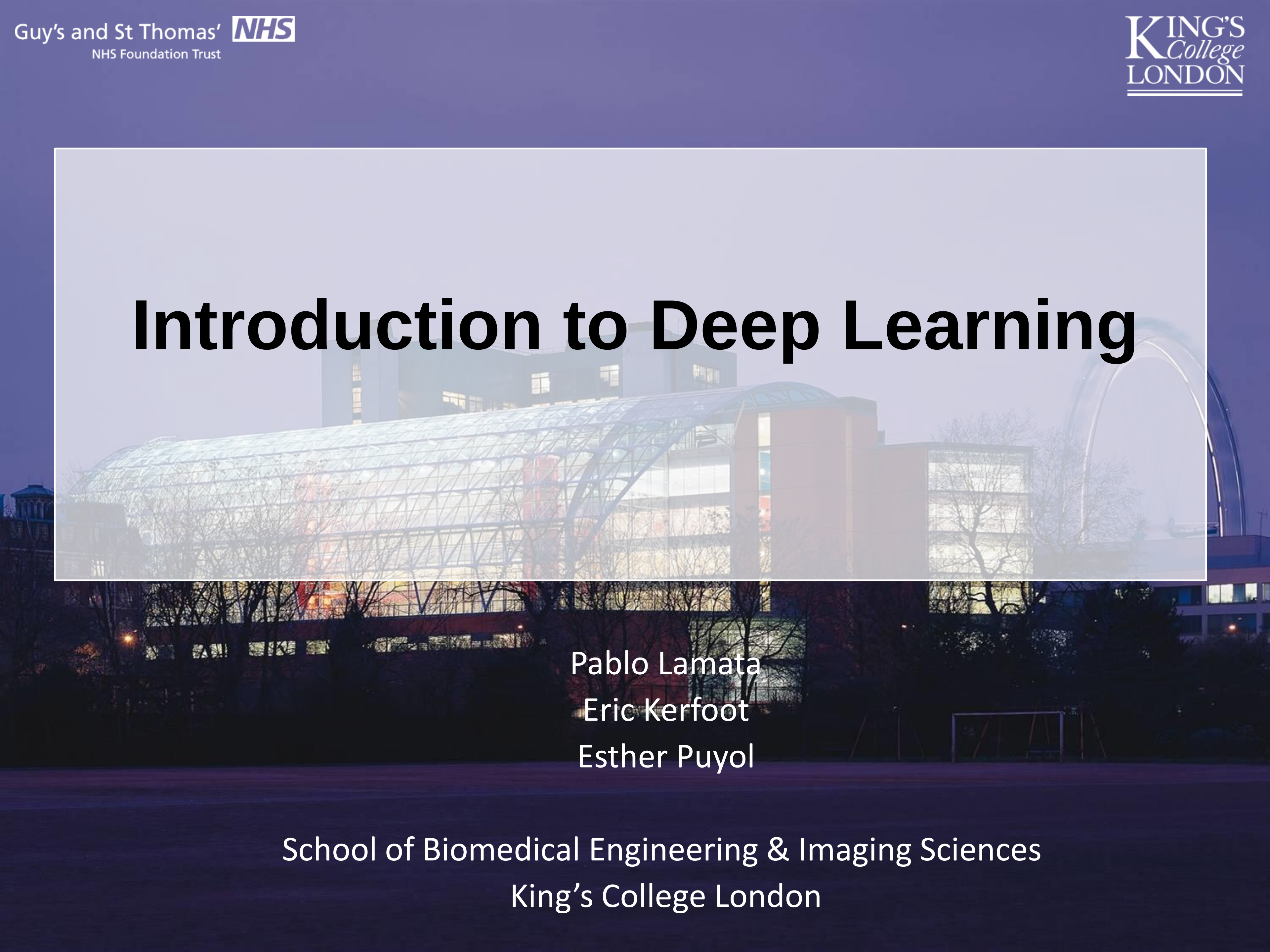
Guy's and St Thomas' NHS
NHS Foundation Trust

KING'S College LONDON



Introduction to Deep Learning

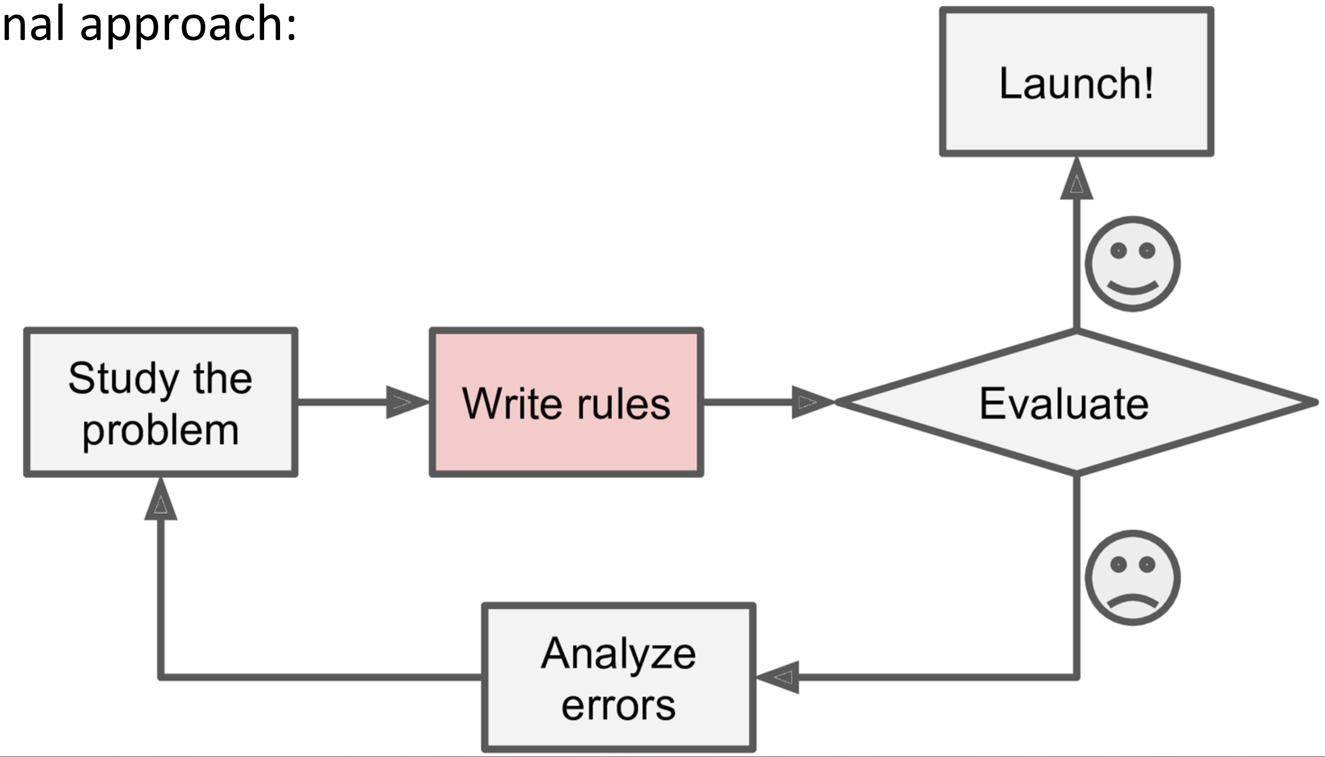
Pablo Lamata
Eric Kerfoot
Esther Puyol

School of Biomedical Engineering & Imaging Sciences
King's College London

What is AI?

The science (and art) of programming computers so they can *"learn from and make predictions on data"*

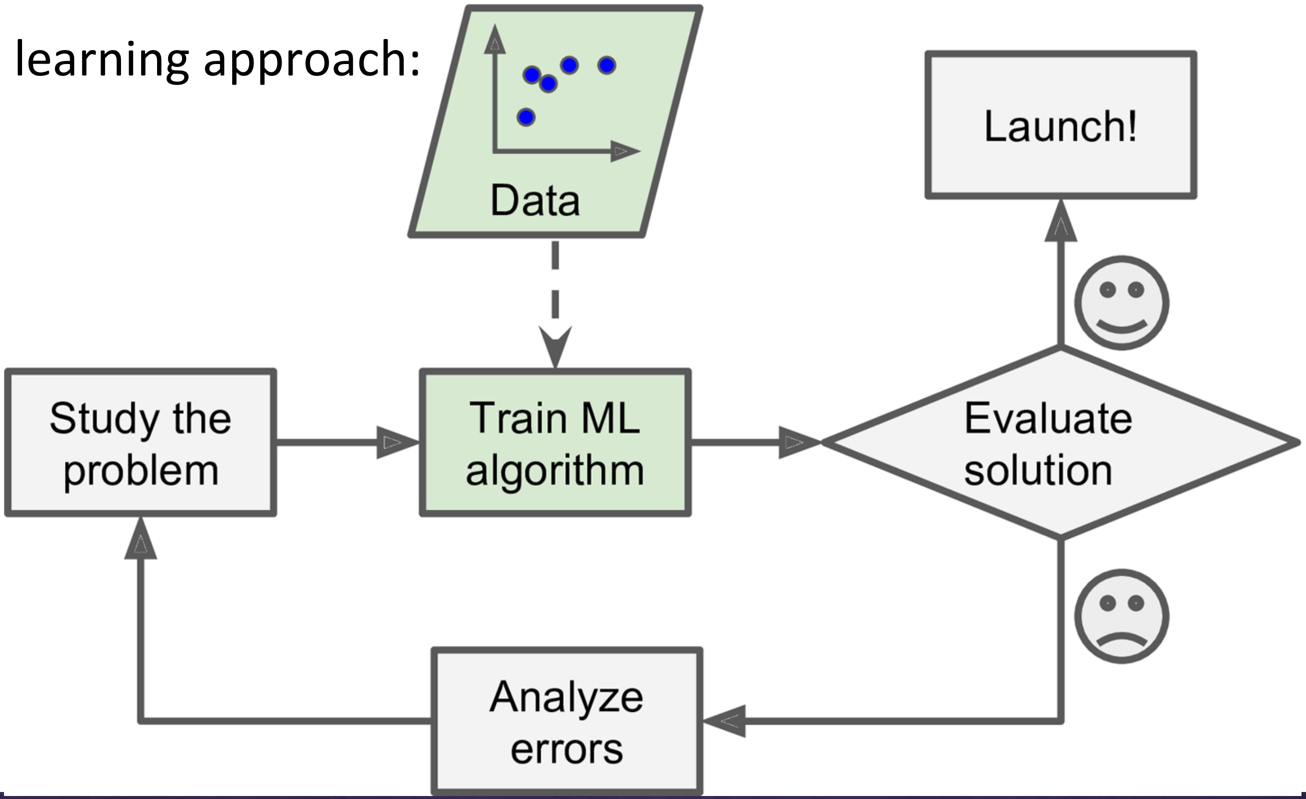
Traditional approach:



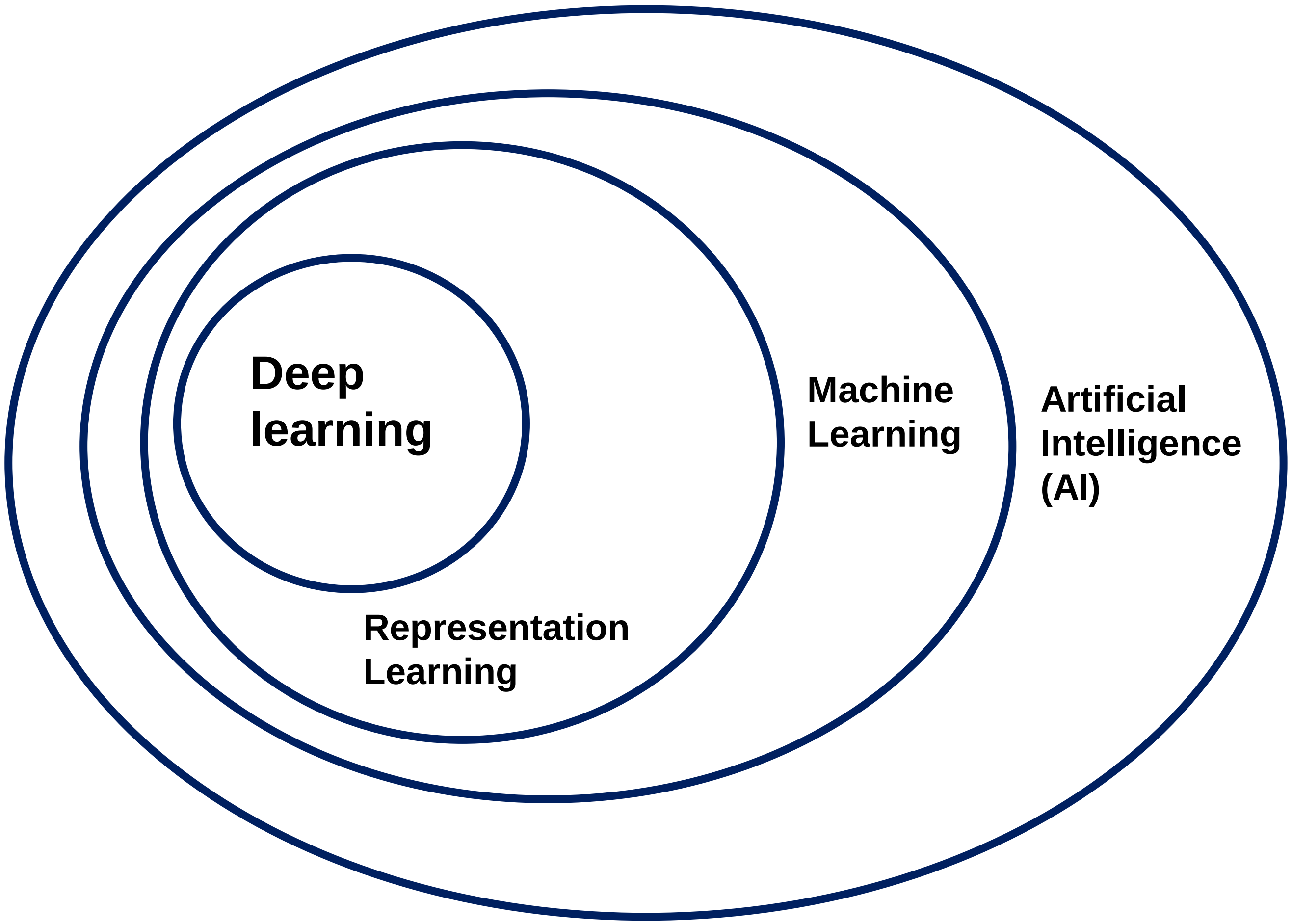
What is machine learning?

The science (and art) of programming computers so they can *"learn from and make predictions on data"*

Machine learning approach:



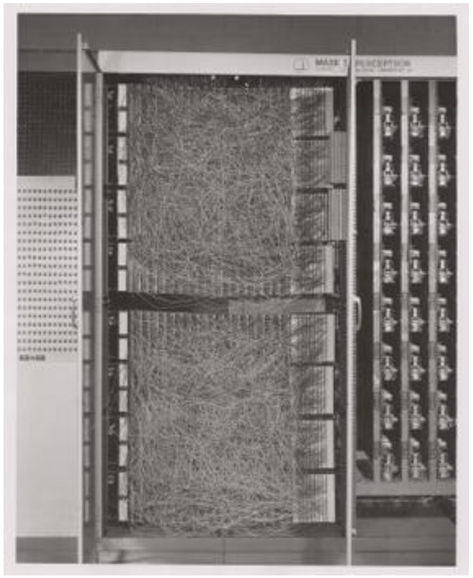
AI, Machine Learning and Deep Learning:



Adapted from: Ian Goodfellow, Yoshua Bengio and Aaron Courville. “Deep Learning” deeplearningbook.org 2

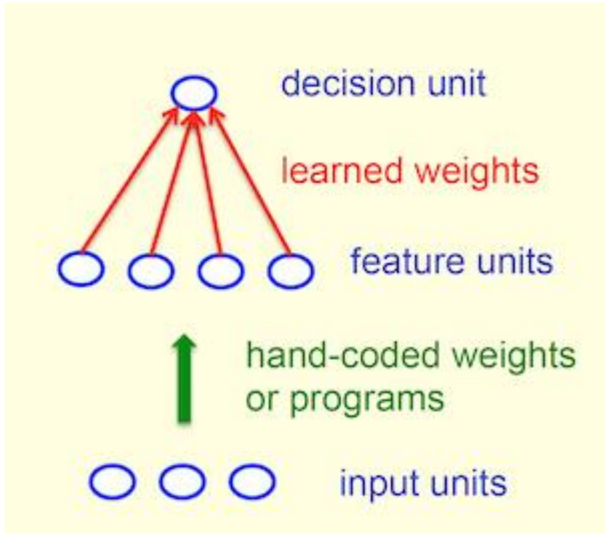
Perceptron

The first neural network (Frank Rosenblatt, 1957)



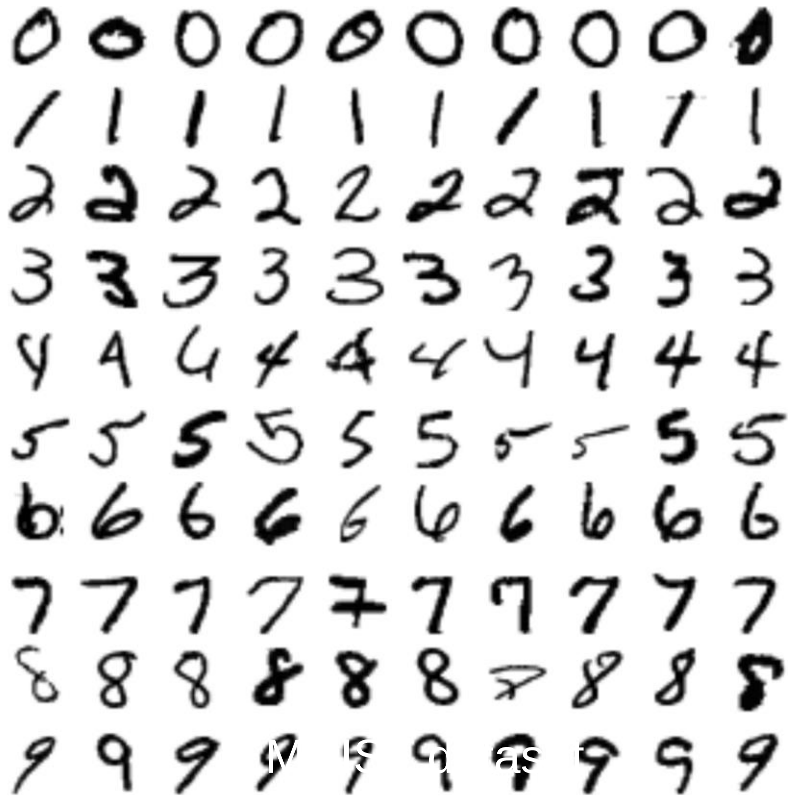
"Mark 1 perceptron“ - machine designed for image recognition: it had an array of 400 photocells, randomly connected to the "neurons". Weights were encoded in potentiometers, and weight updates during learning were performed by electric motors

$$f(x) = \begin{cases} 1 & if \sum_i w_i . x_i > b \\ 0 & otherwise \end{cases}$$



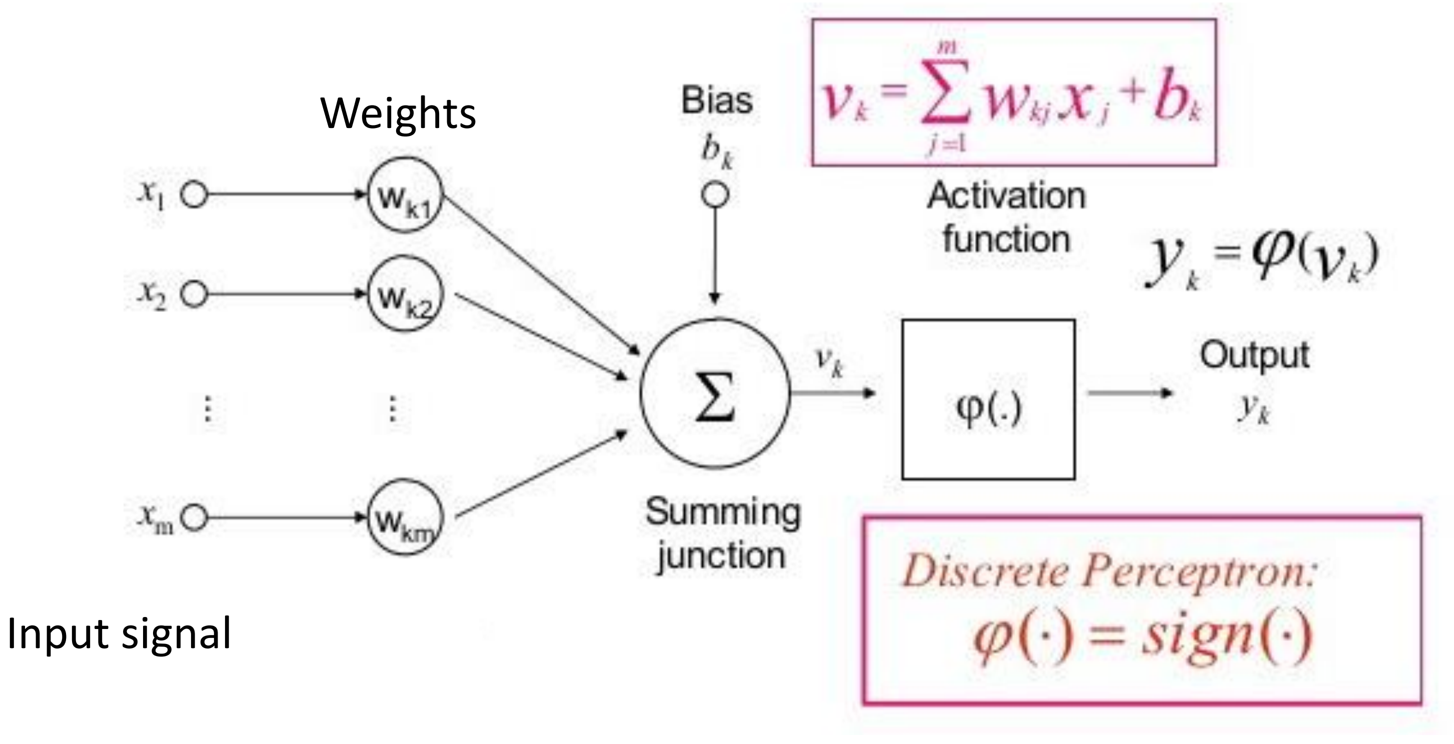
Application of AI

- Optical character recognition
- Spam filtering
- Speech recognition
- Face recognition
- Object recognition
- Robotics
- Recommending products based on what customers bought before
- Detect fraudulent transactions



A single-layer perceptron

A single-layer perceptron looks as follows:



*This and the following slides follow the example on https://hackernoon.com/a-hands-on-introduction-to-neural-networks-6a03afb468b1

A single-layer perceptron

Input:

- Each input to the neuron ($x_1, x_2, \ldots x_n$) is known as a **feature**

Weights:

- Each feature is weighted with a number to represent the strength of that input ($w_{k1}, w_{k2}, \ldots w_{km}$).

Activation function:

- Calculate weighted sum of inputs (v_k) , pass through an **activation function** and threshold result y_k to 0 or 1

A single-layer perceptron can be trained as follows:

1. Ask the neuron* to classify a sample (**forward pass**)
2. Update the neuron's weights based on how wrong the prediction is.
3. Repeat for a set number of times (=**epochs**).

Activation function

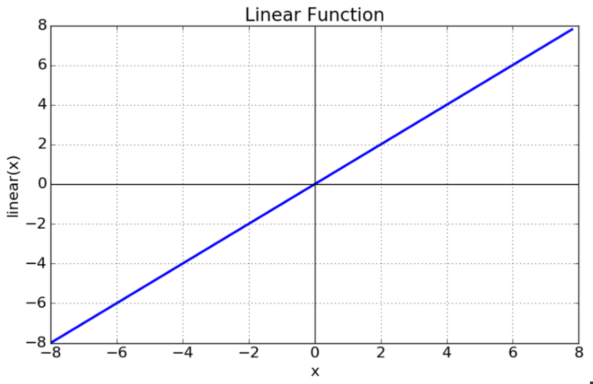
Added to the output end of any neural network

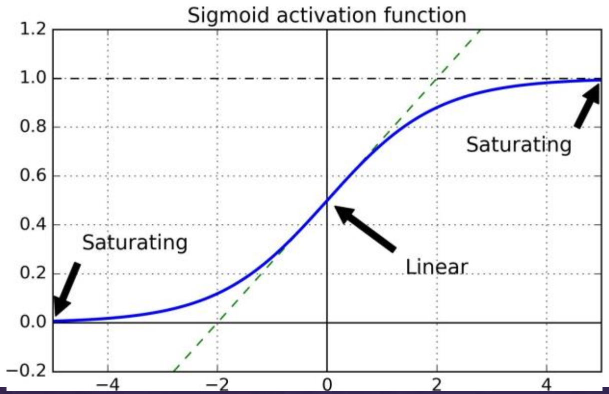
- Can be regarded as a **Transfer Function**
- Can also be attached in between two Neural Networks

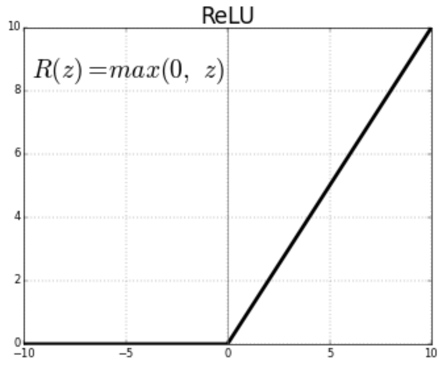
Used to determine whether the output of a neural network is ‘yes’ or ‘no’ (or something in between).

- Maps the resulting values in between 0 to 1 or -1 to 1 (depending upon the activation function)

We distinguish between (piecewise) linear and nonlinear activation functions







Training a perceptron

What else do we need?

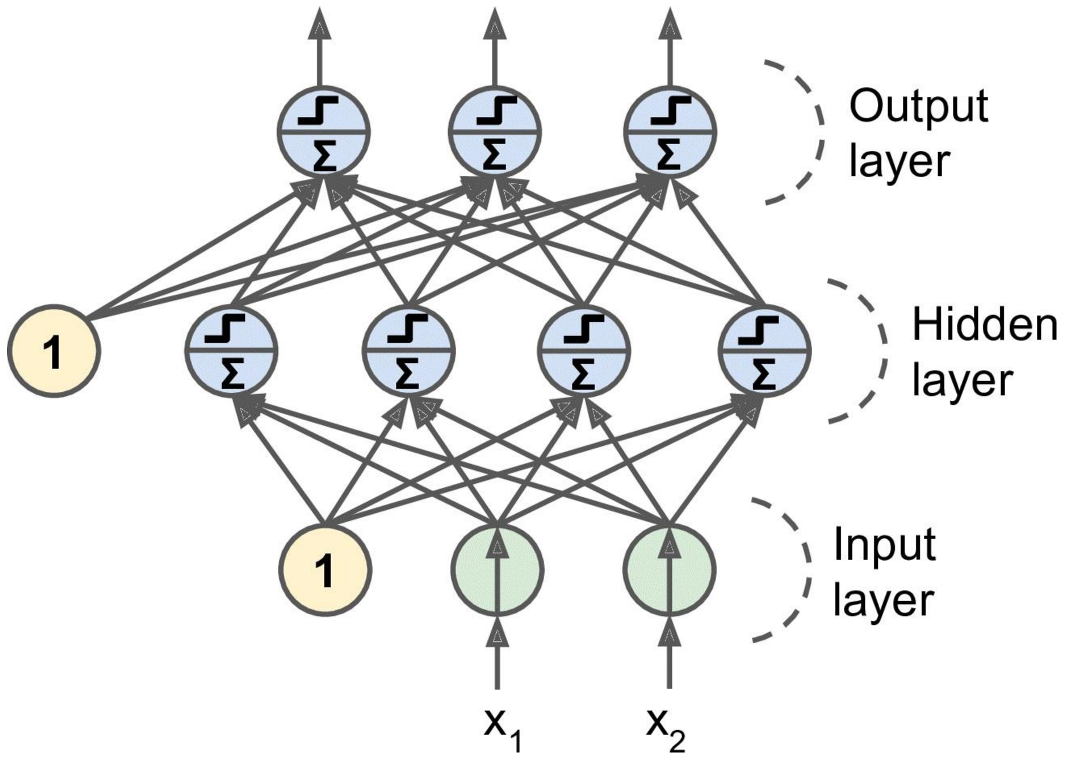
- We need **an error function (cost function** or **loss function):**
  - The cost function must be able to be written as an average over cost functions Ei for individual training examples xi:

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i$$

  - This allows us to compute the gradient (with respect to weights and biases) for a single training example, and run e.g. Gradient Descent
  - Examples of loss function: L1 norm → $E = \sum_i |E_i|$
  - We need an **optimisation method** (e.g. gradient descent)
  - Our activation function should be **differentiable**

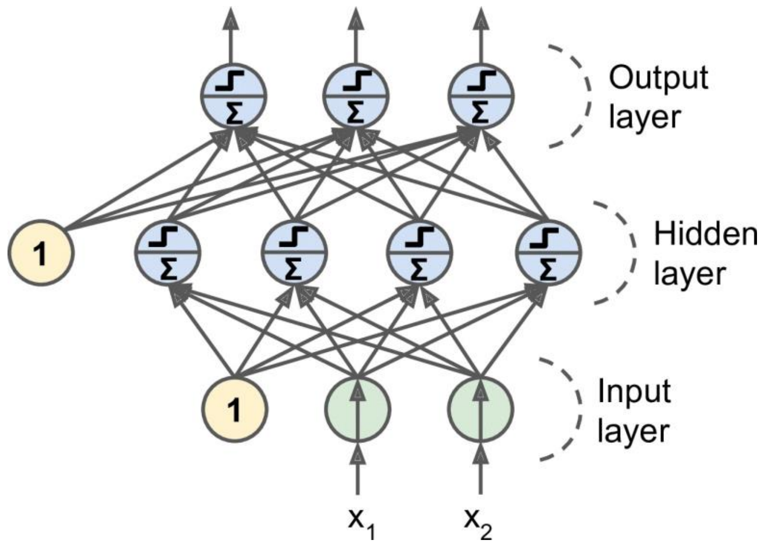
From single-layer to multi-layer perceptron

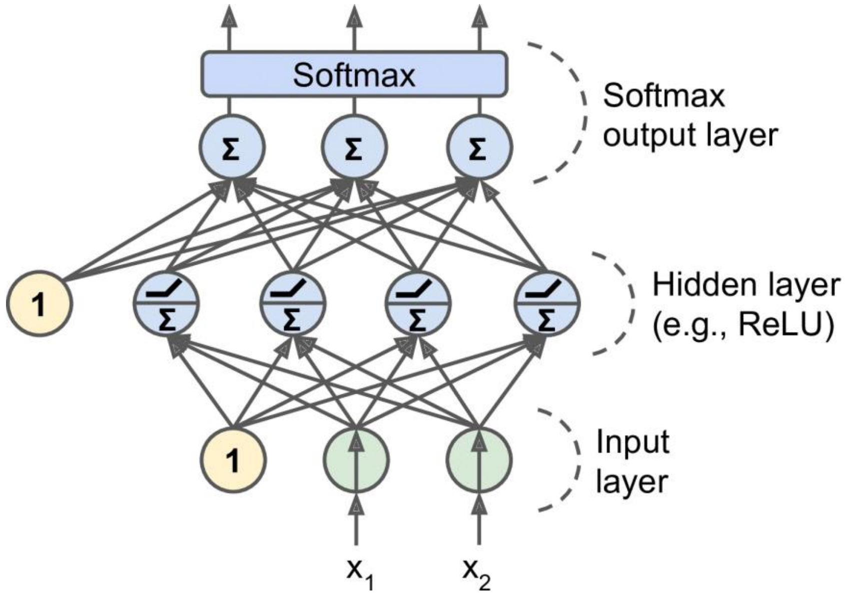
A more complex MLP is shown below (still only 1 hidden layer):



Multi-layer perceptron

We have stacked multiple perceptrons to generate hidden layers:

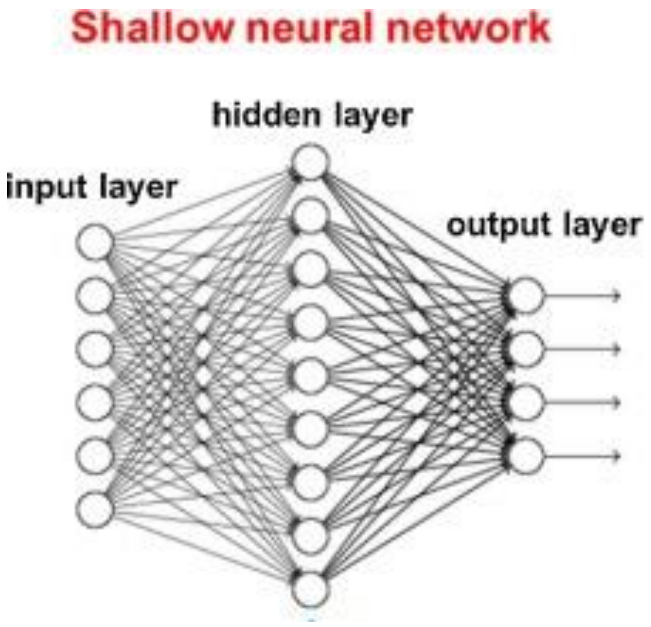


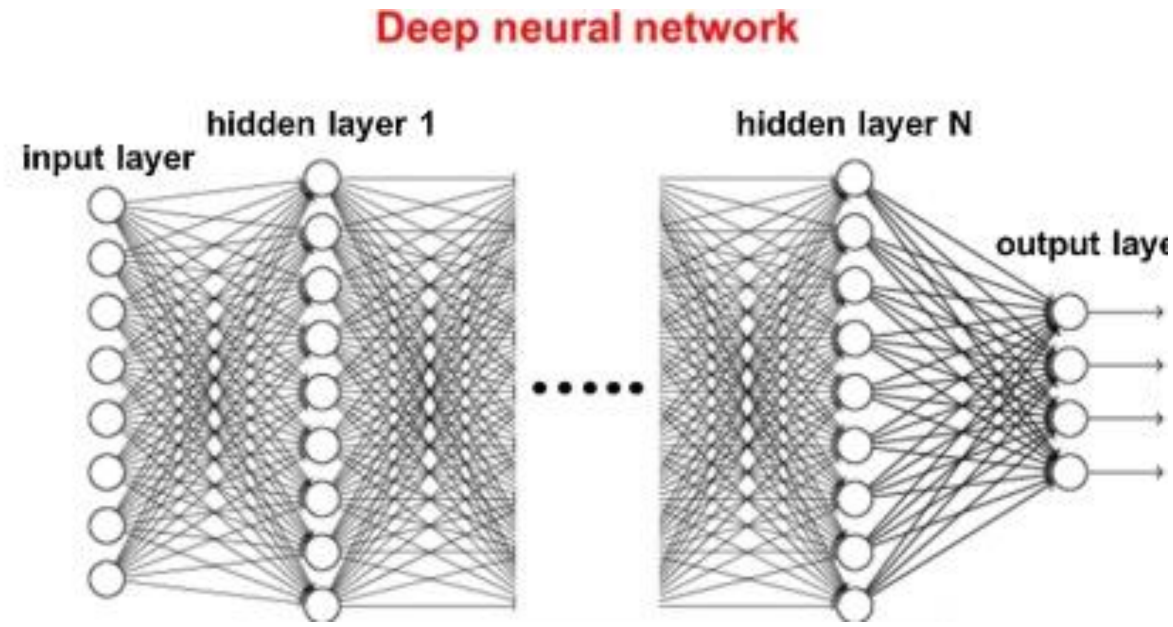


If we have more than one hidden layer, the neural network is considered to be "deep" and we move into **deep learning**

From shallow to deep fully connected networks

Compare this to a deep fully-connected network with N hidden layers:

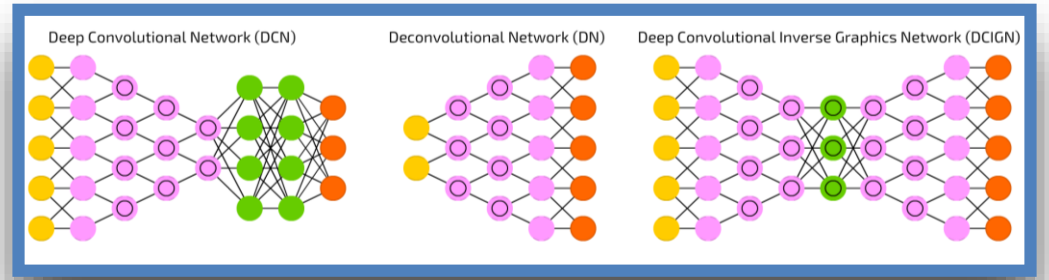




6x9=54 connections to first layer
9x4=36 connections to output layer
54+36=90 connections in total

8x9=72 from input layer, 9x4=36 to output layer
9x9 connections between each hidden layer, ie $(9x9)^{(N-2)}$
8x9 + (N-2)x81 + 9x4
E.g. **108 (N=2), 189 (N=3), 270 (N=3)**

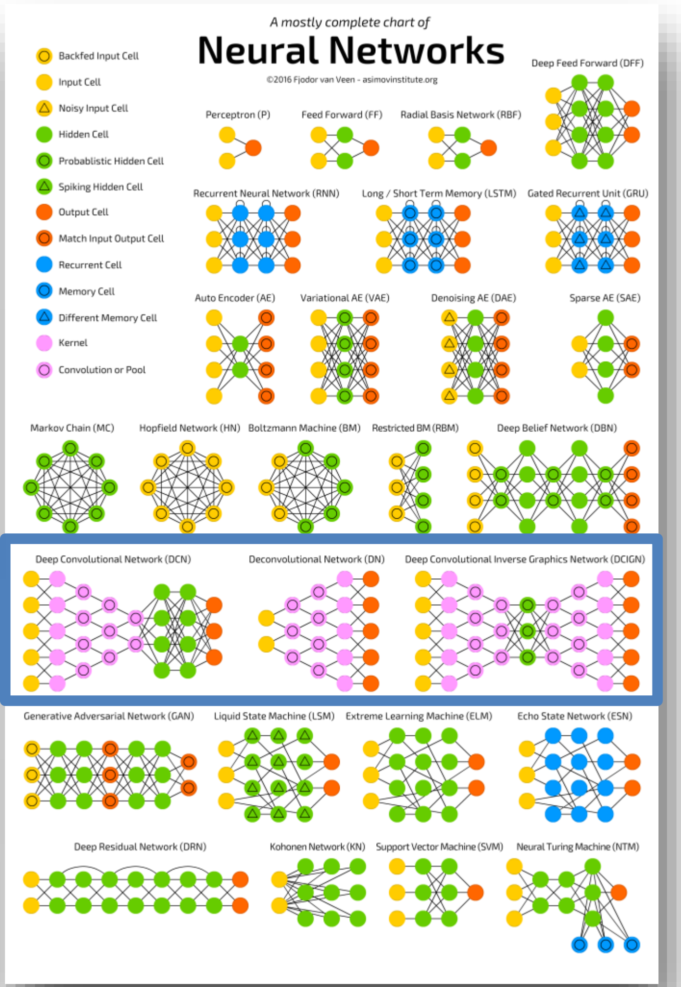
Convolutional neural network (CNN)



Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)

Convolutional neural networks

- ● Input Cell
- ● Hidden Cell
- ◉ Probablistic Hidden Cell
- ● Output Cell
- ● Kernel
- ○ Convolution or Pool



A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Src: towardsdatascience.com

Convolutional Neural Networks (CNNs)

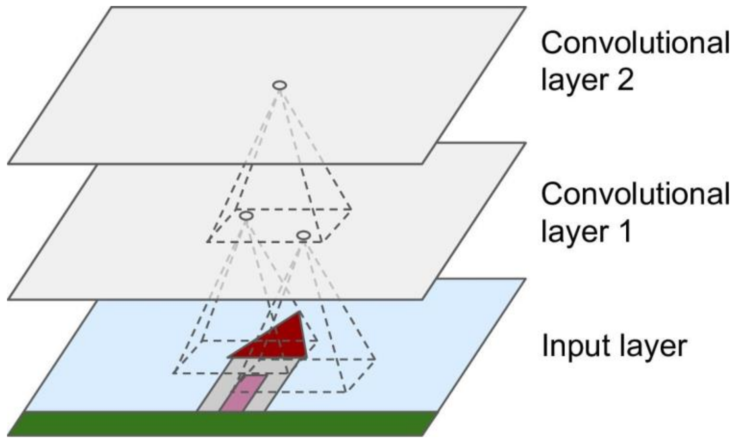
CNNs have several important **building blocks**:

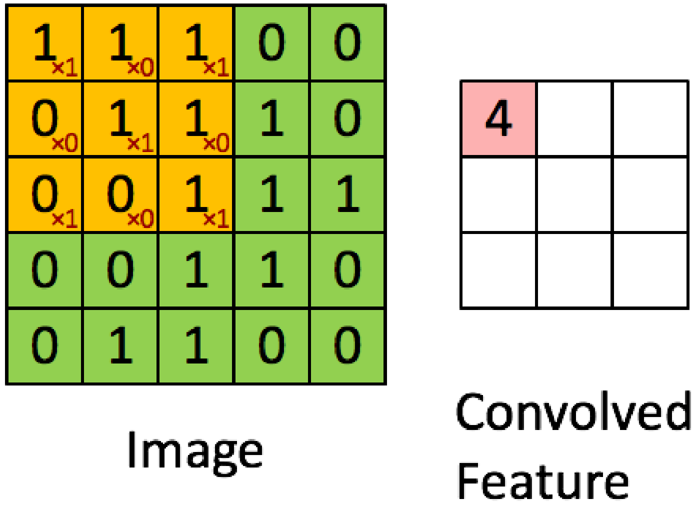
1. **2D (or 3D) input layer**
2. **Convolutional layer**
   - Neurons in the first convolutional layer are not connected to every single pixel, **but only to pixels in their receptive fields**
   - Neurons in the second convolutional layer are only connected to neurons within a small rectangular region in the first layer
3. **Pooling layer**
   - Goal is to **subsample** the input image to reduce computational load, memory usage, numbers of parameters (limits risk of overfitting)
4. **Fully-connected output layer**
   - This is a **regular feed-forward network** which produces final output prediction
   - E.g. **softmax layer** that outputs estimated class probabilities

Convolutional layer

Convolutional layers allow the network to:

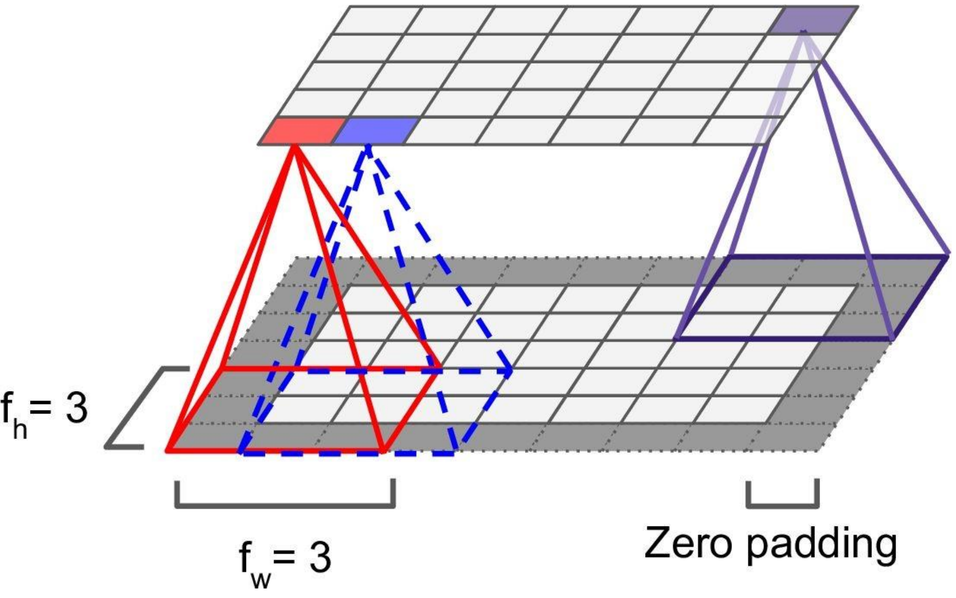
- Concentrate on low-level features in the first hidden layer
- Assemble them to higher-level features in the next hidden layer



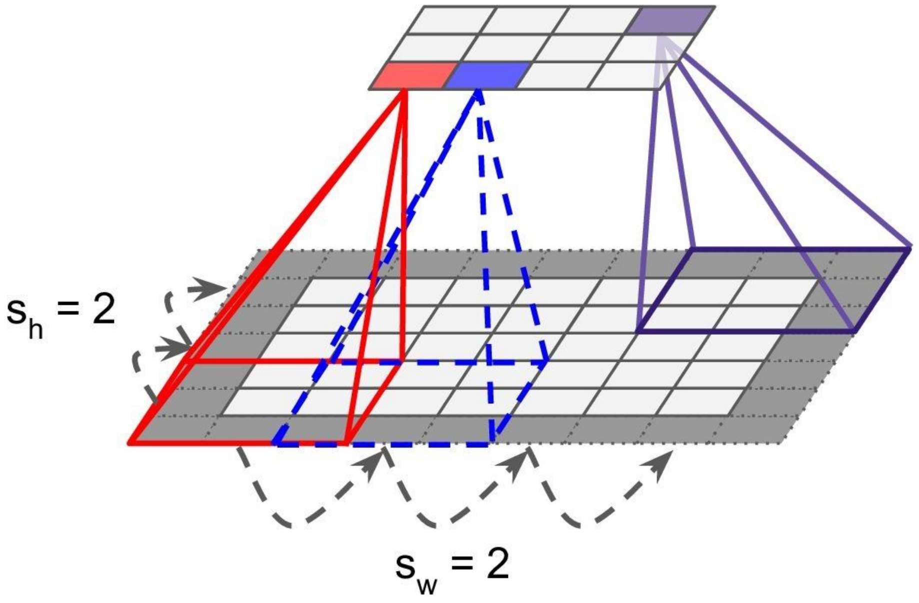


Convolutional layer

Need to apply **zero-padding** at each layer and also apply a stride for further **dimensionality reduction:**



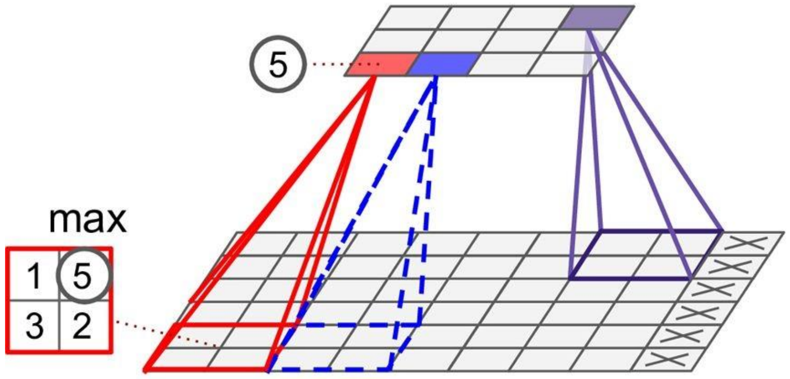
Connections between layers and zero padding

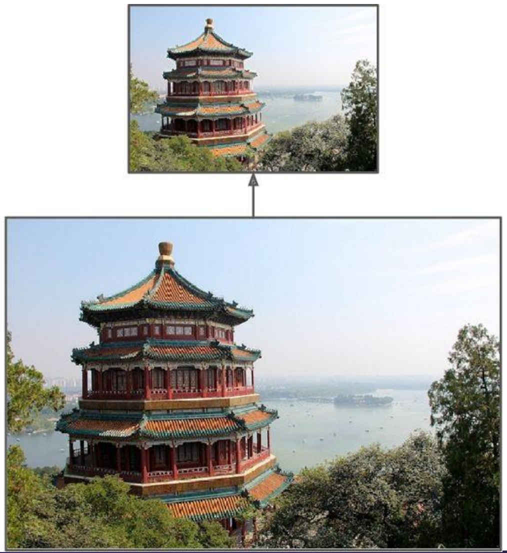


Reducing dimensionality using a stride

Pooling layer

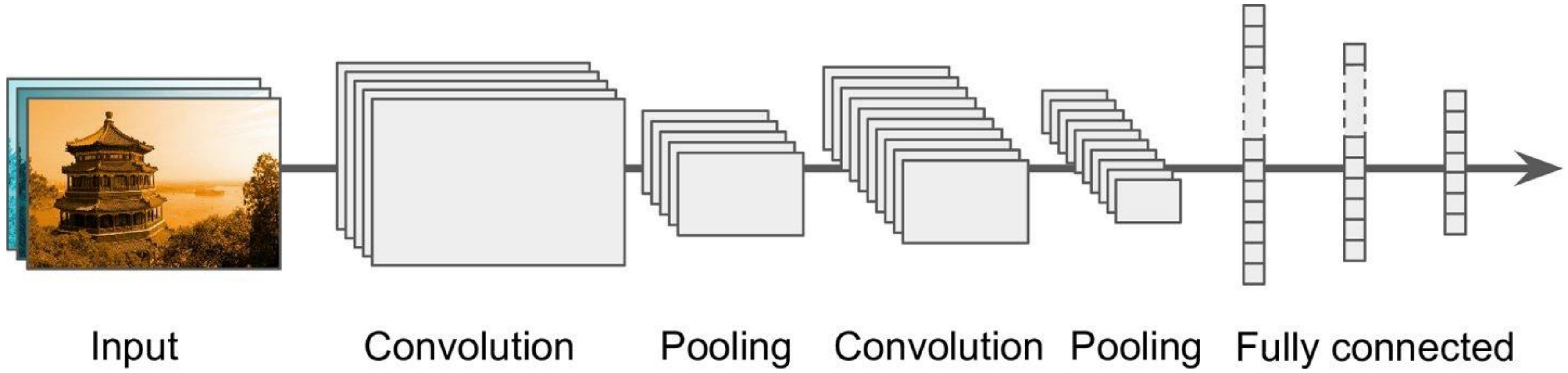
- Goal is to **subsample** the input image to reduce computational load, memory usage, numbers of parameters (limits risk of overfitting)
- Each neuron in pooling layer is connected to the outputs of limited number of neurons in previous layer, again located within a small rectangular receptive field
- Most common type is **max pooling**





Convolutional Neural Networks (CNNs)

Putting it all together:

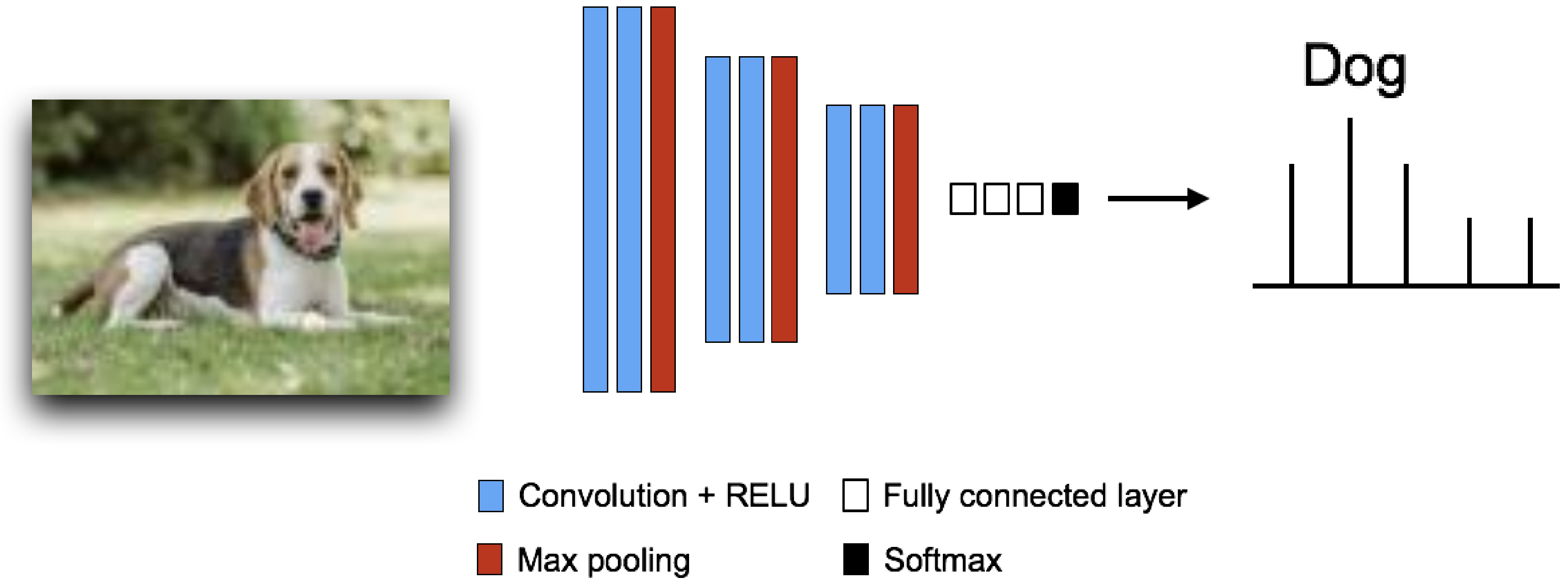


In this case, 3-channel RGB image)

Many variants of CNN exist and have been boosted by the advent of **ImageNet in 2010**

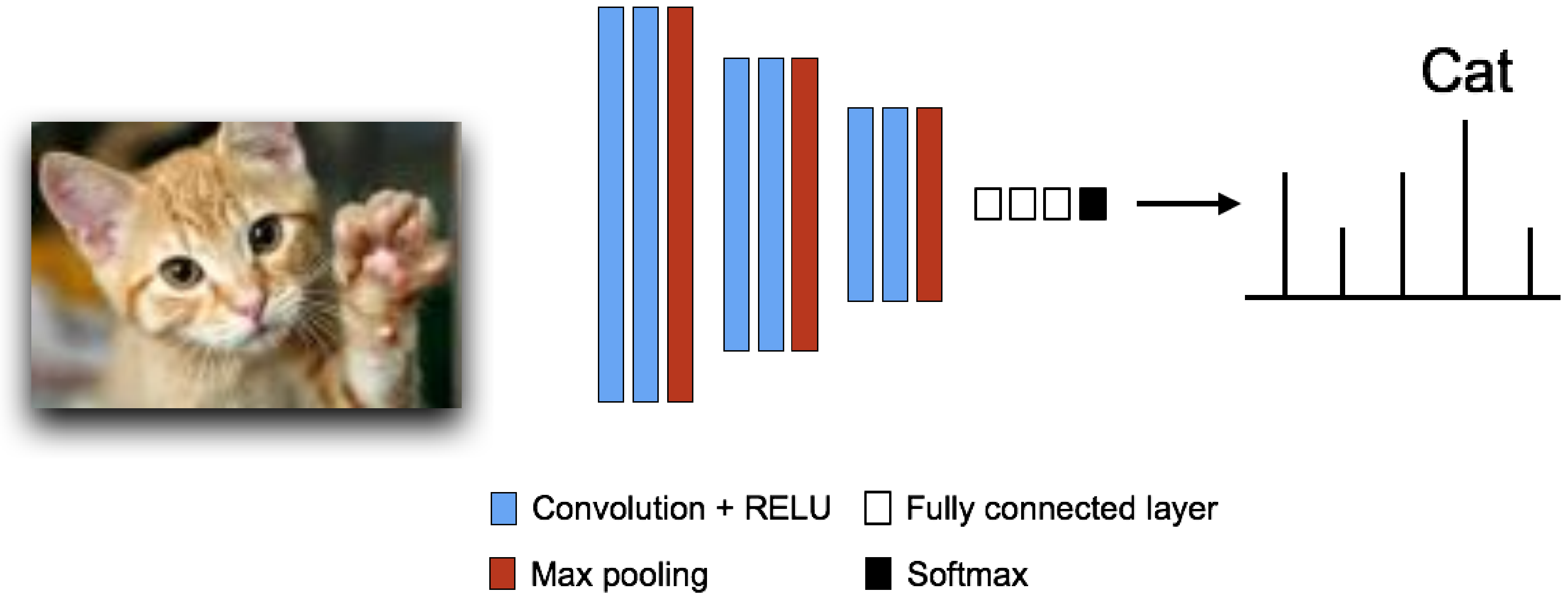
CNNs for image classification

First rewind and look at a simple CNN applied to real image classification



CNNs for image classification

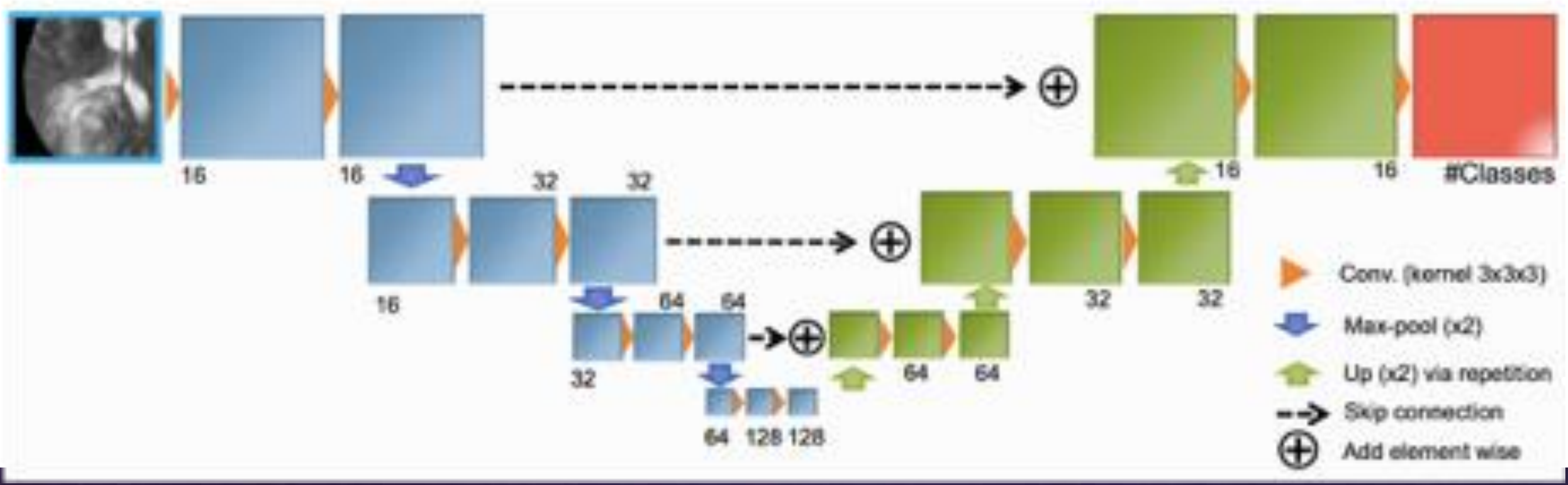
First rewind and look at a simple CNN applied to real image classification



CNNs for image segmentation

One major focus in computer vision and medical imaging is on **image segmentation**

- Challenging in medical imaging due to variability in patient anatomy & pathology, patient pose and motion, image artefacts
- U-net is the most common used network , which was proposed by Ronneberger (Google DeepMind) in 2015:



Bits & Bobs

All of the networks so far (deep neural networks, with our without convolutional layers), need to be carefully designed and trained:

Choice of:

- Number of hidden layers and neurons, stacking (AE)
- Loss function, activation function, learning rate, epochs
- **Optimisation methods**
- **Regularisation methods**
- **Transfer learning**

Optimisation

Choice of faster gradient-based optimisation methods for use with backpropagation:

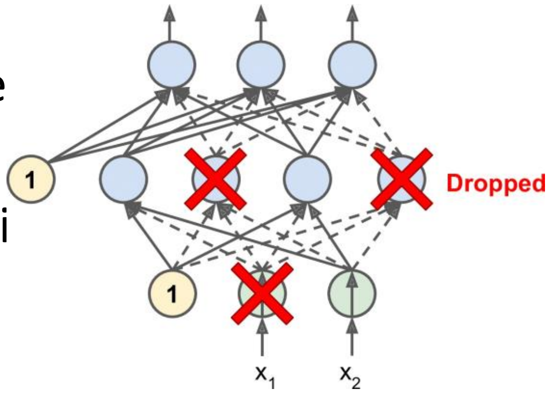
1. **Gradient Descent**
2. **Momentum optimisation**
3. **Nesterov Accelerated Gradient**
4. **AdaGrad**
5. **RMSProp**
6. **Adam Optimiser**

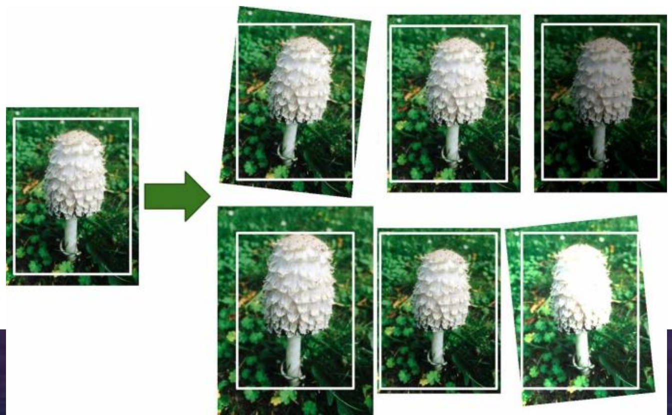
Regularisation

To avoid **overfitting**, we can do the following:

1. **Early stopping**: Interrupt training when its performance on the validation set starts dropping
2. ***l_1* and *l_2* regularization:** Add a regularization term in the cost function.
3. **Dropout**: At every training step, every neuron (input or hidden) has a probability *p* of being temporarily "dropped out"
4. **Data augmentation:** Generate new training instance artificially boosting the size of the training set.

   E.g. you can rotate, shift (translate), resize (scale), fli

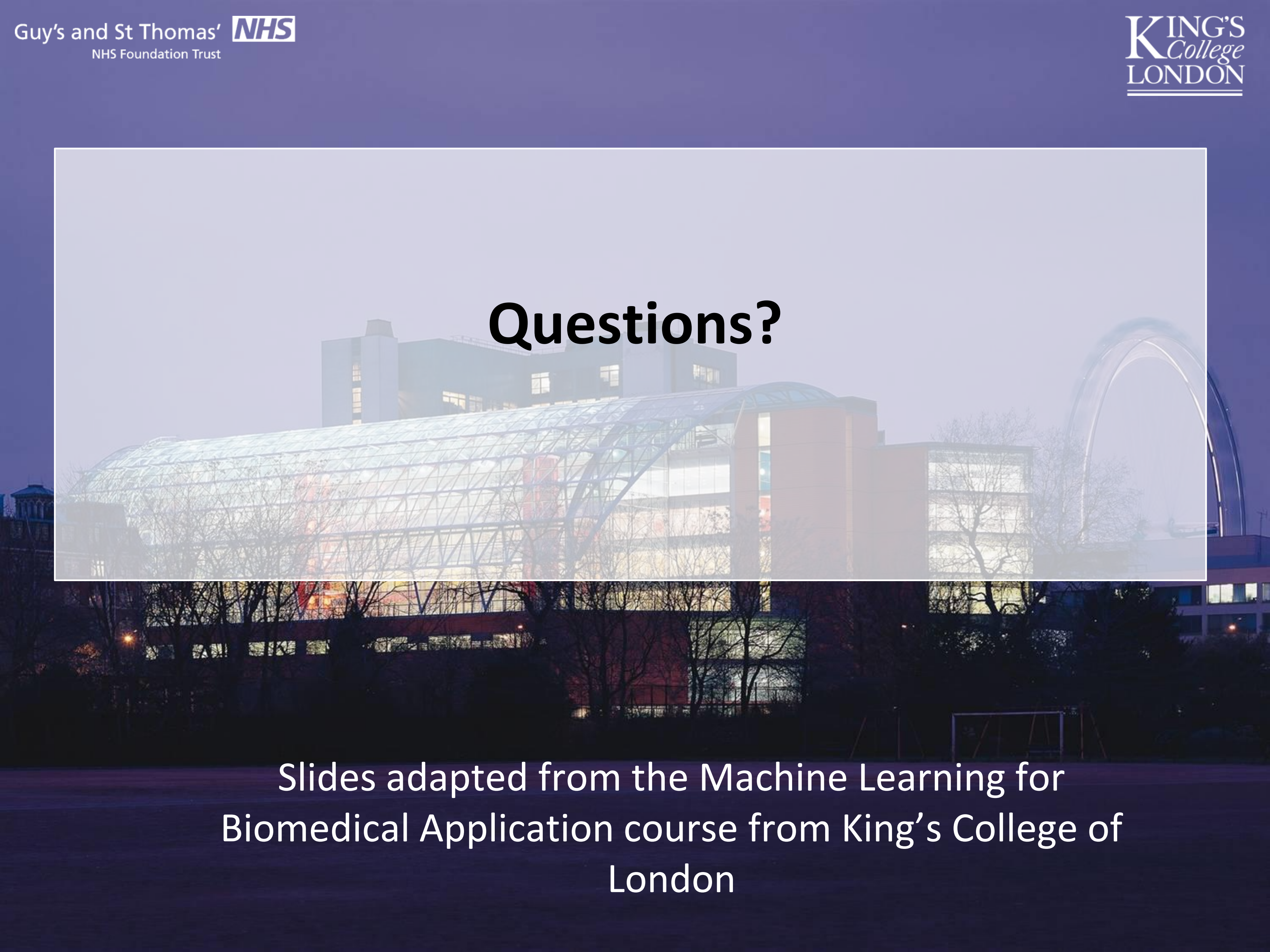




Summary

We can also exploit existing network architectures or parts thereof by:

- **Stacking** different types of layers, such as convolutional and pooling layers – it's very modular
- Creating **ensemble** architectures by aggregating results from different, separately trained architectures
- Using pre-trained networks (e.g. U-net) for similar applications (**transfer learning**)



Guy's and St Thomas' NHS
NHS Foundation Trust

KING'S College LONDON

Questions?

Slides adapted from the Machine Learning for Biomedical Application course from King's College of London