Imperial College London

Introduction to Machine Learning

Part 3: Neural networks and deep learning

John Pinney

November 2020

Intended learning outcomes

After attending this workshop, you will be better able to:

- Explain the difference between supervised and unsupervised learning.
- Select a suitable machine learning method for a given application.
- Prepare your own training and testing data sets.
- Evaluate the performance of a machine learning experiment.

Overview

Neural networks

- Perceptron
- Multiple layers
- Gradient descent
- Backpropagation
- Activation functions

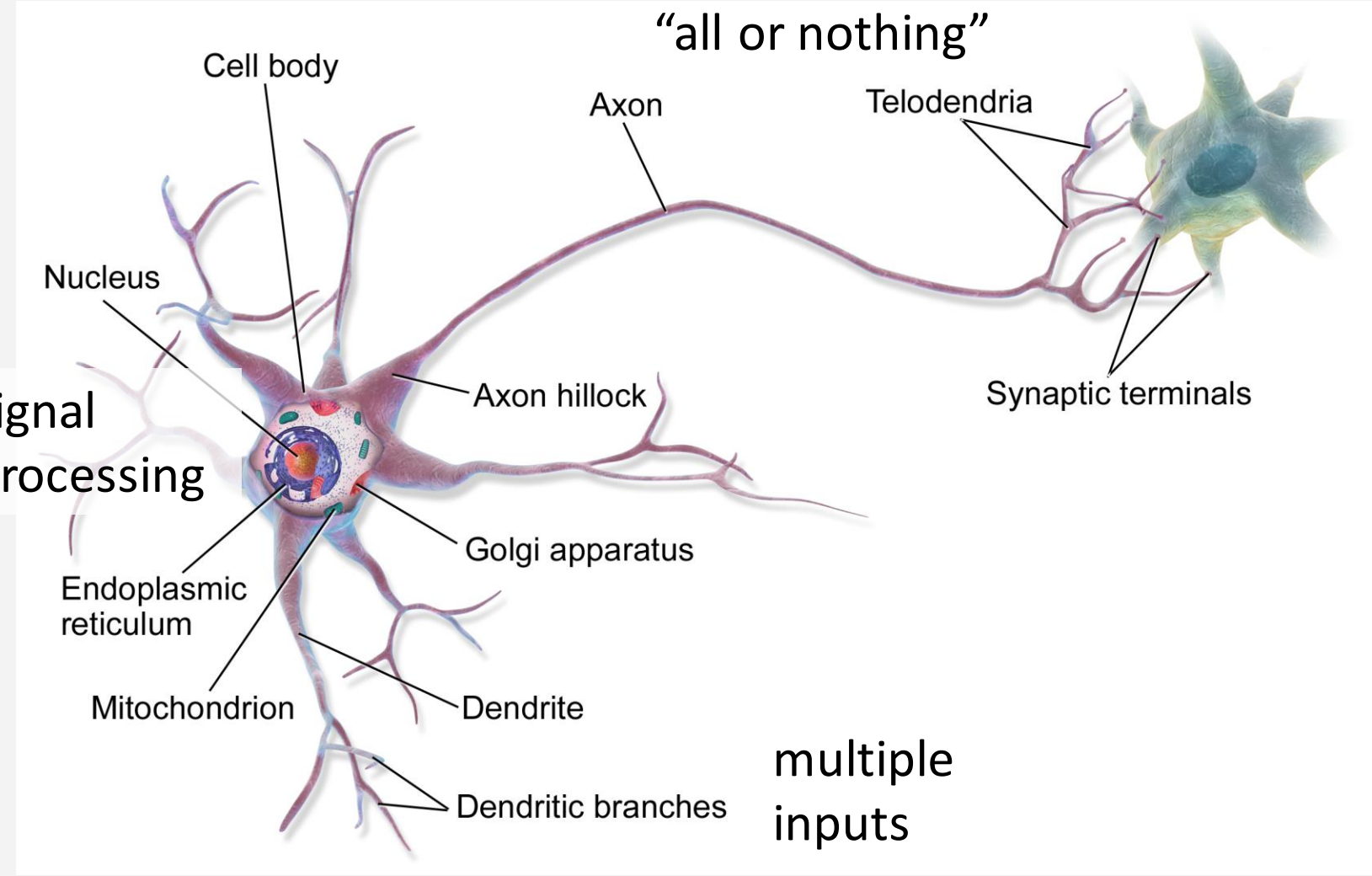
Deep learning

- Deep networks
- Unstructured data
- Convolution
- Embedding
- Special architectures

Neural networks

Biological neuron

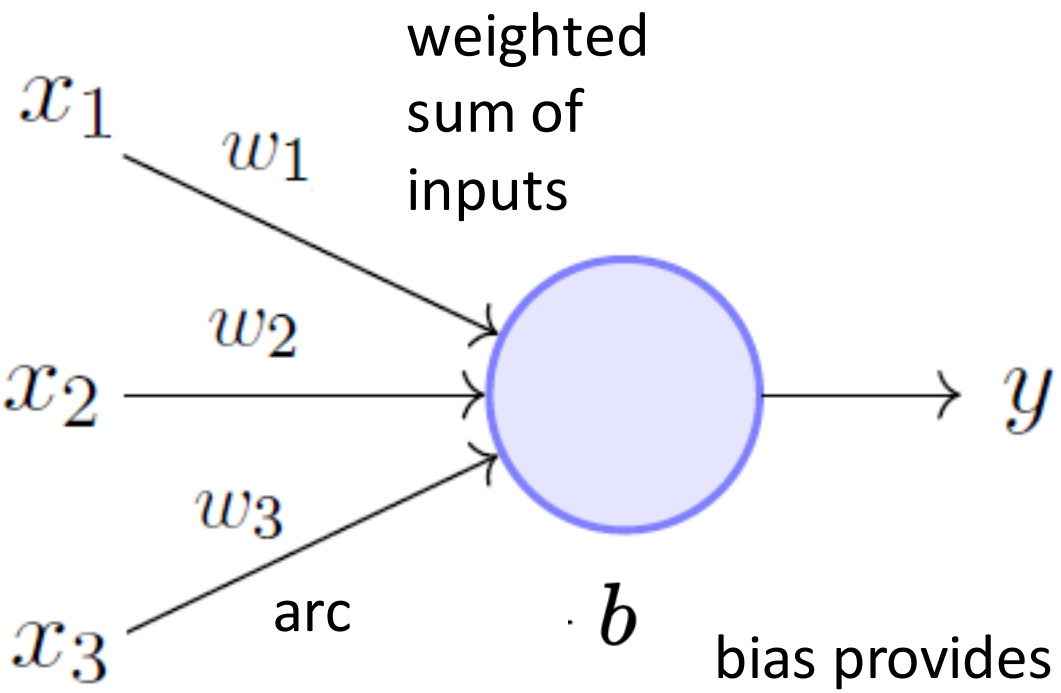
single output:
"all or nothing"



signal processing

multiple inputs

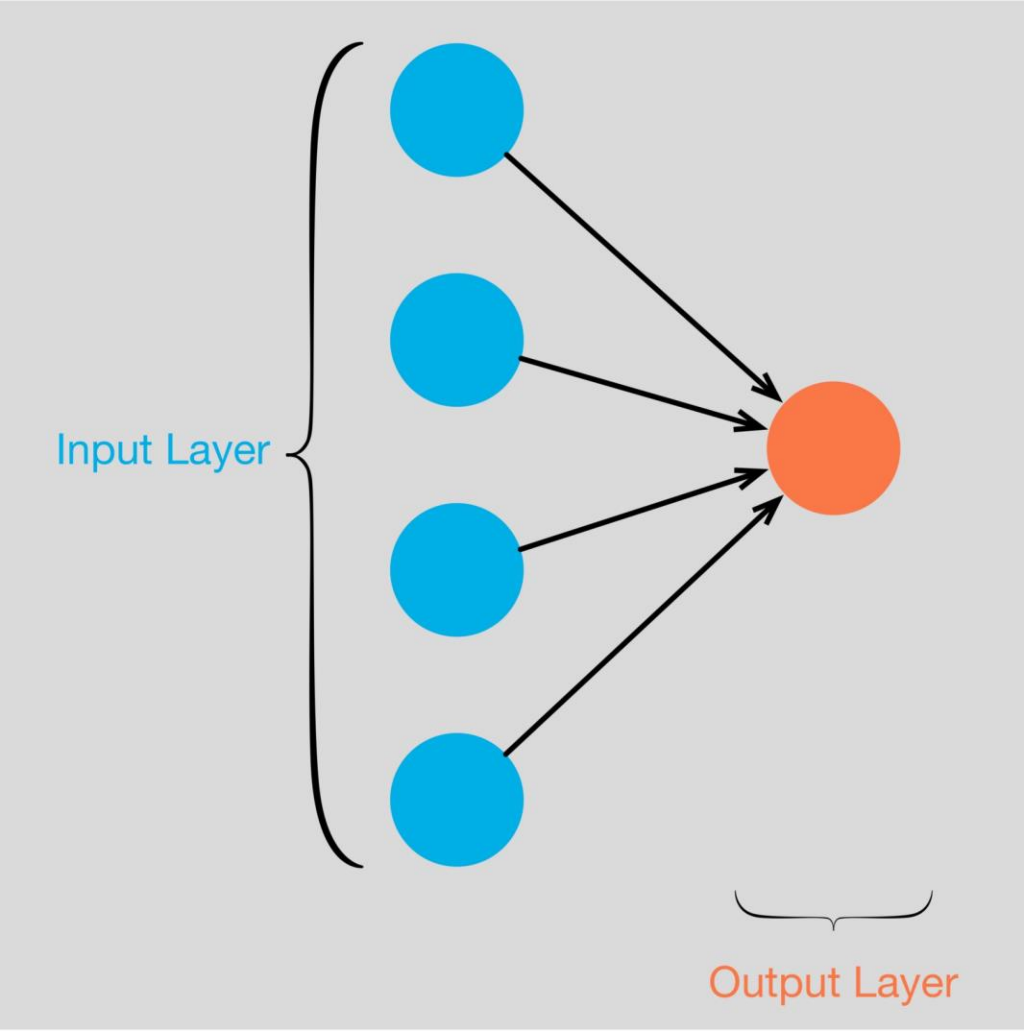
Perceptron (1958)



activation function

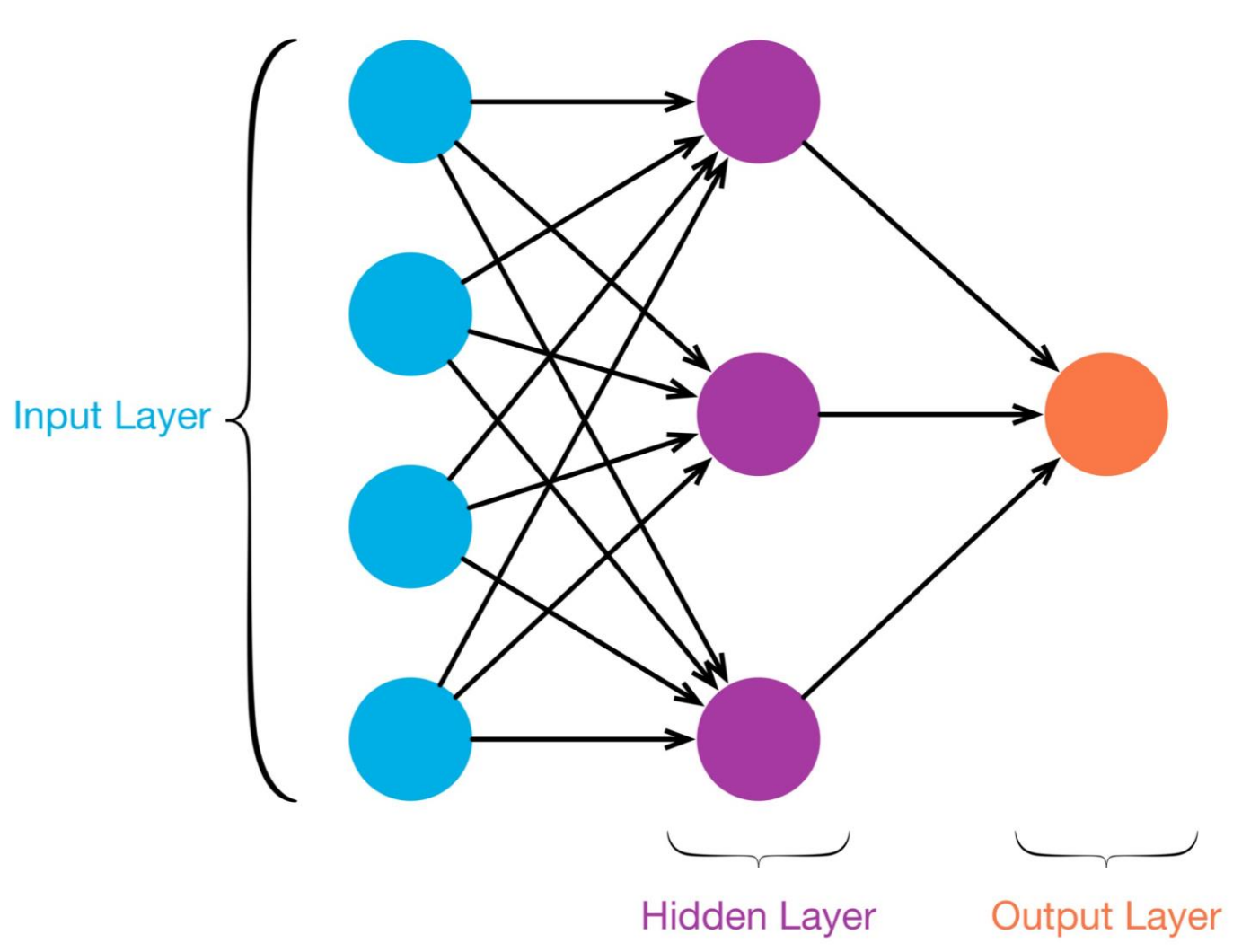
$$y = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

Single-layer network



Multi-layer network

A **feedforward network** contains no directed loops



Multi-layer networks

- Use multiple hidden layers to create more complex models => can solve more complex problems.
- The model will "discover" its own representation of the data in a way that best fits the learning task.
- Neurons in hidden layers can therefore represent complex features without any need to engineer these *a priori*.
- This can be very powerful if enough training data is available.

Fitting a neural network model

- To train our model, we need to find values for the weights, **w**. (We can incorporate the bias *b* as an additional weight w_0).

- The best choice of **w** would minimise an appropriate **loss function** for our predictions compared to the training data labels (e.g. *mean squared error* for a regression task).

- For a network with *m* arcs and *n* neurons, **w** has (*m+n*) components: we will be optimising in a *high-dimensional* parameter space.

Gradient descent

- This is a simple but effective *optimisation method* in high dimensions, which uses the *gradient* (i.e. slope) of the loss function and takes progressive small steps downhill until it finds a minimum.

- **Stochastic gradient descent** includes a small random movement to helps us escape local minima.

Gradient descent example

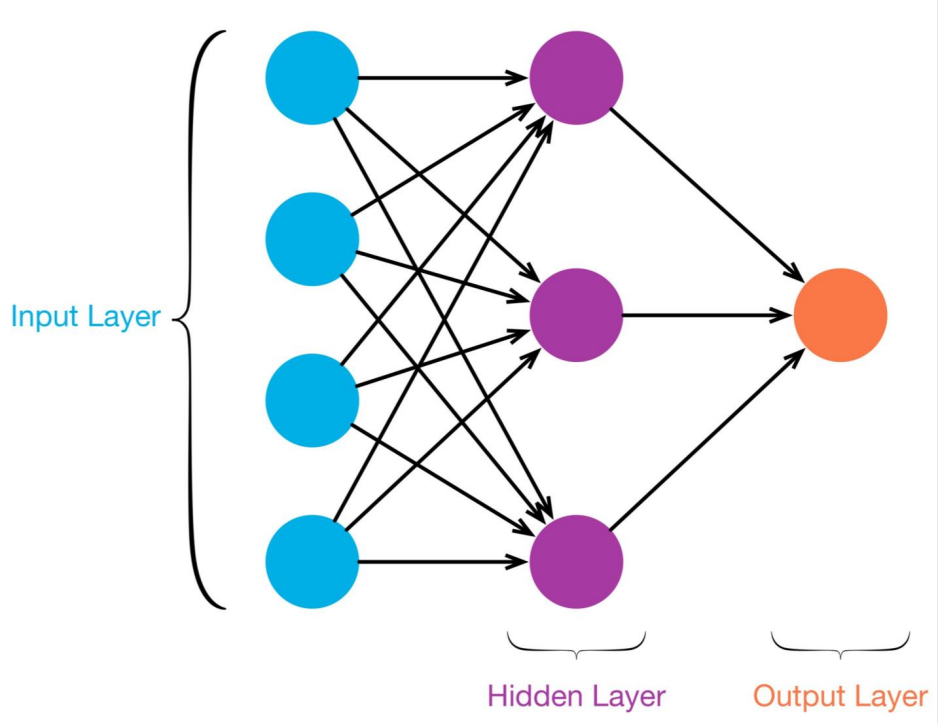
With the **iris** dataset,

Use *gradient descent* to fit a binary *logistic regression* model that can predict **iris versicolor** from only **petal length** and **petal width**.

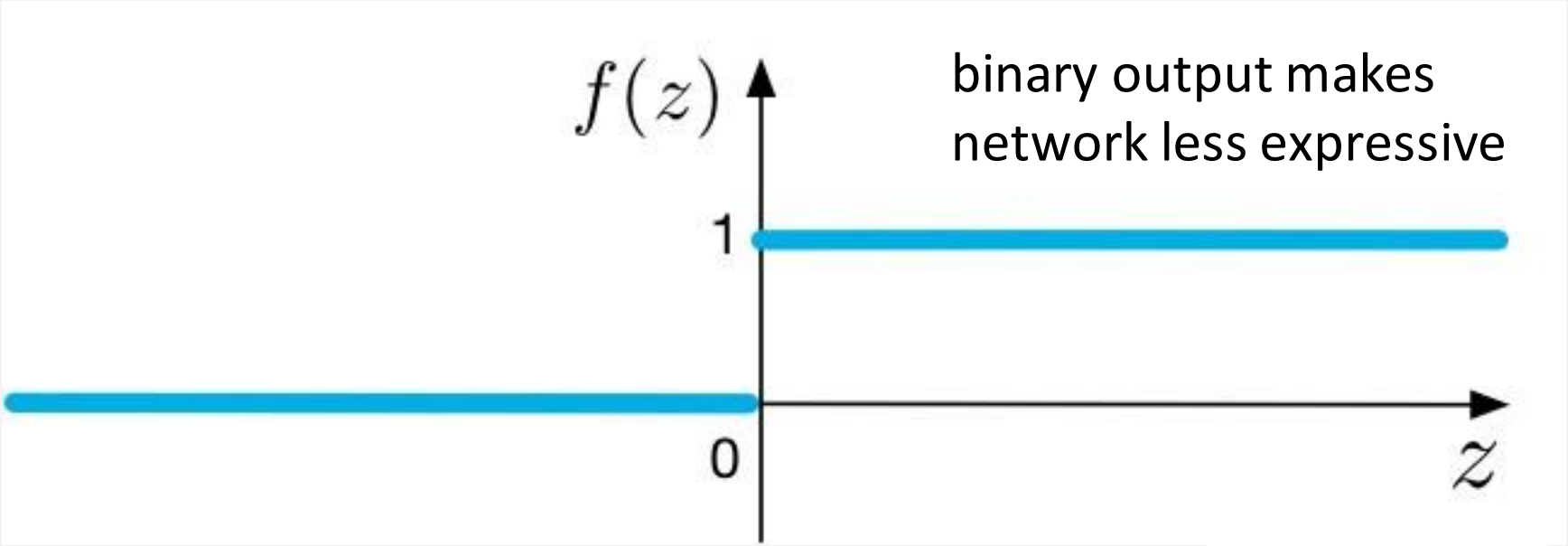
Backpropagation

To calculate the gradient of the loss function,we start from the output layer and work back layer by layer to build up its derivative with respect to each of the weights.

This technique is known as **backpropagation**



Problems with the activation function



binary output makes
network less expressive

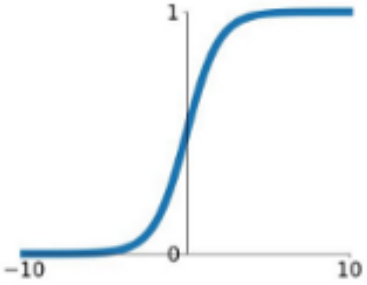
$$z = \mathbf{w} \cdot \mathbf{x} + b$$

zero gradient everywhere (except z=0)
means backpropagation won't work!

Solution: continuous activation functions

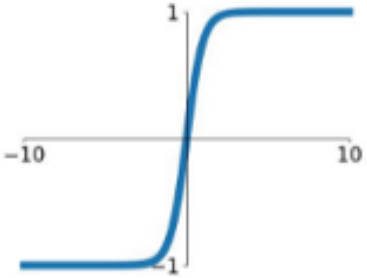
Sigmoid

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



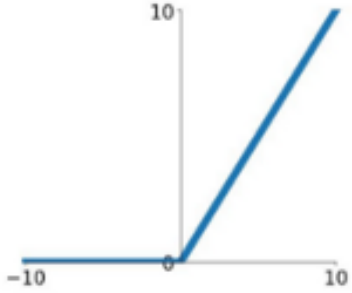
tanh

$$\tanh(x)$$



ReLU

$$\max(0, x)$$



These continuous activation functions (and many others) are used in multi-layer neural networks.

Their different shapes result in different learning characteristics, suitable for different tasks.

Multi-layer network exercise

With the **wine quality - red** dataset,

Use a *neural network* to predict **quality** from the other features.

Explore how performance is affected by adding additional hidden layers to the network.

Deep learning

Deep networks

"Deep learning" concerns neural network models with many hidden layers, used in both unsupervised and supervised learning contexts.

It is difficult to train deep models effectively, so special techniques have been developed for this class of models.

Specialised software packages (e.g. TensorFlow, Keras) and computer hardware are available.

Unstructured data

Deep networks are especially well suited to working with unstructured input data, which doesn't have easily definable informative features.

Images

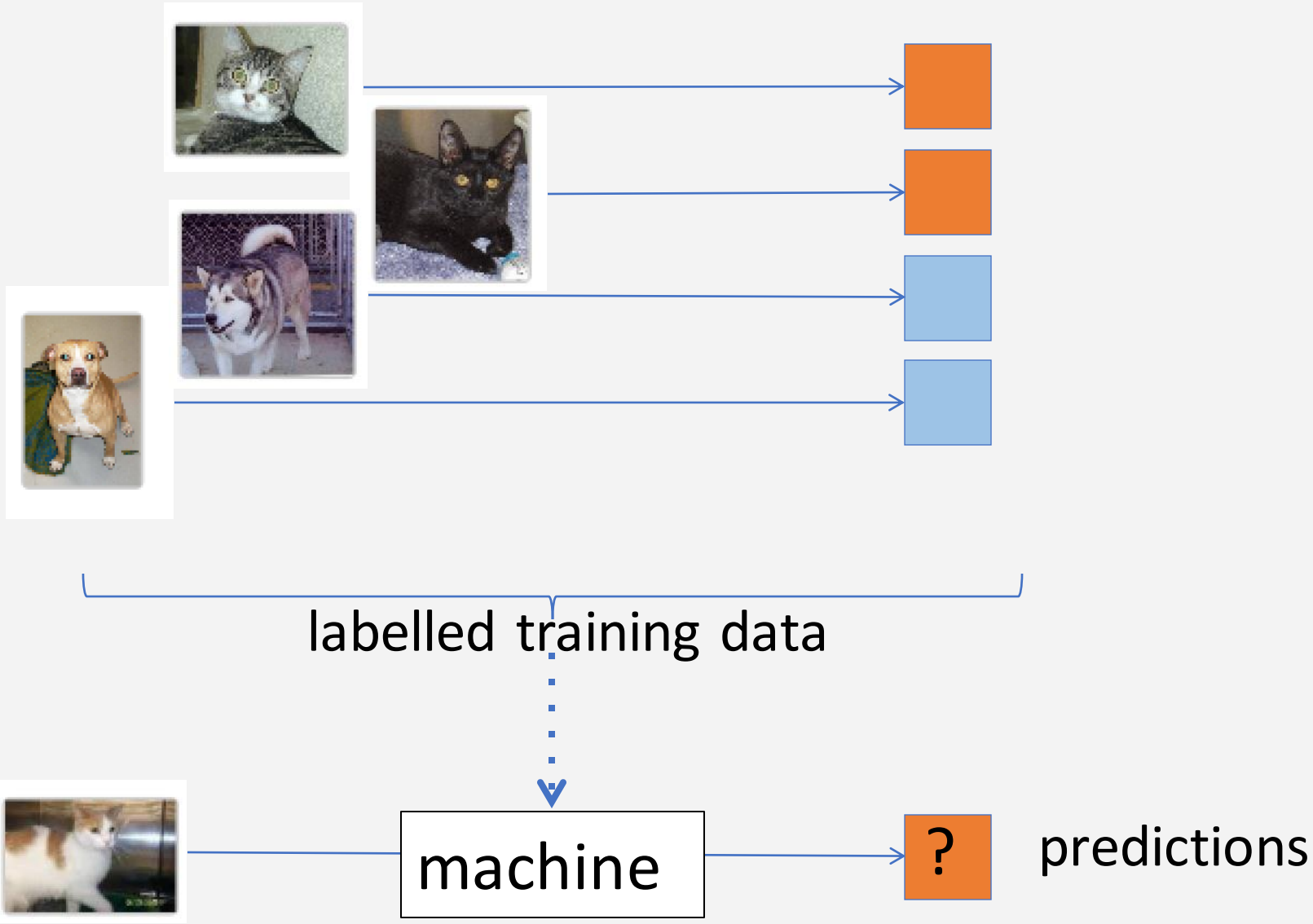
Text

Speech

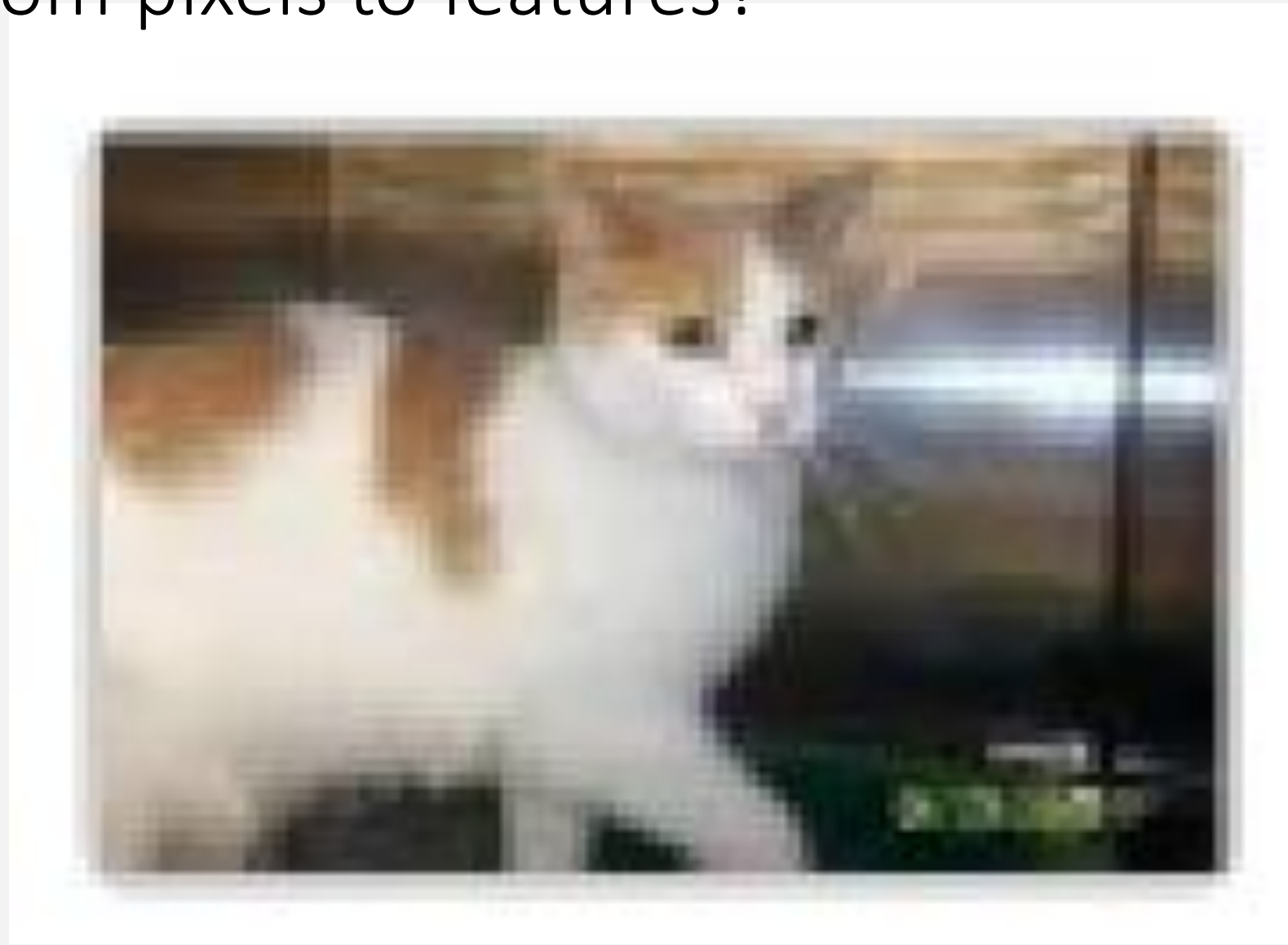
Music

…

Image classification task



From pixels to features?

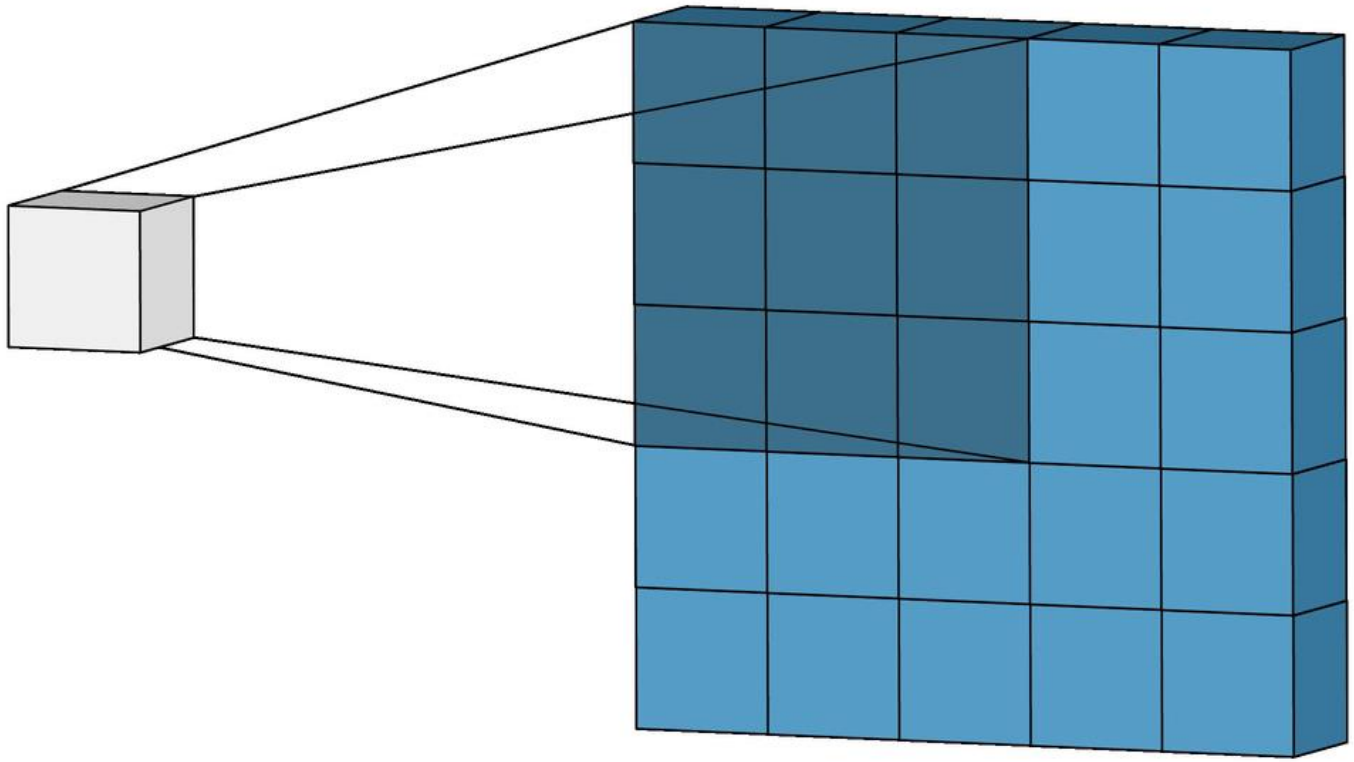


Convolution

- Convolution is a way to transform unstructured data into network inputs in a way that *preserves the relevant relationships* in time and/or space.
- A network architecture that uses this technique is known as a Convolutional Neural Network (CNN).
- In practice, convolutional layers may form part of a larger architecture.

Convolution

The **kernel** is a matrix of weights that passes over the image

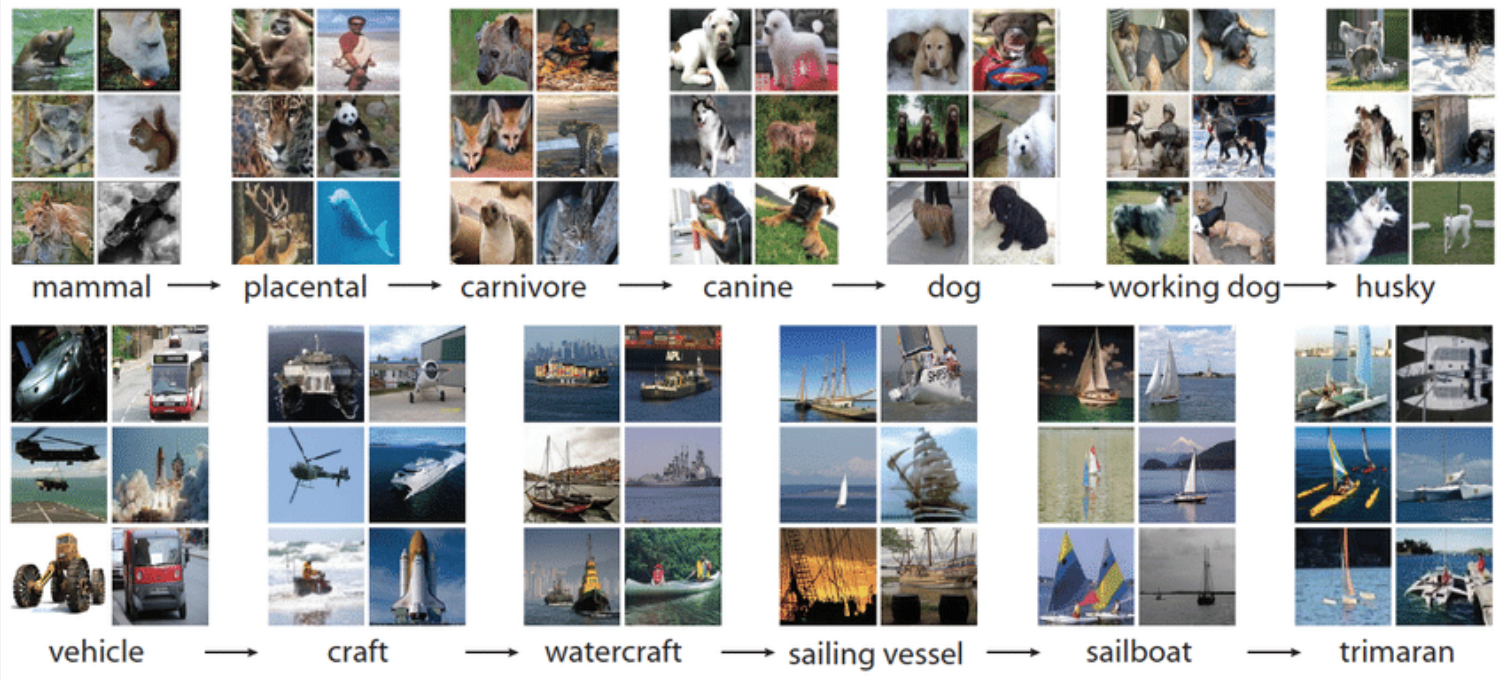


next layer

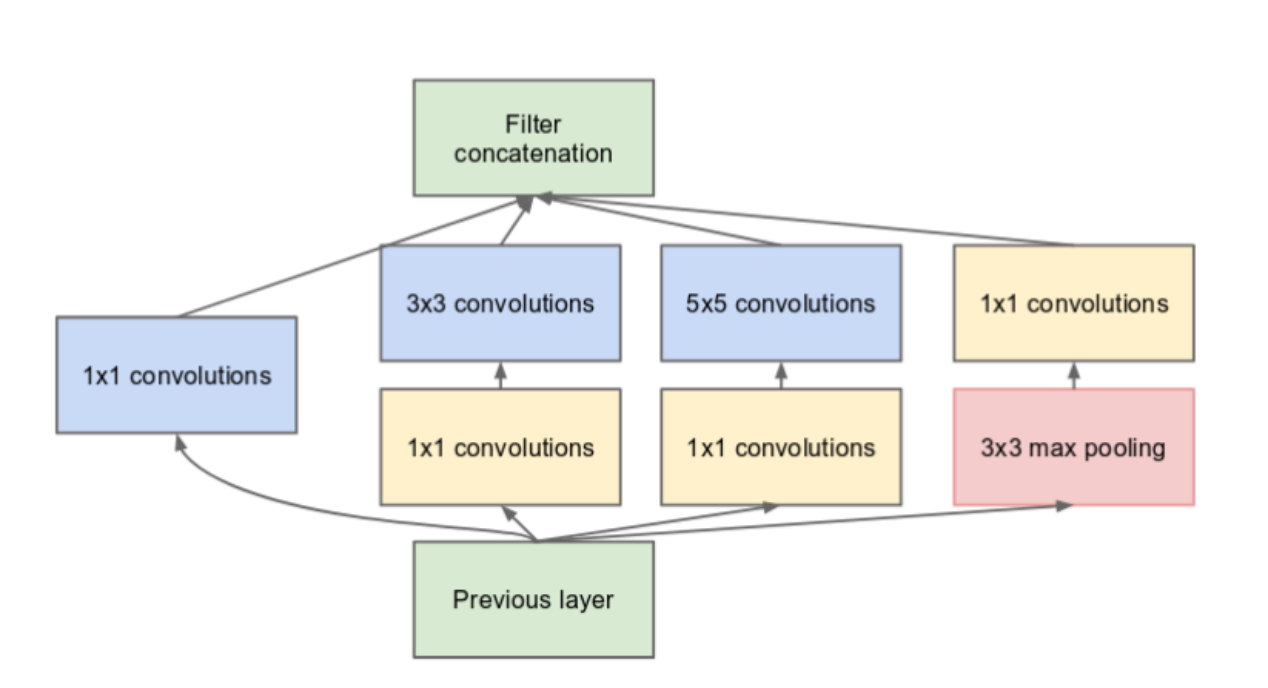
input layer

Google Inception (2015)

An image classification model, trained on the *ImageNet* dataset.

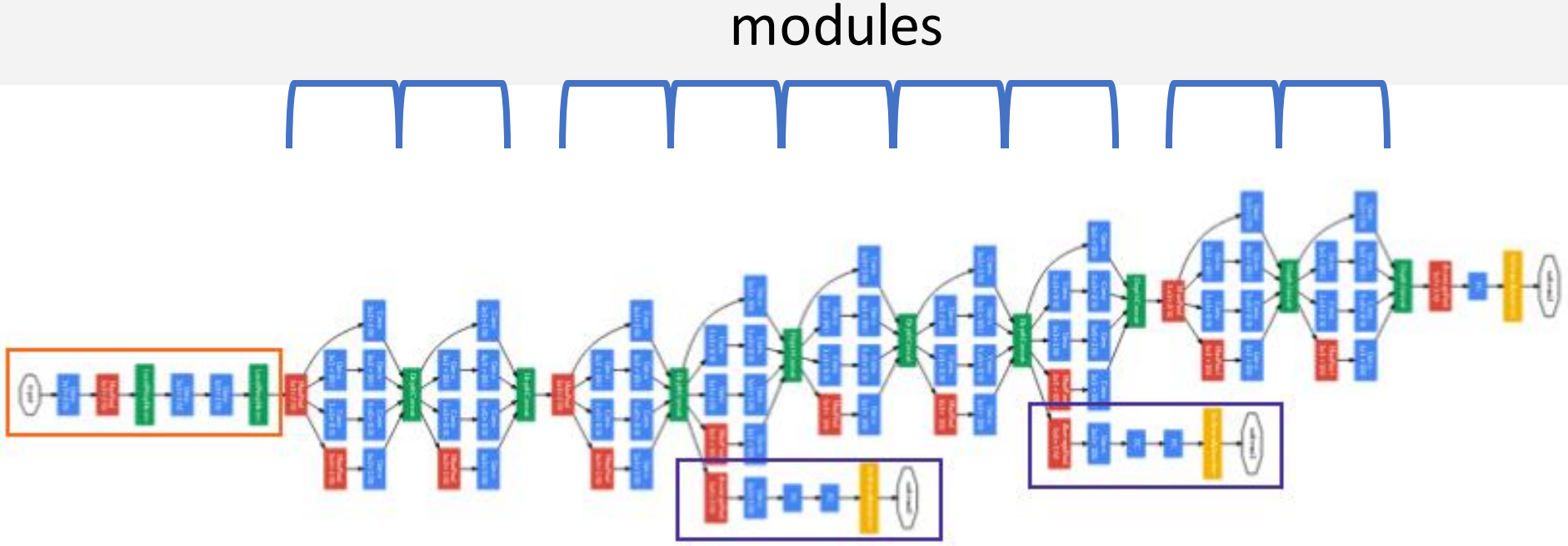


Google Inception module



- Architecture is carefully designed to capture salient features and ensure efficient training.
- Uses kernels of different widths to detect both local and global features

Google Inception model



The overall model for image classification is built from 9 of these modules. It is 27 layers deep in total.

The authors suggest it would take about a week to train on a few high-end GPUs.

Embedding

Once the weights of the Inception model have been trained, we can use the model to classify any image into the ImageNet categories.

The hidden layers of the model must be capturing some highly informative features. We can make use of these features in our own models.

When an image is used as input, the *activations* of the penultimate layer are extracted as features. This is called **embedding**.

Image embedding example

Using *image* embedding, can we train a neural network to distinguish cats from dogs?

Some special architectures

- **Autoencoder**

  learns a compact representation of the data in an unsupervised way (e.g. for dimensionality reduction).
- **Recurrent Neural Network (RNN)**

  represents processes evolving over time.
- **Generative Adversarial Network (GAN)**

  two networks (a *generator* and a *discriminator*) compete against each other. This is often used in **reinforcement learning**.

Summary of Part 3

Artificial neural networks are a diverse family of biologically inspired learning methods.

Neurons in a network receive multiple inputs and combine these as a **weighted sum**. The neuron's output is determined by its **activation function**.

To fit the model, weights need to be *optimised*
e.g. by using **gradient descent.**

Deep learning implies any architecture with multiple hidden layers that can discover relevant features from the data itself.

Convolution layers may be used to work with unstructured inputs such as images or speech.

Where next…?

Müller AC & Guido S, *Introduction to Machine Learning with Python*
http://ebookcentral.proquest.com/lib/imperial/detail.action?docID=4698164

Burger SV, *Introduction to Machine Learning With R*
https://www.oreilly.com/library/view/introduction-to-machine/9781491976432/

Nielsen M, *Neural Networks and Deep Learning*
http://neuralnetworksanddeeplearning.com/index.html

Tutorials and examples at
https://scikit-learn.org/
https://www.kaggle.com