

# Lecture 7: Convolutional Neural Networks

Isabel Casas  
[icasas@deusto.es](mailto:icasas@deusto.es)

# What is deep learning?



Figure 1: Image source:

<https://scherlund.blogspot.com/2017/12/differences-between-ai-machine-learning.html>

- Multilayer Perceptron
  - ▶ regression (bike rent demand forecast)
  - ▶ classification (mnist database)
- Can you understand this **joke** now?

# Deep Learning

- Deep learning is a subset of machine learning that uses artificial neural networks to model and solve complex problems.
- It involves training a neural network with large amounts of data to recognize patterns and make predictions or decisions without being explicitly programmed for them.
- The term “deep” in deep learning refers to the use of deep neural networks, which are neural networks with many hidden layers.
- These layers allow the network to learn and represent complex features in the data, making it possible to tackle tasks like image and speech recognition, natural language processing, and game playing.

- Deep learning has achieved impressive results in many areas: computer vision, speech recognition, and natural language processing.
- It has led to breakthroughs in fields like healthcare, finance, and transportation, where it is used to analyze data, predict outcomes, and make decisions.
- Overall, deep learning is a powerful and rapidly evolving field that has the potential to transform many industries and change the way we live and work.

# Particular Examples of Deep Learning Use in Technology

- *Google's* image search, language translation, voice assistants, and self-driving cars all rely on deep learning algorithms. In addition, Google is using deep learning to improve its search algorithms and make more accurate predictions.
- *Amazon* uses deep learning to personalize its recommendations and search results for customers, as well as to optimize its supply chain and logistics operations. The company also uses deep learning in its voice assistant, Alexa, to understand and respond to natural language queries.
- *Facebook* uses deep learning to improve its image and object recognition, as well as to analyze and classify the content on its platform. The company also uses deep learning in its language translation and voice recognition tools.

# Examples of Deep Learning Use in Cybersecurity

- *Darktrace* is a cybersecurity company that uses unsupervised deep learning algorithms to detect and respond to cyber threats in real-time.
- *Cylance* uses deep learning to develop predictive models that can identify and prevent malware attacks.

# Examples of Deep Learning Use in Healthcare

- *Google Health* is using deep learning to improve medical imaging and diagnostics: its algorithms analyze medical images, such as MRI scans, to detect and diagnose diseases like cancer and eye disorders.
- *BenevolentAI* uses deep learning to accelerate drug discovery and development: its algorithms analyze massive amounts of scientific data to identify potential drug candidates and predict their efficacy.
- *PathAI* uses deep learning to improve cancer diagnosis and treatment: its algorithms analyze tissue samples to identify cancerous cells and predict patient outcomes.



# Other Examples of Deep Learning Use

- *Netflix* uses deep learning to personalize its recommendations for individual users, as well as to optimize its content delivery network and improve the video quality of its streams.
- *Tesla* uses deep learning in its self-driving car technology to recognize and react to objects and obstacles in real-time. The company's Autopilot system uses neural networks to identify and classify objects, such as other cars, pedestrians, and traffic lights.
- *ChatGPT* is a deep learning-based conversational AI model developed by OpenAI. Specifically, ChatGPT is based on a type of deep learning model called a transformer, which was first introduced in a research paper by Google in 2017.

# Image processing

Visual cortex processes information in a hierarchical manner: simple features first and progressing to more complex ones.

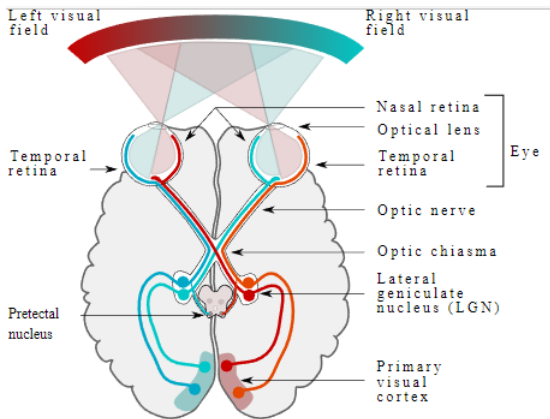


Figure 2: Image source:

"[https://commons.wikimedia.org/wiki/File:Human\\_visual\\_pathway.svg](https://commons.wikimedia.org/wiki/File:Human_visual_pathway.svg)"

# Swan recognition

We want to train a machine to recognise a swan. As humans, we might look at the beak, the long neck and the body shape.



Figure 3: Swan distinctive body parts. Image source : <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>

# Swan recognition

- A machine can learn to identify these body parts from many pictures and classify what is a swan and what is not.
- But... not all pictures are the same



Figure 4: Flying swan distinctive body parts. Image source : <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>

# Swan recognition

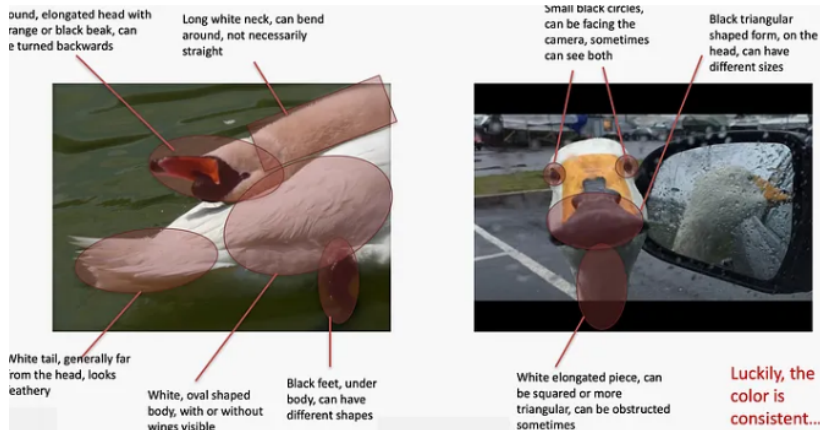


Figure 5: Contorsionist swans. Image source : <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>

# Swan recognition

Even the colour can change.

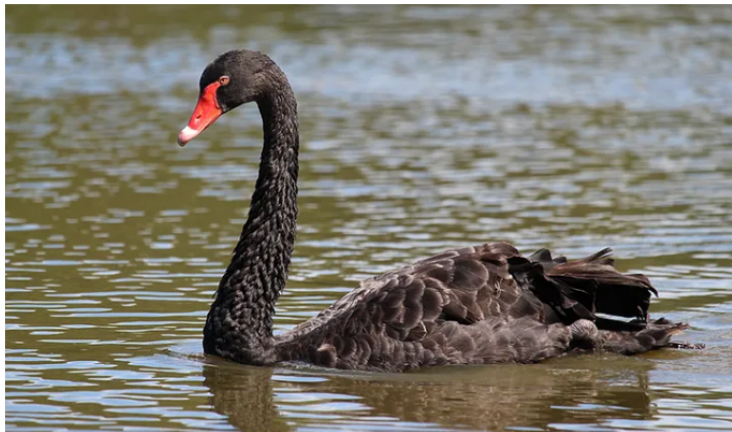
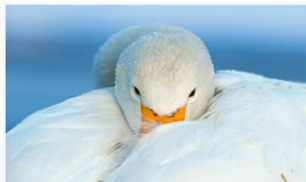
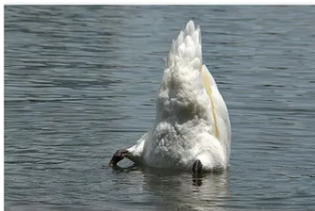


Figure 6: Black swan. Image source : <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>

# Swan recognition

Many other scenarios.



Man in swan tent photographing swans

Figure 7: Other swans. Image source : <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>

# Multilayer Perceptron

Multilayer Perceptron graph.

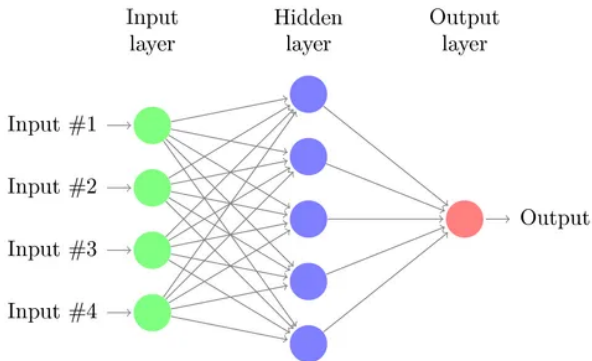


Figure 8: Standard MLP. Image source : <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>



# Multilayer Perceptron for Image Recognition?

- Classical artificial neural networks such as MLP or any of the other methods we have seen are not good for image recognition.
- An image is a matrix with numbers representing the colour/gray of a pixel. If we have only a gray scale like in Lecture 6 with the MNIST database, we need only one matrix per image and can be written into one vector and use the MLP or random forest to classify it.
- But... those were simple images: 10 different numbers in black and white and we had to look at the same part of the picture to identify each number. They are not turned around numbers for instance.
- So the input layer is very similar for every image of the same number. For example, the white areas are in the same place for two different images of the same number and similarly with the darker areas. So this translate well into a vector.
- This is not the case for images like the swan.

# MLP for Image Recognition

- Too large for the input layer. For example, a  $224 \times 224$  image in color (3 colours in the RGB), will need 150,528 weights only to connect the input and hidden layers.
- Those weights will only work if all pictures have the same structure, for example always the beak is on the middle right side of the pictures, etc.

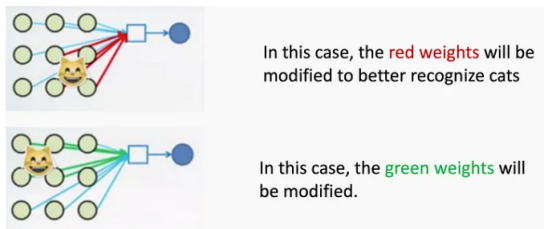


Figure 9: MLP challenge. Image source : <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>

- We resolve this with convolutional neural networks (CNN)

# Convolutional Neural Network

- We use three main types of layers to build CNN architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer
- The simplest CNN has one convolutional layer, activation function, pooling layer and fully connected layer.

INPUT → CONV → RELU → POOL → FC

Figure 10: Simplest CNN.

- We explain the function of these layers below

# Description of layers

- INPUT layer contains the input data. The dimension is number of raw pixel values  $\times 3$  (from 3 RGB)
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as pixel values  $\times 3 \times 12$  if we decided to use 12 filters.
- RELU layer will apply an element-wise activation function, such as the  $\max(0, x)$ . This leaves the size of the volume unchanged.
- POOL layer will perform a down-sampling operation along the spatial dimensions (width, height), resulting in a smaller volume  $\times 12$  filters.
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size  $[1 \times 1 \times \text{number of categories}]$ , for example 10 categories of MINST.

# Setup the layers

The basic building block of a neural network is the layer. Layers extract representations from the data fed into them. And, hopefully, these representations are more meaningful for the problem at hand. Figure 11 is a graphical representation of a general CNN.

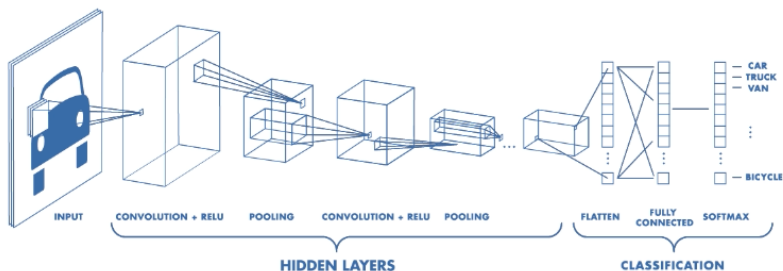


Figure 11: CNN. Image Source : <https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks-1489512765771.html>.

# Convolutional layer: filter

- CNNs were inspired by the structure and function of the visual cortex in the brain.
- CNNs mimic this hierarchical organization by using convolutional layers to learn simple features (for example the beak and the size of swan neck) and combining them at higher layers to learn more complex features (how these are connected).
- A **filter** (or kernel) analyses the relation between nearby pixels. For example, if the beak is red, pixels in a neighbourhood of a beak will be read or will degrade away from red.
- The user specifies the size of the filter (for example 3x3 or 5x5 matrix), which is the window sizes in which the whole picture is divided. This window moves across the figure from top left to bottom right. For each point in the image, a value is calculated based on the filter using a convolution operation.

# Convolution operation

Output = Input \* Filter

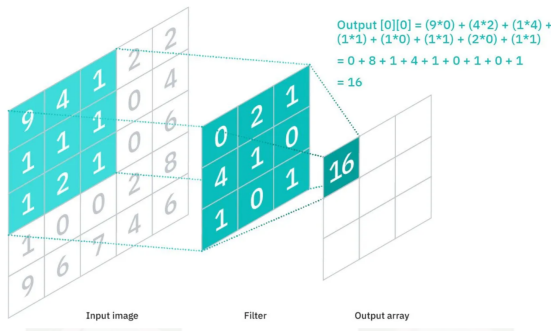


Figure 12: Graph of a convolution. Image source: IBM

- A filter reduces the number of weights that the neural network must learn compared to an MLP, and also means that when the location of these features changes it does not throw the neural network off

## Activity 2

The input volume of one image is the following 6x6 matrix ( $X_i$ ):

$X =$

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

Apply the following filter to it to obtain the result of a convolution layer ( $w_i$ ).

0	2	1
4	1	0
1	0	1



## Activity 2 - Answer



##		[,1]	[,2]	[,3]	[,4]
##	[1,]	57	15	15	5
##	[2,]	40	14	8	6
##	[3,]	19	11	10	7
##	[4,]	14	12	7	5

# Convolutional layer: filter

- A filter is:  $Filter = \sum w_i x_i + b_i$  ( a matrix smaller than  $X_i$ )
- Different features will be learnt by different filters.
- When building the network, we randomly specify values for the filters, which then continuously update themselves as the network is trained (to reduce the error).
- It is very very unlikely that two filters that are the same will be produced unless the number of chosen filters is extremely large.


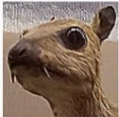
# Example of filters

*Edge detection*


$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$


Kernel

*Sharpen*


$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$


Examples of kernel filters for CNN's.

Figure 13: Two different filters. Image source : <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>

# Convolutional layer

Watch from 43:25-47:30

# RELU (Activation function)

- After a filter have passed over the image, a feature map is generated for each filter (we apply many filters, all different, that will collect different features).
- To add non-linearity, we use the activation function (commonly RELU) to the output of the filters.

**Question:** how did RELU look like? (saw in Lecture 5)

- We can then do a lot of things, such as adding more filtering layers and creating more feature maps, which become more and more abstract as we create a deeper CNN.

# Pooling layer

- Nothing is learnt in this layer (it has no parameters/weights), but it reduces the dimension. Pooling is used to reduce the spatial dimension of the feature map, making the network more computationally efficient and robust to small variations in the input image.
- A problem with the output feature maps is that they are sensitive to the location of the features in the input. Reducing the dimension will help to summarise features.
- Operations in the pooling layers summarise the presence of features in patches of the feature map. For example, we can summarise the output of a feature map (matrix) choosing the average of all numbers in the matrix, or the max.

# Pooling layer

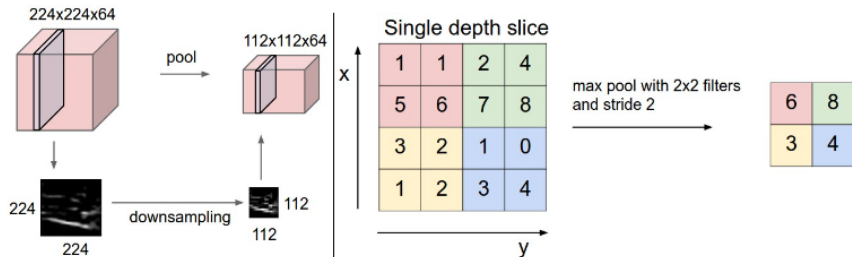


Figure 14: Source in <https://cs231n.github.io/convolutional-networks/>

- Then, we can have a few sets of **Conv + RELU + Pool**

# Fully Connected Layer

- This is the last layer in a CNN.
- This layer takes the output of the previous layer and applies a set of learnable weights to produce a final output.
- This layer is similar to the hidden layers of MLP and is used to map the features learned by the convolutional layers to the final output classes.
- Its output will classify the image into “swan” or “not-swan”.



# Convolutional Neural Network: Summary

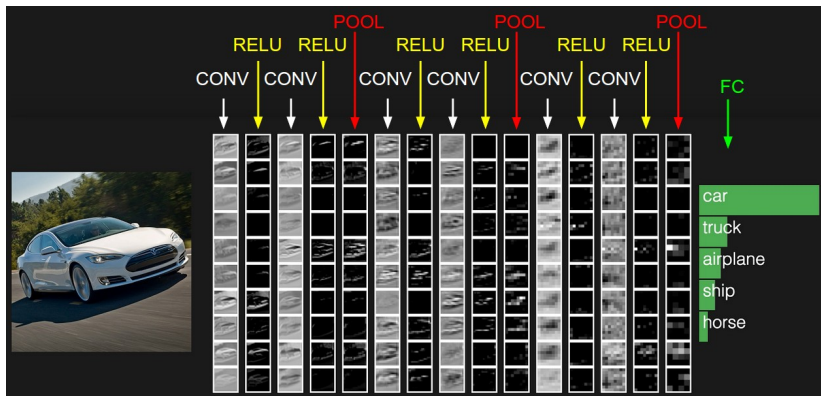
- Convolutional Neural Network (CNN) is a type of deep neural network with three distinctive type of layers: convolutional, pooling and fully connected layers.
- Commonly used to process images.
- How to teach computers to understand pictures, TED Talk, for home watch

# Convolutional Neural Networks

INPUT → CONV → RELU → POOL → FC

Figure 15: CNN with two convolutional layers, 2 pooling layers and a fully connected layer. Image source: <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>

# Example



**Figure 16:** The initial volume stores the raw image pixels (left) and the last volume stores the class scores (right). Each volume of activations along the processing path is shown as a column. Since it's difficult to visualize 3D volumes, we lay out each volume's slices in rows. The last layer volume holds the scores for each class, but here we only visualize the sorted top 5 scores, and print the labels of each one. The full [web-based-demo](https://cs231n.github.io/convolutional-networks/) is shown in the header of our website. The architecture shown here is a tiny VGG Net, which we will discuss later. Image source: <https://cs231n.github.io/convolutional-networks/>

# Example of 2 convolutional layers CNN

- ❶ **Convolutional Layer:** The output of this layer is a feature map, which contains the activations of the filters.
- ❷ **ReLU Activation Layer:** The ReLU activation function is applied to the output of the first convolutional layer to introduce non-linearity into the network.
- ❸ **Pooling Layer** downsamples the feature map by taking the maximum or average of a local patch.
- ❹ **Convolutional Layer:** The second convolutional layer applies another set of filters to the output of the first convolutional layer to extract more complex features.
- ❺ **ReLU**
- ❻ **Pooling Layer.**
- ❼ **Flatten Layer:** This layer flattens the output of the previous layer into a one-dimensional vector to feed into the fully connected layers.
- ❽ **Fully Connected Layer:** This layer produces a final output.
- ❾ **ReLU:** Another ReLU activation function is applied to the output of the fully connected layer to introduce non-linearity into the network.

# Convolutional Neural Network in R

- Algorithms for CNN were first programmed in package *Tensorflow* and *keras* in Python.
- There are versions of these packages in R now, but we have to do the following in our computer:
  - 1 Install Rtools (make sure it is included in the PATH) - this is already done in your computers (our computers in the lab have it)
  - 2 Open R and install package *remotes*
  - 3 Type in R command line:  
`remotes::install_github("rstudio/tensorflow")`, hit *enter* and choose *1. All*
  - 4 Type in R command line: `library(tensorflow)`
  - 5 Type in R command line: `install_tensorflow()`, hit *enter* and say **Y** to install *miniconda*
  - 6 R will restart and after it does we type again: `library(tensorflow)`
  - 7 Type in R command line: `tf_config()`
  - 8 Install *keras* as usual: `install.packages("keras")`

# Build a Convolutional Neural Network in R

Building a CNN consists of three steps:

- ❶ Configuring the layers of the model. We decide the number and type of layers in our network and tell it to the computer using R.
- ❷ Compiling the model. The software makes a general architecture from the blueprint and hyperparameters give in 1.
- ❸ Training the model. Using the architecture above and software, we find the best model through optimisation.

We cannot do this with methods in the *caret* package, we are using functionality in the *keras* packages instead.

We will see how to use them in Lab 5

Learn more about [CNN](#)