Training neural networks using Tensorflow

Lars Mennen

Cambridge Wireless, 23rd November 2017

Training neural networks using Tensorflow

- Two parts today:
  - General part about neural networks and how to train them
  - Training neural networks using Tensorflow
- Keep your laptops ready!
- Follow along with Python notebooks:

https://github.com/larsmennen/intro_to_tensorflow

(most code adopted from Tensorflow tutorials - tensorflow.org)

FIVE AI

Part I:
Training neural networks

Supervised learning

- Today's focus: **supervised** learning

Given input pairs

$$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

where x_i comes from some input space X and y_i comes from some output space Y, we try to learn a function f such that:

$$f(x') = y'$$

for some unseen pair (x', y') (but from the **same** spaces!)

FIVE AI

Classifying **images**

- Basic example: image classification

Training data: 1.2M images + their categories
(container ship, motor scooter, mushroom, ...)

Predict category for **unseen** image

2012 ImageNet classification challenge: only 16% top-5 error!

Image source: ImageNet competition

Source: ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012) - http://www.image-net.org/challenges/LSVRC/2012/

Supervised learning **methods**

- Various algorithms that attempt to solve this problem:
  - Nearest neighbour
  - Decision tree learning
  - Support vector machines
  - **Neural Networks**
  - ...
- There are **many** approaches to supervised learning
- Neural networks are not always the answer

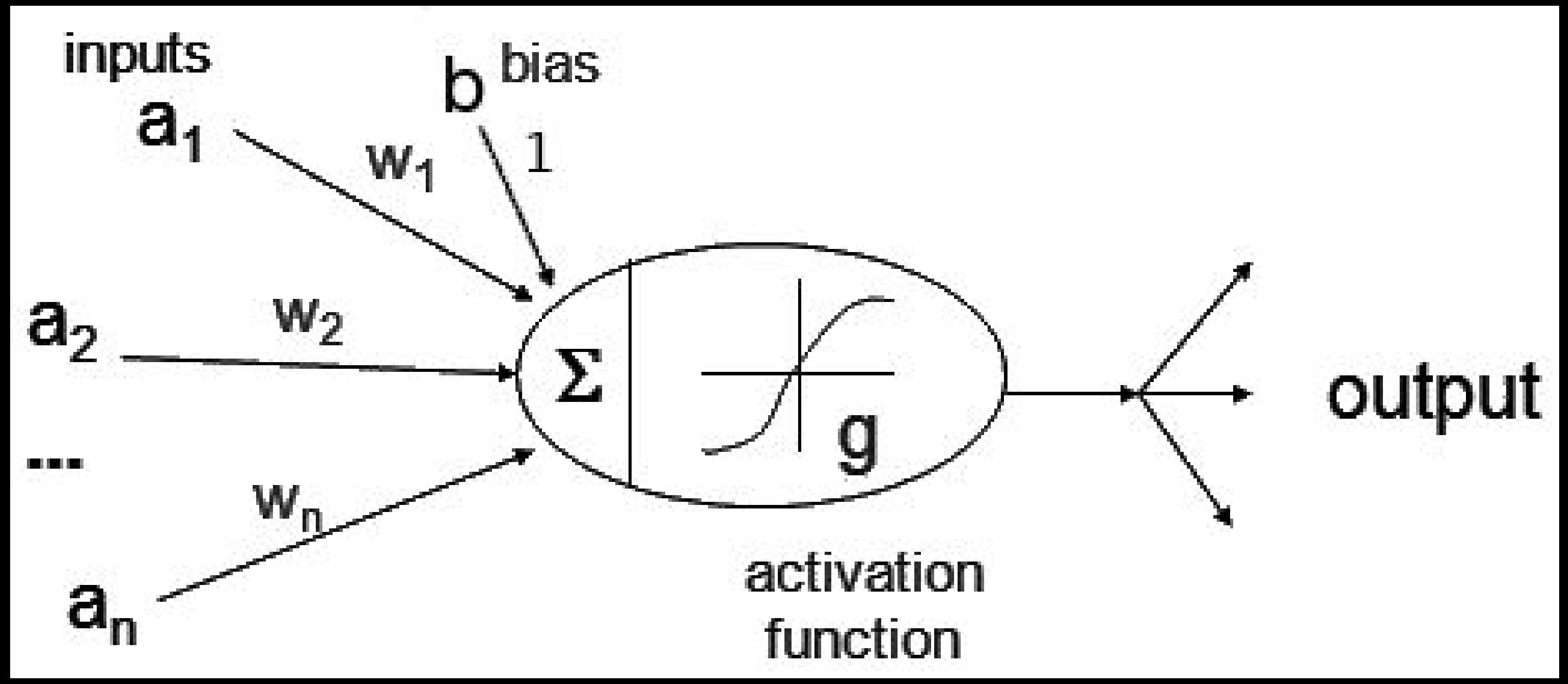
Neural **networks**

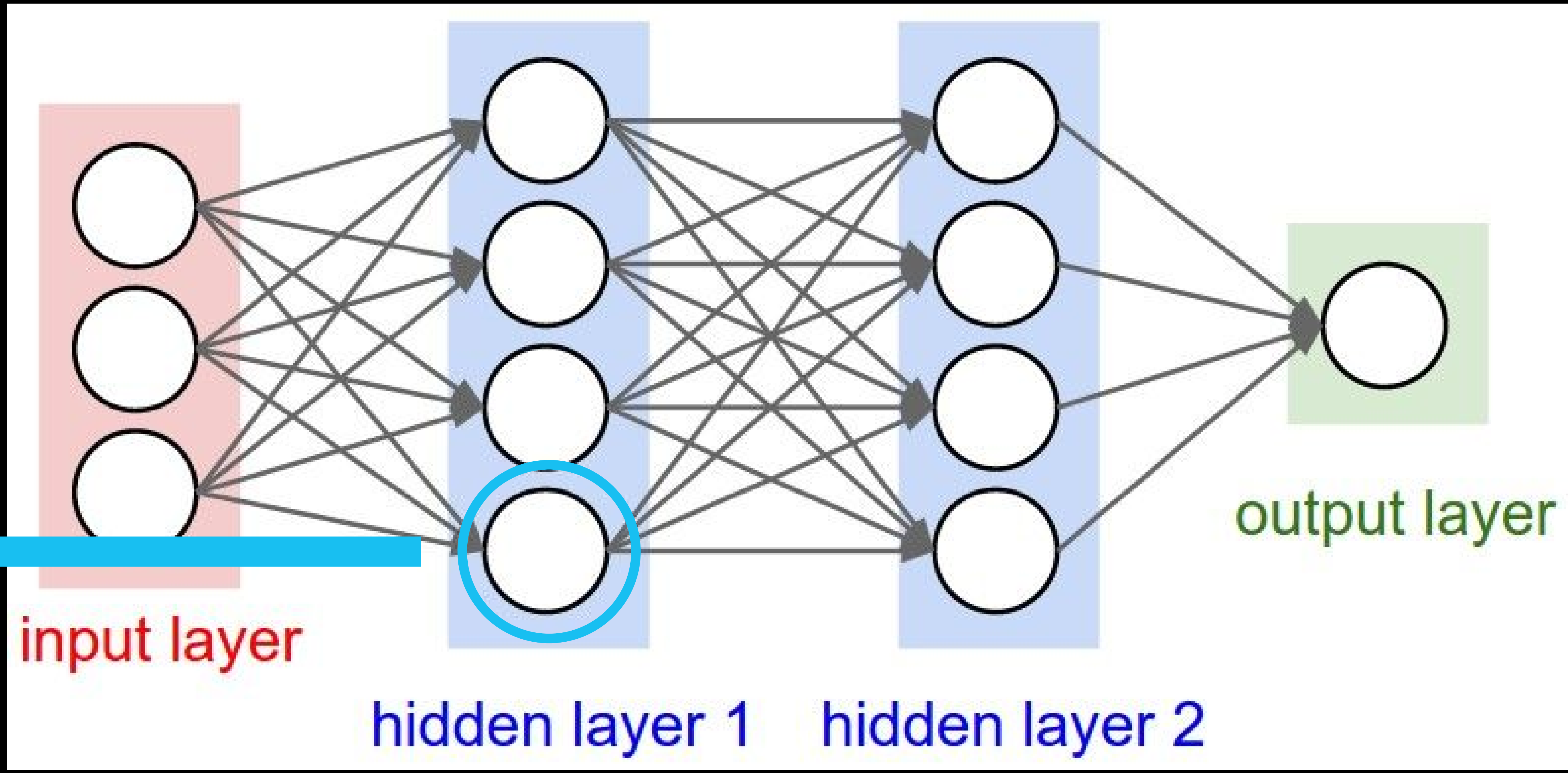
- Inspired by biological **neurons**
- Main structure:
  - Relatively simple neurons that compute a function given some inputs
  - Structured in ordered layers, where the neurons in each layer have as input a weighted sum of outputs of neurons in the previous layers.

FIVE AI

Neural **networks**

- So, we want to learn *f(x)* = *y*. Usually *x* and *y* are represented as vectors.
- We feed *x* to the input layer.





For each neuron: $$\text{activation} = g\left(\sum_{i=1}^{n} w_i a_i + b\right)$$

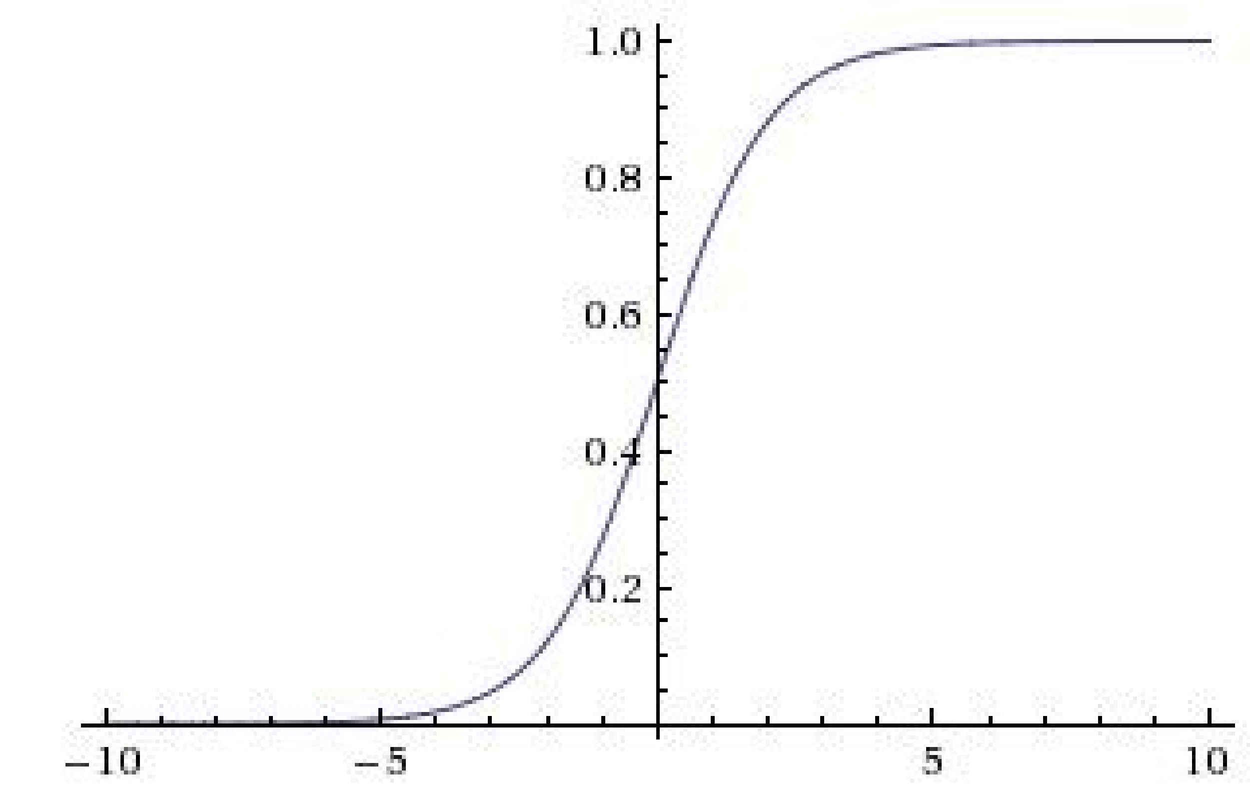
FIVE AI

Image source: Stanford

Neural **networks**

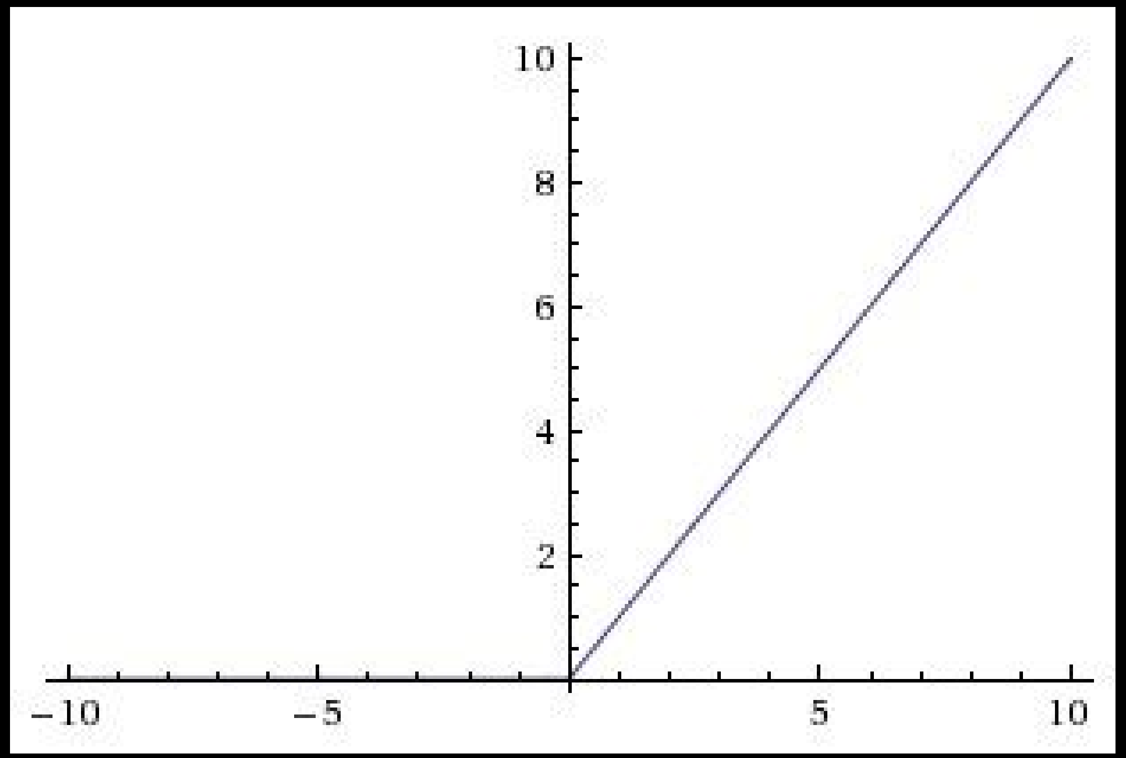
- Popular choices for activation function *g*:
  - Sigmoid:

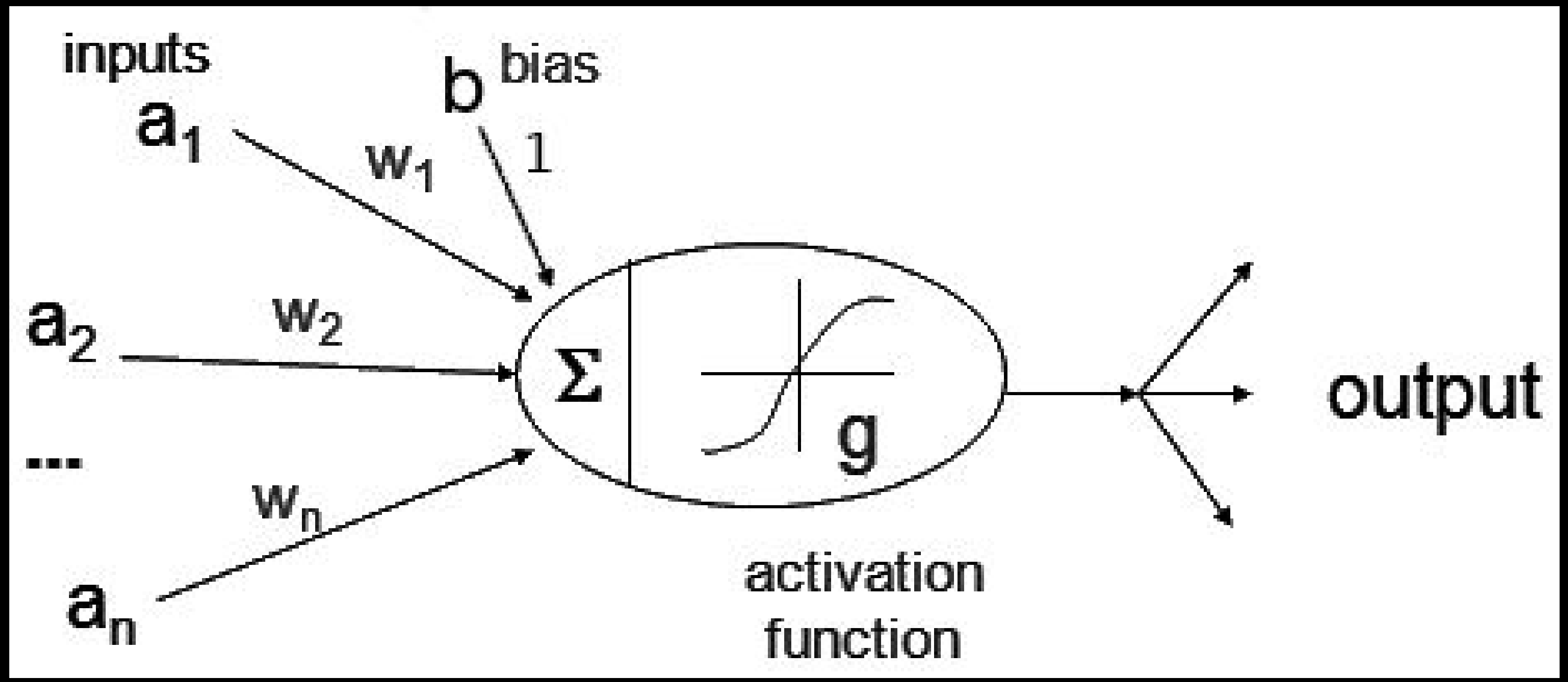
$$g(x) = \frac{1}{1+e^{-x}}$$



  - ReLU (rectified linear unit):

$$g(x) = \max(0, x)$$





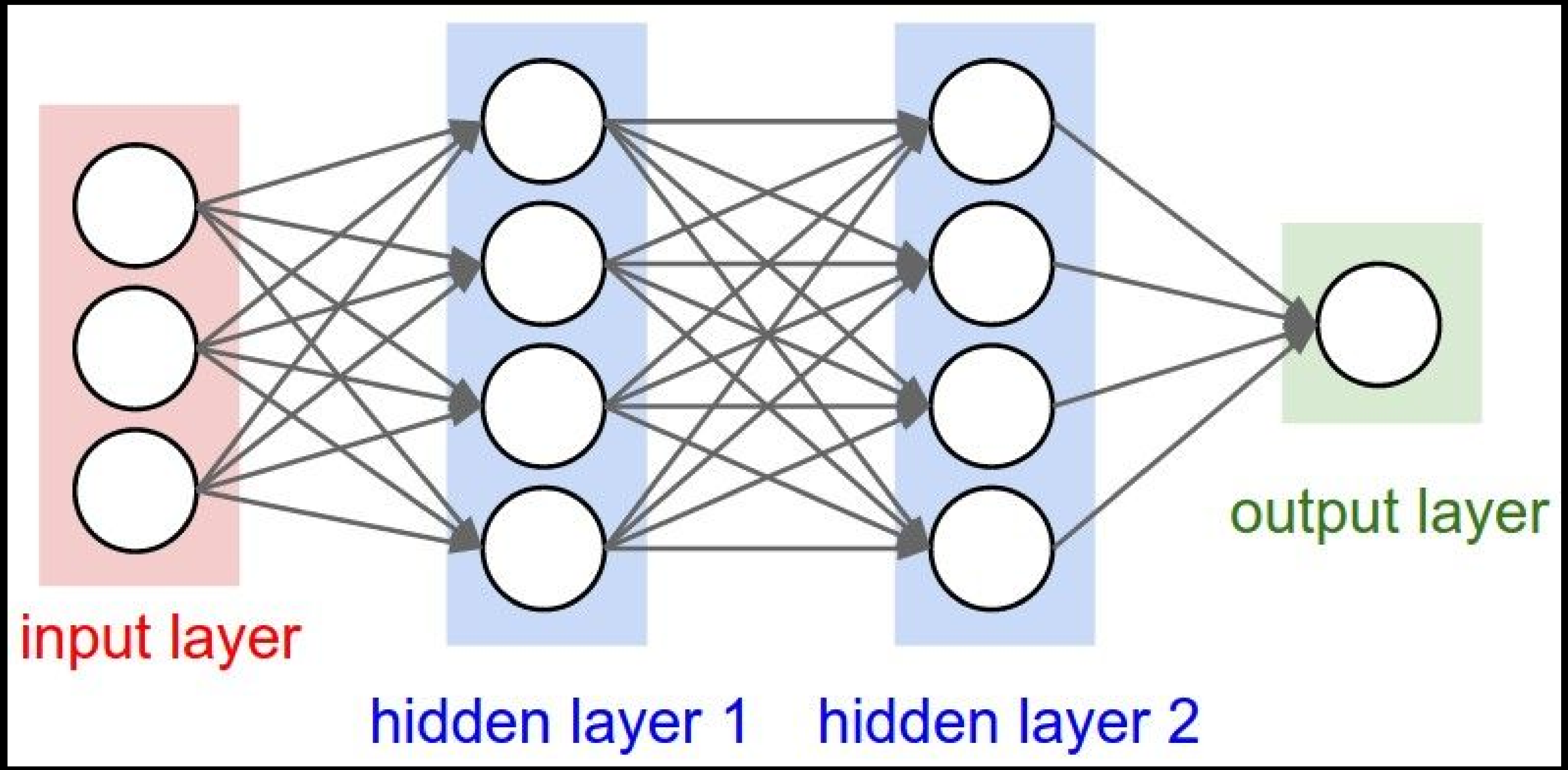
$$\text{activation} = g\left(\sum_{i=1}^{n} w_i a_i + b\right)$$

Note that these are **non-linear!**

Image source: Stanford

Neural **networks**

- Given a neural network as on the right, an input *x* and a function *g* we can now compute the value of the node(s) in the output layer!
- We want this value to correspond to the label *y* in the pair *(x,y)*, as then the network is computing *f(x) = y*.



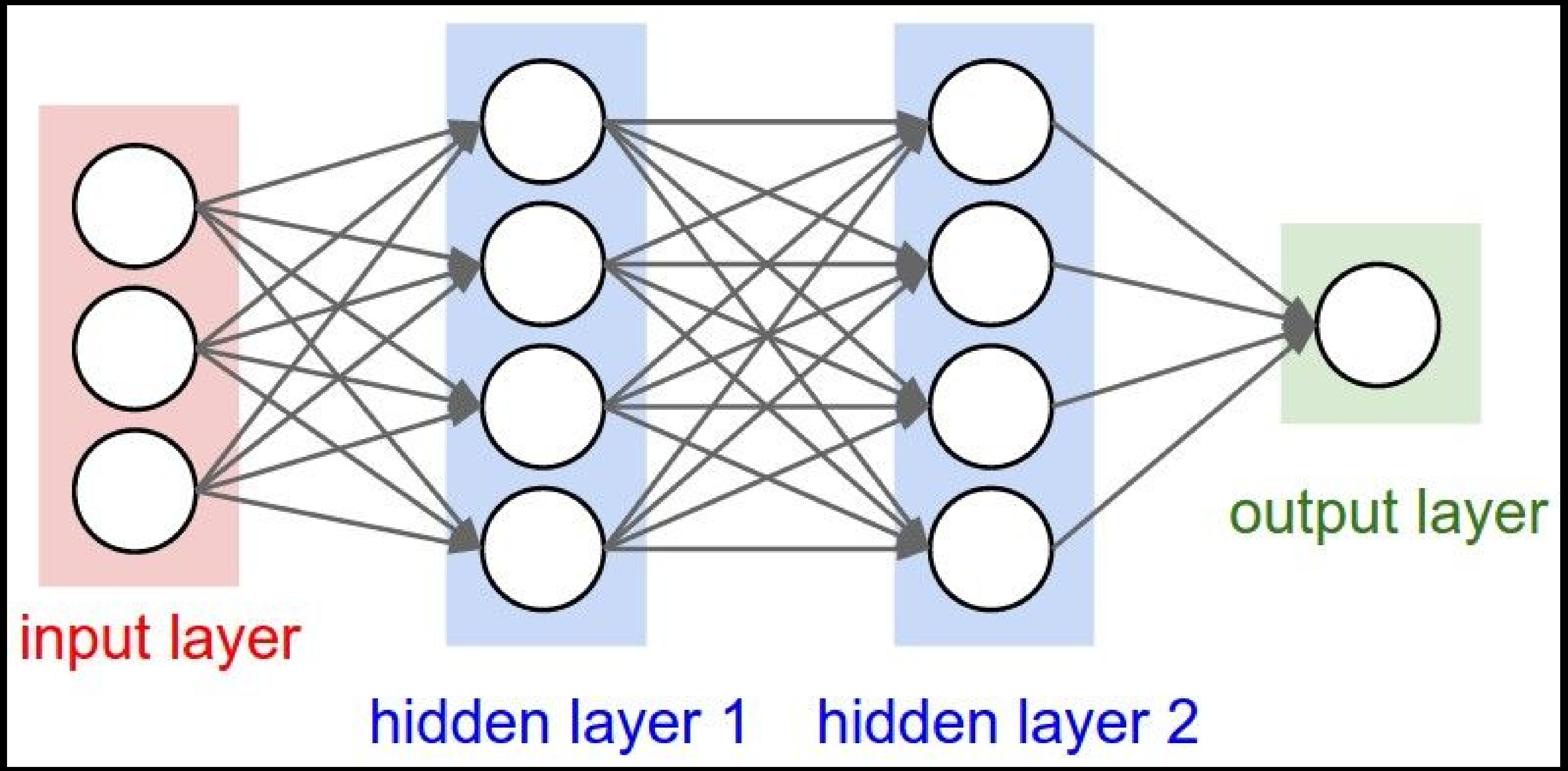
$$\text{activation} = g\left(\sum_{i=1}^{n} w_i a_i + b\right)$$

FIVE AI

Image source: Stanford

Neural **networks**

- However, how do we know which:
  - Layer structure
  - Activation function *g*
  - Values for weights for each layer
- we need to pick so that this network computes the function *f* that we want?



$$\text{activation} = g\left(\sum_{i=1}^{n} w_i a_i + b\right)$$

FIVE AI

Image source: Stanford

Neural **networks**

- Unfortunately, don't have learning algorithm to find layer structure for hidden layers and/or activation function g.
- Found by reasoning, experimentation and building on previous research.
- In this talk we assume the structure and activation function are given.

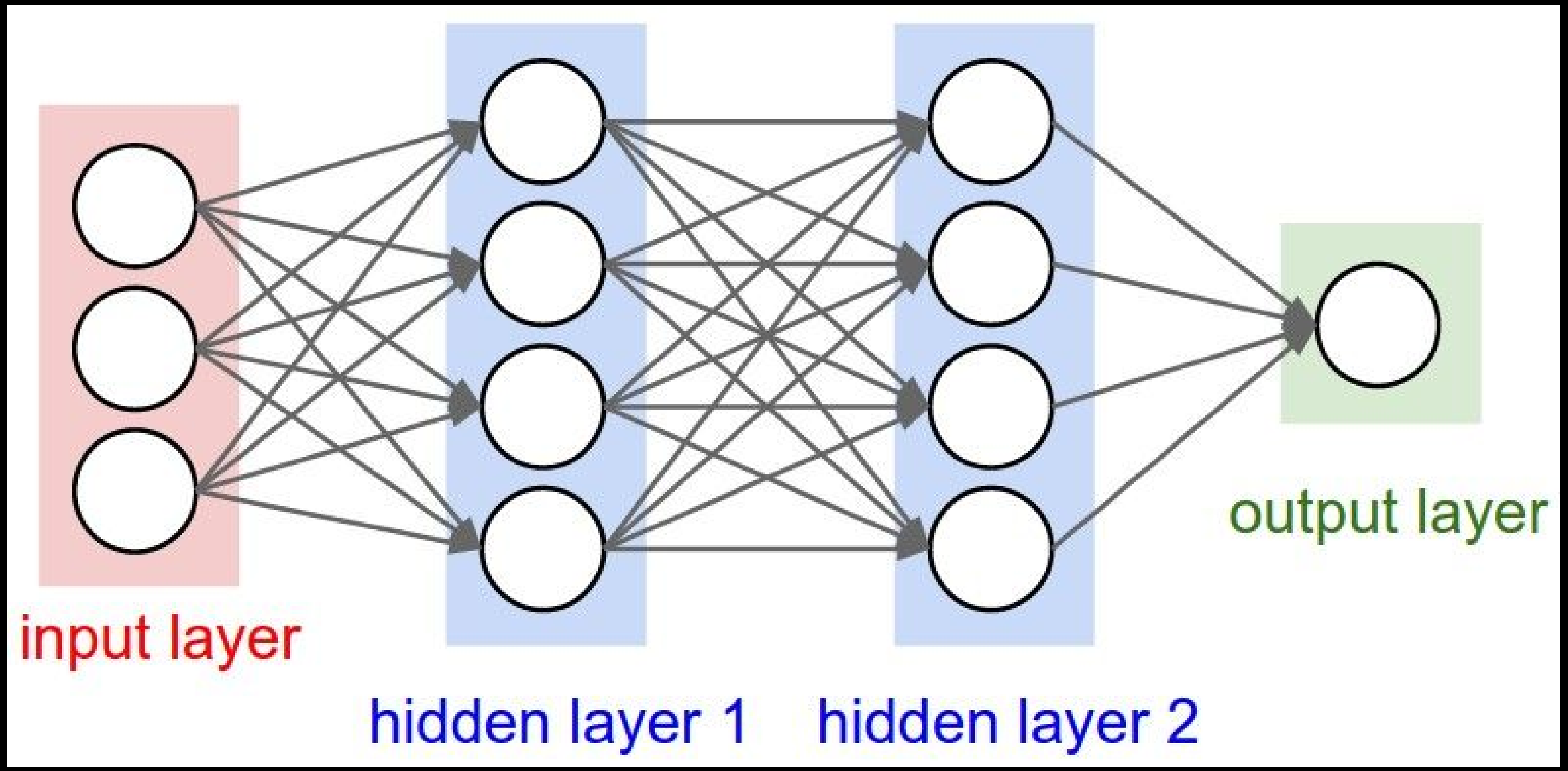
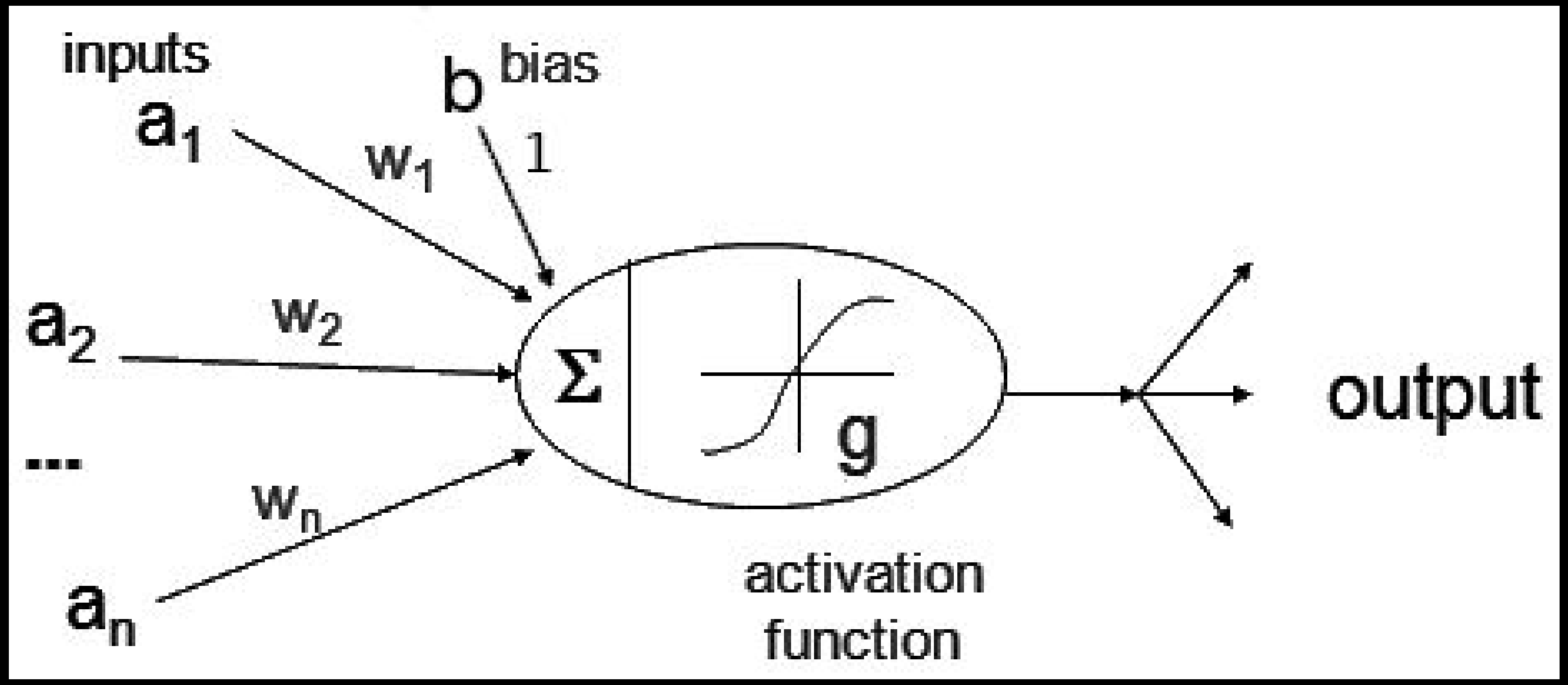


Image source: Stanford

FIVE AI

Neural **networks**

- We **do** have an algorithm to learn the weights:
  - Backpropagation
- The backpropagation algorithm, together with large amounts of data, powerful GPUs and *convolutional* neural networks (see later) is what makes modern NNs so popular and effective.



$$\text{activation} = g\left(\sum_{i=1}^{n} w_i a_i + b\right)$$

Image source: Stanford

Backpropagation **in a nutshell**

- We define a *loss function*:
  - Tells us "how far our prediction $f(x) = y'$ is off" from the label y
  - E.g. if we have a training example (x_i, y_i), one possible loss function is:

$$L = \frac{1}{2}(y_i - f(x_i))^2$$

- We want to **minimise L**
- Parameter space is way too big to set derivatives equal to 0!

FIVE AI

Backpropagation in a nutshell

- So we do it iteratively.
- How do we know in which direction we have to change weights to minimize *L*?
  - Look at the derivatives!
- Every single step in the neural network and the loss function are **differentiable**, which is key to the backpropagation algorithm.
- For every weight w_{ij} (from neuron *i* to neuron *j* in the next layer), we'd like to know: $\frac{\partial L}{\partial w_{ij}}$ as then we know in which direction we should move w_{ij}.

FIVE AI

Backpropagation **in a nutshell**

- I'll skip the exact maths here (great explanation on http://neuralnetworksanddeeplearning.com), but main idea:
- If you know all the intermediate activations (i.e. all outputs of the activation function *g*) for an input x_i, you can compute $\frac{\partial L}{\partial w_{ij}}$ for all weights using the **chain rule for derivatives**.
- We can express gradients in a layer in terms of gradients of the next layer.
- So if we start at the last layer, we can **backpropagate** to find gradients in previous layers.

FIVE AI

Backpropagation in a nutshell

- Given these expressions, we can use an optimisation method, e.g. **gradient descent**, to adjust the weights in a way that will minimise the loss ***L***.
- **Forward pass**: compute all activations for a given input x_i.
- **Backward pass:** compute gradients and change weights according to optimisation method

FIVE AI

Backpropagation in a nutshell

Main algorithm:

1. initialize all weights randomly
2. repeat until stopping criterion is met:
   a. for (x_i, y_i) in dataset D:
      i. **Forward pass:** compute $f(x_i)$, store intermediate activations, and compute L
      ii. **Backward pass:** compute gradients w.r.t. L and update weights according to optimisation method

Note that updating the weights **changes** f!

FIVE AI

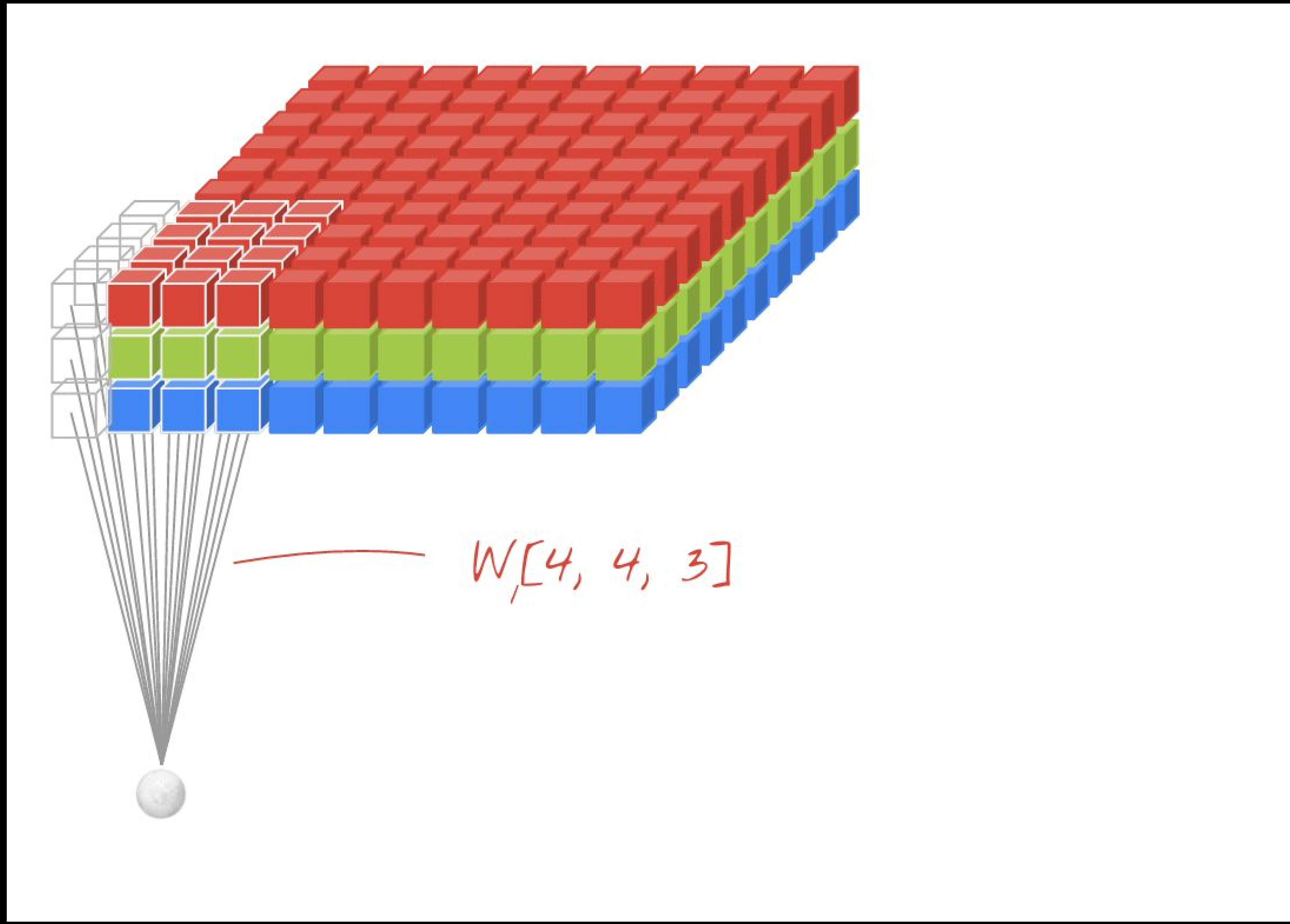
Backpropagation **in a nutshell**

- In practice, do forward pass for multiple training examples at once (batching):
  - More efficient
  - Less noisy gradients
- Which stopping criterion to use?
  - Loss doesn't drop anymore
  - Better: look at loss on held-out validation set
    - Otherwise you might **overfit** on the particular training set, and hence **fail to generalise** to new examples.

FIVE AI

Using convolutions

- **Fully connected** layers don't scale well to images
- **Convolutional** layers:
  - Convolve learned weights with input
  - Weights **shared** along spatial dimensions
  - $k^2 \times c_i \times c_o$ weights for k x k kernel from c_i input channels to c_o output channels



FIVE AI

Image source: Google CodeLabs

Training **neural networks**

In summary:

- Given a supervised learning problem and a dataset D:
  - Define the structure of a neural net, an activation function and a loss function.
  - Learn the weights using the algorithm described before
  - You now have a function **f: X -> Y** which you can use to compute *f(x)* on unseen *x*.

FIVE AI

Part 2:
Tensorflow

Google Tensorflow

What is Tensorflow?

- *"TensorFlow™ is an* ***open source*** *software library for numerical computation using data flow graphs."*
- Probably **the most popular** open-source framework for training neural nets (but it's more general than that!)
- Large community, easy to use Python interface
- Used extensively in industry and research
- Development moves extremely fast!

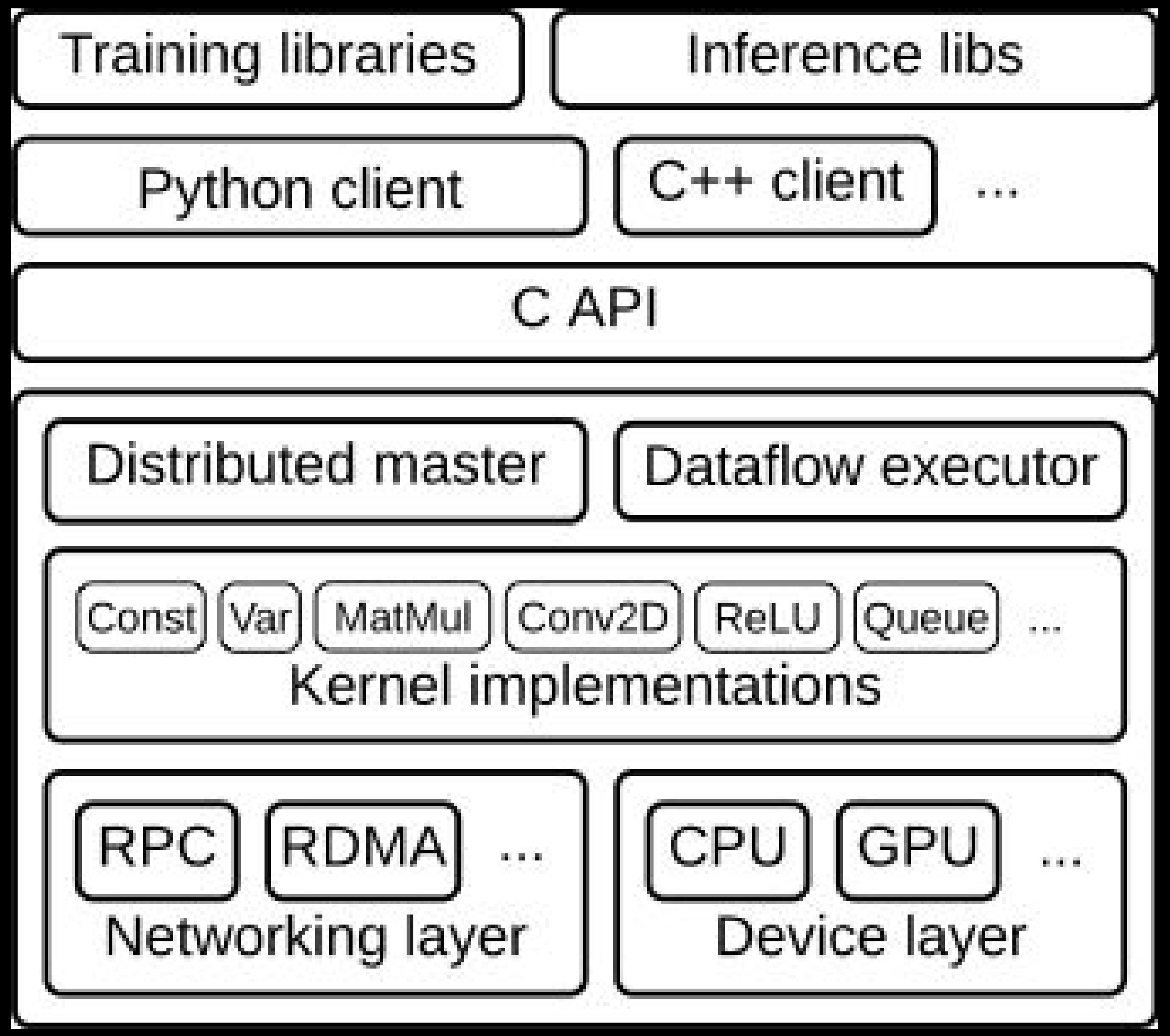
FIVE AI

Tensorflow **Overview**

- Tensorflow allows you to define, train, evaluate and perform inference on neural networks.
- Lots of extra functionality:
    - Tensorboard - visualising neural networks and training
    - Serving - serving models in production
    - Training on HPC clusters
    - Preprocessing data
    - Quantization of neural networks
    - ...
- APIs for C++, **Python**, Java and Go

Tensorflow **Architecture**

- Main implementations in C(++)
- Every operation can have a CPU and/or GPU implementation
- Most GPU code uses NVIDIA CUDA (proprietary)
  - CuDNN for common neural net operations
  - Efforts to get OpenCL support
- Relies heavily on **Eigen** and **Protobuf**



FIVE AI

Image source: tensorflow.org

Concepts: Tensors

- Computations in Tensorflow are done on **tensors**
- Generalisation of matrices to higher dimensions
- E.g. a tensor of rank 4 of dimensions (10,2,2,5) would have 10 * 2 * 2 * 5 = 200 elements
- Tensors have strong typing
- For input data, usually the first dimension is the **batch size**
  - E.g. feedforward pass for 4 images at once: (4, ...)

FIVE AI

Concepts: Computation Graph

- **All computations** in Tensorflow are represented in the computation graph
  - Neural network, optimiser, ...
- The majority of code you'll write in Python **does not** actually execute the network on data; it constructs the computation graph
- Graph consists of **Operations** whose inputs and outputs are **Tensors.**
- Input data is represented by **placeholders**

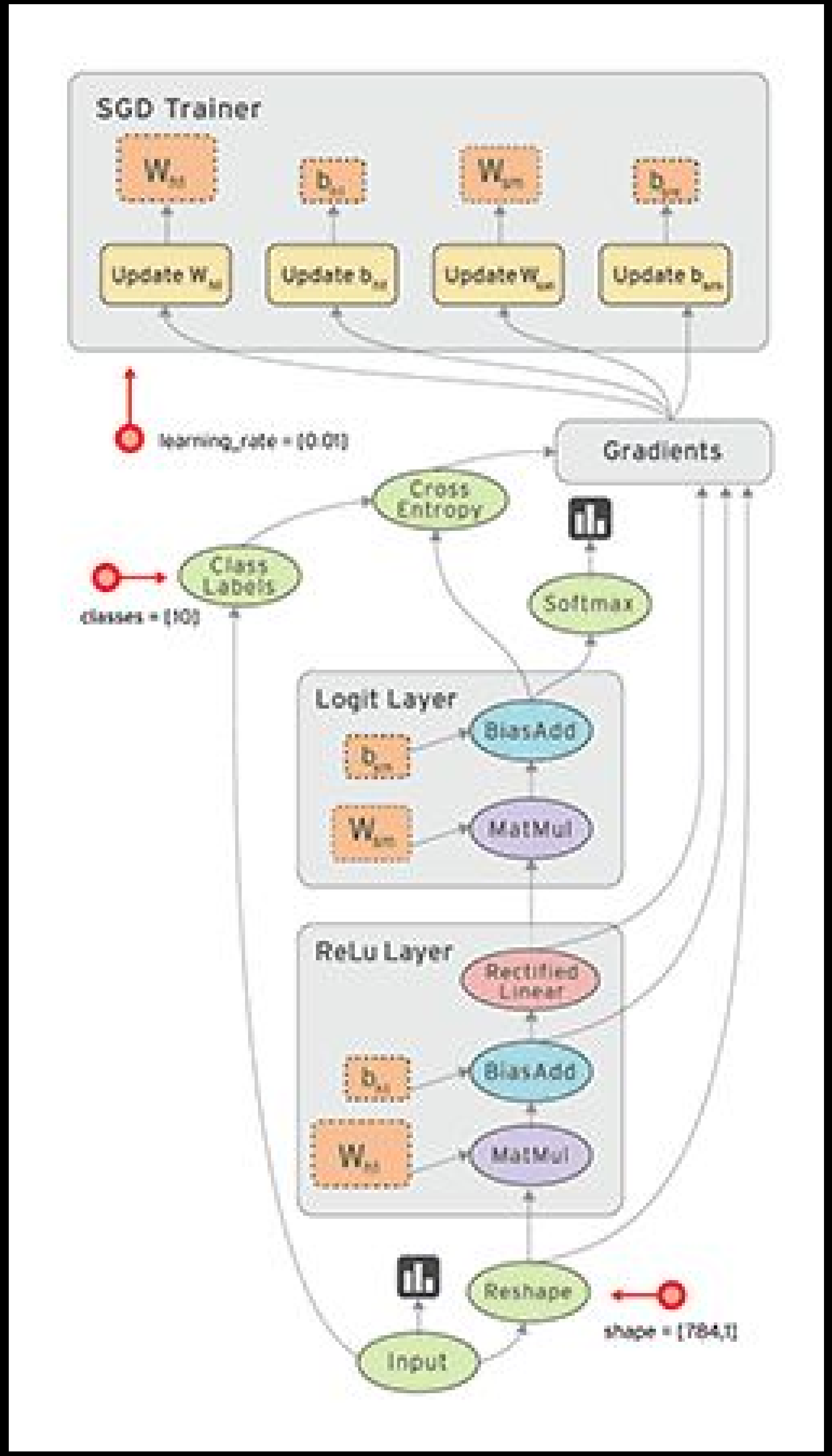


Image source: tensorflow.org

Concepts: Operations and Kernels

- **Operations** run **kernels**
- Operations:
  - Metadata
  - Shape and type inference
    - Can work with partially defined shapes
  - Central registry
- Kernels:
  - Actual implementations on CPU or GPU
  - Can work for only certain types
  - Often Eigen for CPU kernels, NVIDIA CUDA/CuDNN for GPU kernels

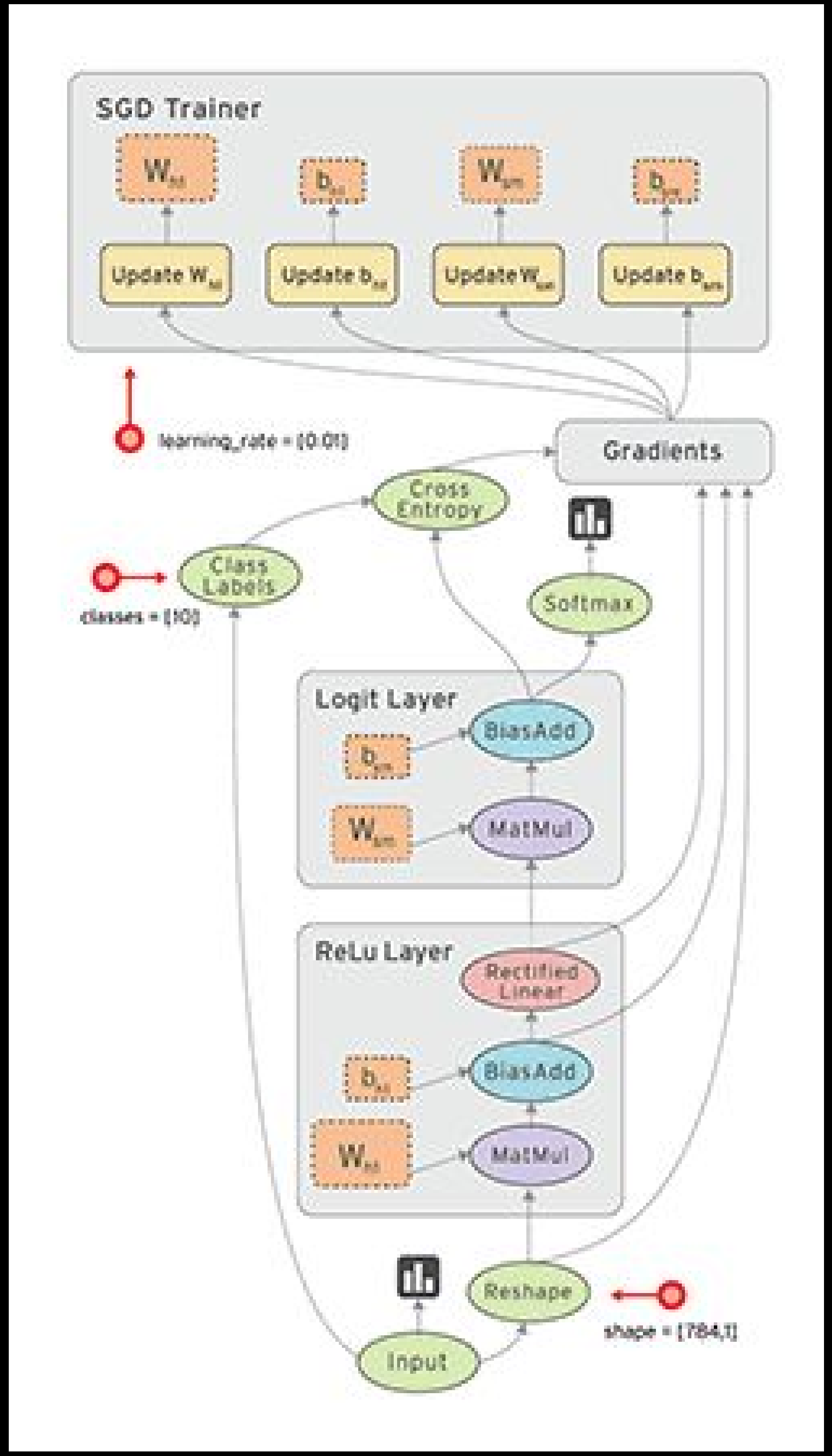


Image source: tensorflow.org

Concepts: Operations and Kernels

- Usually NN operations need **gradient operations**
- Tensorflow deduces which kernel to use and handles memory management for you
  - E.g. CPU-only operation after GPU-only operation
  - Possible to force placement on a specific CPU or GPU
- You can implement your own operations.
  - Python: as a combination of existing operations
  - C++: load at runtime as shared library

FIVE AI

Concepts: **Session**

- Represents the connection between the client (Python) and the C(++) runtime
- Provides access to the CPU and GPU device(s), which may be remote
- Allows to evaluate (parts of) the graph on data

FIVE AI

Time to code!

Installation

These instructions can be found on:
https://github.com/larsmennen/intro_to_tensorflow

Vanilla Python or virtualenv (CPU only):
```
pip3 install tensorflow
```

Vanilla Python or virtualenv (GPU, CUDA and CuDNN present):
```
pip3 install tensorflow-gpu
```

Anaconda (NVIDIA GPU)
```
conda install tensorflow-gpu
```

Anaconda (no NVIDIA GPU)
```
conda install tensorflow
```

FIVE AI

Recognising Handwritten Digits

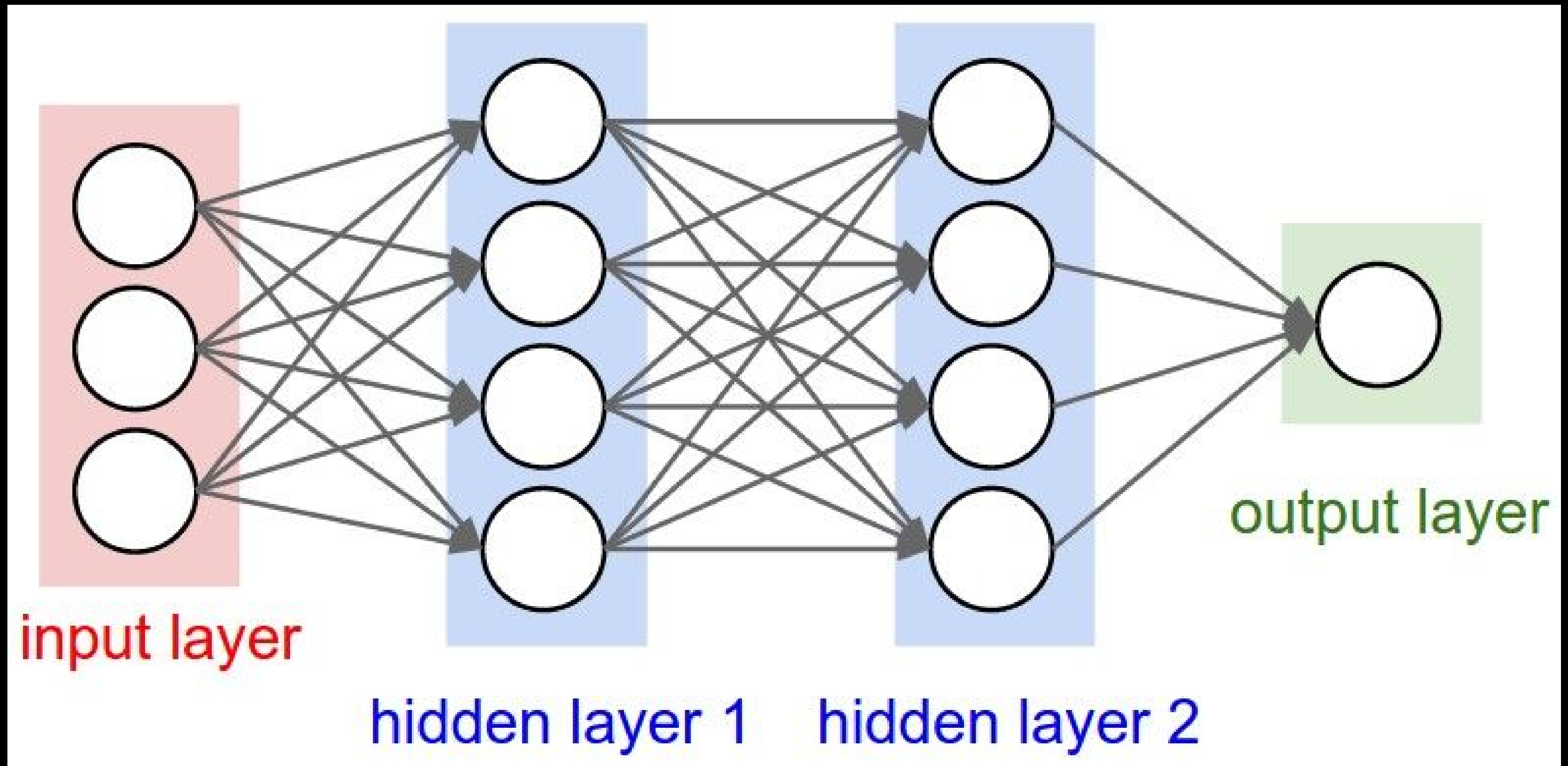
- We'll follow the Tensorflow tutorial on MNIST, but more in-depth
- Recognising handwritten digits
- **Classification problem**, 10 classes
- Data: pairs (**x**,**y**) where **x** is a 28x28 pixel image (which we'll flatten to a 784-element vector) of a handwritten digit and **y** is a 10-element one-hot vector representing the label
- 55k training, 5k validation, 10k test

Network definition in Tensorflow

mnist.py and the inspecting_mnist notebook

Network definition in Tensorflow

784 128 32 10



Gives a total of:

784 * 128 + 128 * 32 + 32 * 10 = 104 768

weights we need to train

FIVE AI

Image source: Stanford

Let's train our network!

Network training in Tensorflow

training_mnist notebook

FIVE AI

Higher level interfaces

- Using Tensorflow as we did today can get cumbersome
- There are higher level interfaces that make development easier and cleaner:
  - tf.slim
  - tf.estimator
  - Keras
- Keras provides a **clean, functional** API and only uses Tensorflow as a **backend** (can also use Microsoft CNTK or Theano) https://keras.io/

FIVE AI

More **resources**

- Explanation of (convolutional) neural networks:
  http://neuralnetworksanddeeplearning.com
  http://cs231n.stanford.edu/
- Tensorflow:
  https://www.tensorflow.org/
- OpenCL support for Tensorflow:
  https://github.com/tensorflow/tensorflow/issues/22

FIVE AI

We're hiring!
five.ai/careers

DevOps Engineer

Engineering Manager

Graduate Engineer

Internship

Machine Learning for Visual Scene Understanding

Machine Vision and Image Processing

Office Manager - Edinburgh

Research Engineer - Computer Vision and Deep Learning

Research Engineer - Robotics

Research Scientist

Research Scientist - Activity Understanding and Prediction

Research Scientist - Machine Learning for Driving Decisions

Research Scientist - Motion Prediction

Simulation Developer

Software Engineer - Machine Vision + Image Processing

Software Engineer - Platform

Questions?

Operation **implementation**

Let's have a look under the hood. How is an operation actually implemented?

Simple example: **tf.sigmoid** $g(x) = \frac{1}{1 + e^{-x}}$

tf.sigmoid

Aliases:

- `tf.nn.sigmoid`
- `tf.sigmoid`

```
sigmoid(
    x,
    name=None
)
```

Defined in `tensorflow/python/ops/math_ops.py`.

See the guide: Neural Network > Activation Functions

Computes sigmoid of `x` element-wise.

Specifically, `y = 1 / (1 + exp(-x))`.

Image source: tensorflow.org

Operation **implementation**

`tensorflow/python/ops/math_ops.py`

```python
def sigmoid(x, name=None):
  """Computes sigmoid of `x` element-wise.

  Specifically, `y = 1 / (1 + exp(-x))`.

  Args:
    x: A Tensor with type `float32`, `float64`, `int32`,
`complex64`, `int64`,
      or `qint32`.
    name: A name for the operation (optional).

  Returns:
    A Tensor with the same type as `x` if `x.dtype !=
qint32`
      otherwise the return type is `quint8`.

  @compatibility(numpy)
  Equivalent to np.scipy.special.expit
  @end_compatibility
  """
  with ops.name_scope(name, "Sigmoid", [x]) as name:
    x = ops.convert_to_tensor(x, name="x")
    return gen_math_ops._sigmoid(x, name=name)
```

Namespacing

If *x* is not already a tensor, convert it to a tensor

FIVE AI

Operation **implementation**

`tensorflow/python/ops/gen_math_ops.py`

```
def _sigmoid(x, name=None):
  r"""Computes sigmoid of `x` element-wise.

  Specifically, `y = 1 / (1 + exp(-x))`.

  Args:
    x: A `Tensor`. Must be one of the following types:
`half`, `float32`, `float64`, `complex64`, `complex128`.
    name: A name for the operation (optional).

  Returns:
    A `Tensor`. Has the same type as `x`.
  """
  result =  op def_lib.apply_op("Sigmoid", x=x, name=name)
  return result
```

Just invokes the "Sigmoid" operation from the *operation registry* on *x*

Which will bring us to C++, so in Python only some conversions and checks!

FIVE AI

Operation **implementation**

`tensorflow/core/ops/math_ops.cc`

```
REGISTER_OP("Sigmoid").UNARY_COMPLEX().Doc(R"doc(
Computes sigmoid of `x` element-wise.

Specifically, `y = 1 / (1 + exp(-x))`.
)doc");

REGISTER_OP("SigmoidGrad").UNARY_GRADIENT_COMPLEX().Doc(R"doc(
Computes the gradient of the sigmoid of `x` wrt its input.

Specifically, `grad = dy * y * (1 - y)`, where `y = sigmoid(x)`, and
`dy` is the corresponding input gradient.
)doc");
```

```
#define UNARY_COMPLEX()                                                   \
  Input("x: T")                                                           \
      .Output("y: T")                                                     \
      .Attr("T: {half, float, double, complex64, complex128}")            \
      .SetShapeFn(shape_inference::UnchangedShape)

#define UNARY_GRADIENT_COMPLEX()                                          \
  Input("x: T")                                                           \
      .Input("y: T")                                                      \
      .Output("z: T")                                                     \
      .Attr("T: {half, float, double, complex64, complex128}")            \
      .SetShapeFn(shape_inference::UnchangedShape)
```

This registers the **Operation**, but **no kernels** yet. So this is just "metadata".
Note both Sigmoid and SigmoidGrad.

FIVE AI

Operation **implementation**

`tensorflow/core/kernels/cwise_op_sigmoid.cc`

```
REGISTER5(UnaryOp, CPU, "Sigmoid", functor::sigmoid, float, Eigen::half, double,
          complex64, complex128);
#if GOOGLE_CUDA
REGISTER3(UnaryOp, GPU, "Sigmoid", functor::sigmoid, float, Eigen::half,
          double);
#endif
```

```
REGISTER5(SimpleBinaryOp, CPU, "SigmoidGrad", functor::sigmoid_grad, float,
          Eigen::half, double, complex64, complex128);
#if GOOGLE_CUDA
REGISTER3(SimpleBinaryOp, GPU, "SigmoidGrad", functor::sigmoid_grad, float,
          Eigen::half, double);
#endif
```

```
#define REGISTER(OP, D, N, F, T)                                             \
  REGISTER_KERNEL_BUILDER(Name(N).Device(DEVICE_##D).TypeConstraint<T>("T"), \
                          OP<D##Device, F<T>>);
```

This registers the **kernels**. Separate for CPU and GPU, may have different supported features.

FIVE AI