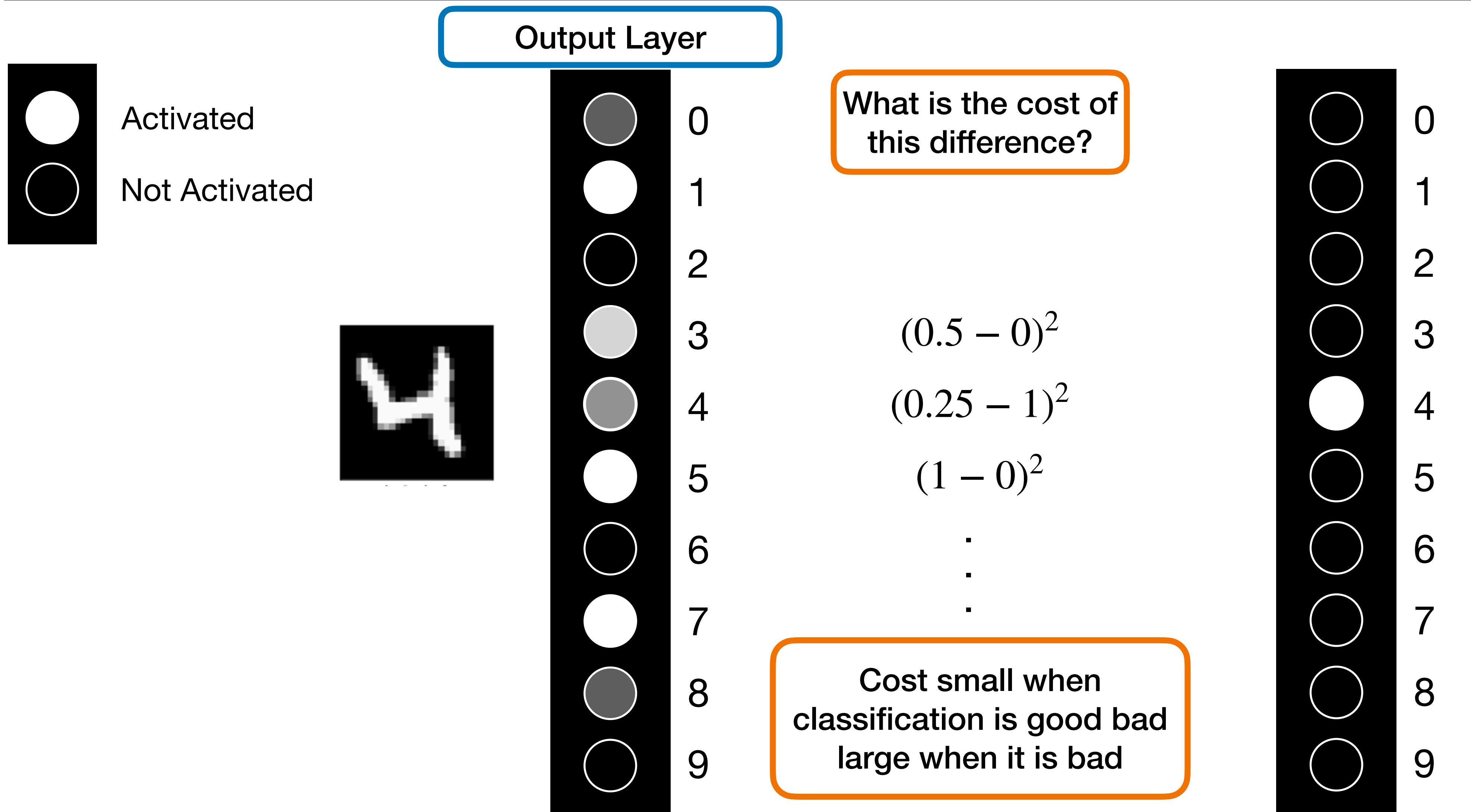
Second step: Tell the computer that is doing it wrong



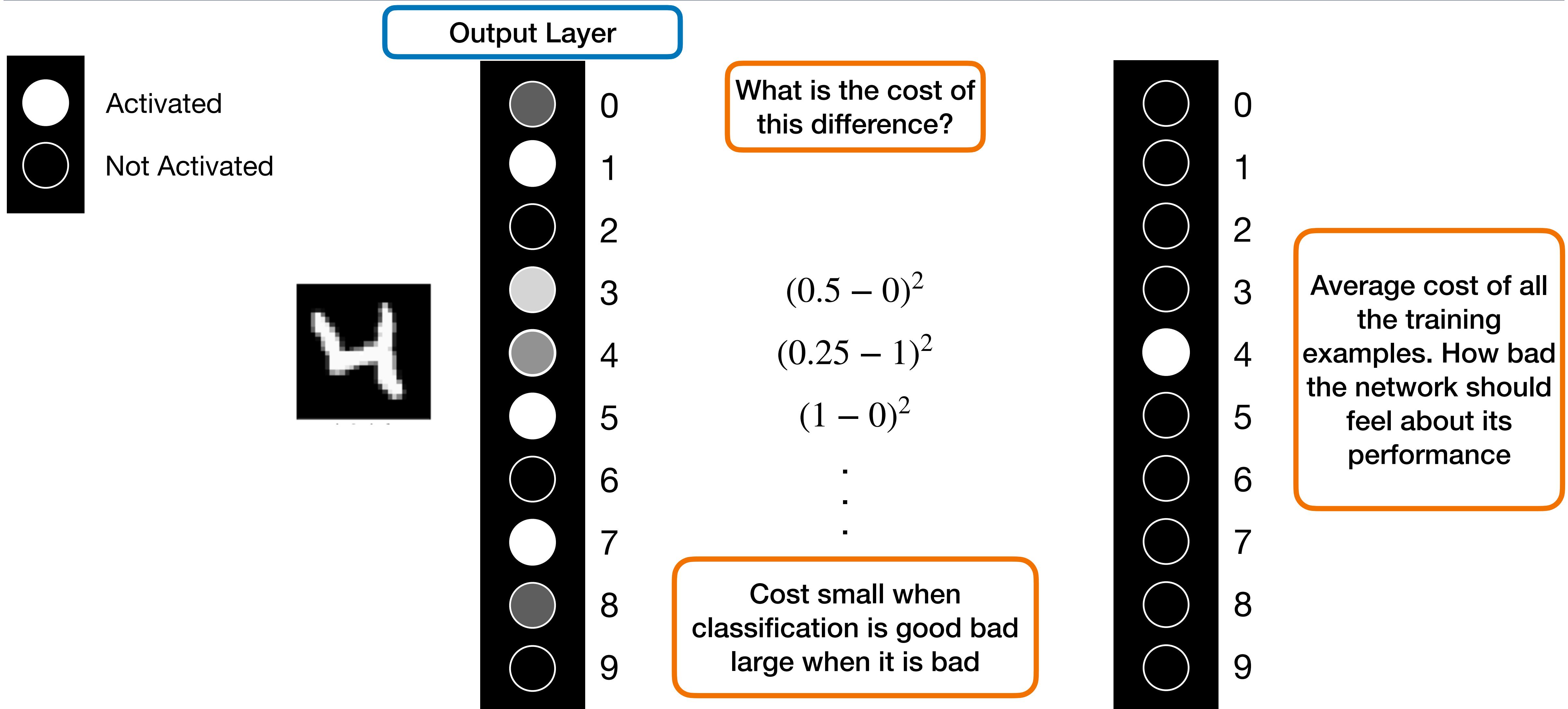
Second step: Tell the computer that is doing it wrong



Second step: Tell the computer that is doing it wrong

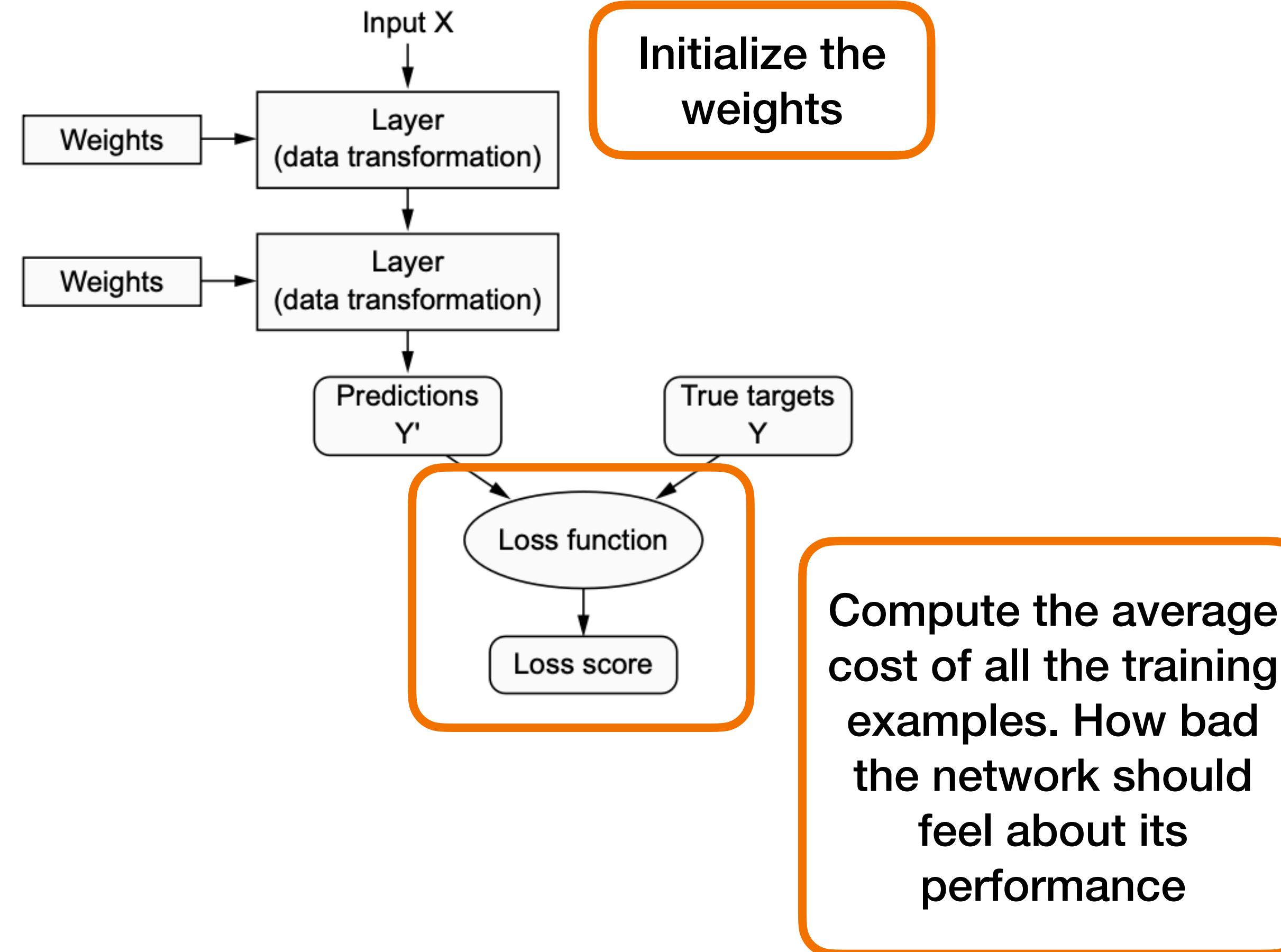


Second step: Tell the computer that is doing it wrong



Small Recap: We found a number to tell how the model is doing

Forward Pass

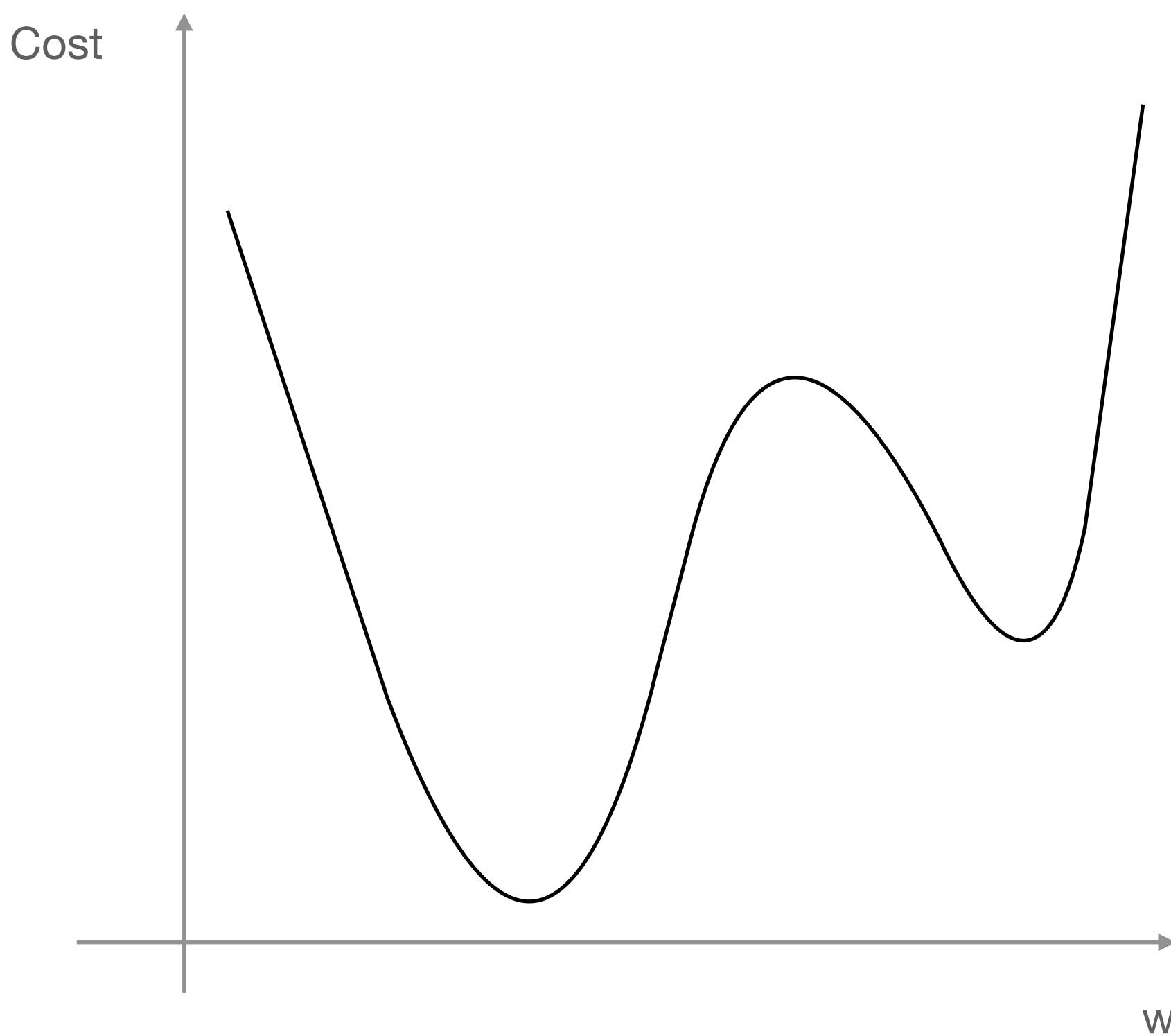


How do we improve the network (weights and biases) now that we know how is it doing it?

How do we propagate this cost backwards?

How do we propagate this cost backwards?

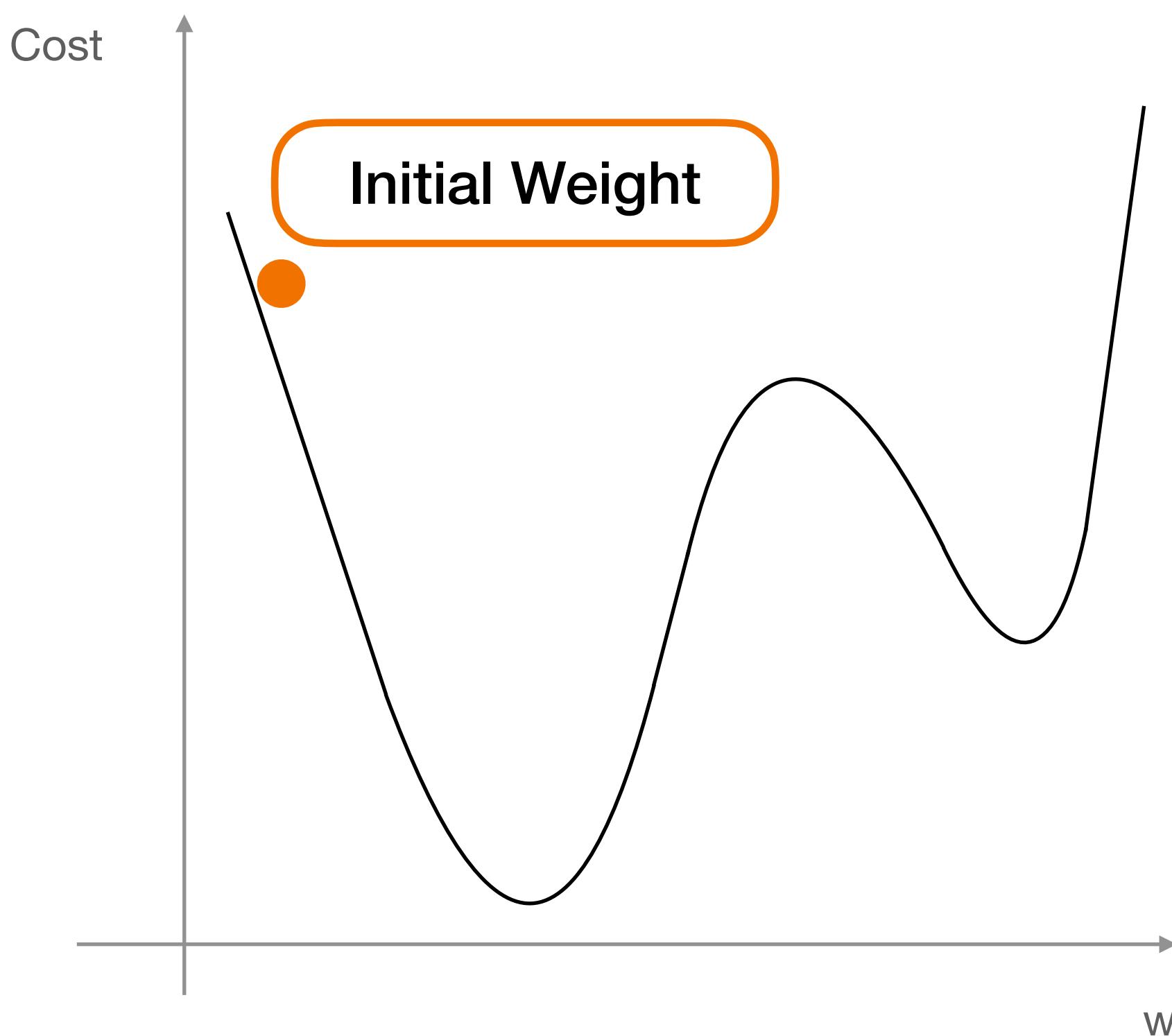
First, we need to look at the cost function



How do we propagate this cost backwards?

First, we need to look at the cost function

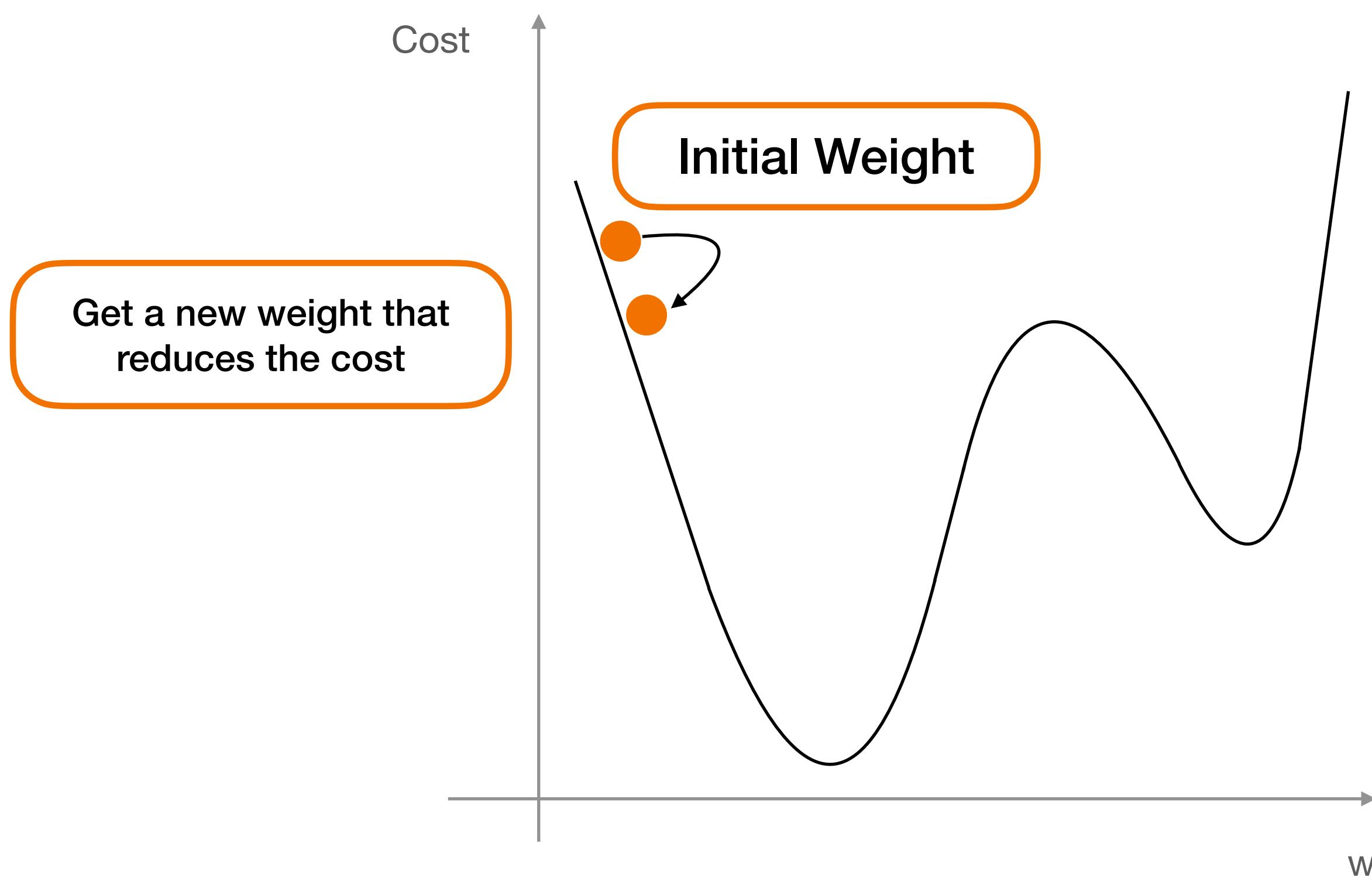
Objective: Reduce the cost



How do we propagate this cost backwards?

First, we need to look at the cost function

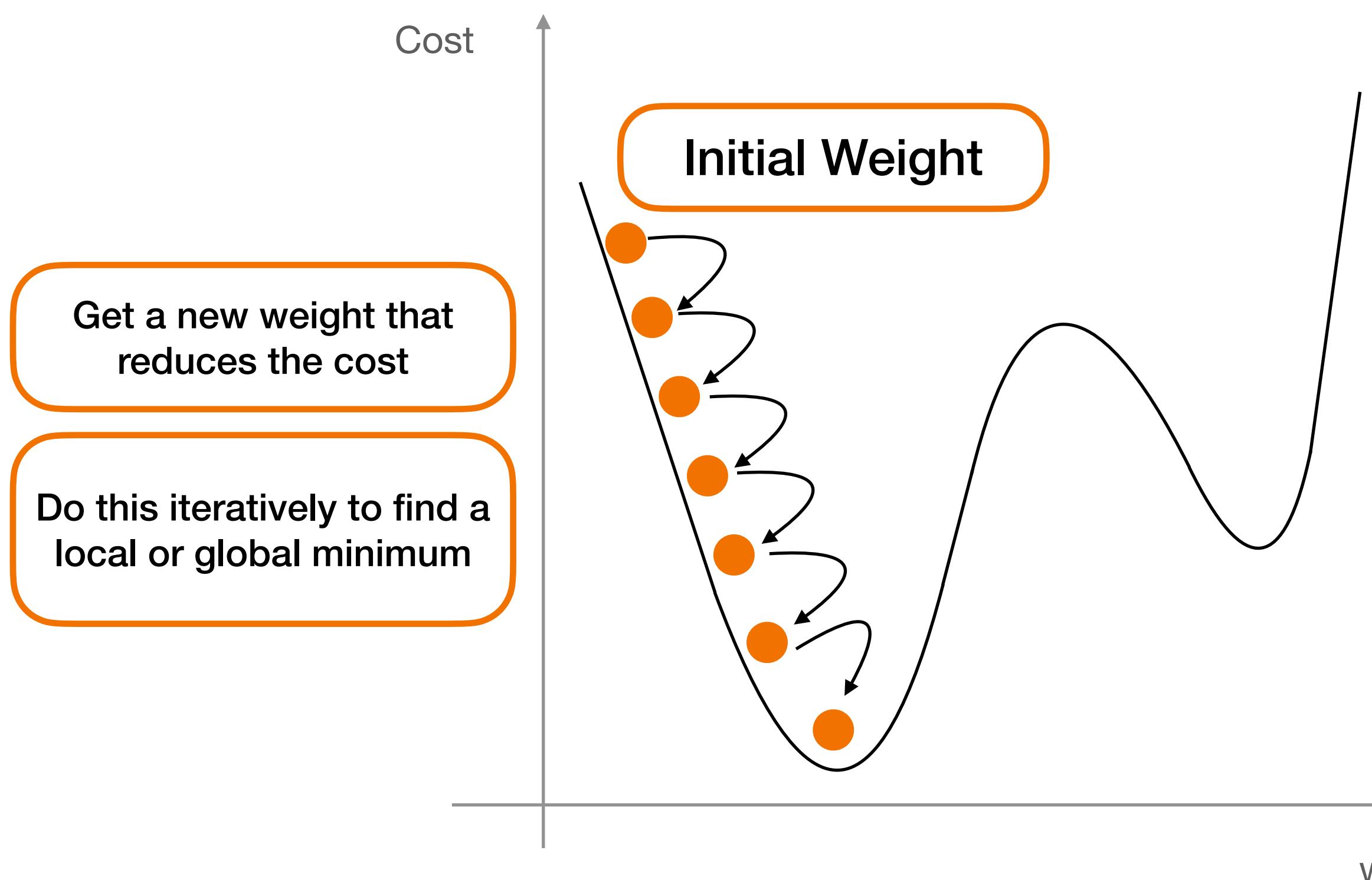
Objective: Reduce the cost



How do we propagate this cost backwards?

First, we need to look at the cost function

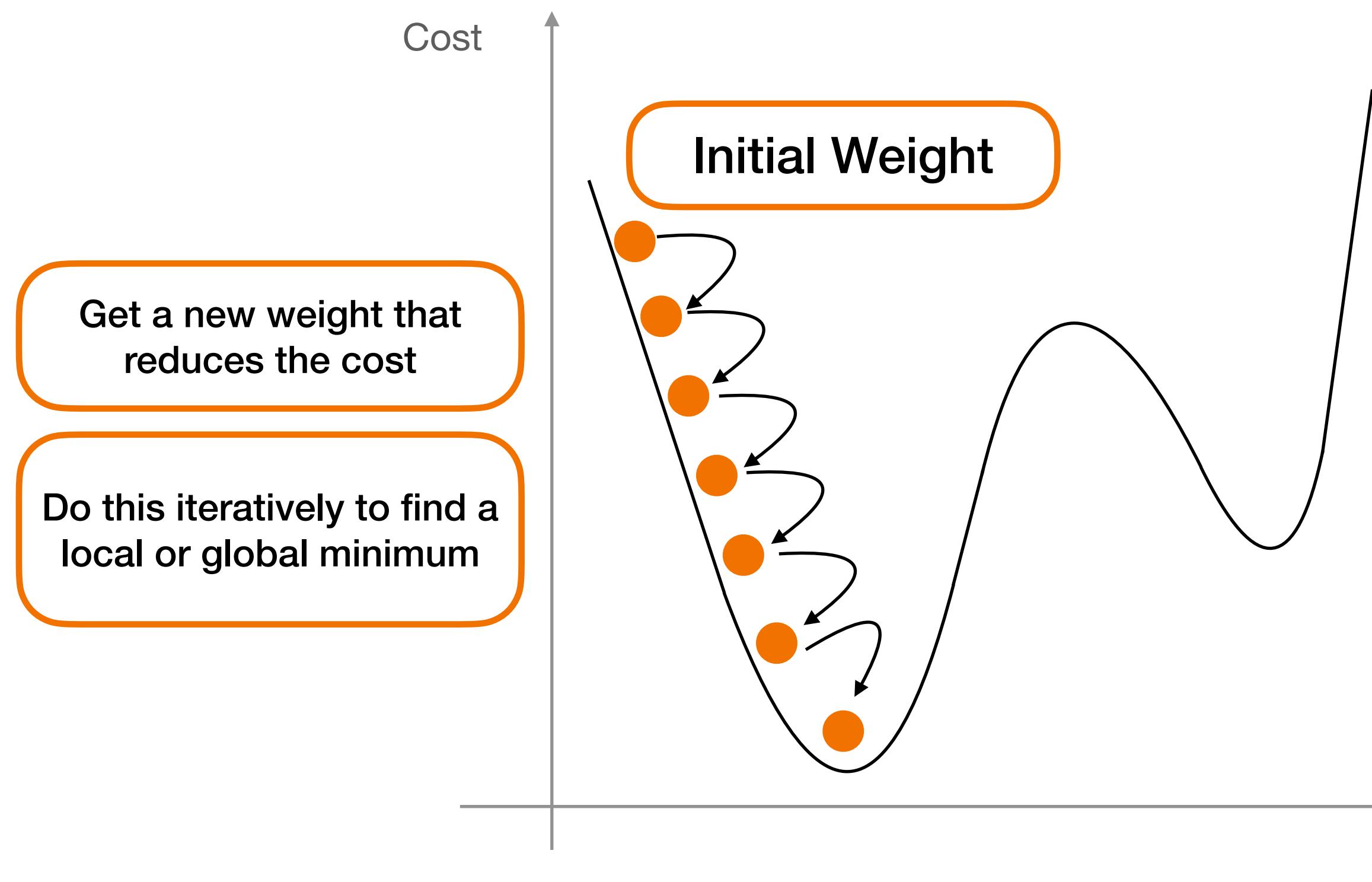
Objective: Reduce the cost



How do we propagate this cost backwards?

First, we need to look at the cost function

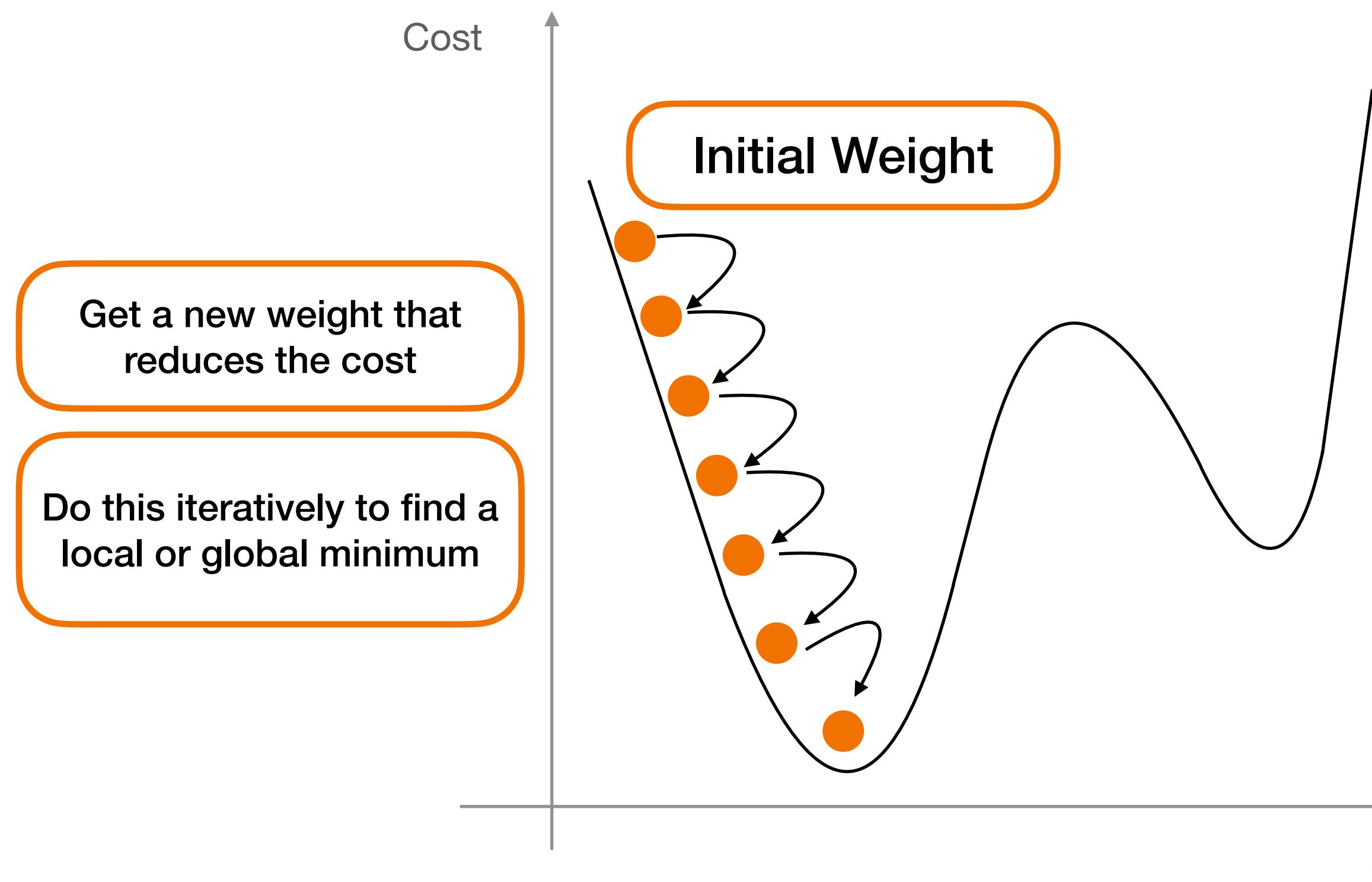
Objective: Reduce the cost



How do we propagate this cost backwards?

First, we need to look at the cost function

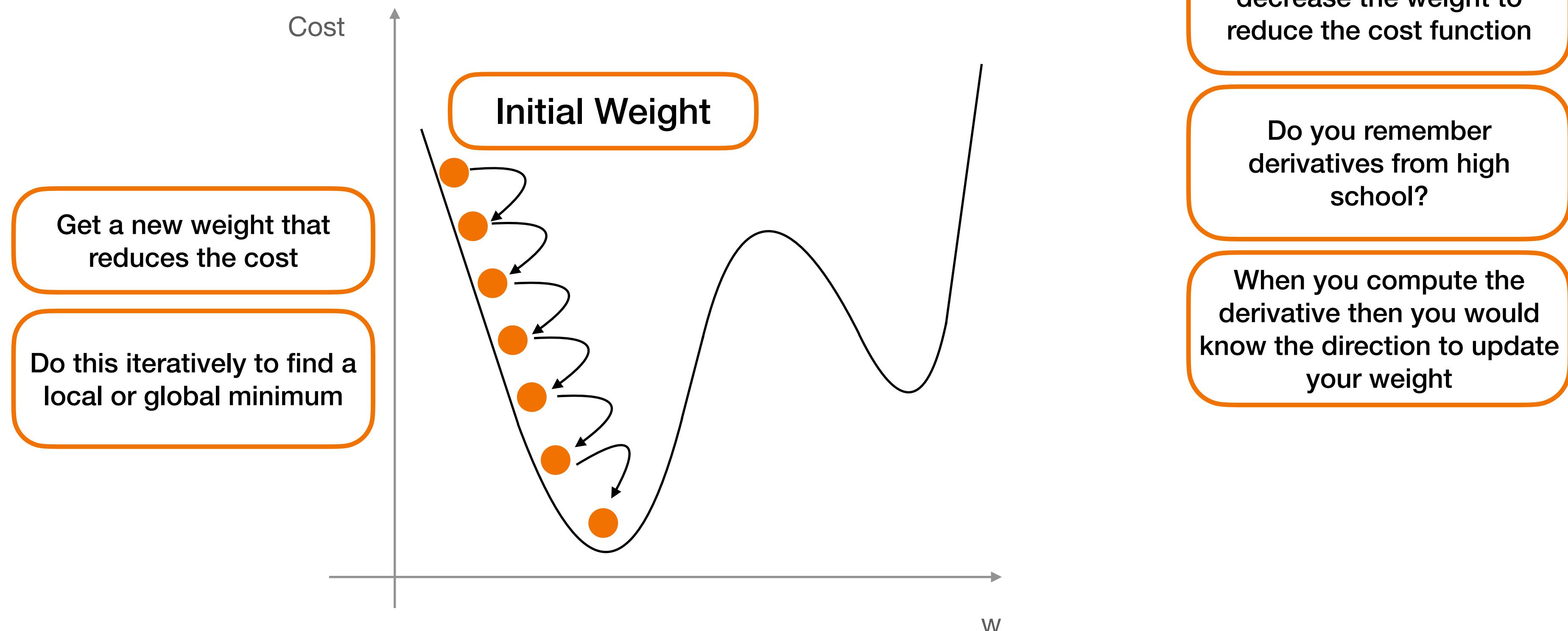
Objective: Reduce the cost



How do we propagate this cost backwards?

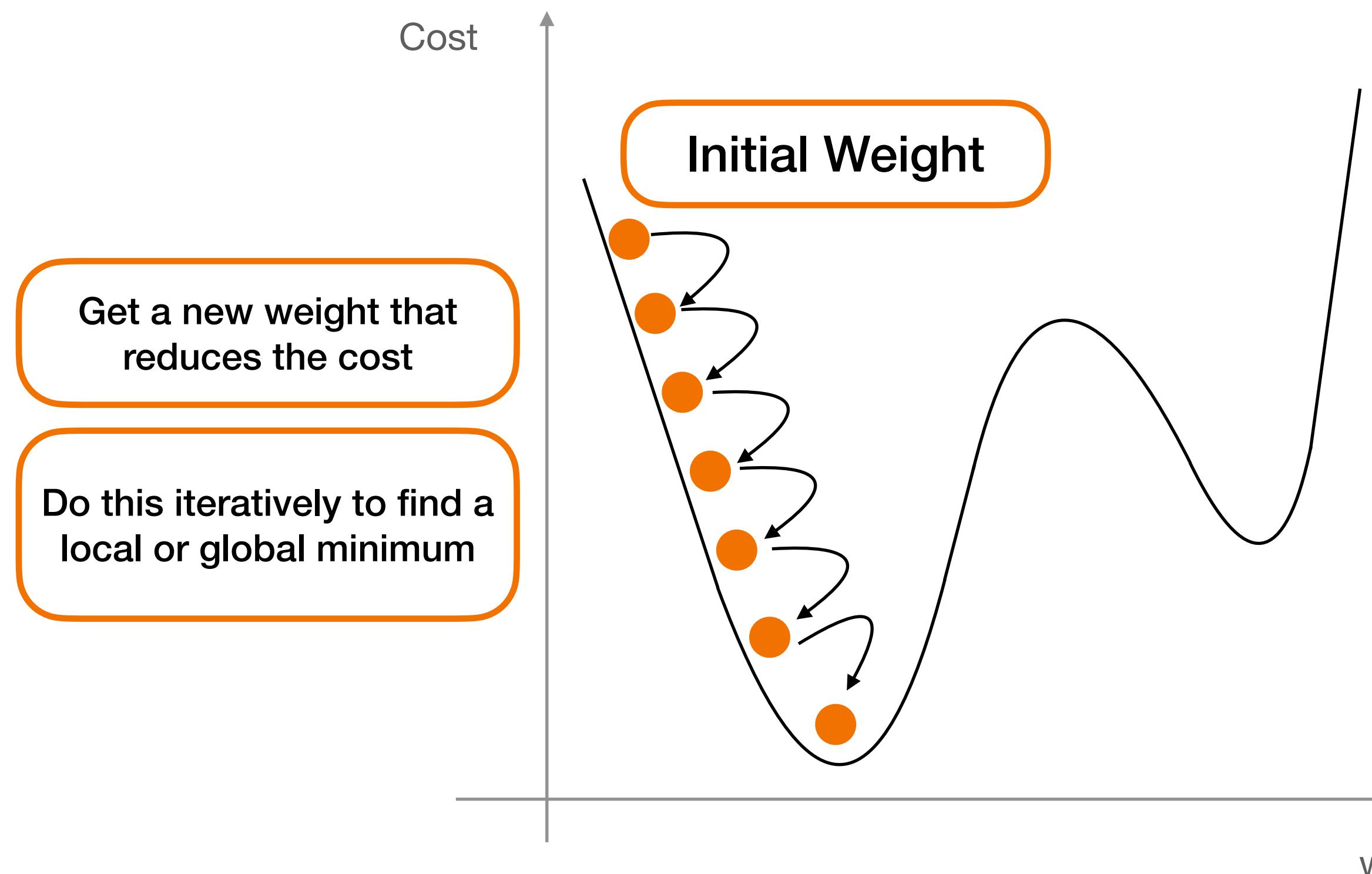
First, we need to look at the cost function

Objective: Reduce the cost



How do we propagate this cost backwards?

First, we need to look at the cost function



Objective: Reduce the cost

Essentially you increase or decrease the weight to reduce the cost function

Do you remember derivatives from high school?

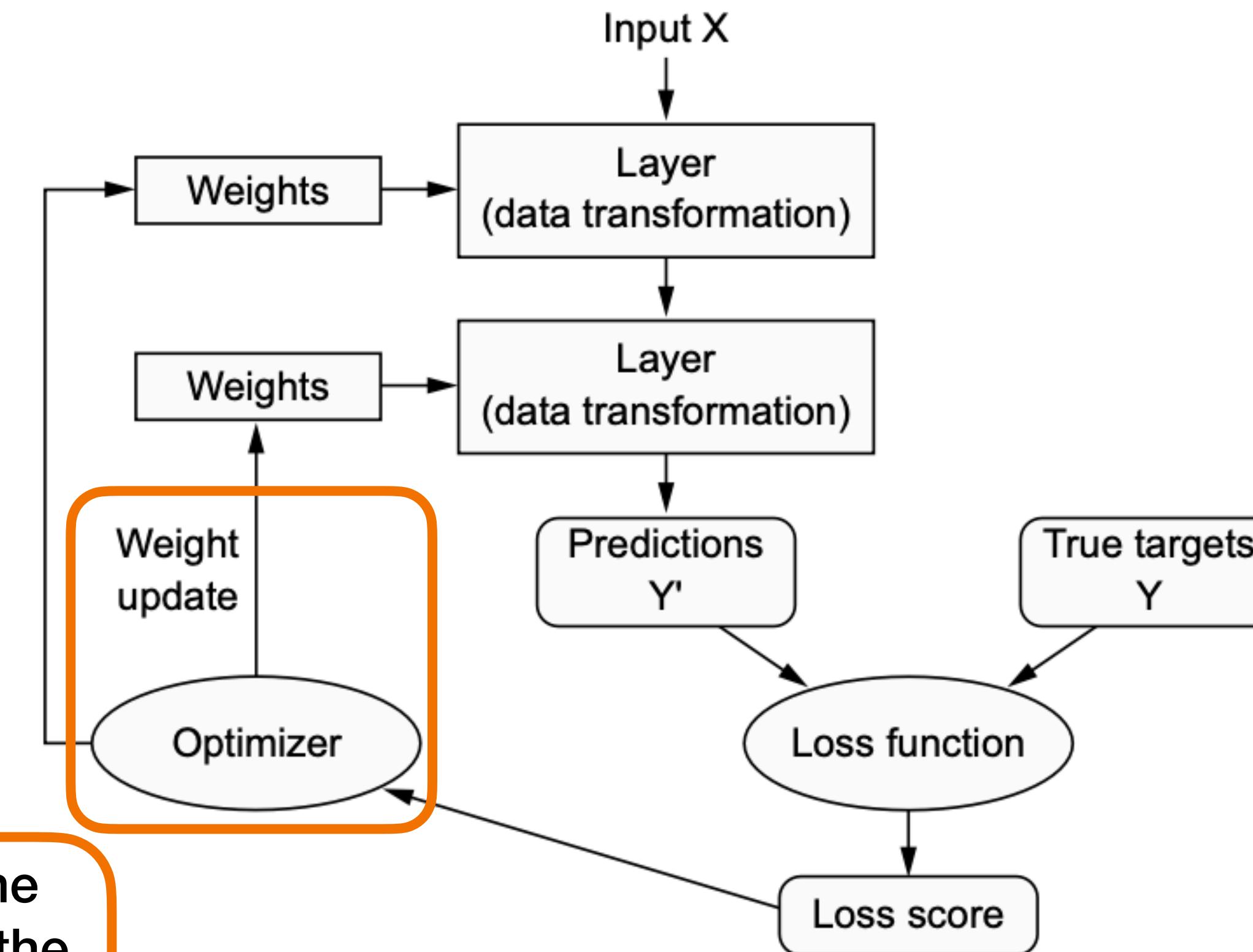
When you compute the derivative then you would know the direction to update your weight

As we have a lot of weights:
Multidimensional function,
we use the Gradient

Gradient Descent

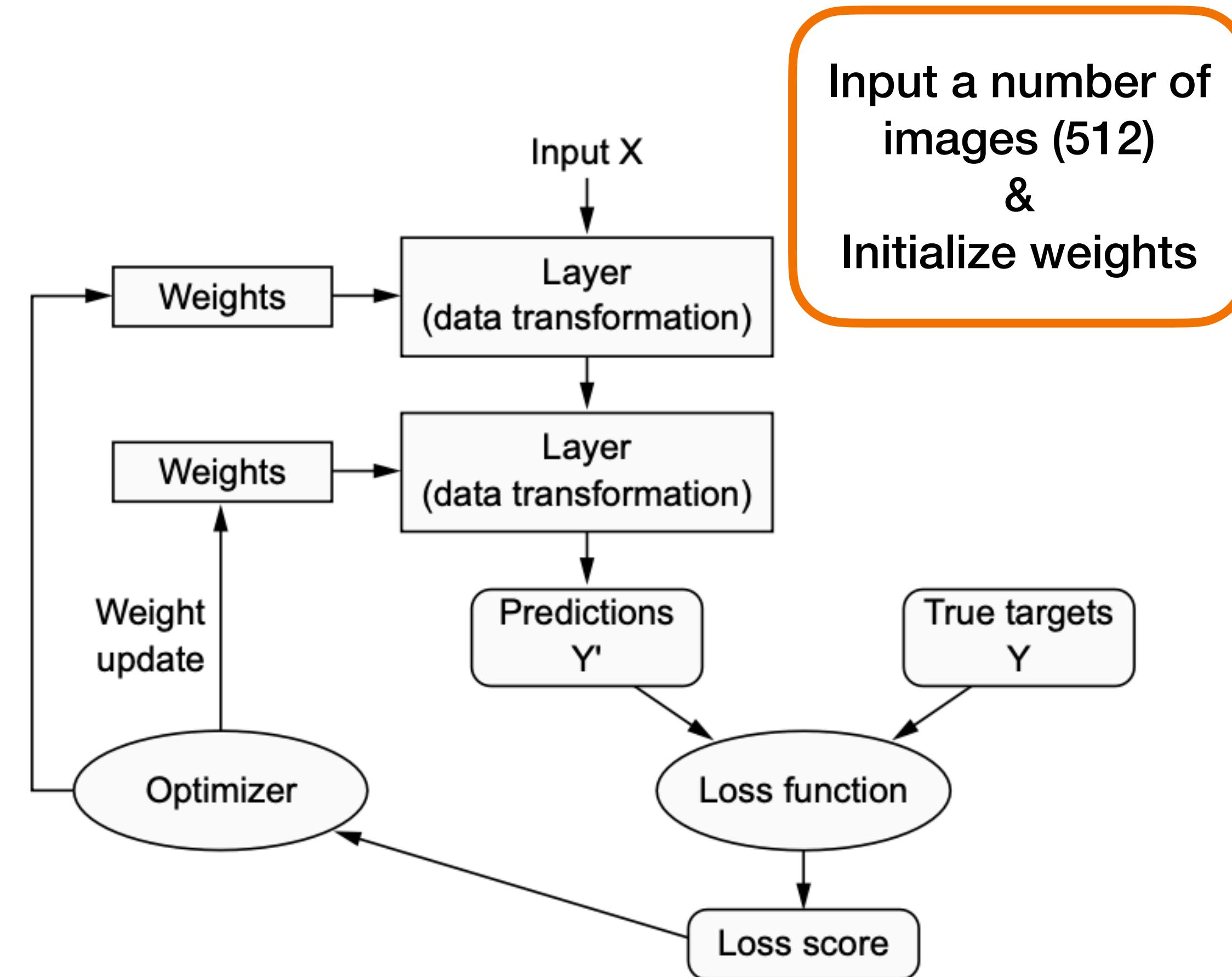
How do we propagate this cost backwards?

Backward Pass & Weight Update

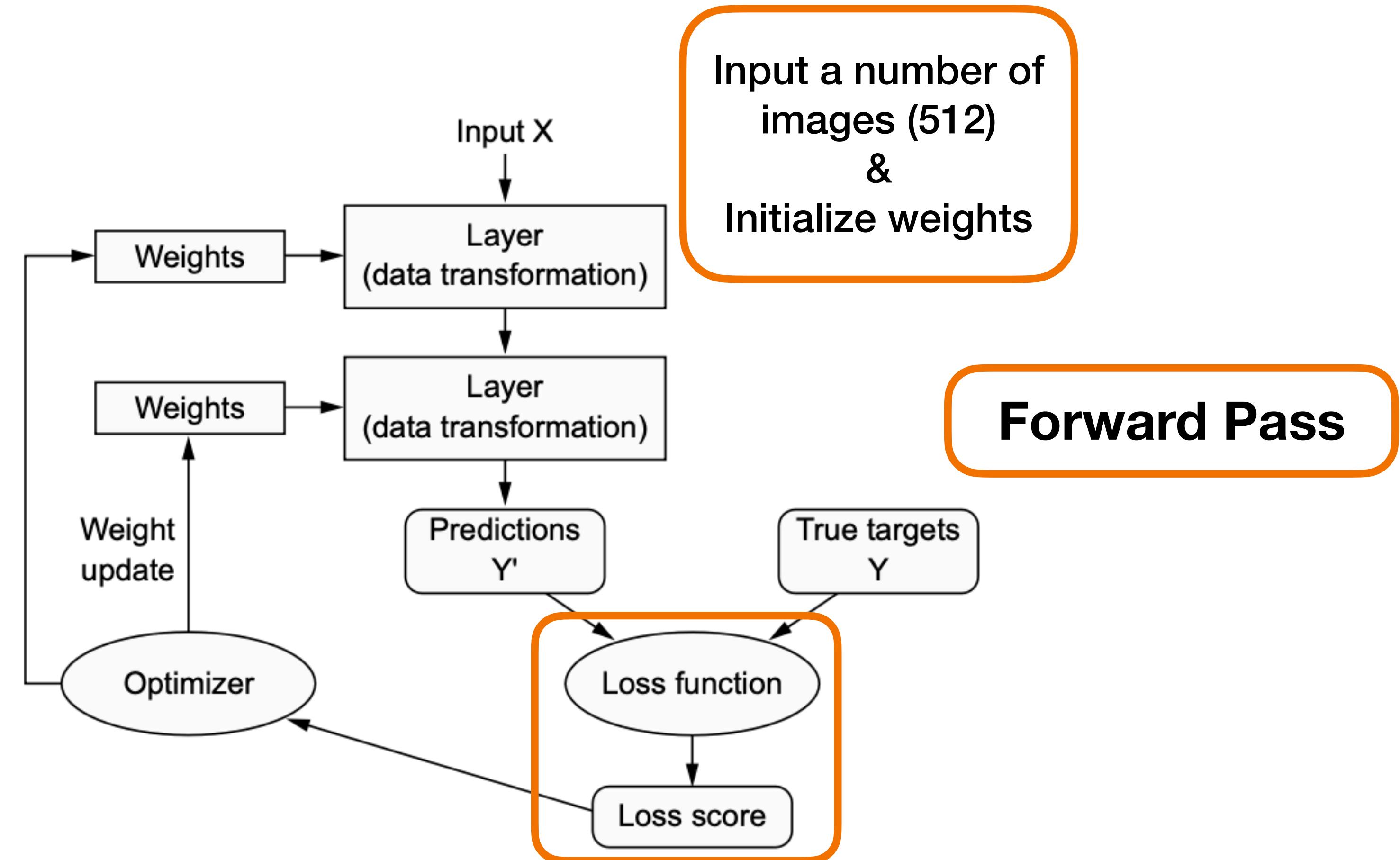


The algorithm to compute the gradients for every weight in the network efficiently is called backpropagation

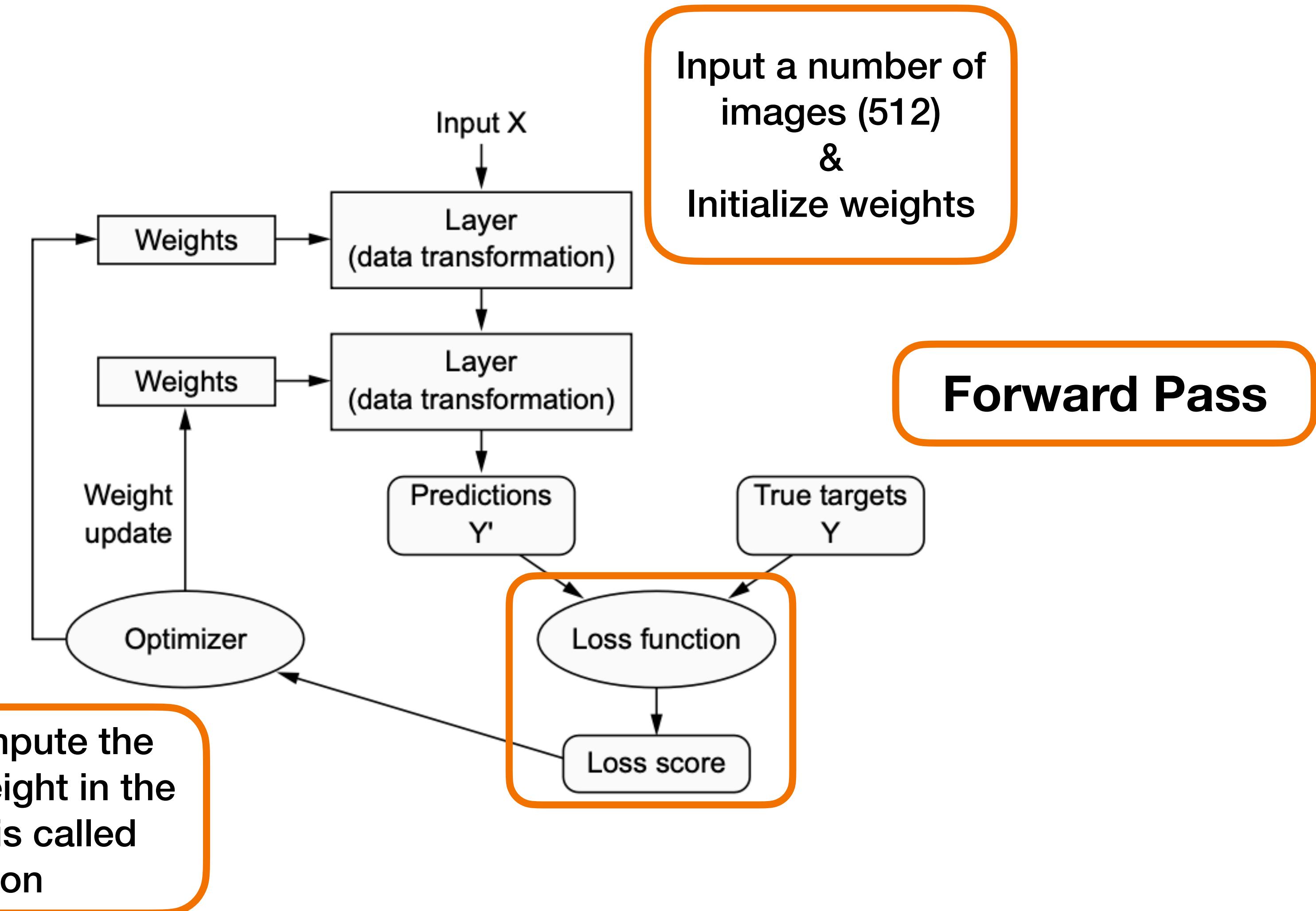
In reality: Stochastic Gradient Descent



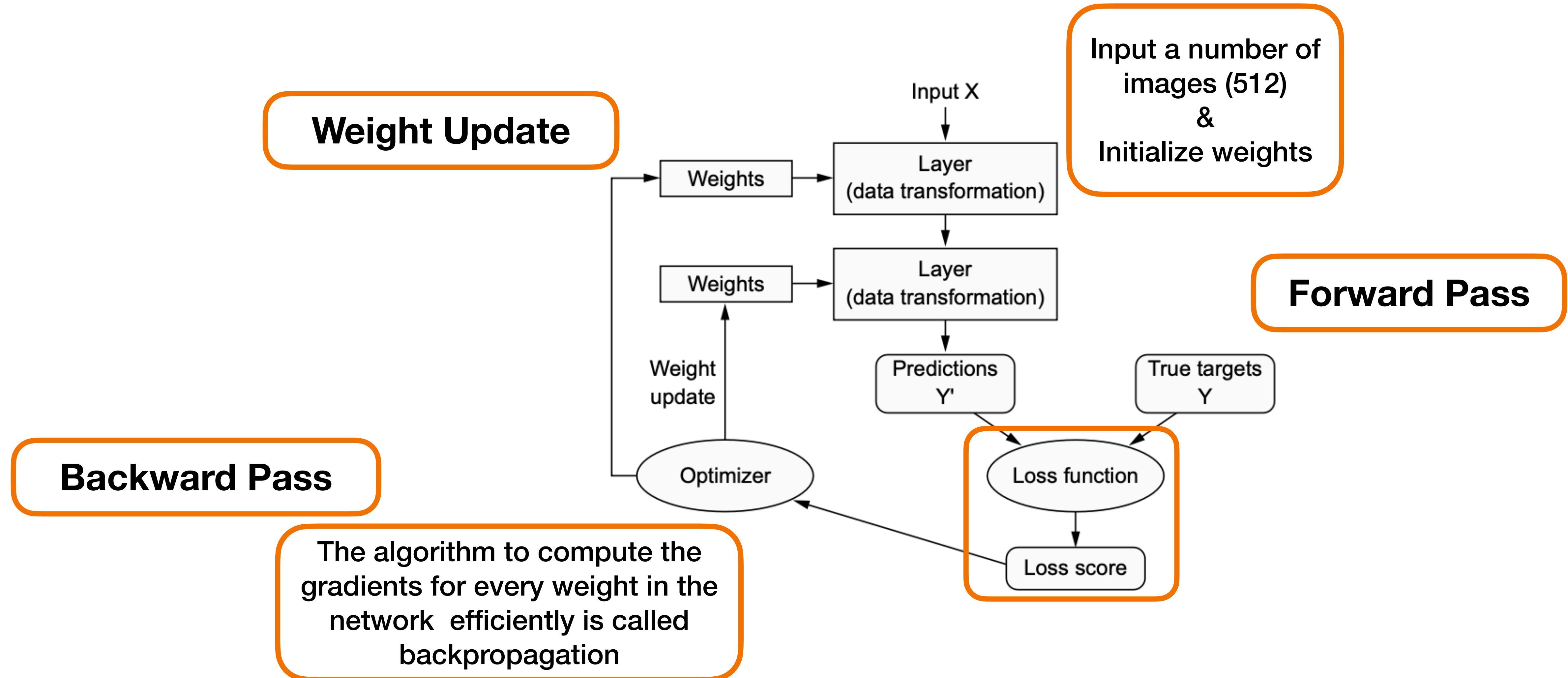
In reality: Stochastic Gradient Descent



In reality: Stochastic Gradient Descent



In reality: Stochastic Gradient Descent





Introduction to Keras. The MNIST dataset

What is Keras?

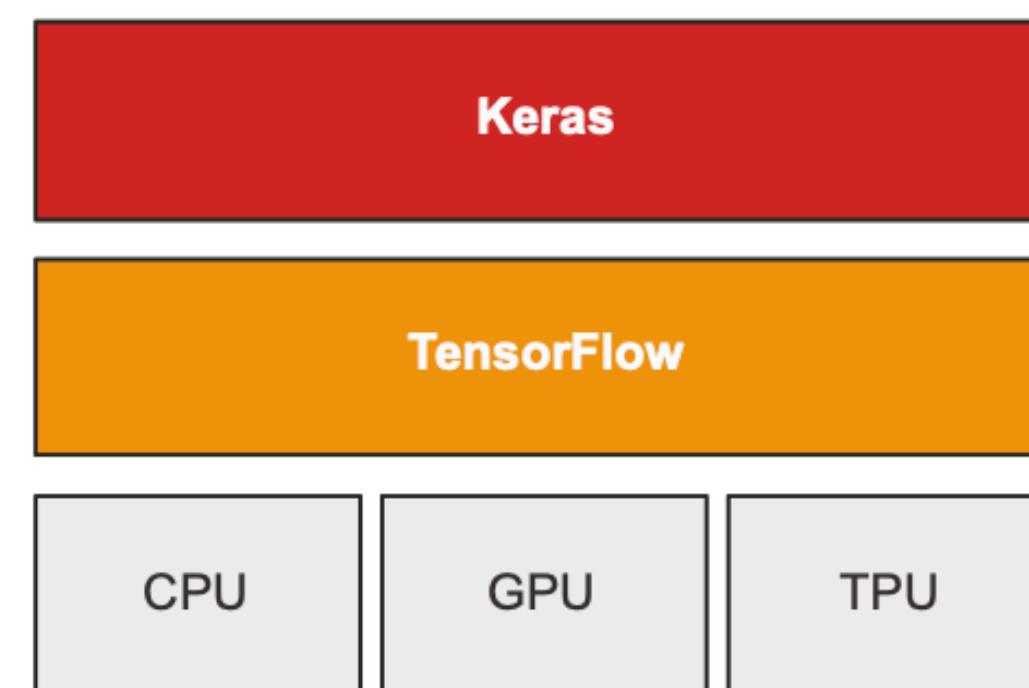
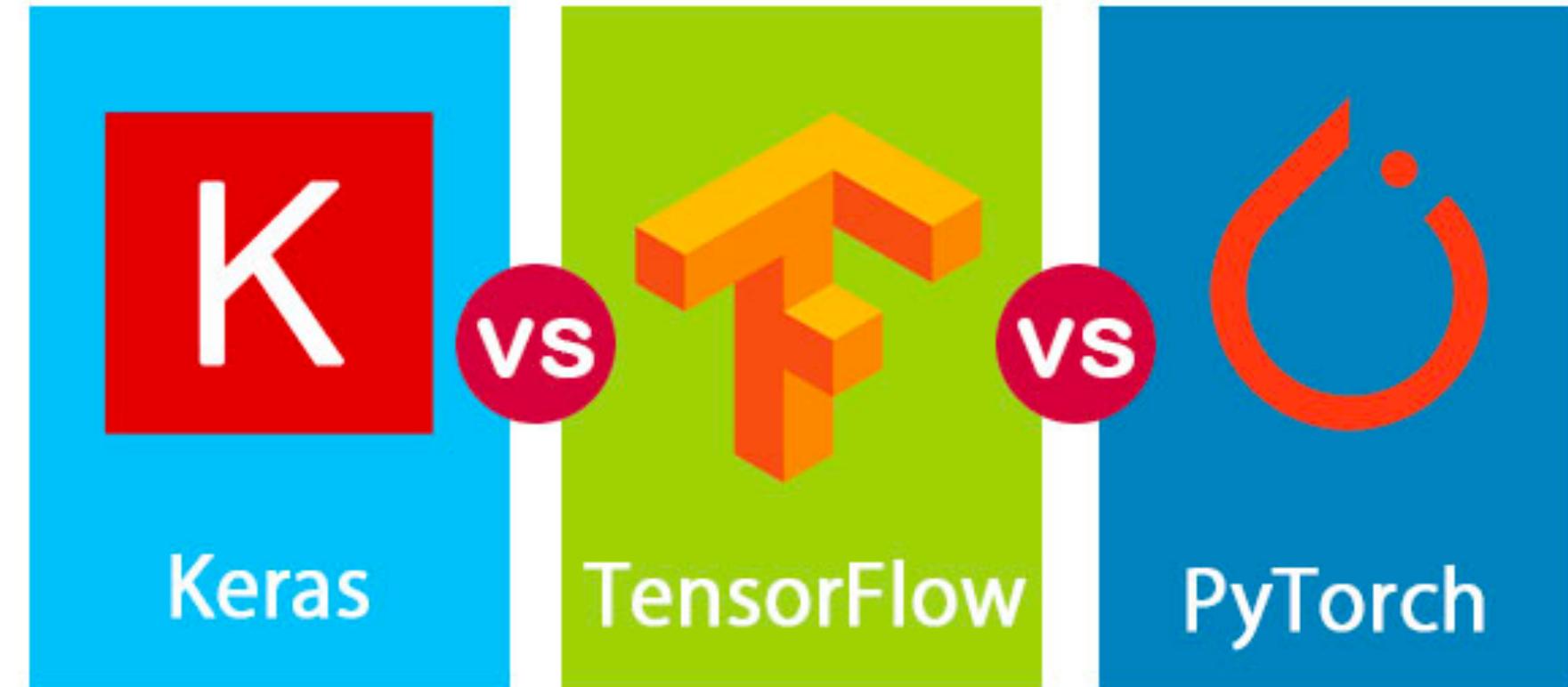


```
>>> from sklearn.datasets import make_regression
>>> from sklearn.ensemble import GradientBoostingRegressor
>>> from sklearn.model_selection import train_test_split
>>> X, y = make_regression(random_state=0)
>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, random_state=0)
>>> reg = GradientBoostingRegressor(random_state=0)
>>> reg.fit(X_train, y_train)
GradientBoostingRegressor(random_state=0)
>>> reg.predict(X_test[1:2])
array([-61...])
>>> reg.score(X_test, y_test)
0.4...
```

What is Keras?



```
>>> from sklearn.datasets import make_regression
>>> from sklearn.ensemble import GradientBoostingRegressor
>>> from sklearn.model_selection import train_test_split
>>> X, y = make_regression(random_state=0)
>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, random_state=0)
>>> reg = GradientBoostingRegressor(random_state=0)
>>> reg.fit(X_train, y_train)
GradientBoostingRegressor(random_state=0)
>>> reg.predict(X_test[1:2])
array([-61...])
>>> reg.score(X_test, y_test)
0.4...
```

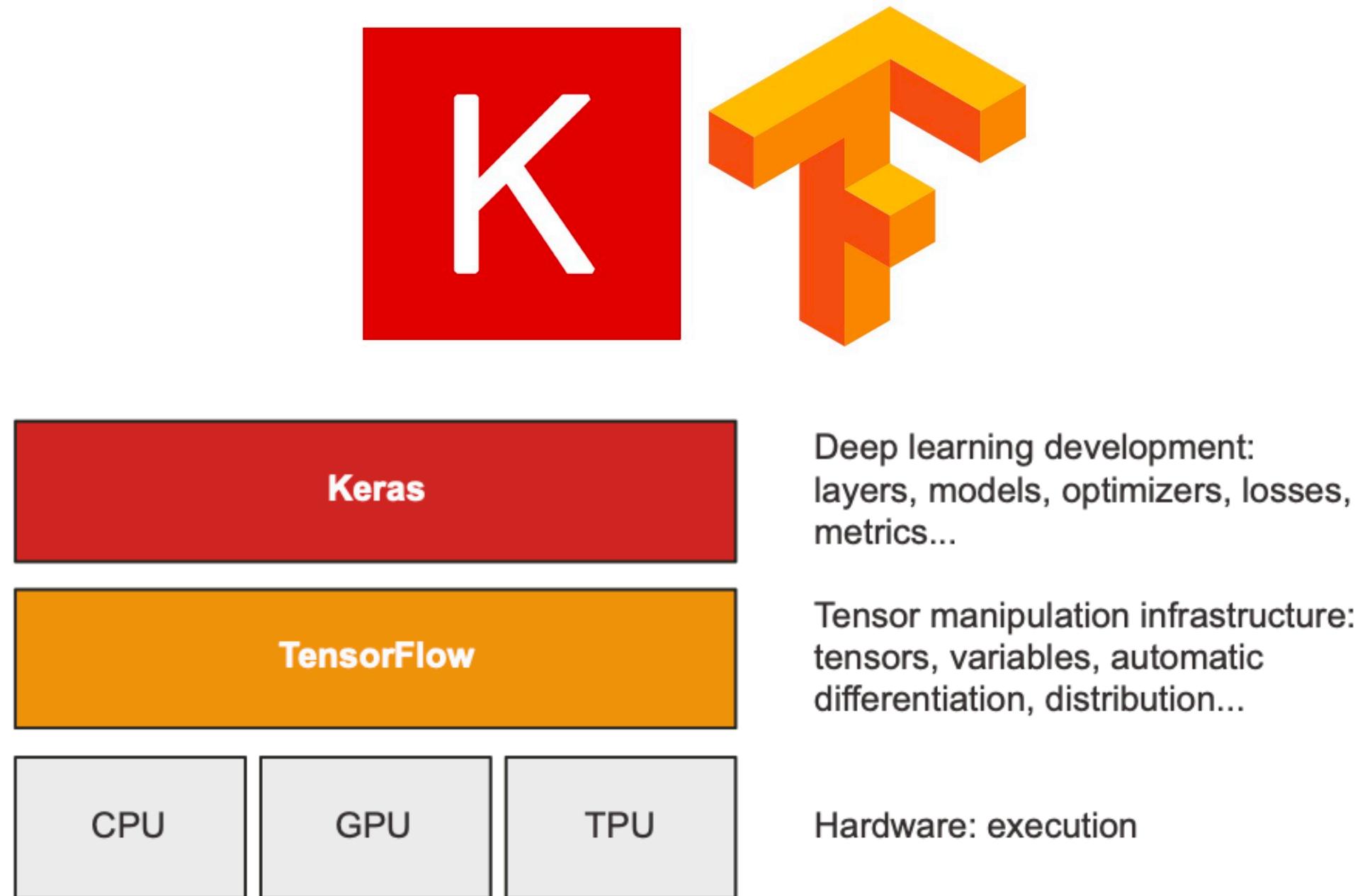


Deep learning development:
layers, models, optimizers, losses,
metrics...

Tensor manipulation infrastructure:
tensors, variables, automatic
differentiation, distribution...

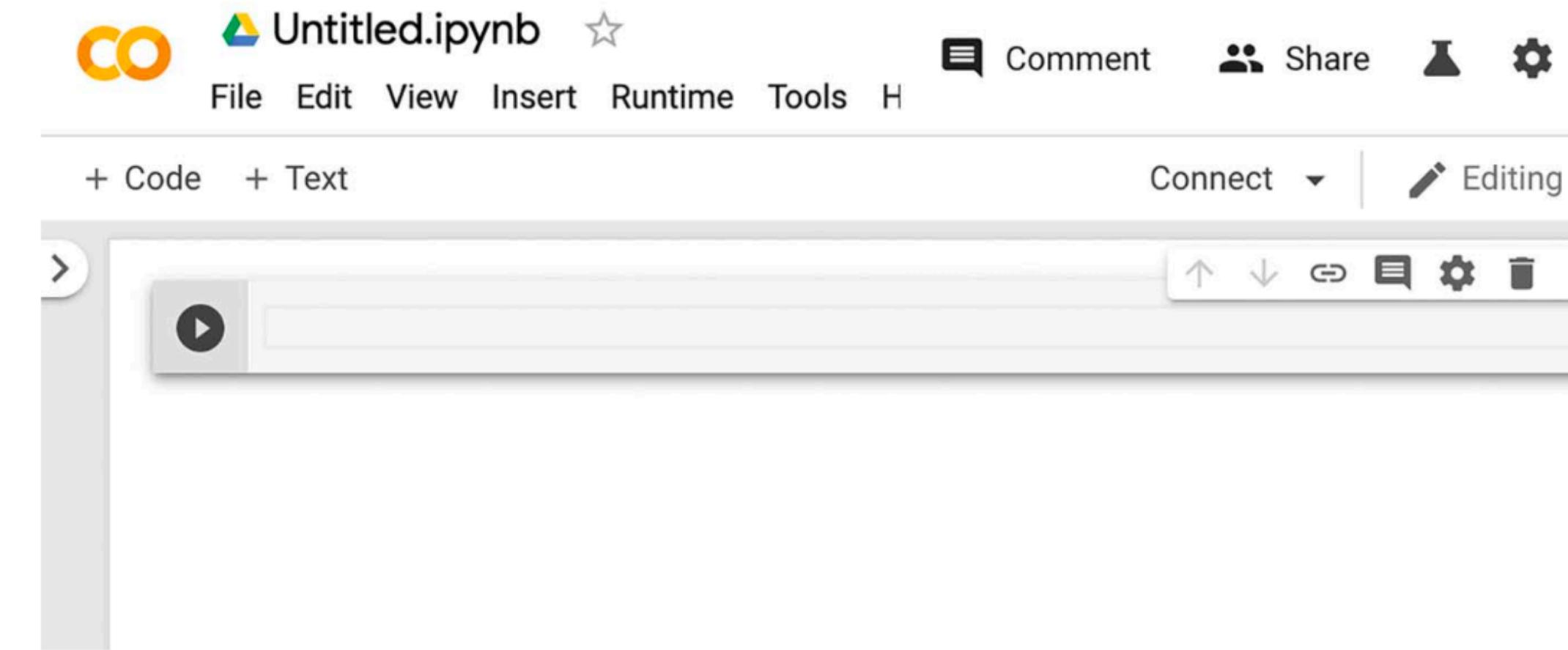
Hardware: execution

Why Keras & Tensorflow? The best way to start

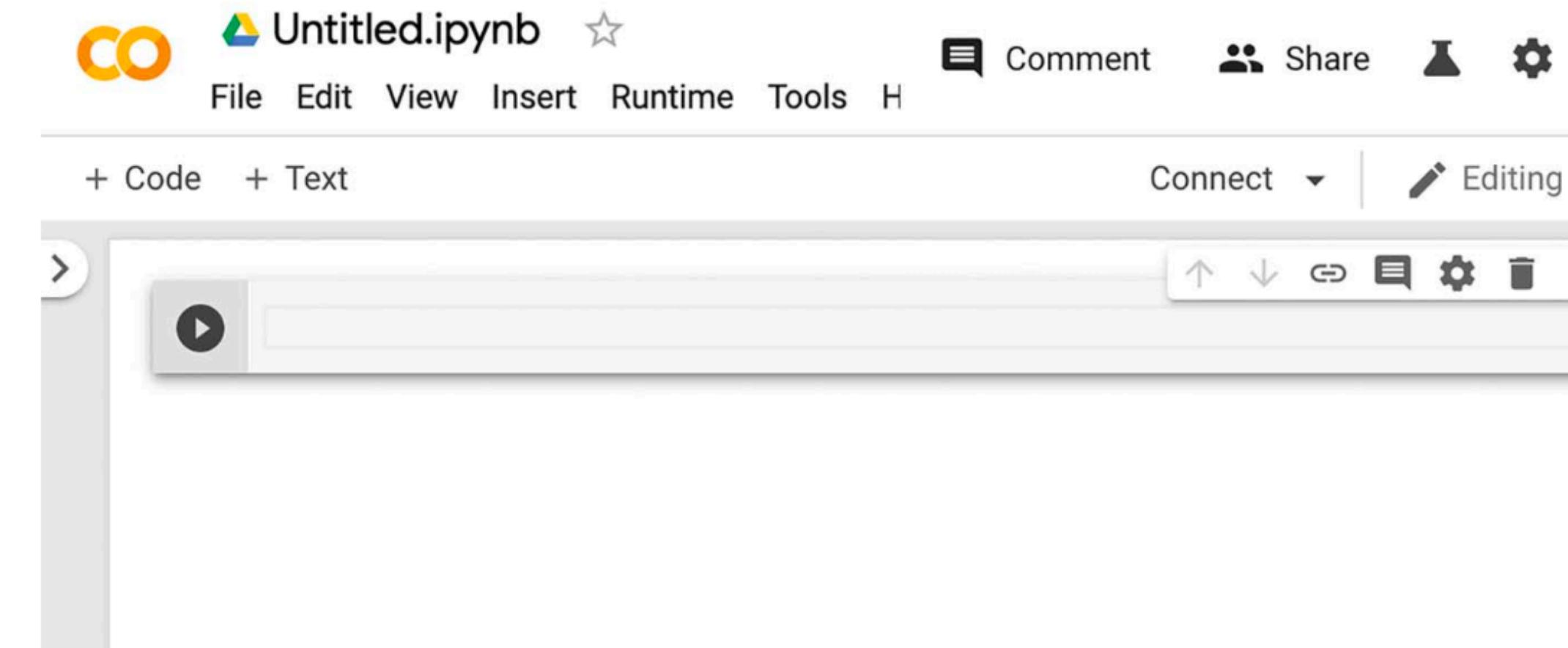


- Tensorflow:
 - Computes gradients Automatically
 - Runs on GPUs
 - Easy to parallelize
- Keras is easy to use and allows you to build NN in a straightforward manner

The best way to start coding Neural networks

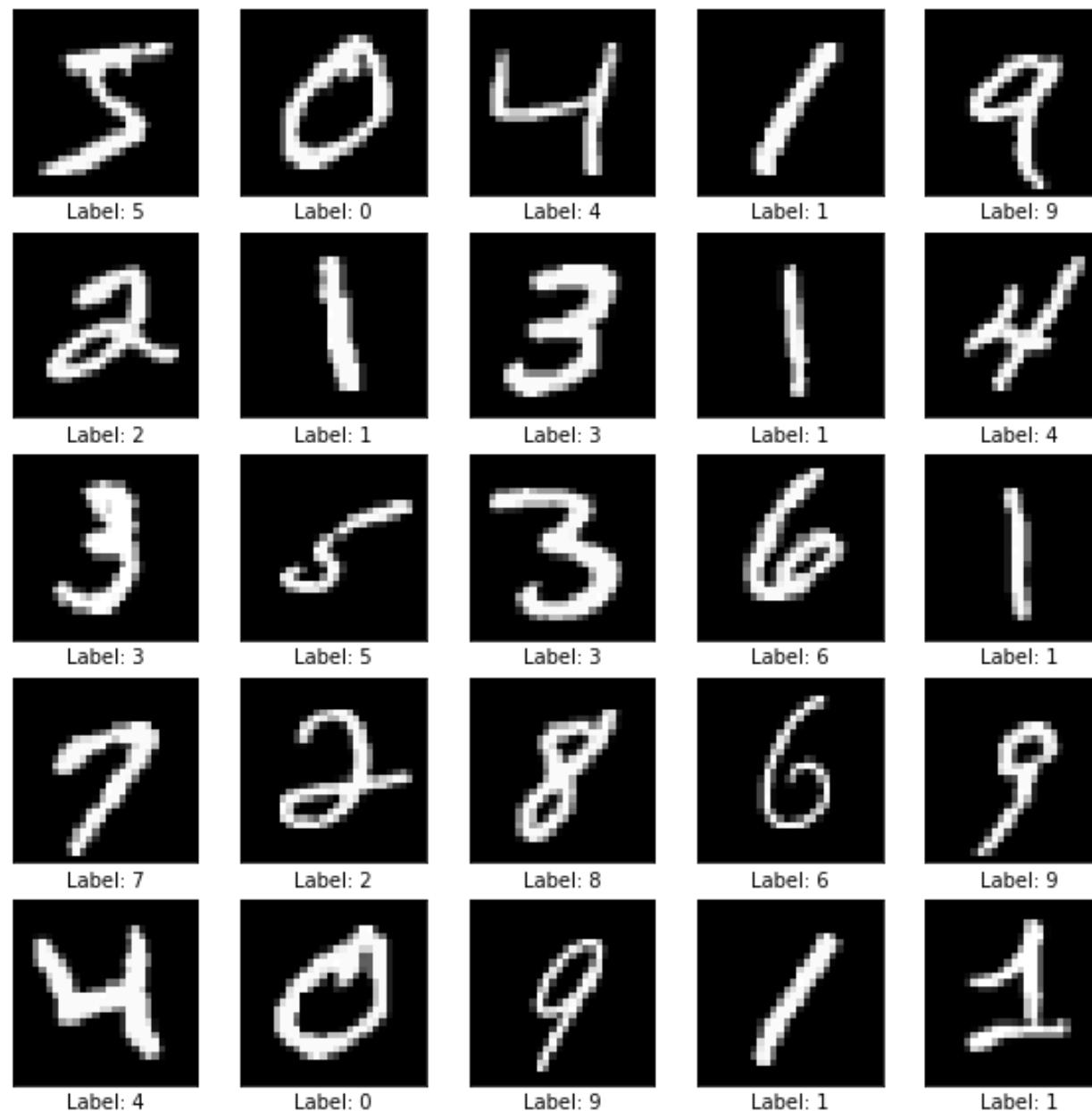


The best way to start coding Neural networks



Let's build our first Neural Network!

Open Colab, Load Keras, and load the MNIST dataset



Load Keras

```
from tensorflow.keras.datasets import mnist  
from tensorflow import keras  
from tensorflow.keras import layers
```

Load Dataset

```
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

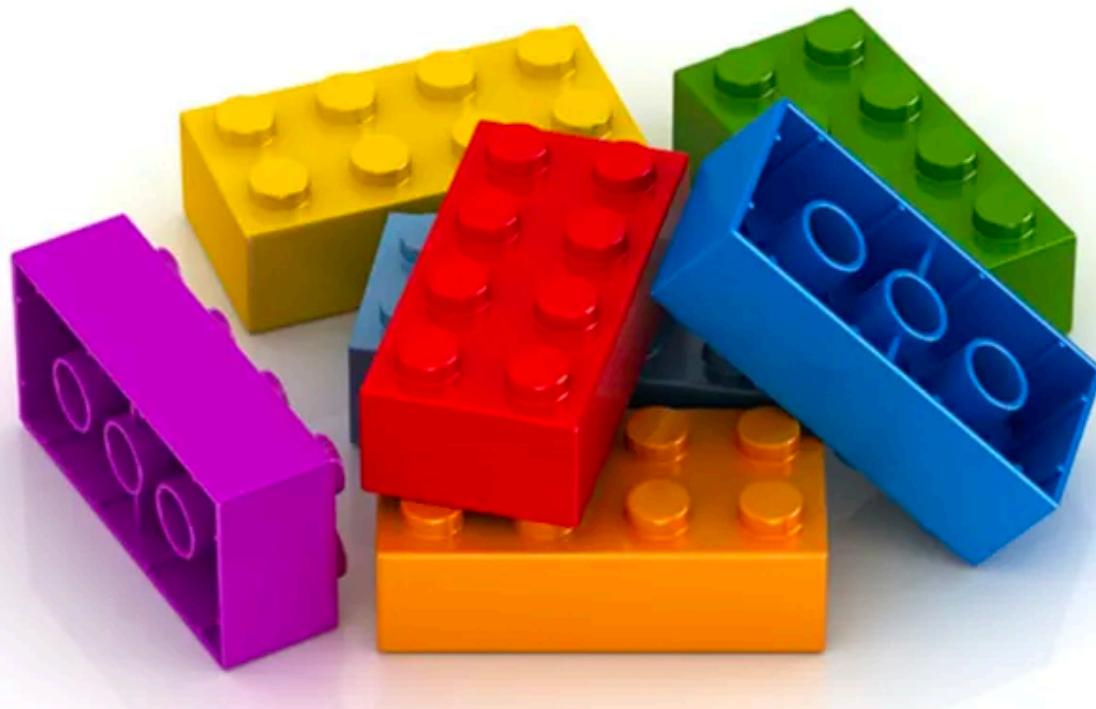
Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>
11490434/11490434 [=====] - 0s 0us/step

```
train_images.shape, test_images.shape
```

```
((60000, 28, 28), (10000, 28, 28))
```

Keras building blocks: Layers (Build the structure)

- Layers are the building blocks of deep learning
- You can think of them as the LEGO bricks of deep learning (Keras uses this analogy to stack layers)



Core layers

- Input object
- Dense layer
- Activation layer
- Embedding layer
- Masking layer
- Lambda layer

Convolution layers

- Conv1D layer
- Conv2D layer

Layer activations

- relu function
- sigmoid function
- softmax function
- softplus function
- softsign function
- tanh function
- selu function
- elu function
- exponential function

```
model = keras.Sequential([
    layers.Dense(784, activation='sigmoid'),
    layers.Dense(16, activation='sigmoid'),
    layers.Dense(16, activation='sigmoid'),
    layers.Dense(10, activation='softmax'),
])
```

Keras building blocks: Compile (Tell it how to learn)

- **Loss Function** (Tell the computer how good/bad is doing it): Objective function to be minimized during training.
- **Optimizer**: Updates to the network. A version of SGD (Stochastic Gradient Descent)
- **Metrics**: Measure of success to monitor during training, i.e. Accuracy.

```
model.compile(  
    optimizer="rmsprop",  
    loss="sparse_categorical_crossentropy",  
    metrics=[ 'accuracy' ]  
)
```

Keras building blocks: Compile (Tell it how to learn)

- **Loss Function** (Tell the computer how good/bad is doing it): Objective function to be minimized during training.
- **Optimizer**: Updates to the network. A version of SGD (Stochastic Gradient Descent)
- **Metrics**: Measure of success to monitor during training, i.e. Accuracy.

```
model.compile(  
    optimizer="rmsprop",  
    loss="sparse_categorical_crossentropy",  
    metrics=[ 'accuracy' ]  
)
```

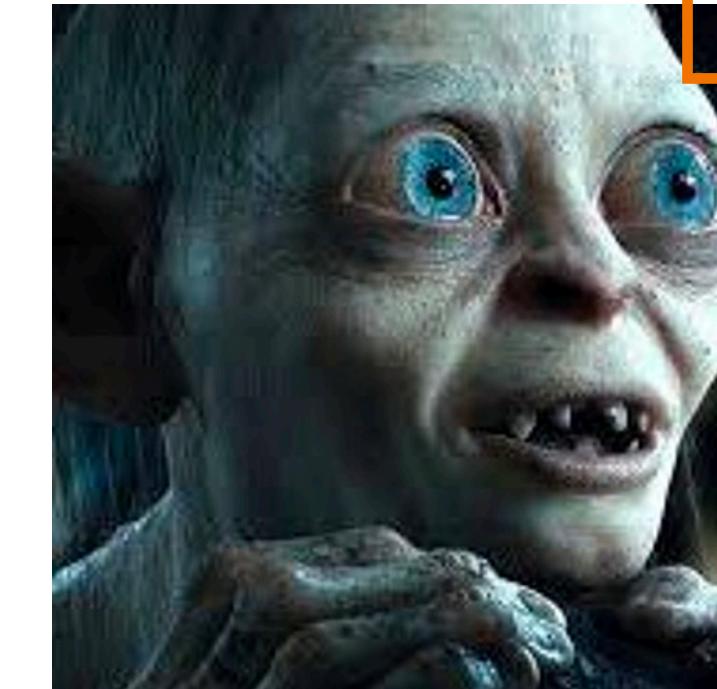
$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Keras building blocks: Compile (Tell it how to learn)

- **Loss Function** (Tell the computer how good/bad is doing it): Objective function to be minimized during training.
- **Optimizer**: Updates to the network. A version of SGD (Stochastic Gradient Descent)
- **Metrics**: Measure of success to monitor during training, i.e. Accuracy.

```
model.compile(  
    optimizer="rmsprop",  
    loss="sparse_categorical_crossentropy",  
    metrics=[ 'accuracy' ]  
)
```

$$\text{Loss} = - \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$



Loss Function

The network will do anything to lower the loss (Example of well-being and fake news). Choose wisely

Keras building blocks: Fit (Make it learn)

- **The Data** (Inputs & Targets) to train on. Numpy arrays or a TensorFlow Dataset object
- **Number of Epochs**: Number of times the training loop should iterate over the data.
- **Batch Size**: Number of training examples considered to compute the gradients for one weight update.

```
model.fit(  
    train_images,  
    train_labels,  
    epochs=10,  
    batch_size=128)
```

Keras building blocks: Fit (Make it learn)

- **The Data** (Inputs & Targets) to train on. Numpy arrays or a TensorFlow Dataset object
- **Number of Epochs**: Number of times the training loop should iterate over the data.
- **Batch Size**: Number of training examples considered to compute the gradients for one weight update.

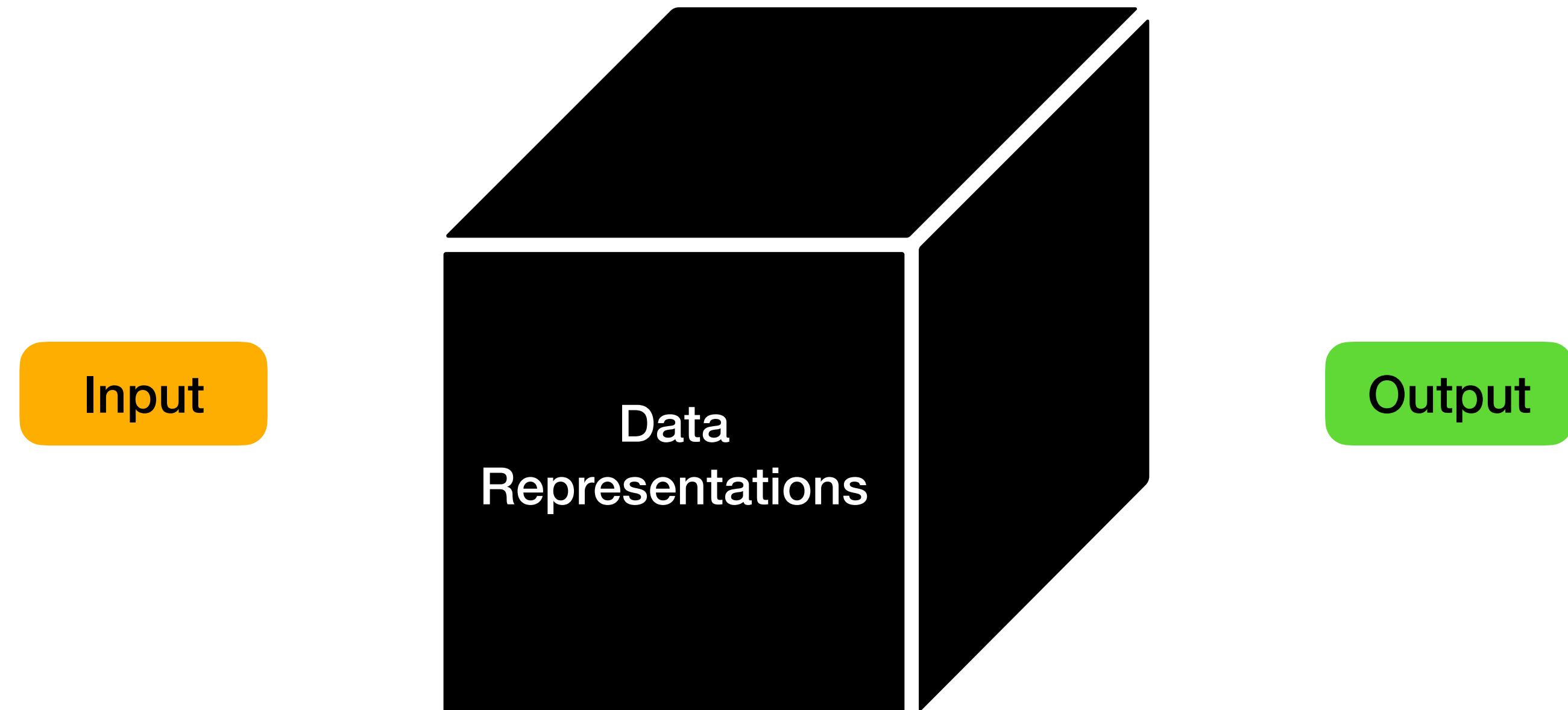
```
model.fit(  
    train_images,  
    train_labels,  
    epochs=10,  
    batch_size=128)
```

We are able to recognize hand-written digits with
~96% Accuracy!

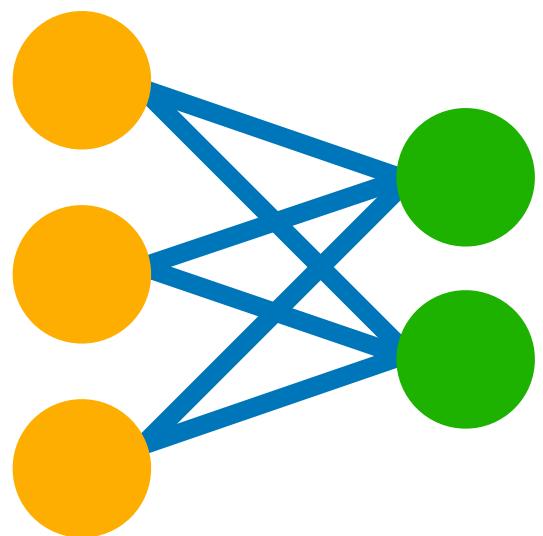
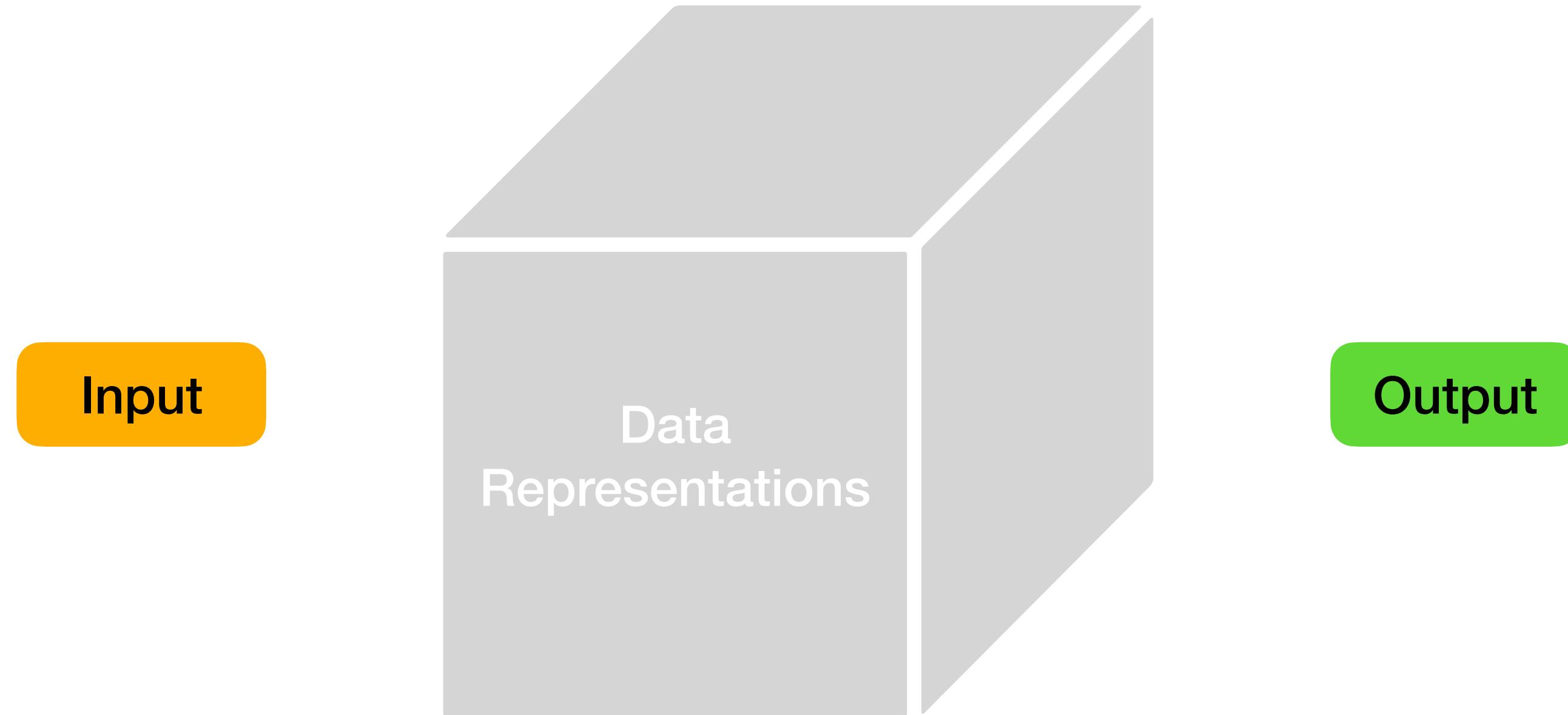


Types of Neural Networks and their applications

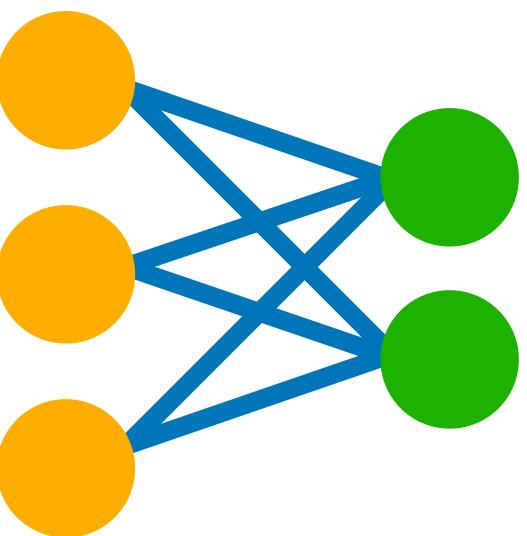
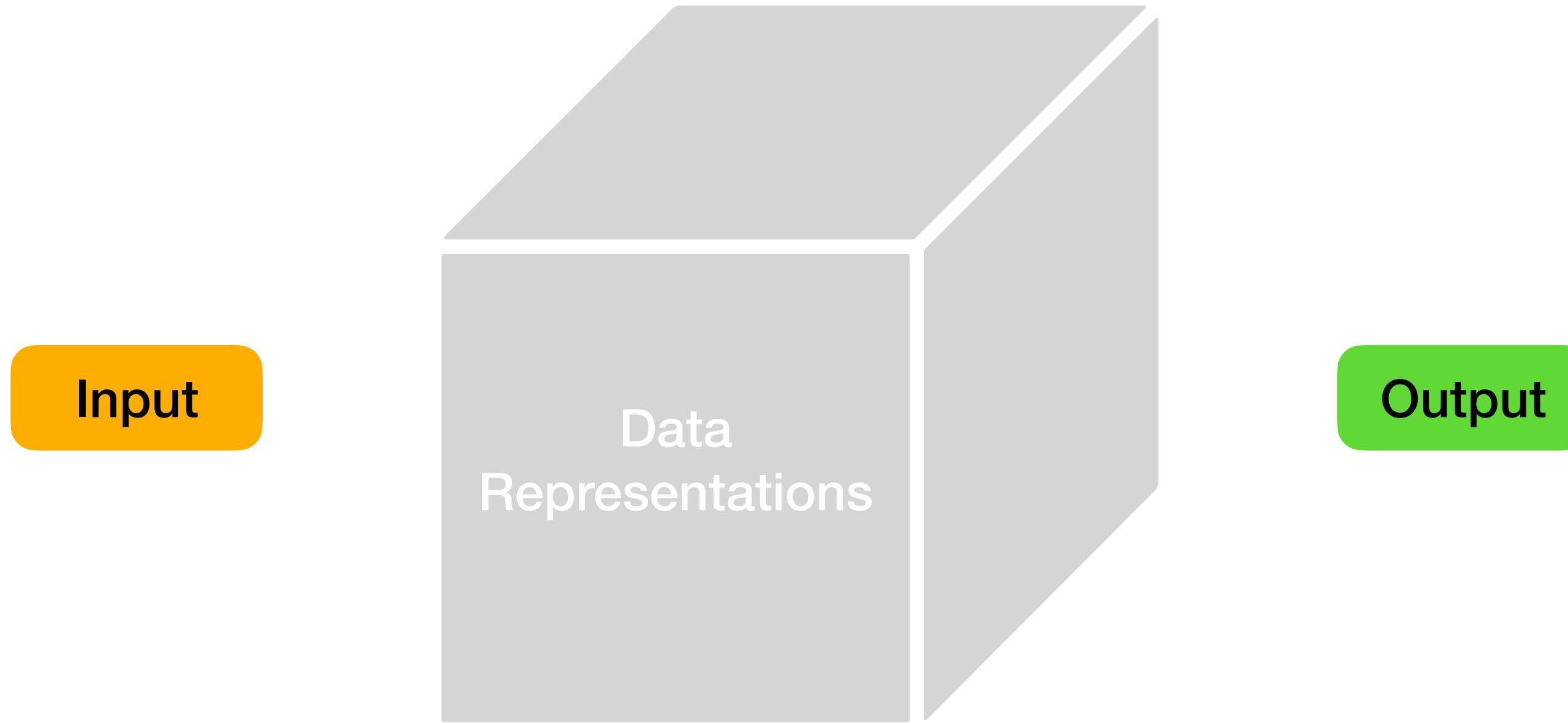
Building upon the same idea



Building upon the same idea

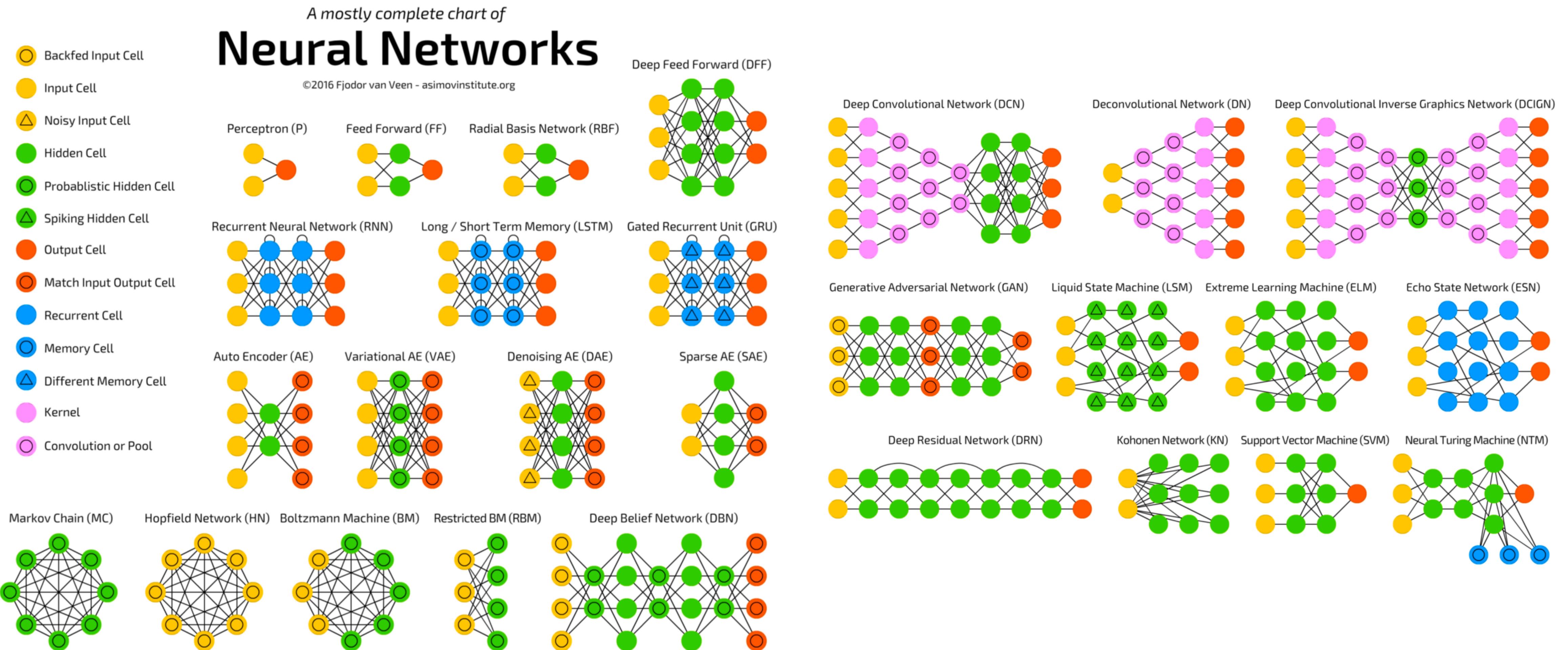


Building upon the same idea



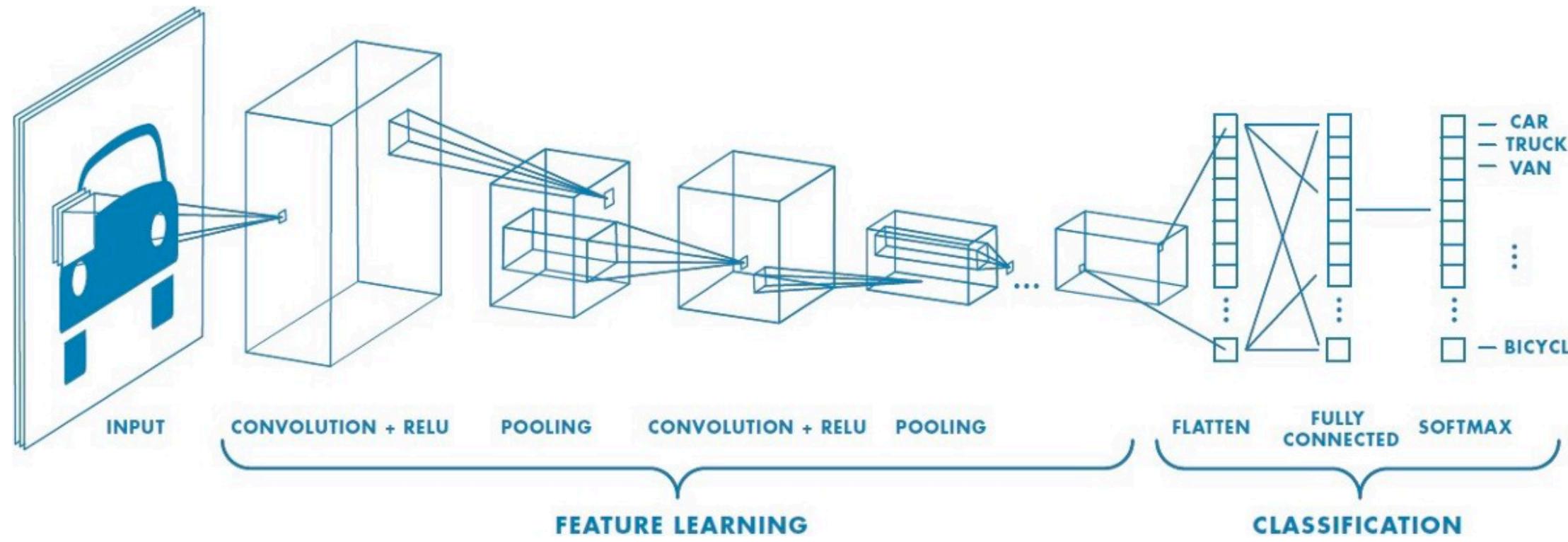
However, what we saw was
only one type of Neural Network

It's a huge field

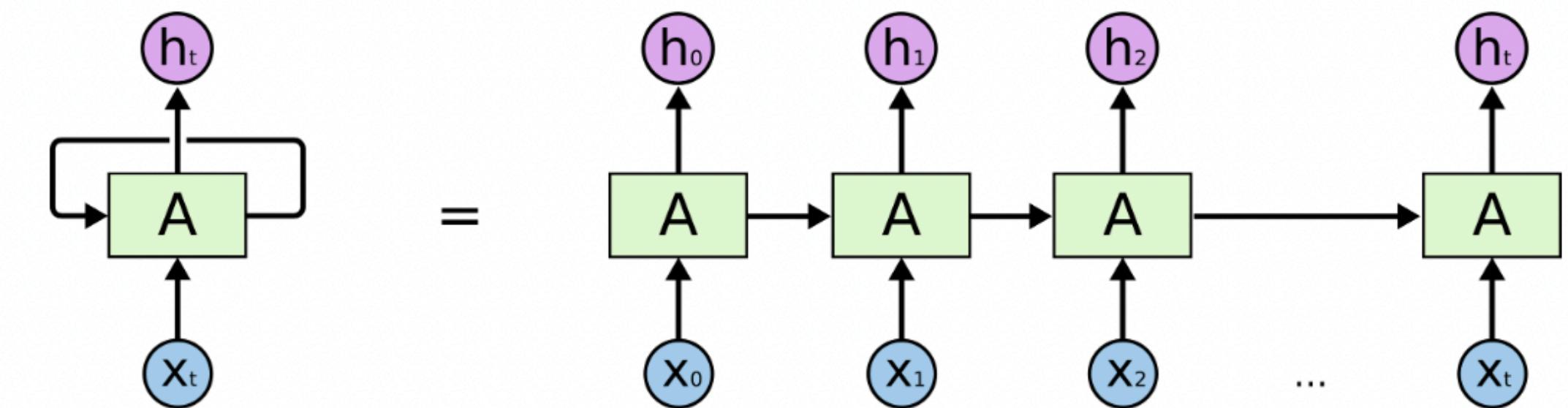


Let's talk and try to code two of the most important ones

Convolutional Neural Networks (CNNs)



Recurrent Neural Networks (RNNs)



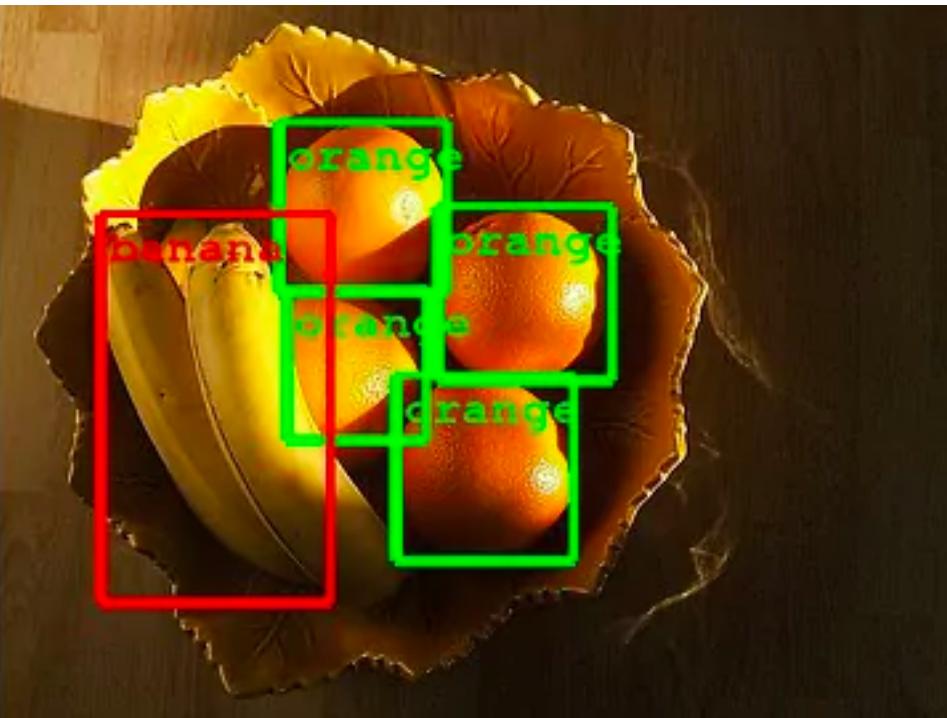
An unrolled recurrent neural network.

Computer Vision

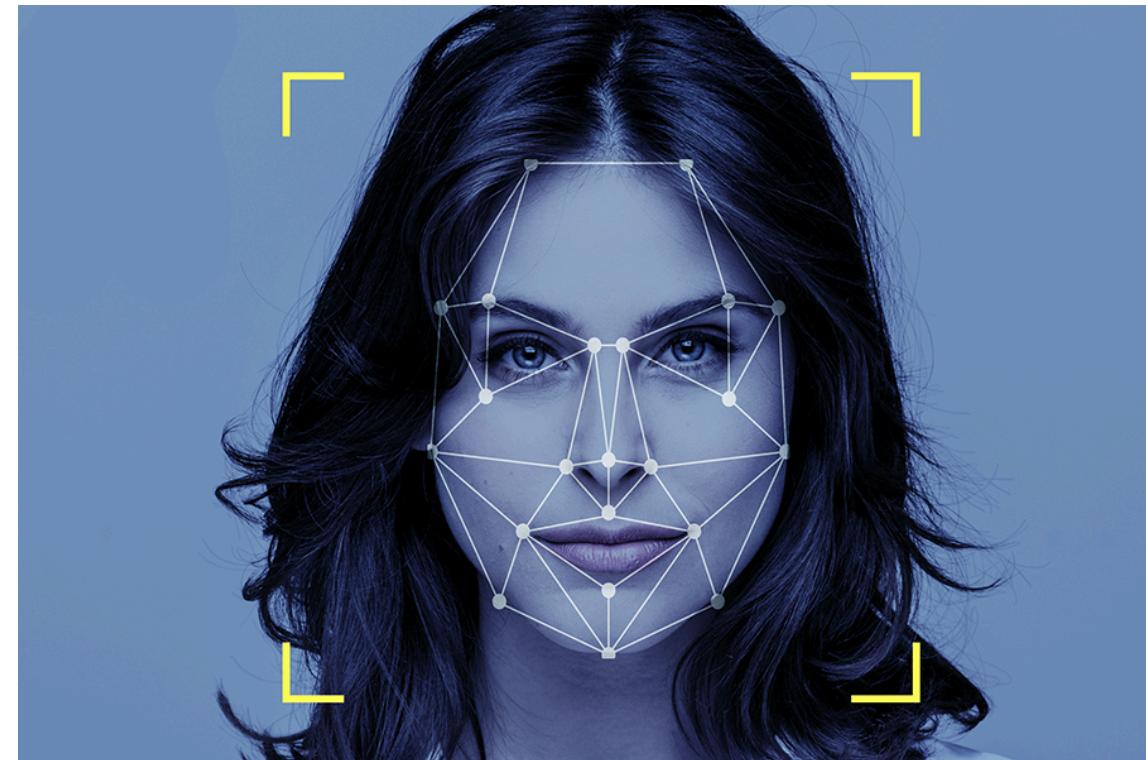
Natural Language Process (NLP)

Applications of Convolutional Neural Networks (CNNs)

Image Recognition & Optical Character Recognition



Face Recognition on social media



Object detection for self-driving cars

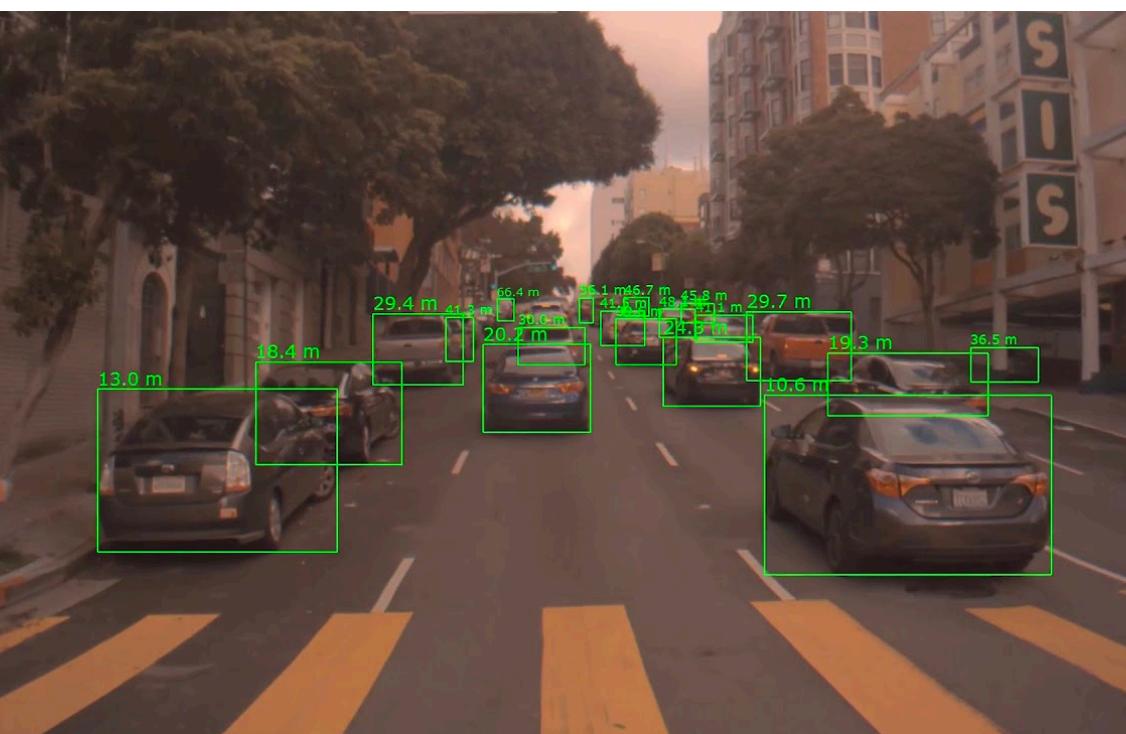
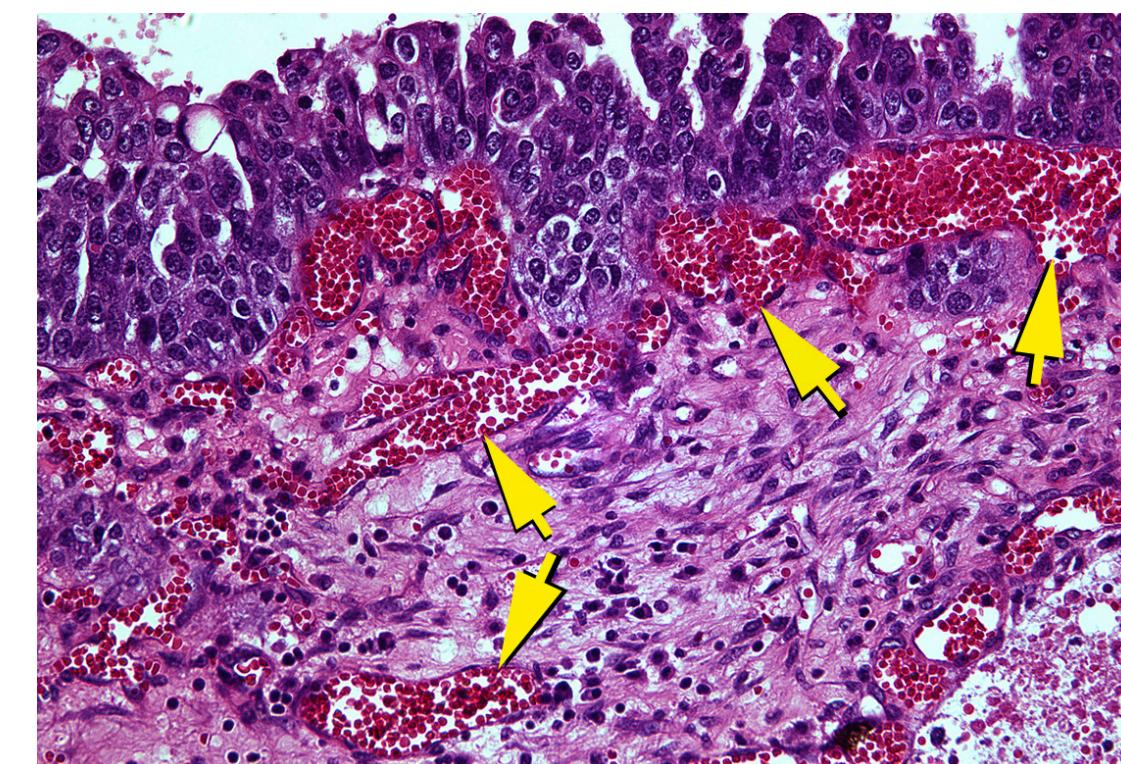
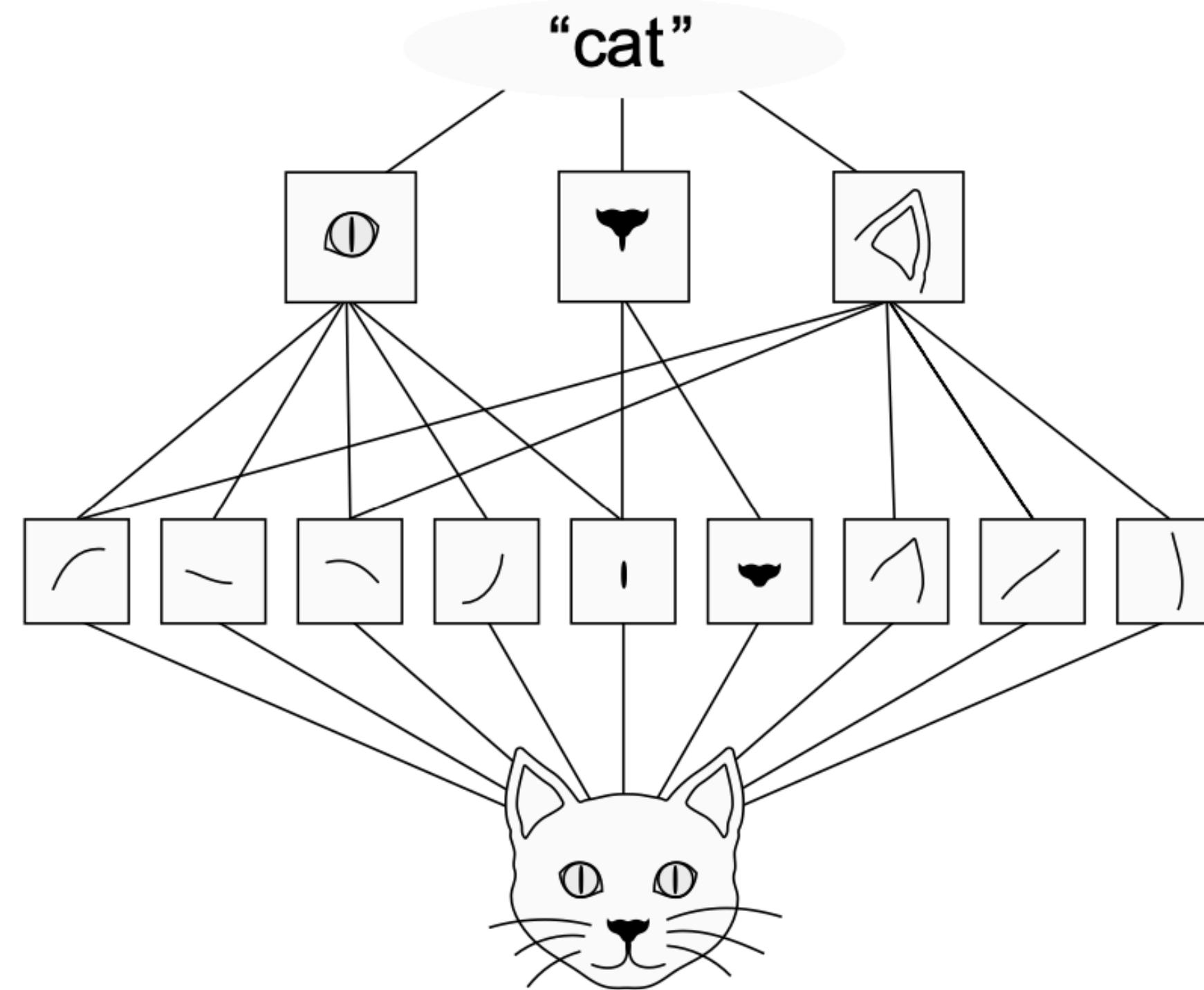
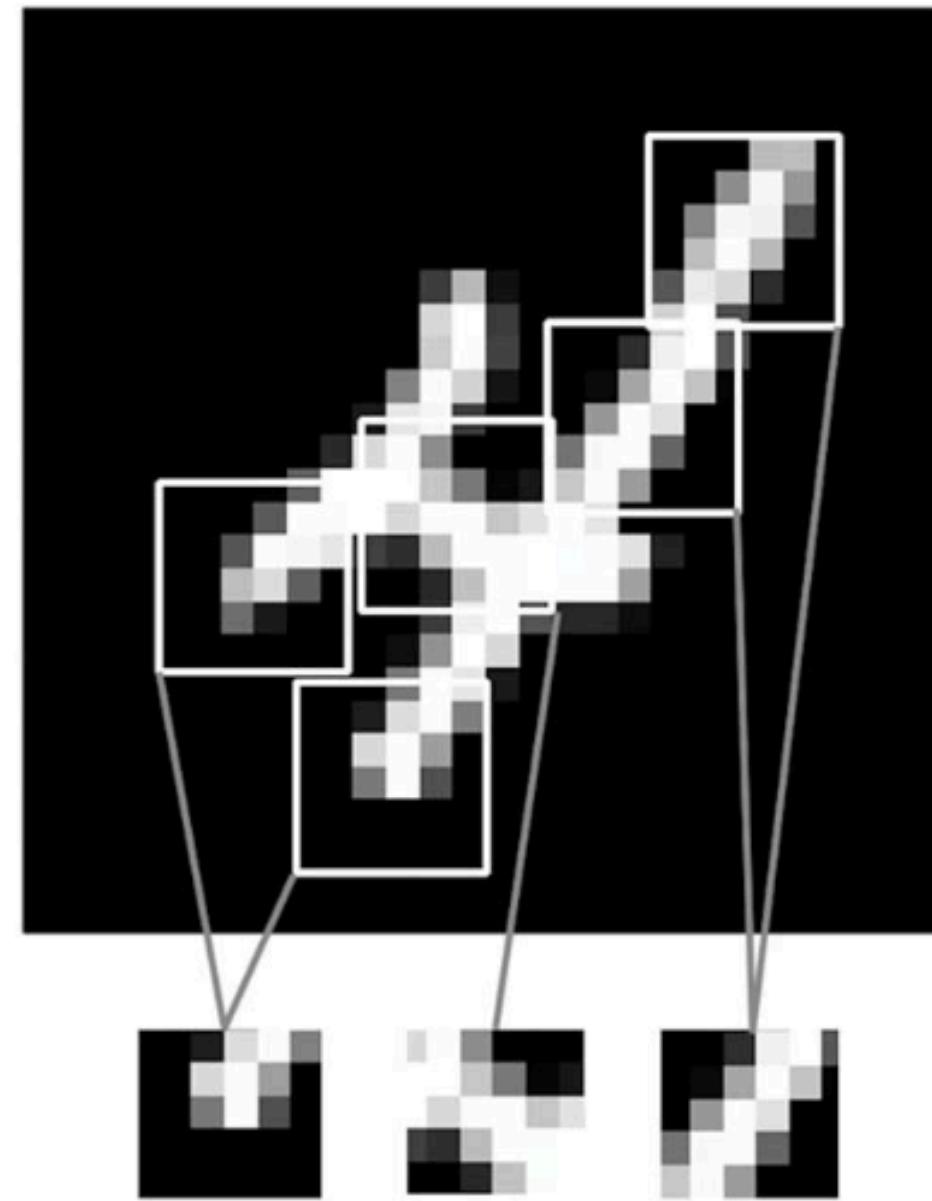


Image analysis in healthcare



A bit of understanding to the Network

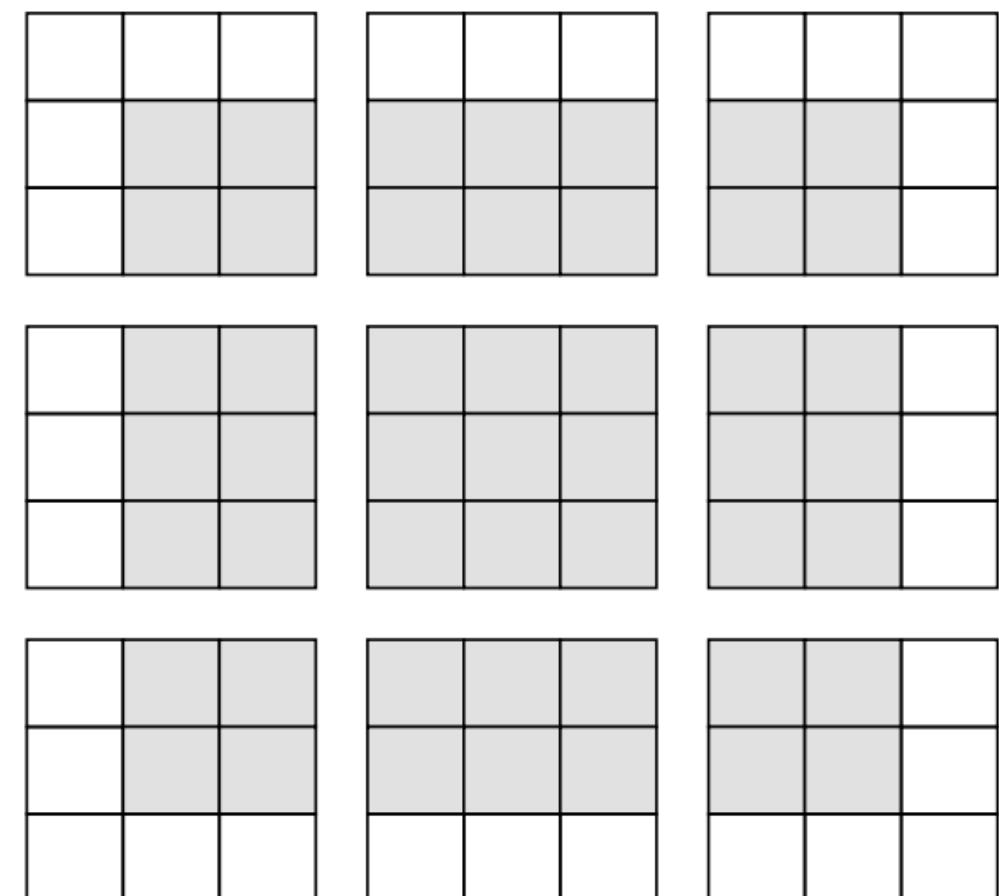
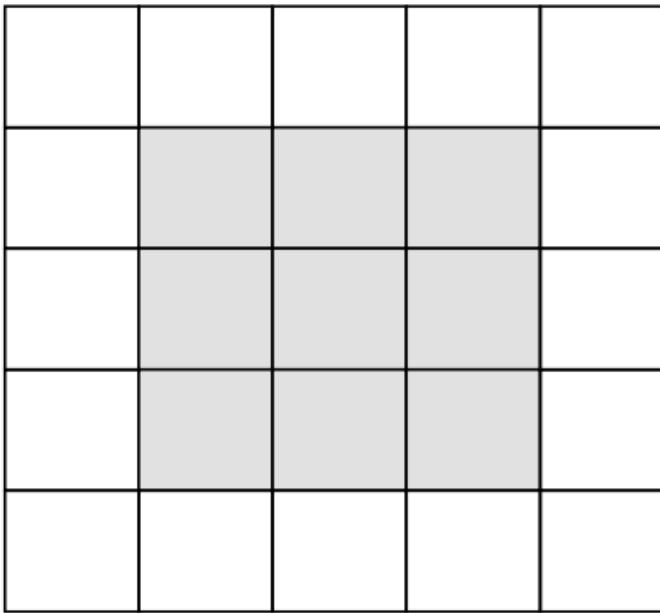
Images can be broken into local patterns such as edges, textures, and so on



A bit of understanding of the Network

In this case, the parameters (weights) of the network are filters

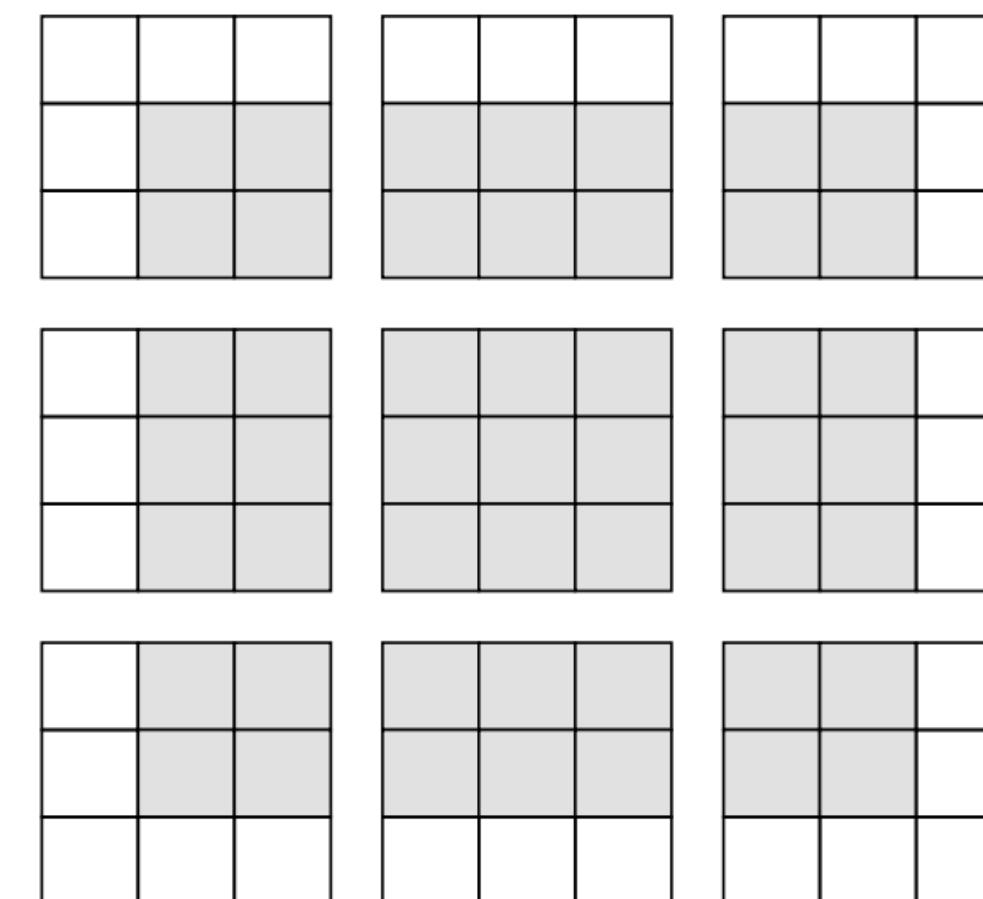
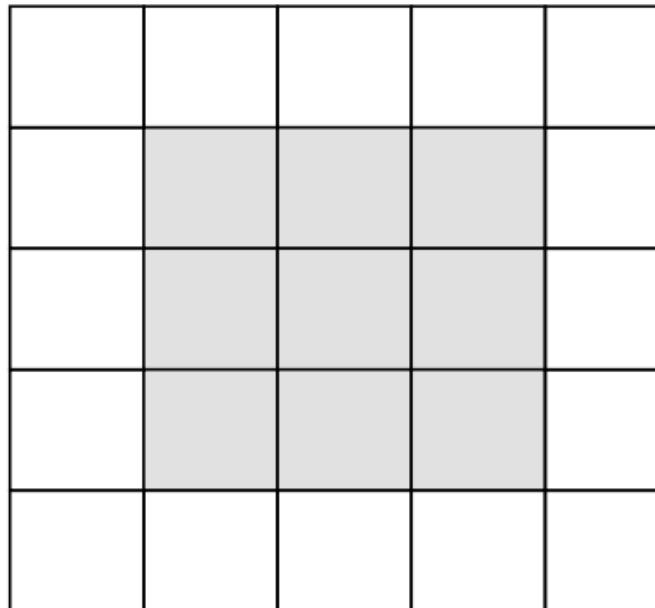
Locations of 3x3 patches in a 5x5 input feature image



A bit of understanding of the Network

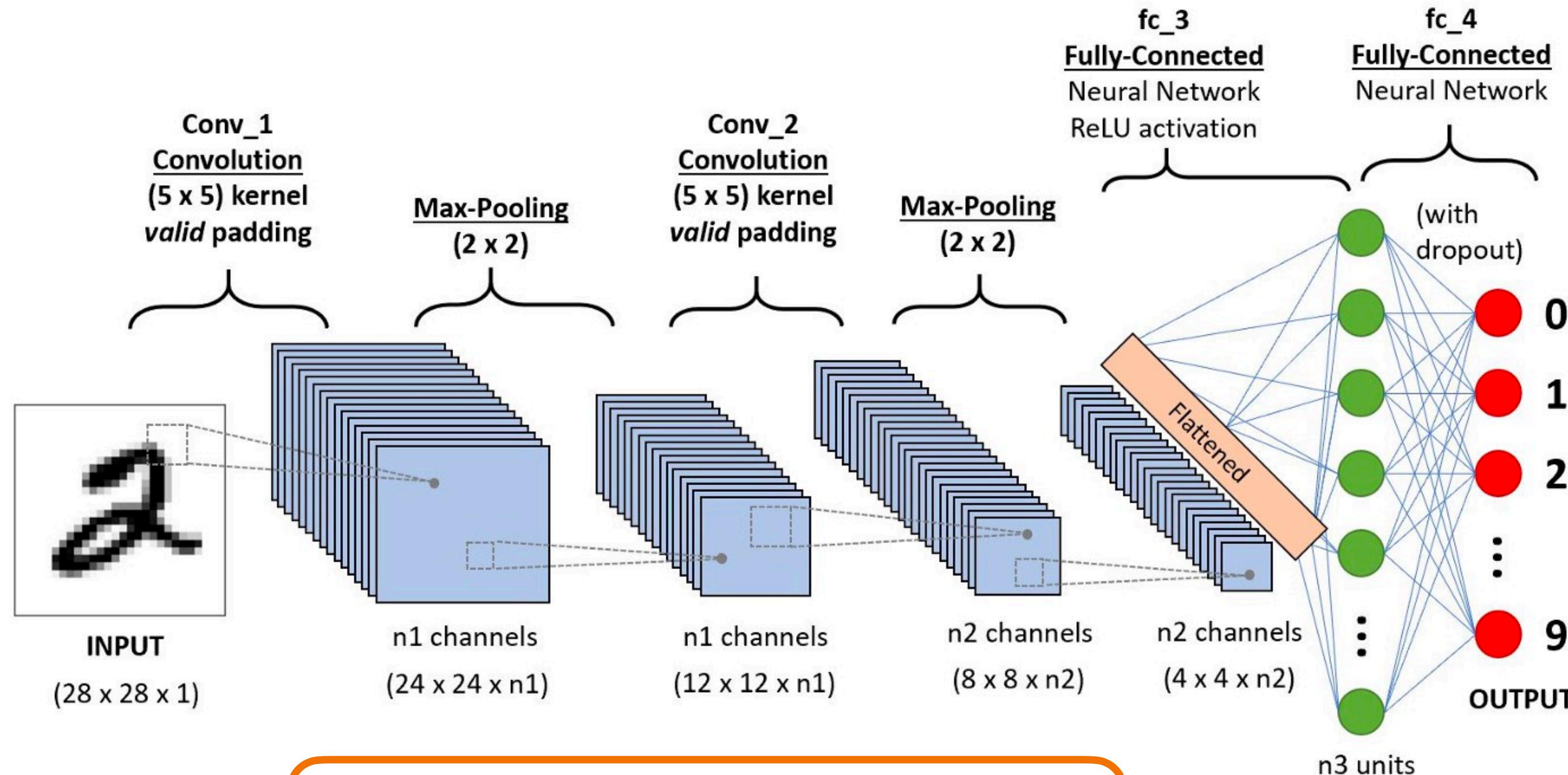
In this case, the parameters (weights) of the network are filters

Locations of 3x3 patches in a 5x5 input feature image

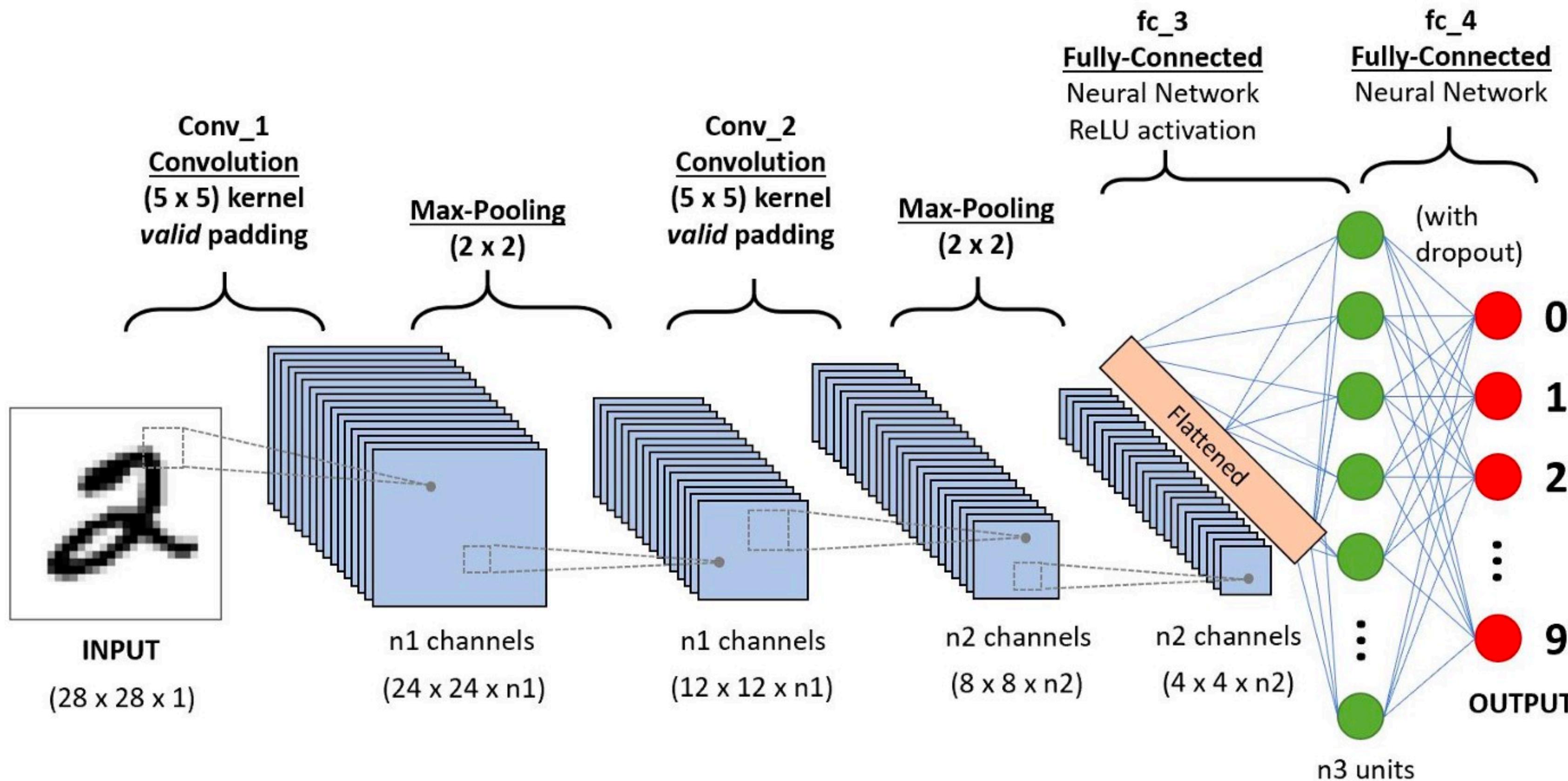


Input	Kernel	Output
$\begin{matrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{matrix}$	$*\quad \begin{matrix} 0 & 1 \\ 2 & 3 \end{matrix}$	$= \quad \begin{matrix} 19 & 25 \\ 37 & 43 \end{matrix}$

A bit of understanding of the Network



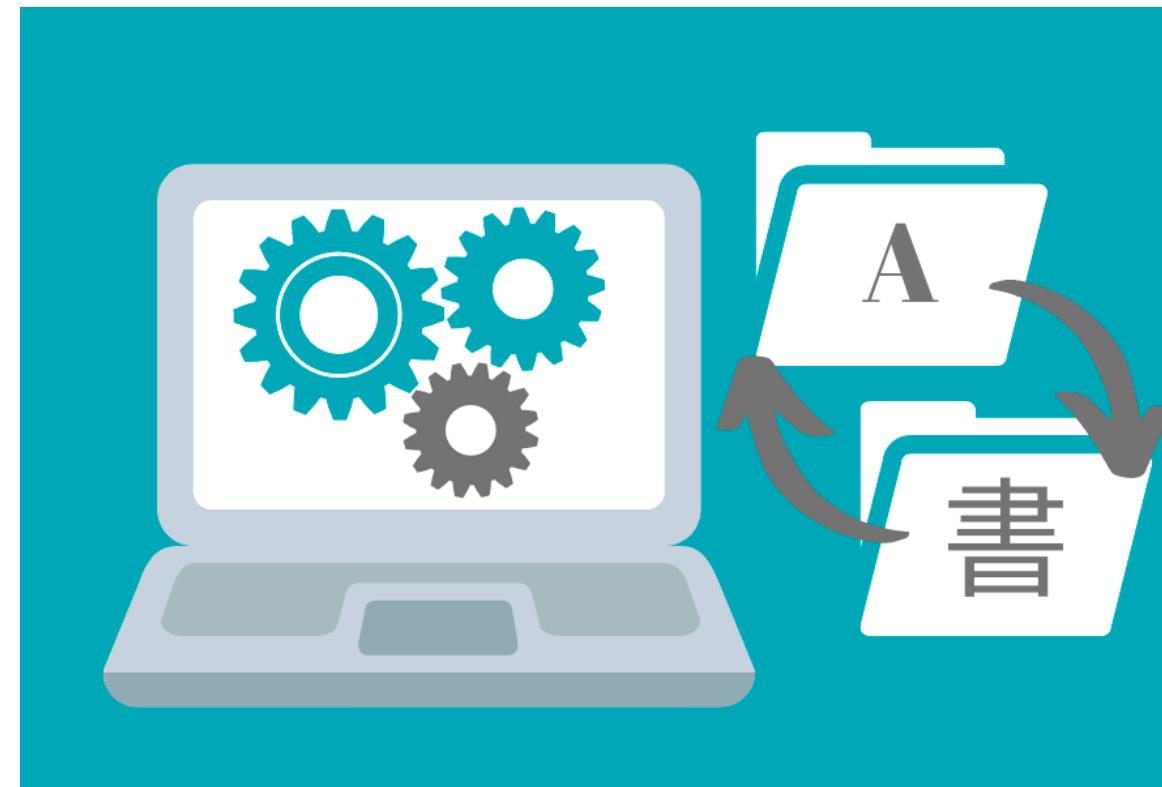
A bit of understanding of the Network



Let's build a CNN!

Applications of Recurrent Neural Networks (RNNs)

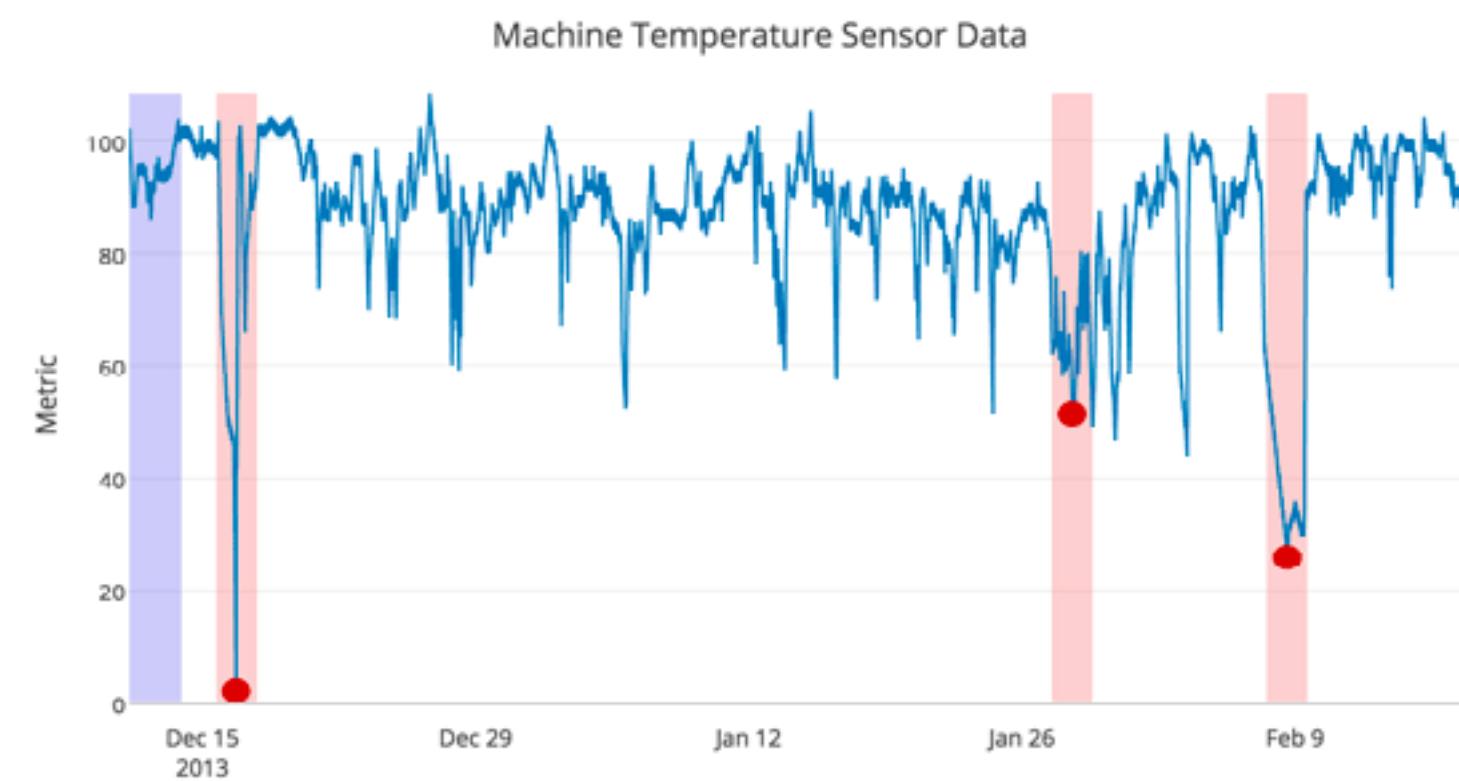
Machine Translation



Stock prediction



Time series anomaly detection



Speech Recognition

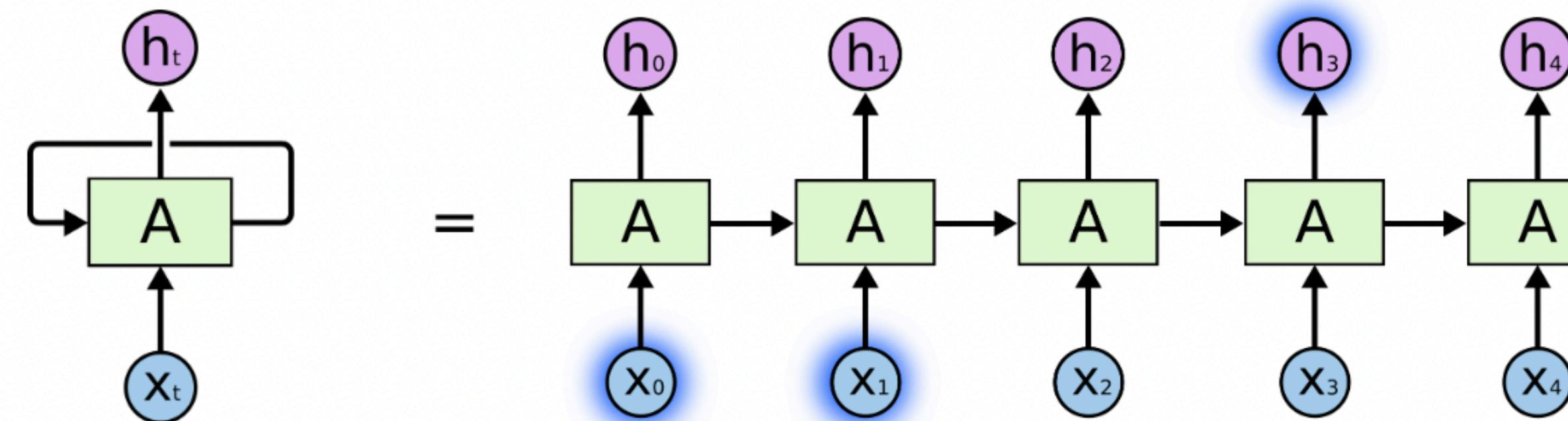


A glimpse of how the network works (RNNs)

Densely connected networks and CNNs have no memory. Inputs are processed independently. However, that is not the case of RNNs.

A glimpse of how the network works (RNNs)

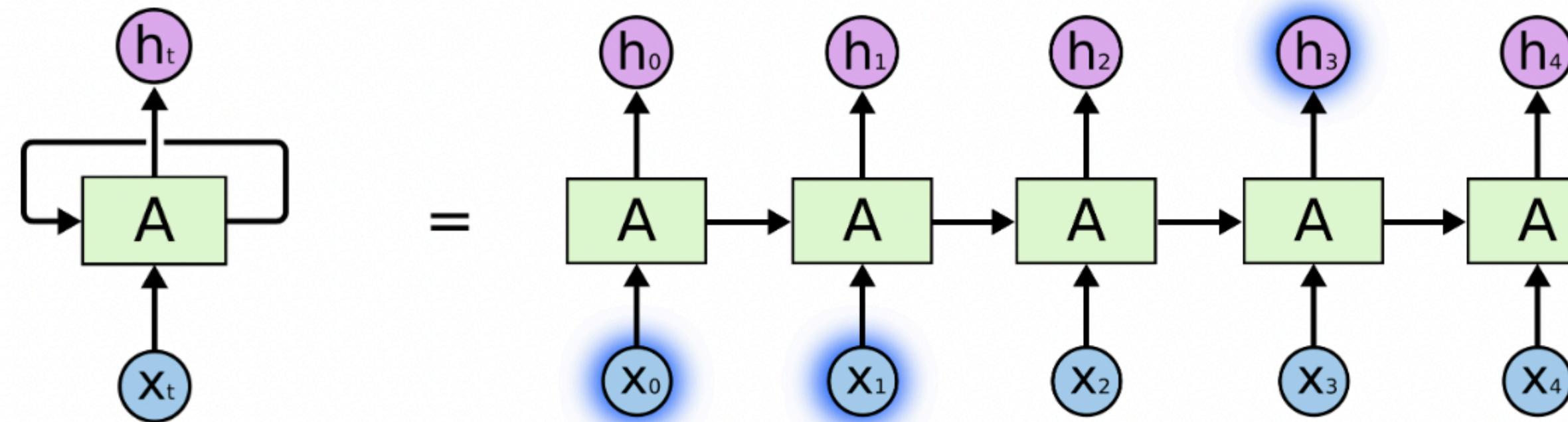
Densely connected networks and CNNs have no memory. Inputs are processed independently. However, that is not the case of RNNs.



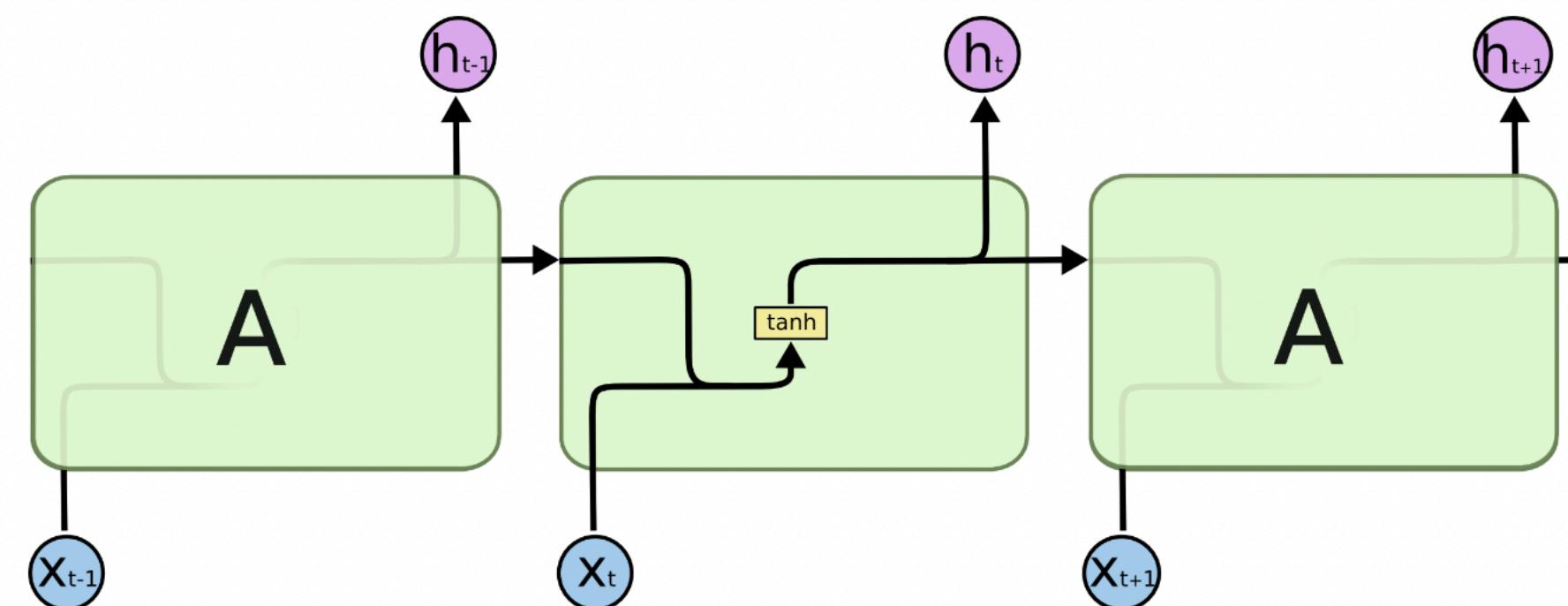
They retain information from previous neurons

A glimpse of how the network works (RNNs)

Densely connected networks and CNNs have no memory. Inputs are processed independently. However, that is not the case of RNNs.



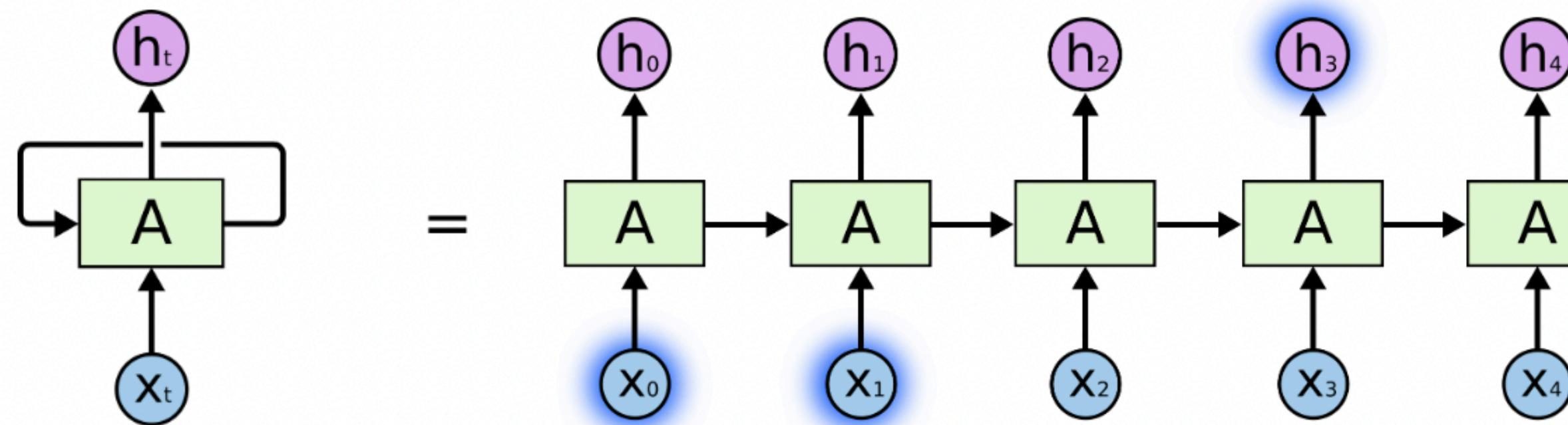
They retain information from previous neurons



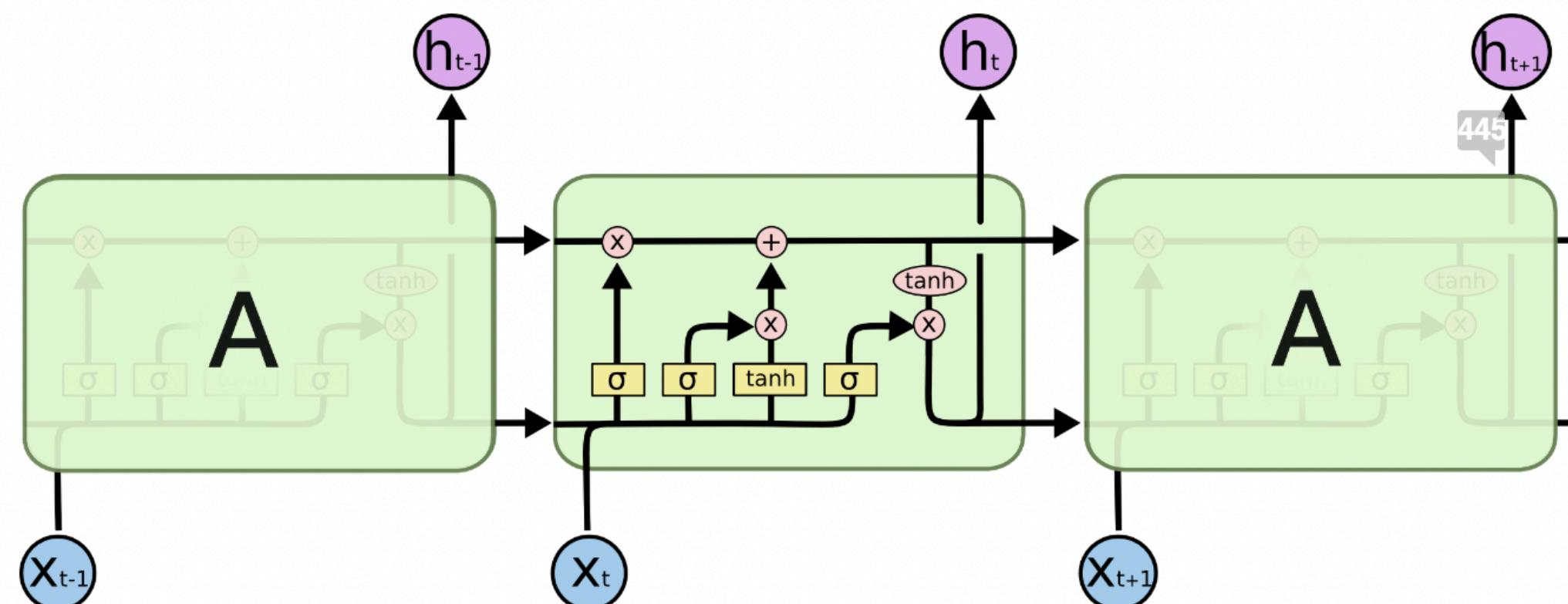
They keep the state of the previous neuron

A glimpse of how the network works (LSTMs)

Densely connected networks and CNNs have no memory. Inputs are processed independently. However, that is not the case of RNNs.



They retain information from previous neurons

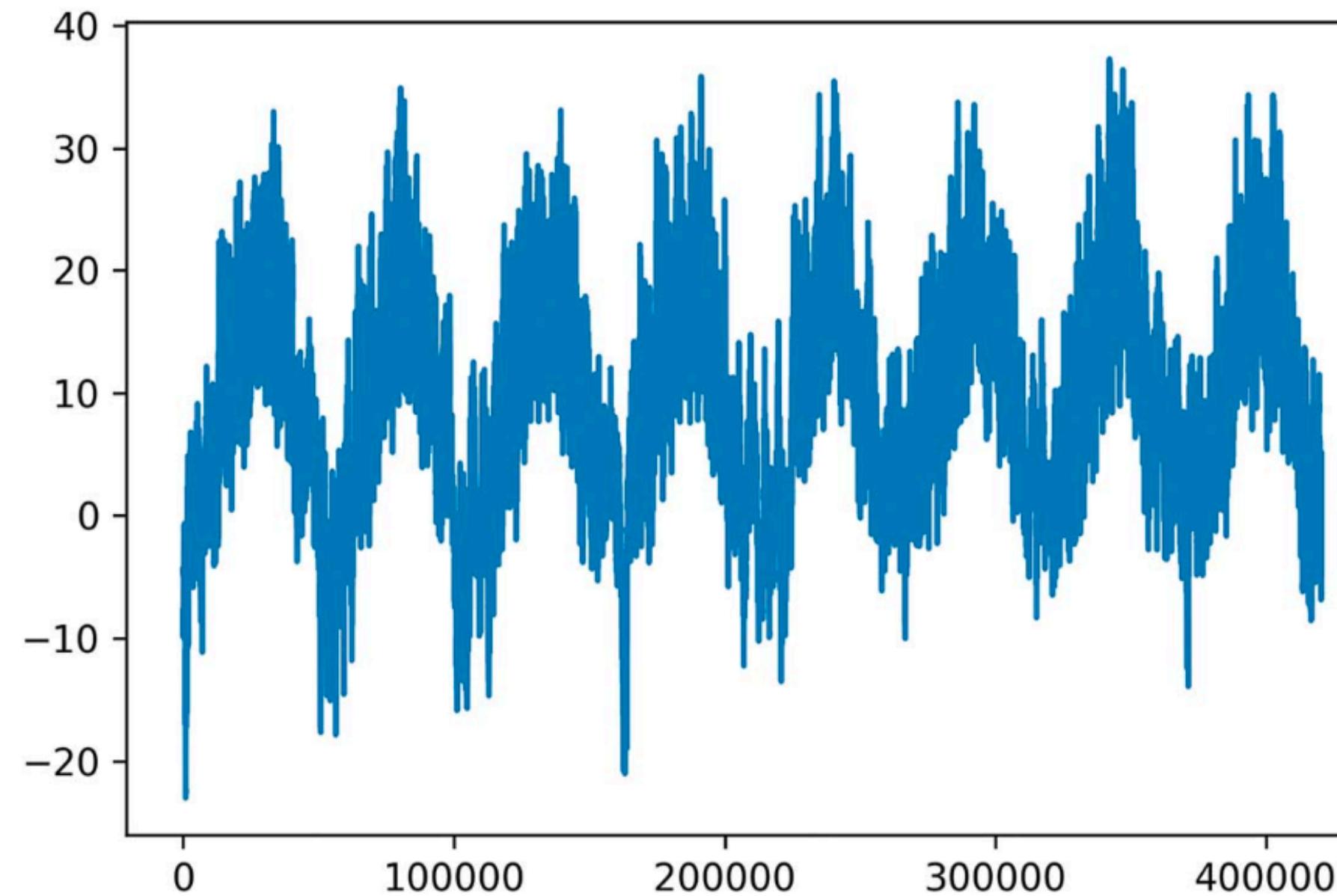


They keep the state of the previous neuron

A practical example (LSTMs): Temperature Forecasting

Time series forecasting

Temperature Forecasting
example



```
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))  
x = layers.LSTM(16)(inputs)  
outputs = layers.Dense(1)(x)  
model = keras.Model(inputs, outputs)
```



How Deep Learning is changing / will change the world?

Some examples using novel networks that are revolutionizing the AI world



New AI Model Translates 200 Languages, Making Technology Accessible to More People

July 6, 2022

Meta AI Introduces 'Make-A-Video': An Artificial Intelligence System That Generates Videos From Text

By **Ashish kumar** - October 3, 2022

<https://makeavideo.studio/>

Some examples using novel networks that are revolutionizing the AI world



New AI Model Translates 200 Languages, Making Technology Accessible to More People

July 6, 2022

Meta AI Introduces 'Make-A-Video': An Artificial Intelligence System That Generates Videos From Text

By **Ashish kumar** - October 3, 2022

<https://makeavideo.studio/>



Discovering novel algorithms with AlphaTensor

October 5, 2022

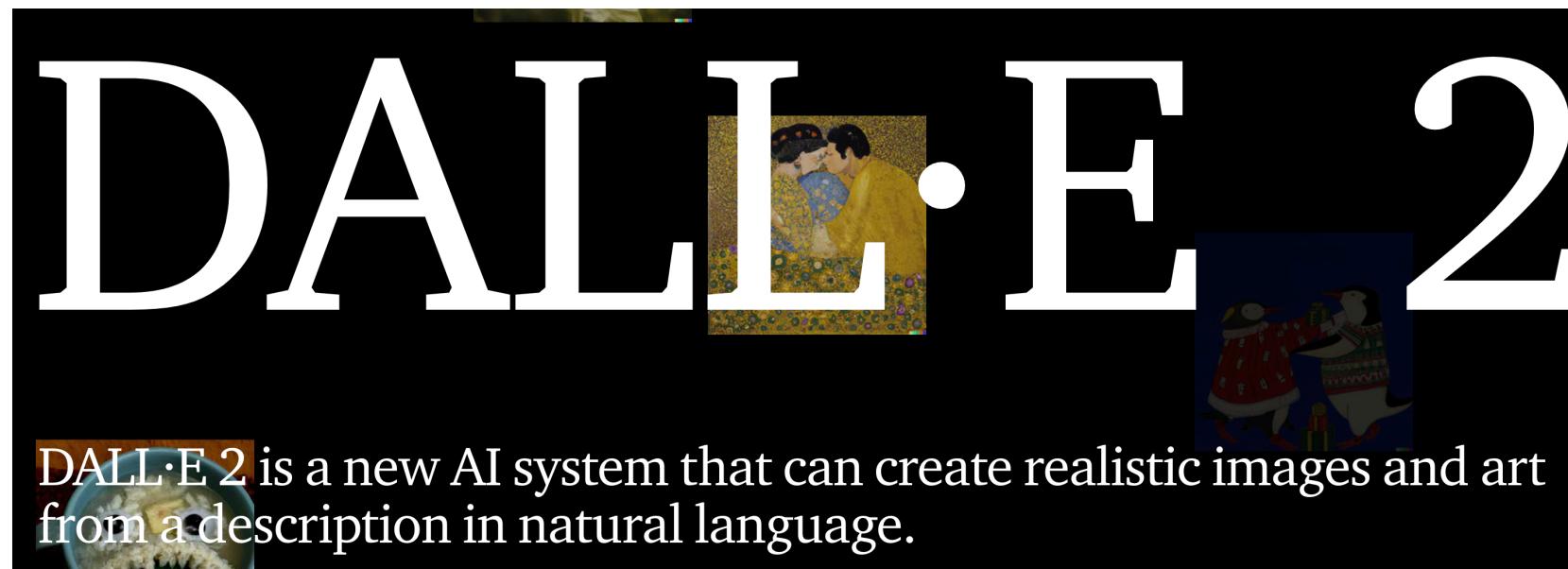
INTELIGENCIA ARTIFICIAL

Una IA de DeepMind halla una nueva forma de multiplicar números y acelerar los ordenadores

Article | **Open Access** | Published: 15 July 2021

Highly accurate protein structure prediction with AlphaFold

Some examples using novel networks that are revolutionizing the AI world



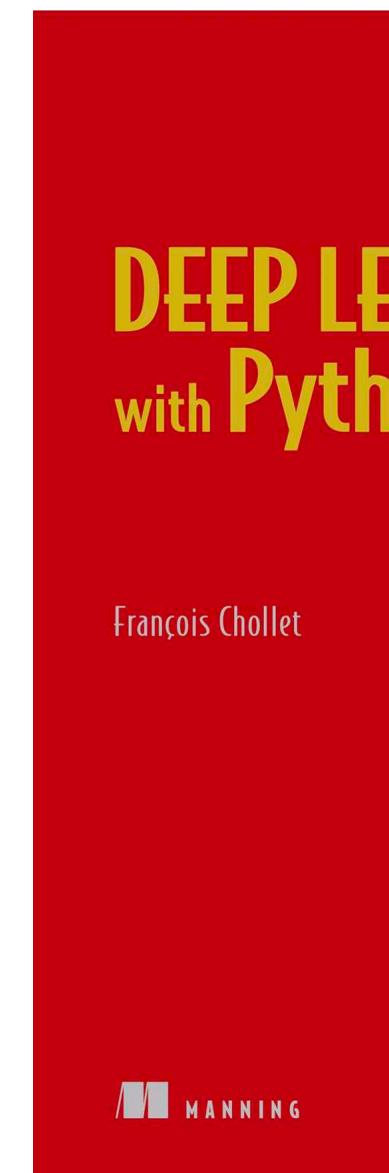
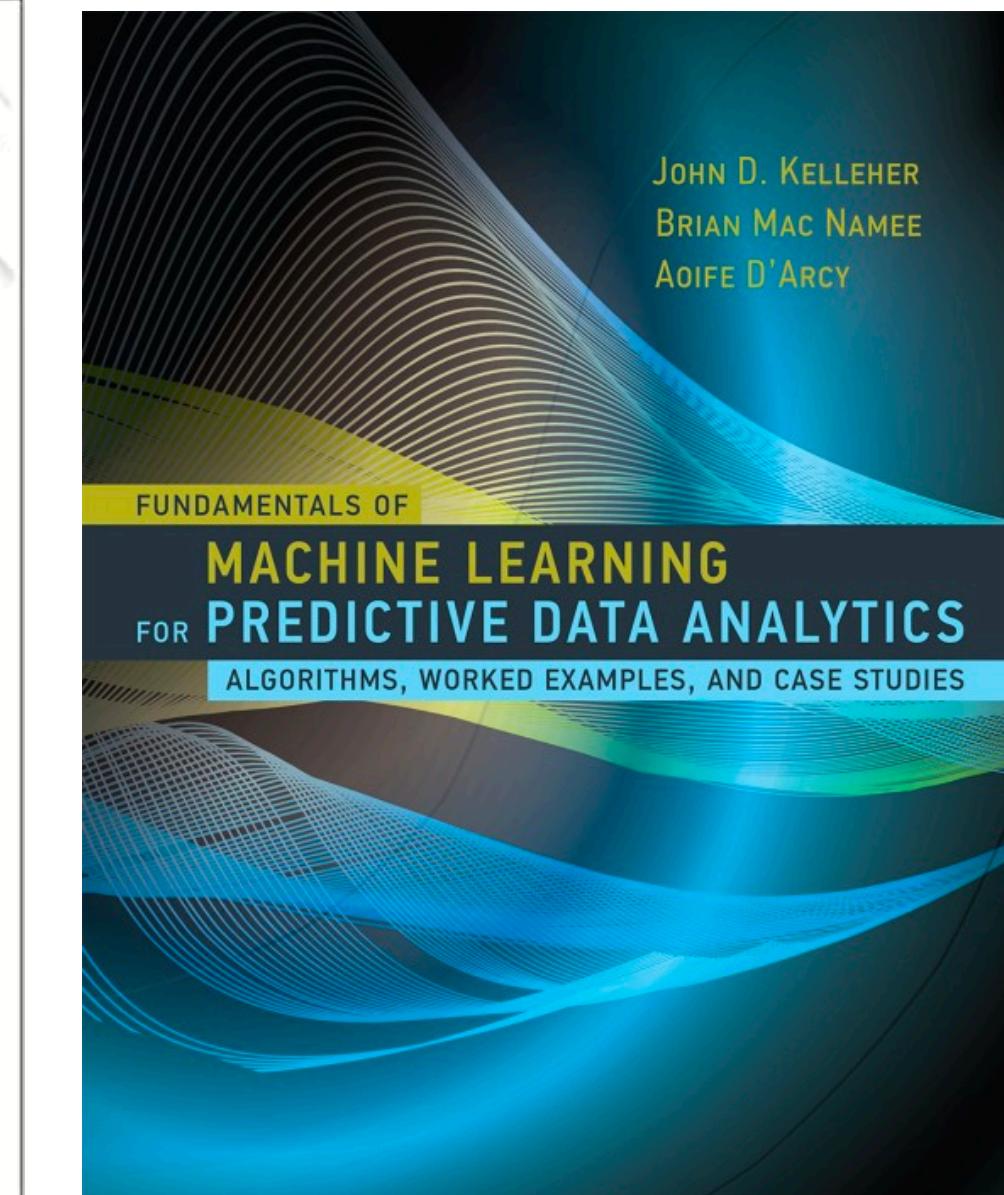
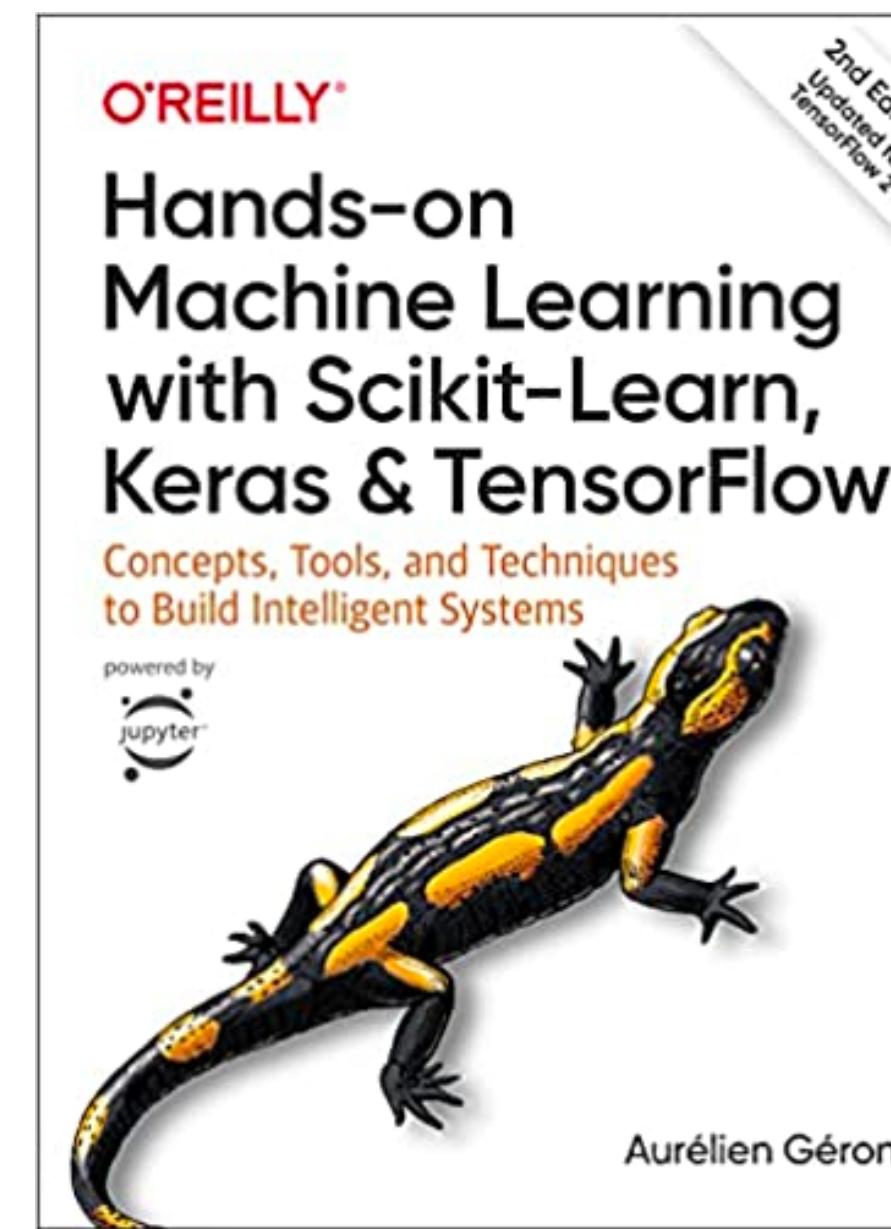
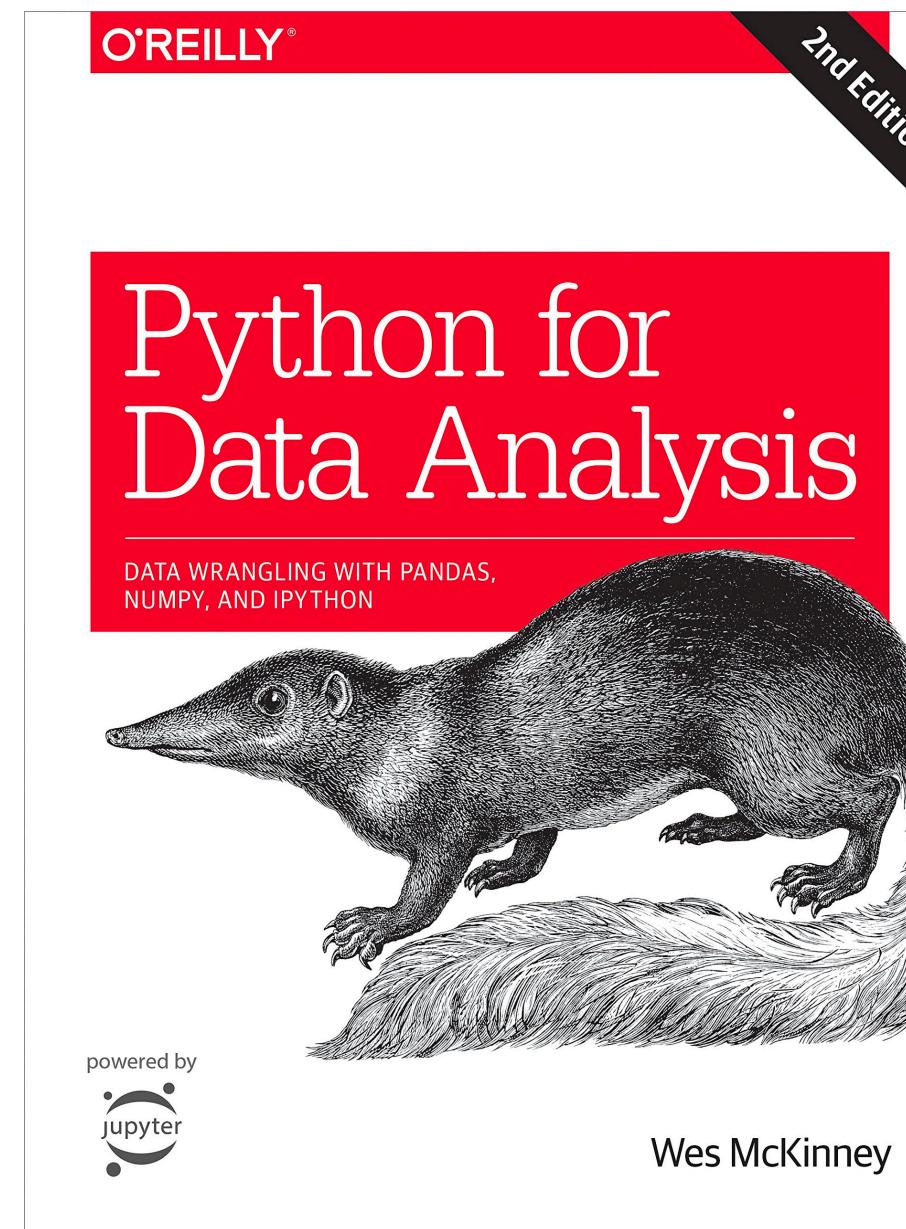
A beautiful landscape in the mountains, in the middle of the trees, with snow around and a big husky laying around





Last words

Free Books to learn Python, ML & DL



3Blue1Brown in YouTube

You are in the right place at the right moment (My Opinion)

- Computational analytical skills are in demand
- Companies need people who can make sense of their data and give them an edge with that.
- AI, ML, and DL are booming, and being able to apply such techniques will be increasingly important
- Math knowledge is not “strictly” necessary, instead having a general understanding and being able to build solutions on the cloud will be the key to the future (my take)
- ML engineer / Data engineer > Data Scientist



Thank you!



Feel Free to contact me!

- Linkedin: Jose Bonet Giner
- Personal mail: pepebogi5@gmail.com
- Company website: hyntsanalytics.com
- Blog: pepesjourney.com
- Twitter: [pepeb_5](https://twitter.com/pepeb_5)

The hynts logo consists of the word "hynts" in a lowercase sans-serif font. The letter "n" has a small orange dot positioned above its middle vertical stroke.