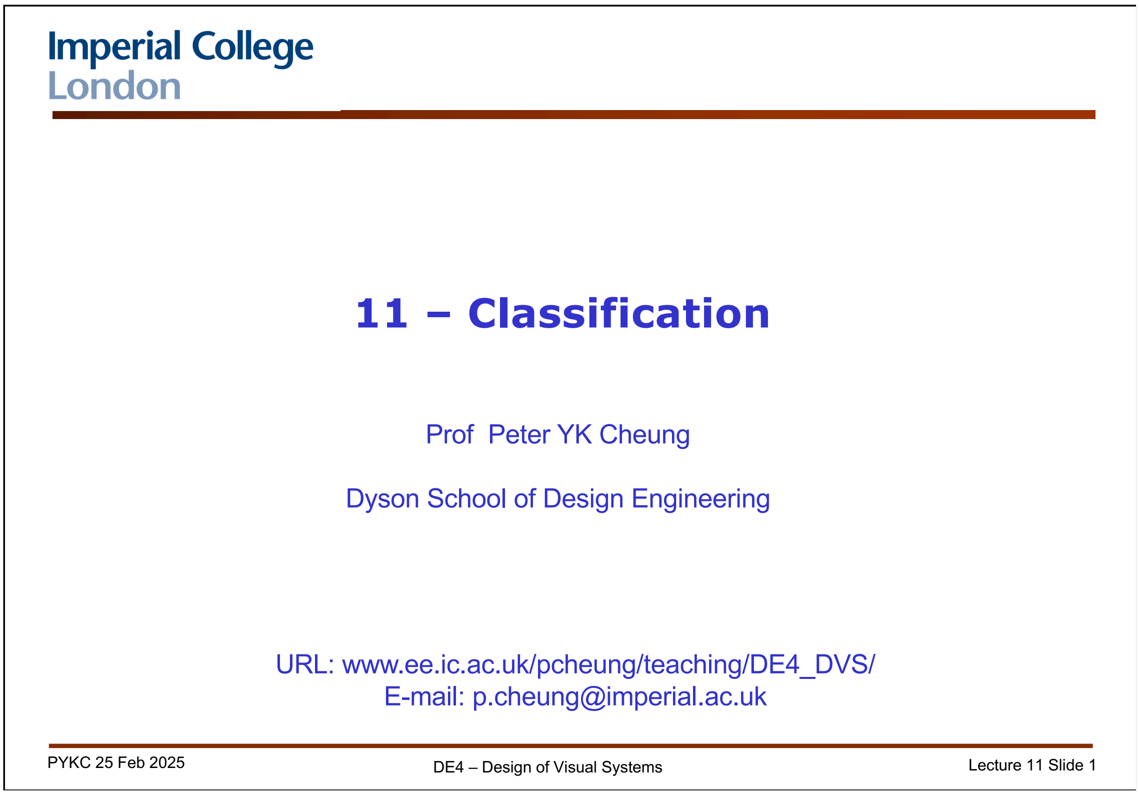


Imperial College London

11 – Classification

Prof Peter YK Cheung

Dyson School of Design Engineering

URL: www.ee.ic.ac.uk/pcheung/teaching/DE4_DVS/
E-mail: p.cheung@imperial.ac.uk

The materials in this Lecture is supplemented with Chapter 10 of the recommended textbook, “Digital Image Processing” by Gonzalez and Woods. As an Imperial College student, you can access the electronic version of this book via the following Imperial College library link:

https://imperial.alma.exlibrisgroup.com/leganto/public/44IMP_INST/lists/44618090290001591?auth=SAML

Approaches to image classification can be divided into three categories:

1. Classification by prototype matching,
2. Classification based on an optimal statistical formulation, and
3. Classification based on artificial neural networks.

With recent advancement in artificial neural networks and their amazing effectiveness, we will only consider the third approach in this lecture.

Recognition and Classification Problem

Face recognition

Handwriting recognition

Lorem ipsum dolor sit amet, consetetur sadipsaing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et
dolore magna aliquyam erat, sed diam voluptua. At vero
eos et accusam et justo duo dolores et ea rebum. Stet
clita kasd gubergren, no sea takimata sanktus est
Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet,
Les lavandes sont un genre de plantes
de la famille des lamiaceae.
Ce sont des arbrisseaux dicotylédones
a fleurs le plus souvent mauves ou
violettes disposees en epis dans la plupart
BRUNO'S

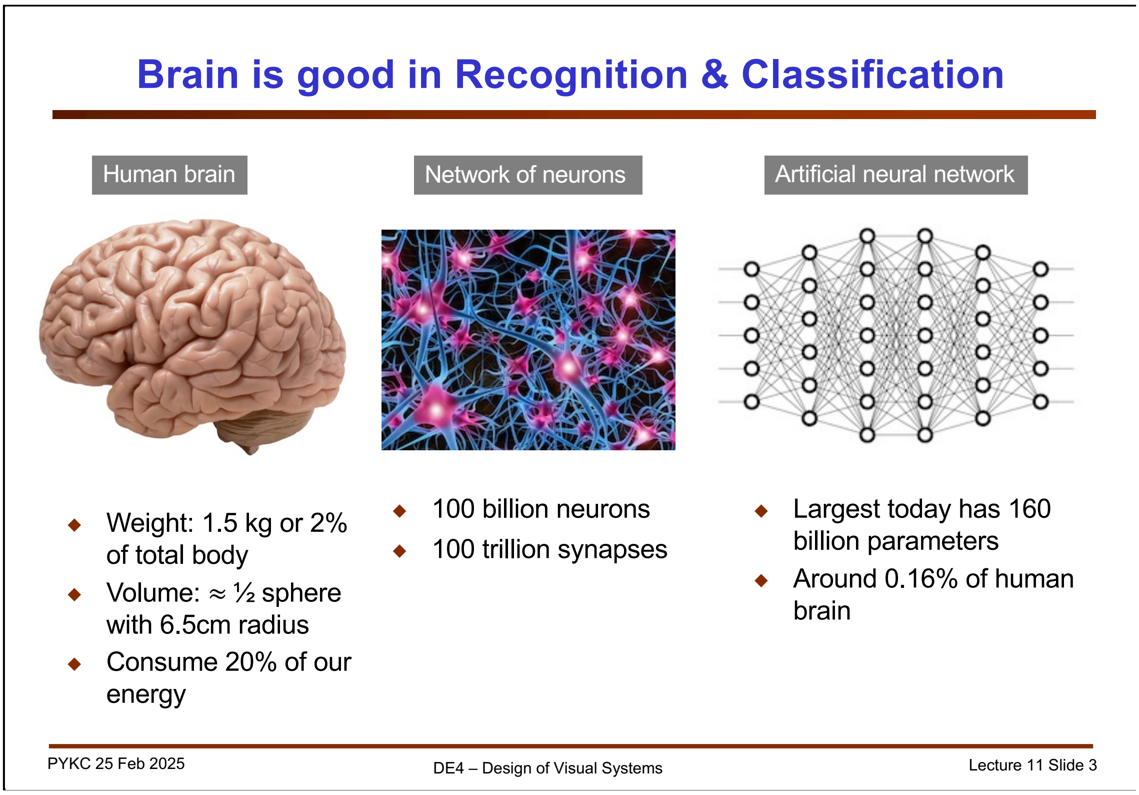
ImageNet (14 million images) – object recognition

There are many situations where image recognition and classification are required. Here are three examples.

We now use face recognition in many situations: border and immigration control at airports and secure login via face recognition, automatic handwriting recognition, handwritten text-to-speech translation, object and animal recognition etc..

The input to such recognition and classification system are raw or segmented images, and the output are the categories (or classes) with a figure of merit reflecting the confidence of the system.

In many of these problems, neural network is the go-to approach. The following slides will explain how a neural network works.



Brain is good in Recognition & Classification

Human brain
Network of neurons
Artificial neural network

- Weight: 1.5 kg or 2% of total body
- Volume: ≈ ½ sphere with 6.5cm radius
- Consume 20% of our energy

- 100 billion neurons
- 100 trillion synapses

- Largest today has 160 billion parameters
- Around 0.16% of human brain

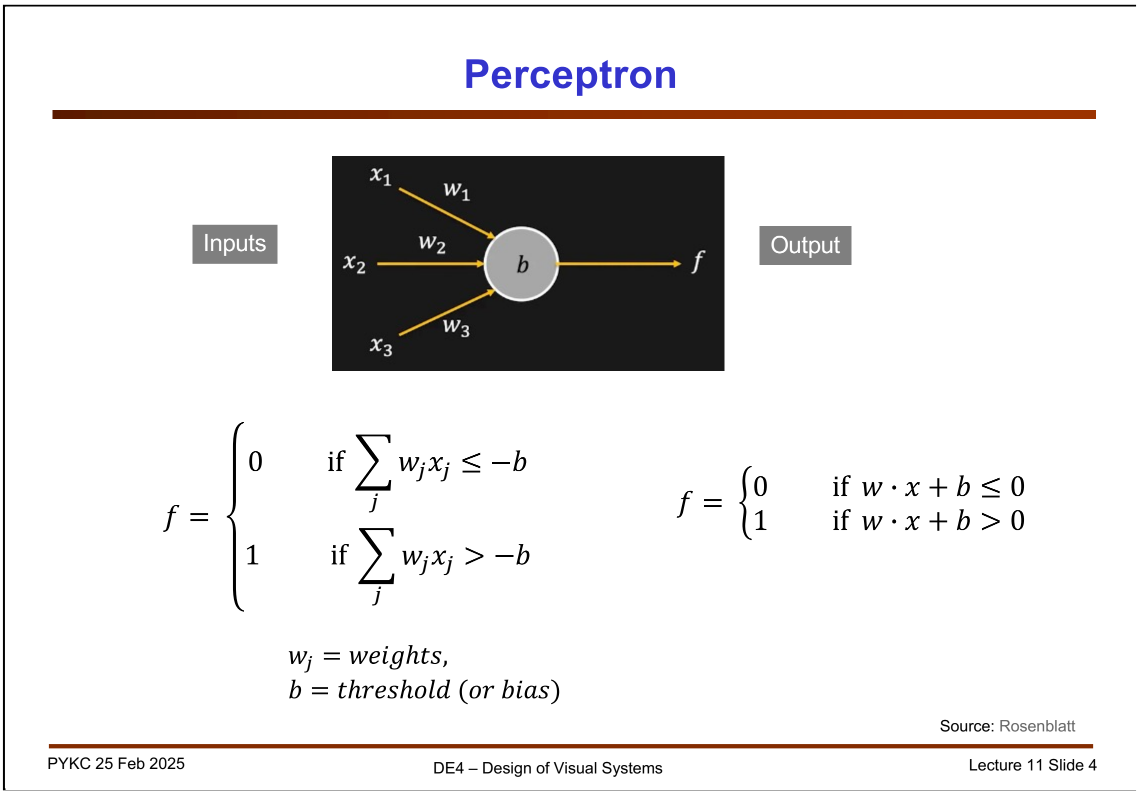
Artificial neural networks mimic how our brain works.

Human brains are remarkable. A typical brain weighs only 1.5kg, account for no more than 2% of our average body weight. Its volume is rough that of half a sphere with a radius of 6.5cm. Yet, it consumes around 20% of our energy.

This high energy consumption is because our brain typically has over 100 billion neurons with 100 trillion synapses (connections), all working simultaneously to make every part of the body functioning. (See Lecture 2.)

Artificial neural network (ANN) imitates the neural connections in the brain. However, with 160 billion connections (or parameters), even the largest artificial neural network to date (in 2023) is still less than 0.2% of a human brain's capability.

Even so, artificial neural network has transformed how we tackle many problems including those of image classification and recognition.



Perceptron

Inputs

Output

$$f = \begin{cases} 0 & \text{if } \sum_j w_j x_j \le -b \\ 1 & \text{if } \sum_j w_j x_j > -b \end{cases}$$

$$f = \begin{cases} 0 & \text{if } w \cdot x + b \le 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

$w_j = weights,$
$b = threshold\ (or\ bias)$

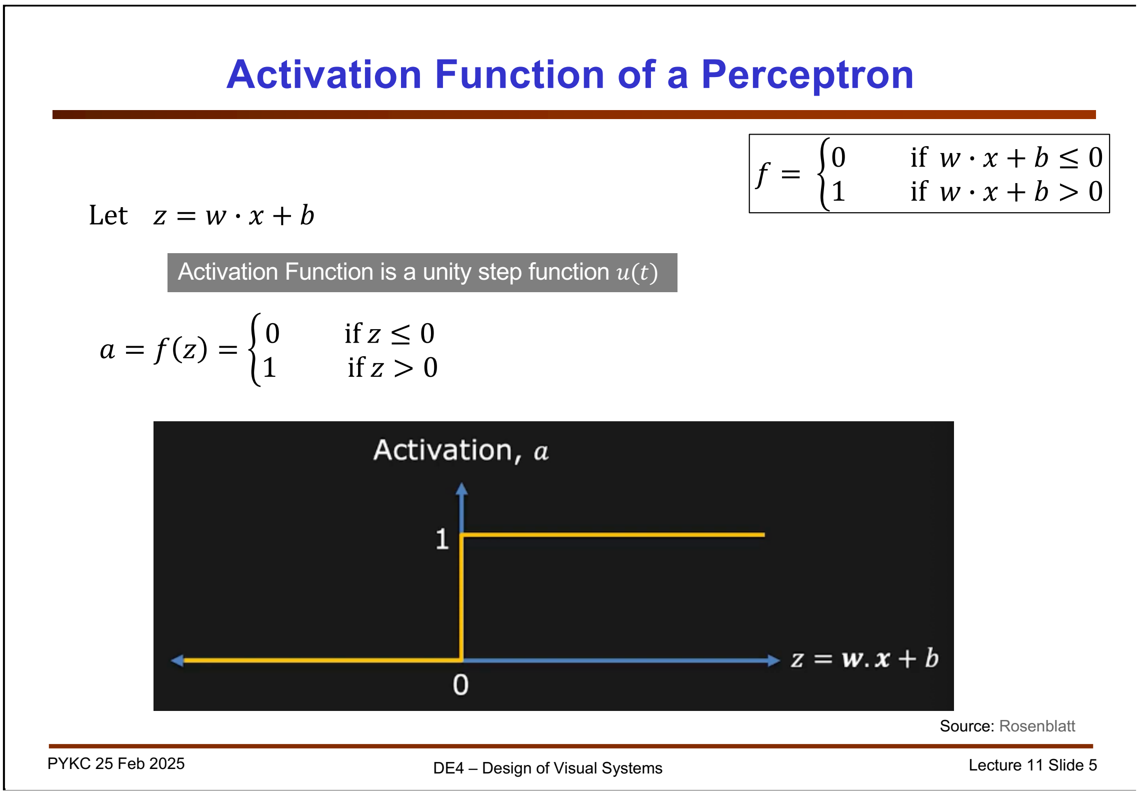
Source: Rosenblatt

The building block of the original ANN is a perceptron, first introduced by Rosenblatt in 1958. It is a processing element with inputs x_i and output f.

Every input is multiplied by their corresponding weights w_i and the products are summed together.

If the sum-of-products, after adding a bias b, is above 0, then the perceptron outputs a '1'. This mimics how a biological neuron fires an action potential when certain threshold is exceeded.

This can be formulated mathematically as shown above.



Activation Function of a Perceptron

$$f = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Let $z = w \cdot x + b$

Activation Function is a unity step function $u(t)$

$$a = f(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$$

Source: Rosenblatt

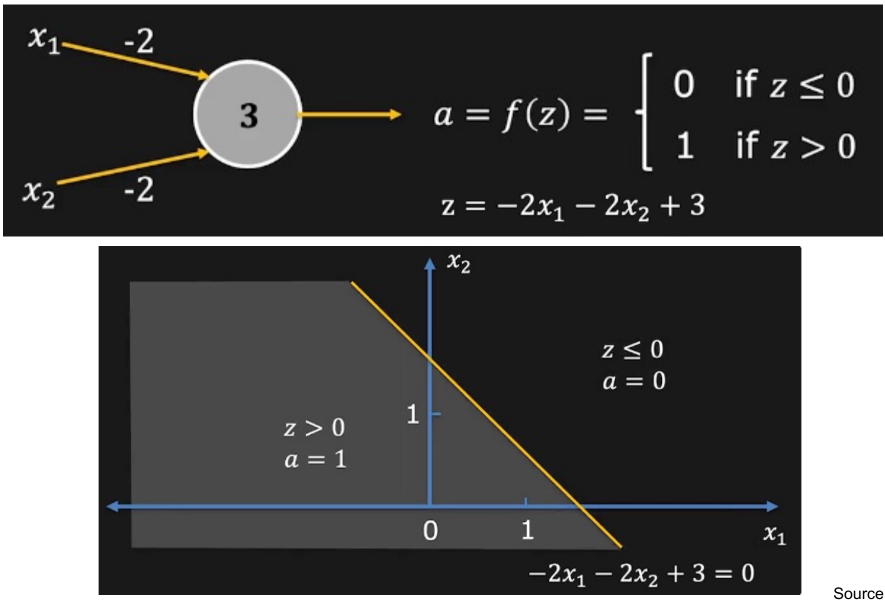
The ”firing” mechanism can be modelled mathematically as a function:

$$z = w \cdot x + b$$

Activation a, is therefore a function of z, $f(z)$, such that if z > 0, then activation occurs. Otherwise, there is no activation.

f is known as the **activation function**. In this case, it is a unit step function (similar but not exactly the same as $u(t)$ in DE2 Electronics 2).

Perceptron is a Linear Classifier



Source: Rosenblatt

A perceptron can be used as a **linear classifier**.

Let us consider a case of a perception having only two inputs. The weights in both input branches are -2, and the bias threshold is 3.

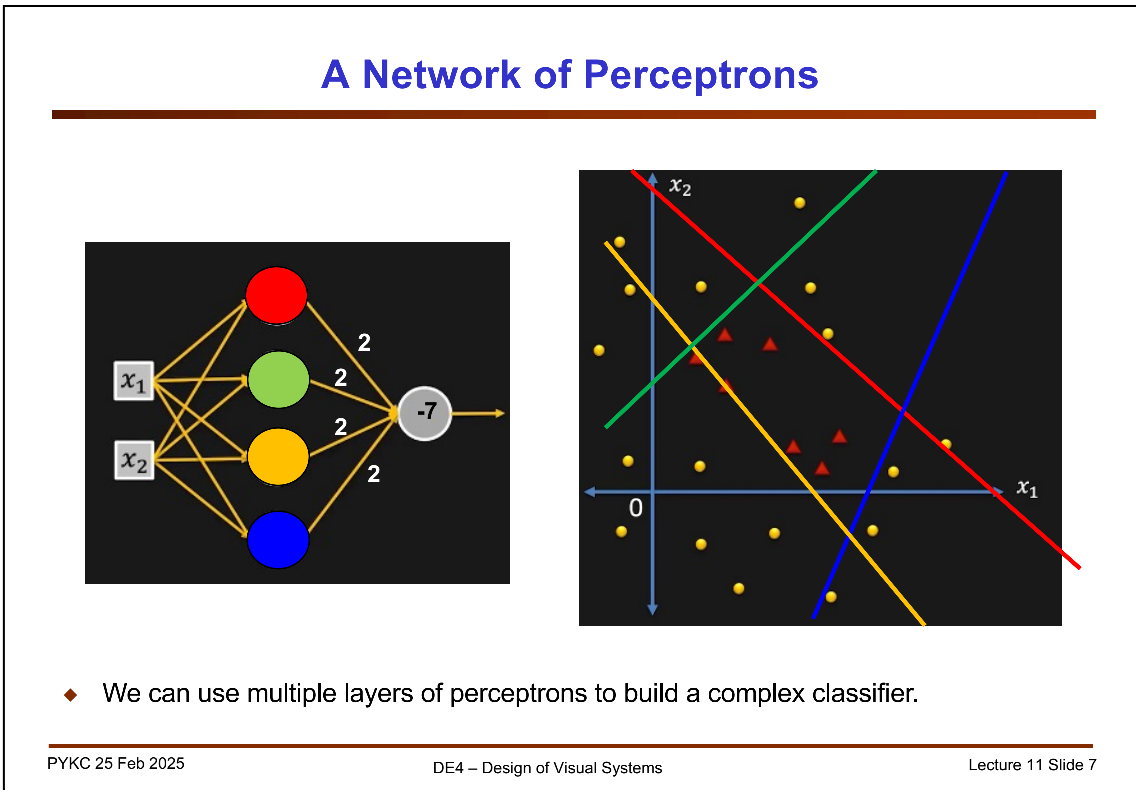
We can write the equation for z, the output as:

$z = -2x_1 - 2x_2 + 3$, and activation occurs if z > 0.

If we plot the function $-2x_1 - 2x_2 + 3 = 0$ on a plane with x_1 on the x-axis, and x_2 on the y axis, we get the straight line as shown in the slide.

This line divides the (x_1, x_2) space into two regions. The region above the line represent no activation, and the region below the line represent activation.

The entire (x_1, x_2) is therefore successfully classified into two categories.



A Network of Perceptrons

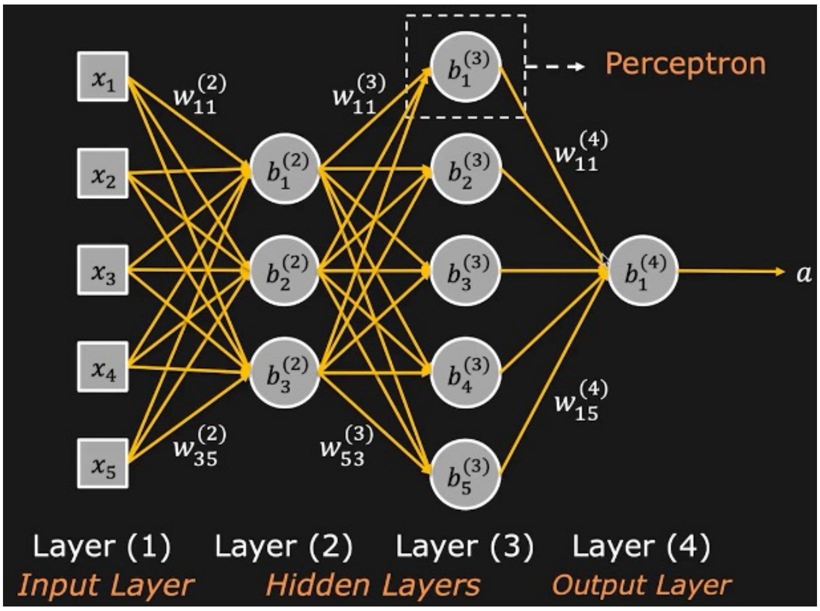
- We can use multiple layers of perceptrons to build a complex classifier.

That was one perceptron. What if we have a network of perceptron as shown here?

If we wish to divide the (x_1, x_2) space into two complex regions, one represented by the red triangles, and the other as yellow dots. This can be achieved by having a layer of four perceptrons. Each perceptron imposes a line boundary as shown in the corresponding colour.

Then the output layer perceptron combines the results from the middle (hidden) layer of perceptrons, and activation is achieved only if all four middle layer perceptrons fire.

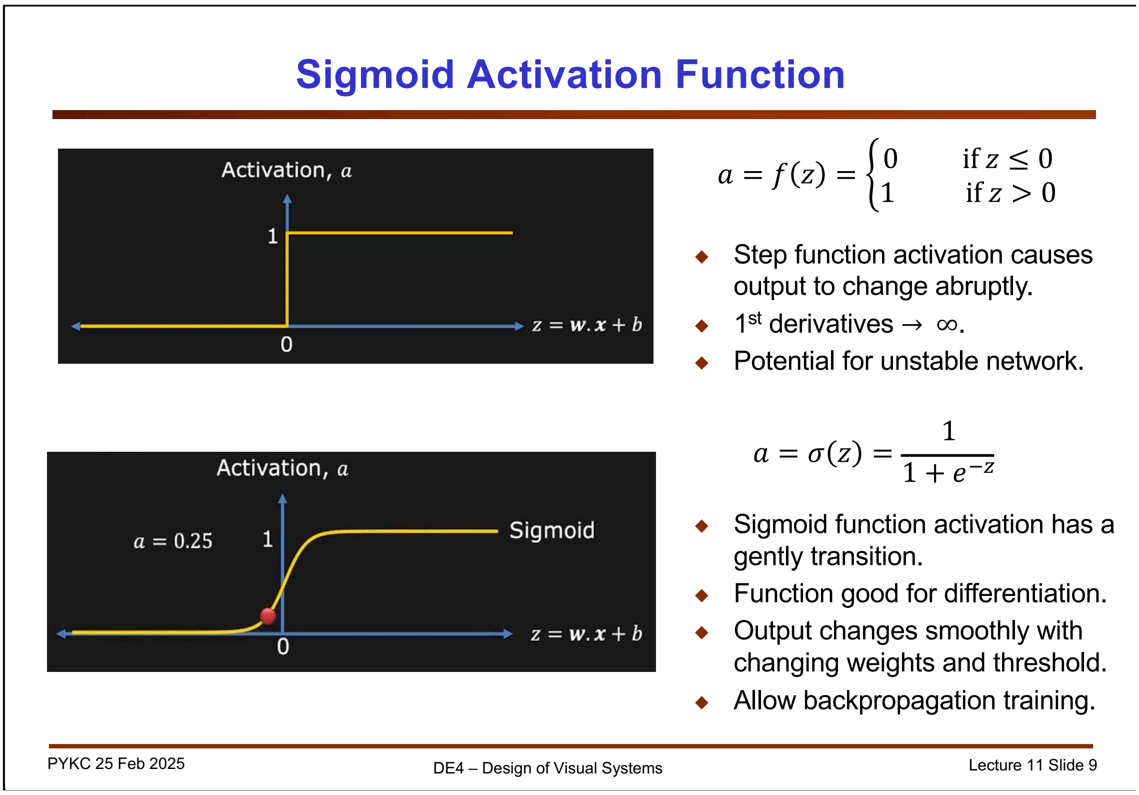
A Multi-layer Perceptron



This slide shows a typical multi-layer perceptron. The left most is called the input layer (layer 1). This just provides the input signals.

There follows two hidden middle layers 2 and 3. Then there is the output layers that merges all the hidden layer results to produce a final activation.

In this network, everything are connected – it is called a fully-connected network. In practice, many of the connection are missing – or the weight is zero.



Sigmoid Activation Function

Activation, a
1
0
$z = \boldsymbol{w}.\boldsymbol{x} + b$

$$a = f(z) = \begin{cases} 0 & \text{if } z \le 0 \\ 1 & \text{if } z > 0 \end{cases}$$

- Step function activation causes output to change abruptly.
- 1^{st} derivatives → ∞.
- Potential for unstable network.

Activation, a
$a = 0.25$
1
Sigmoid
0
$z = \boldsymbol{w}.\boldsymbol{x} + b$

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

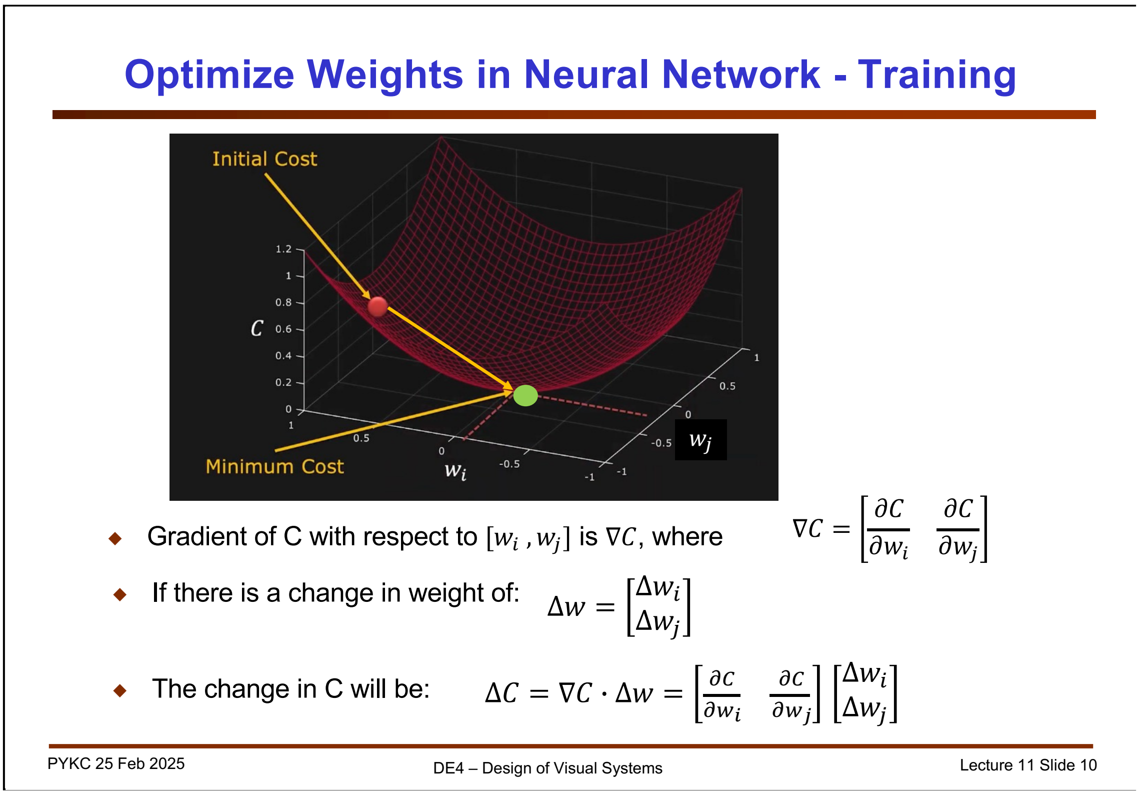
- Sigmoid function activation has a gently transition.
- Function good for differentiation.
- Output changes smoothly with changing weights and threshold.
- Allow backpropagation training.

PYKC 25 Feb 2025 DE4 – Design of Visual Systems Lecture 11 Slide 9

So far, we have only considered activation function which is a step function. Such an activation function is not particularly useful.

A step function is not differentiable because at the step, the gradient is infinite. This causes many undesirable effects in a typical system. The network can potentially become unstable because small changes in weights or threshold values can flip the output. Therefore, it would be better if we can avoid the discontinuity in the activation function.

A much better activation function to use is the "**sigmoid**" function as defined in the slide. This function has a nice 1^{st} derivative that makes it much easier to perform backpropagation training. The output now changes smoothly with small changes in weights and thresholds.



Optimize Weights in Neural Network - Training

Initial Cost
Minimum Cost
C
1.2
1
0.8
0.6
0.4
0.2
0
1
0.5
0
-0.5
-1
w_i
1
0.5
0
-0.5
-1
w_j

- Gradient of C with respect to $[w_i\,, w_j]$ is ∇C, where $\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w_i} & \frac{\partial C}{\partial w_j} \end{bmatrix}$
- If there is a change in weight of: $\Delta w = \begin{bmatrix} \Delta w_i \\ \Delta w_j \end{bmatrix}$
- The change in C will be: $\Delta C = \nabla C \cdot \Delta w = \begin{bmatrix} \frac{\partial c}{\partial w_i} & \frac{\partial c}{\partial w_j} \end{bmatrix} \begin{bmatrix} \Delta w_i \\ \Delta w_j \end{bmatrix}$

Given a neural network with weights at the connections, how should these weights be determined so that the network produces the desired output?

This is done through training. The idea is that a network output may differ from the desired output. This is modelled by a cost function C. The goal of training is to minimize C by optimizing the weights.

Shown in the slides is a simple example of two weights w_1 and w_2. The relationship between the cost C and the two weights is a surface as shown in RED. The red dot is the initial cost for initial weights. The goals is to find the values of w_1 and w_2 for the C value at the location of the green dot, i.e. the bottom of the surface with minimum cost.

To do this, we need to calculate the gradient of C with respect to w_1 and w_2:

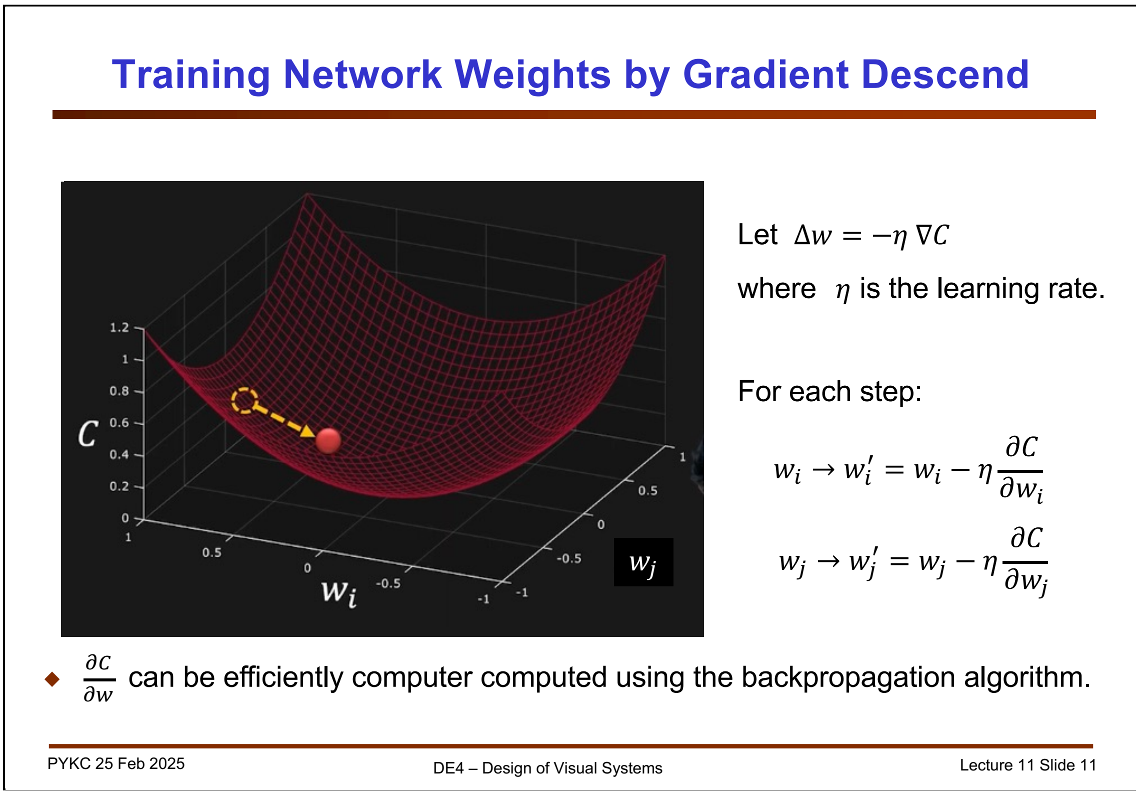
$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w_i} & \frac{\partial C}{\partial w_j} \end{bmatrix}$$

Now if there are small changes in the weights, i.e.

$$\Delta w = \begin{bmatrix} \Delta w_i \\ \Delta w_j \end{bmatrix}$$

Then the change in the cost will be:

$$\Delta C = \nabla C \cdot \Delta w = \begin{bmatrix} \frac{\partial c}{\partial w_i} & \frac{\partial c}{\partial w_j} \end{bmatrix} \begin{bmatrix} \Delta w_i \\ \Delta w_j \end{bmatrix}.$$



Training Network Weights by Gradient Descend

Let $\Delta w = -\eta \, \nabla C$

where η is the learning rate.

For each step:

$$w_i \rightarrow w_i' = w_i - \eta \frac{\partial C}{\partial w_i}$$

$$w_j \rightarrow w_j' = w_j - \eta \frac{\partial C}{\partial w_j}$$

- $\frac{\partial C}{\partial w}$ can be efficiently computer computed using the backpropagation algorithm.

PYKC 25 Feb 2025 DE4 – Design of Visual Systems Lecture 11 Slide 11

A common technique to finding the value of w_1 and w_2 to reach to minimum cost is to apply the **gradient descend algorithm**.

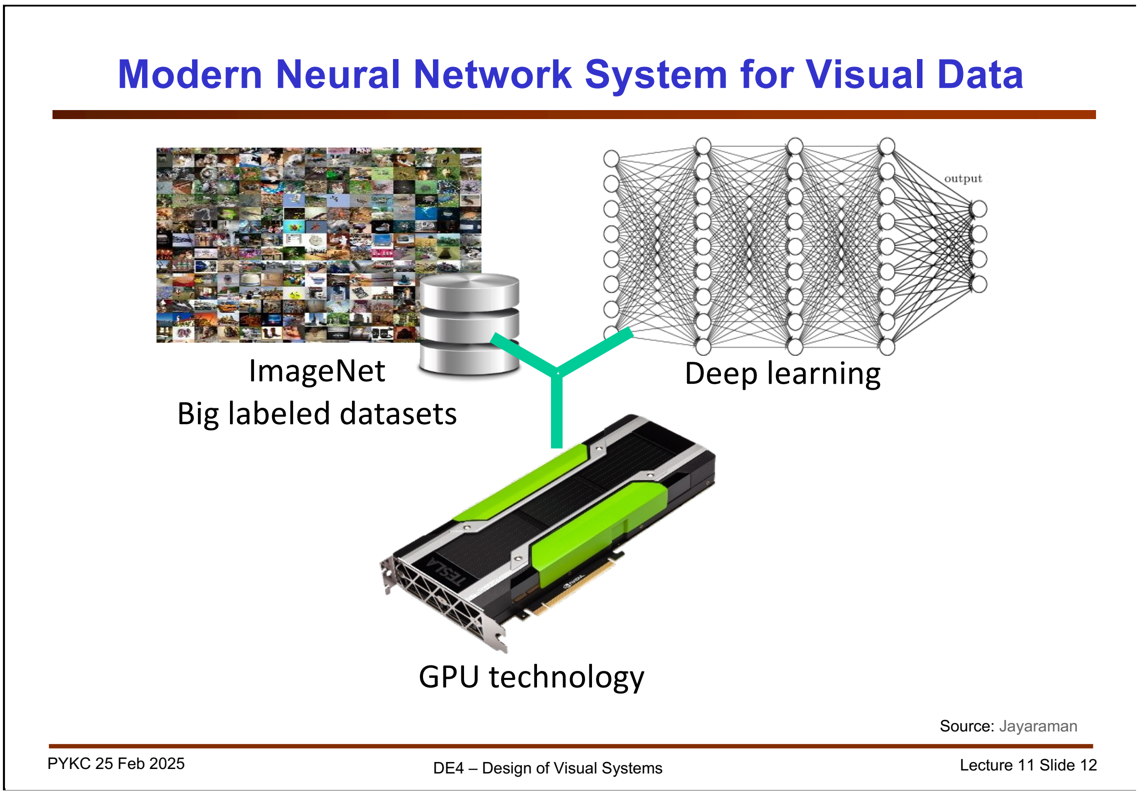
Assume we already calculated the gradient of C with respect to w_1 and w_2 , i.e. ∇C, we can modify the weights according to:

$$\Delta w = -\eta \, \nabla C$$

That is, we move in the opposition direction to the gradient (so that we go downhill instead of uphill) by an amount determined by η. η is called the **learning rate**: a small η means that we may reach the bottom after many steps. However, too large an η could result in overshooting the target and oscillating around it.

The above discussion assumes that we know the value of ∇C. Calculating the gradient of the cost function turns out to be a computationally expensive problem. However, a technique was introduced in 1986 by Rumelhart, Hinton and Williams known as the **backpropagation algorithm** which makes finding the gradients very efficient. This is the general method used in the process of training neural networks.

The paper on this can be found here:
https://www.cs.toronto.edu/~hinton/absps/naturebp.pdf



Modern Neural Network System for Visual Data

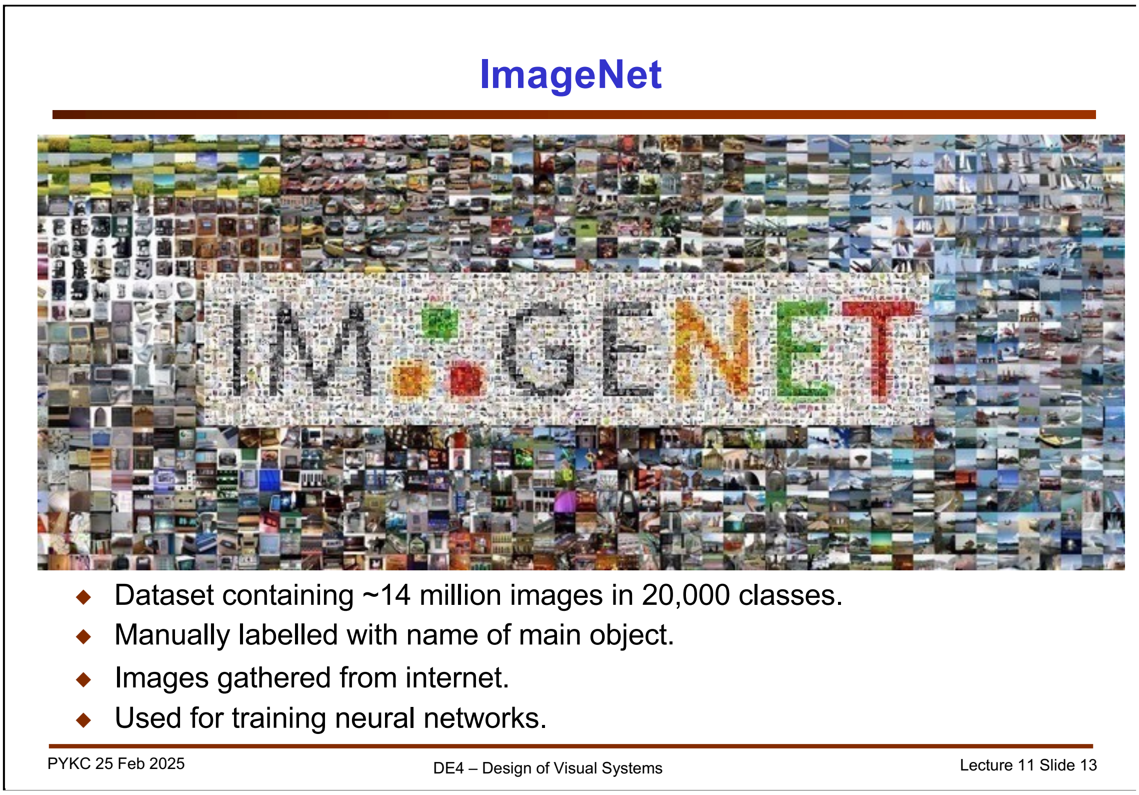
ImageNet
Big labeled datasets

Deep learning

GPU technology

Source: Jayaraman

Modern neural networks for image classification now relies on having a very large image data set. The data is used to train a very large and deep neural network, which has many hidden layers. The entire training process is perform using high performance GPUs.

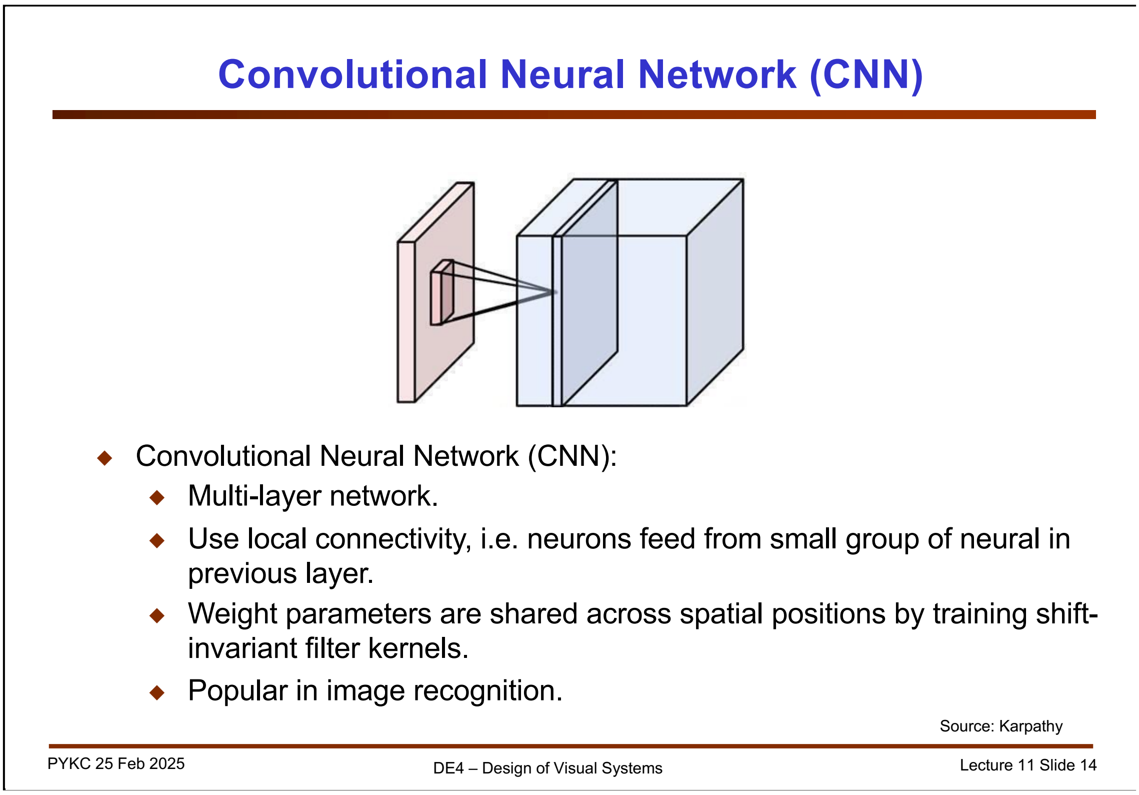


ImageNet

- Dataset containing ~14 million images in 20,000 classes.
- Manually labelled with name of main object.
- Images gathered from internet.
- Used for training neural networks.

ImageNet started in 2006, and is the largest dataset available to the research community for neural network training.

It contains over 14 million images in 20,000 classes, and these are manually labelled – i.e. the class or category an image belongs is known.

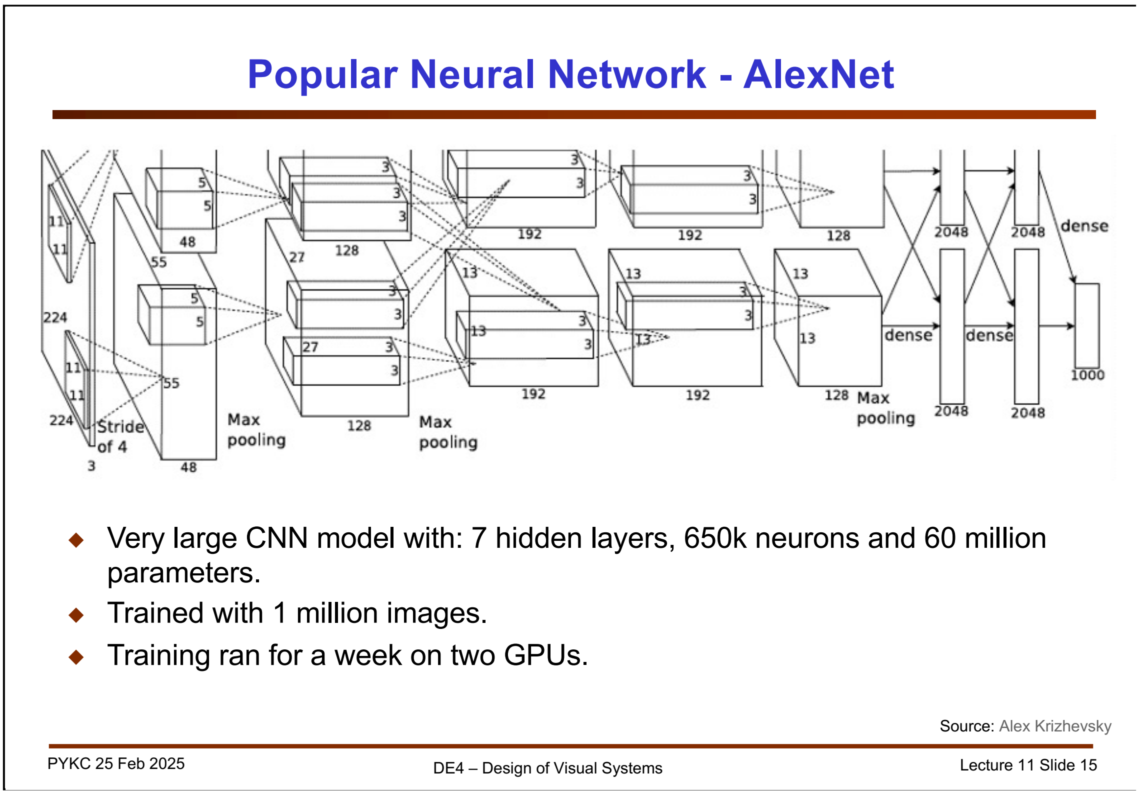


Convolutional Neural Network (CNN)

- Convolutional Neural Network (CNN):
  - Multi-layer network.
  - Use local connectivity, i.e. neurons feed from small group of neural in previous layer.
  - Weight parameters are shared across spatial positions by training shift-invariant filter kernels.
  - Popular in image recognition.

Source: Karpathy

Many neural networks for image recognition are based on the **convolutional neural network** (CNN) architecture. Unlike the multi-layer perceptron, which is fully connected, the CNN has multiple layers, but many of the middle layer neurons are connected to small groups of 'local' neurons in its neighbourhood. Further, the weights are often shared across different locations in a layer, relying on the shift-invariant properties of filter kernels. We will not go into CNN in details, but we will be using a number of CNNs to perform recognition during the laboratory session this week.



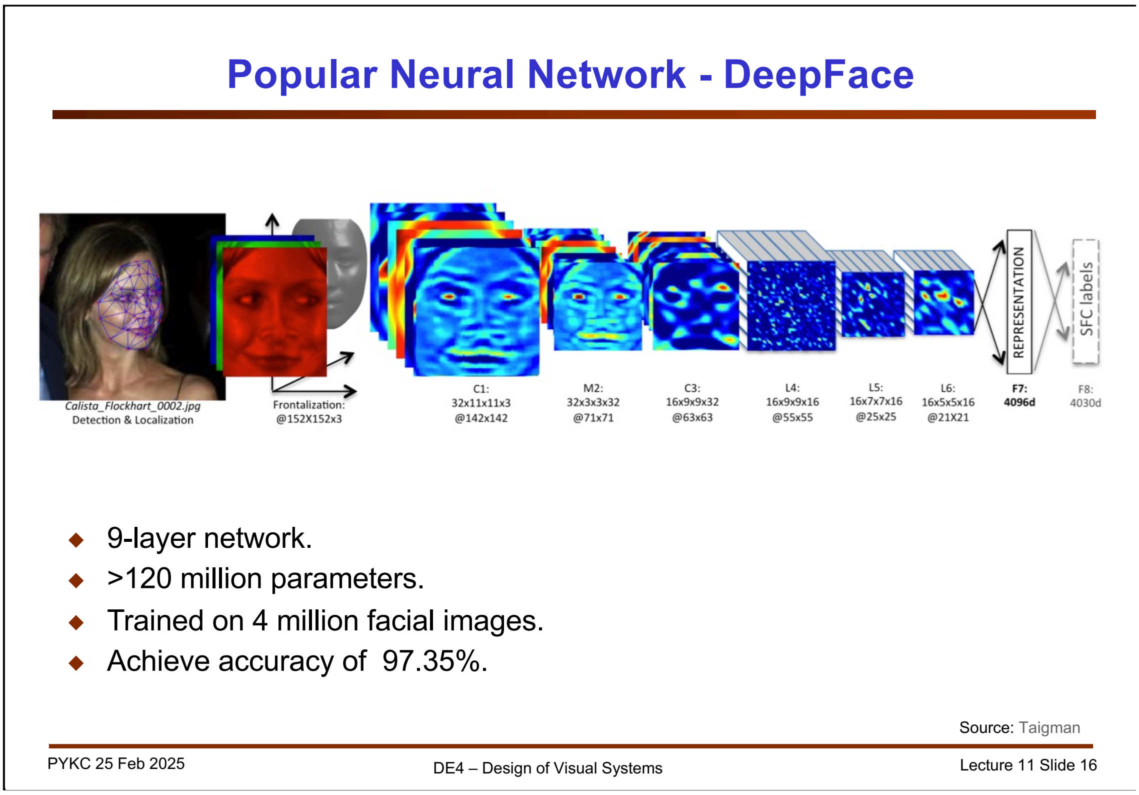
Popular Neural Network - AlexNet

- Very large CNN model with: 7 hidden layers, 650k neurons and 60 million parameters.
- Trained with 1 million images.
- Training ran for a week on two GPUs.

Source: Alex Krizhevsky

One of the first successful CNN for image recognition is the **AlexNet** created by Alex Krizhevsky in 2012, when he and his team won the ImageNet challenge that year.

This is a deep CNN (deep meaning that there are many (7) hidden layers) with 650k elements (neurons) and 60 million connections (weights). The network was trained on 1 million images. This took over a week to train on two GPUs.

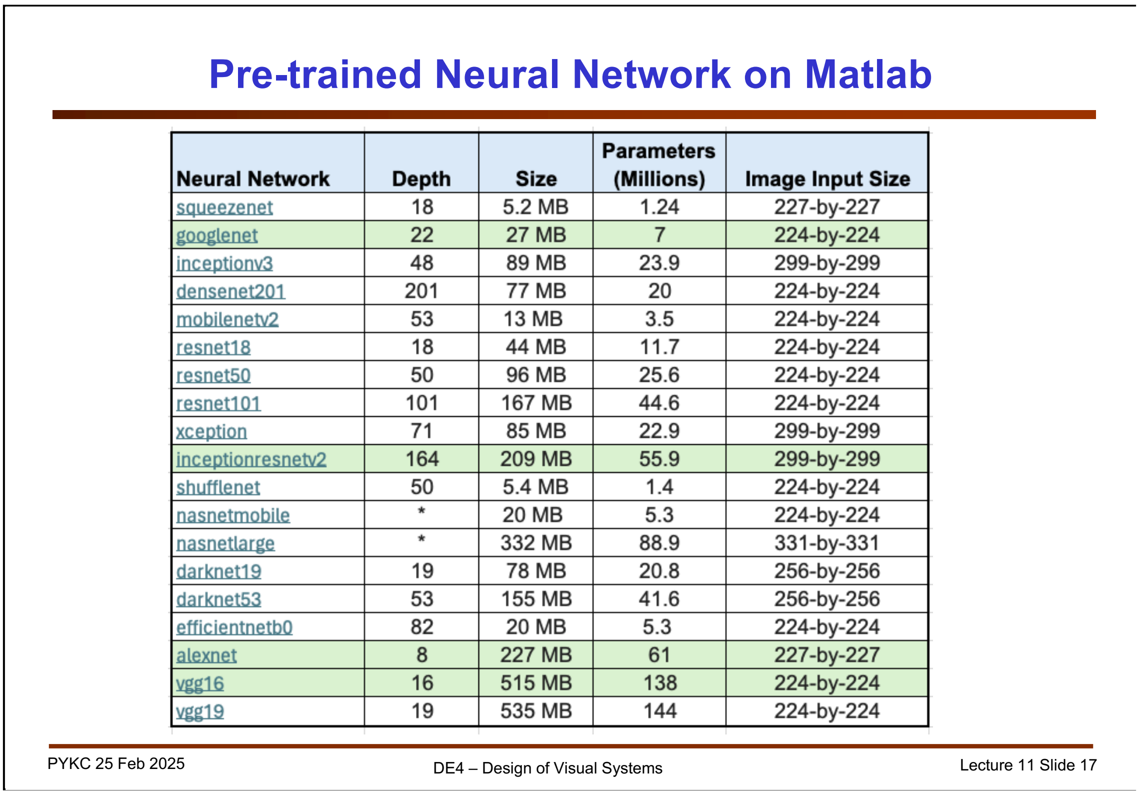


Popular Neural Network - DeepFace

- 9-layer network.
- >120 million parameters.
- Trained on 4 million facial images.
- Achieve accuracy of 97.35%.

Source: Taigman

Another famous CNN is Facebook's DeepFace network. You can read about this work here: https://openaccess.thecvf.com/content_cvpr_2014/papers/Taigman_DeepFace_Closing_the_2014_CVPR_paper.pdf



Pre-trained Neural Network on Matlab

Neural Network	Depth	Size	Parameters (Millions)	Image Input Size
squeezenet	18	5.2 MB	1.24	227-by-227
googlenet	22	27 MB	7	224-by-224
inceptionv3	48	89 MB	23.9	299-by-299
densenet201	201	77 MB	20	224-by-224
mobilenetv2	53	13 MB	3.5	224-by-224
resnet18	18	44 MB	11.7	224-by-224
resnet50	50	96 MB	25.6	224-by-224
resnet101	101	167 MB	44.6	224-by-224
xception	71	85 MB	22.9	299-by-299
inceptionresnetv2	164	209 MB	55.9	299-by-299
shufflenet	50	5.4 MB	1.4	224-by-224
nasnetmobile	*	20 MB	5.3	224-by-224
nasnetlarge	*	332 MB	88.9	331-by-331
darknet19	19	78 MB	20.8	256-by-256
darknet53	53	155 MB	41.6	256-by-256
efficientnetb0	82	20 MB	5.3	224-by-224
alexnet	8	227 MB	61	227-by-227
vgg16	16	515 MB	138	224-by-224
vgg19	19	535 MB	144	224-by-224

Finally, you can find many pre-trained networks available on Matlab. The table above shows the namse of these pre-trained networks for image recognition and the size of the network (via the number of parameters). You will be trying those highlighted in green during your Laboratory session on Thursday.