

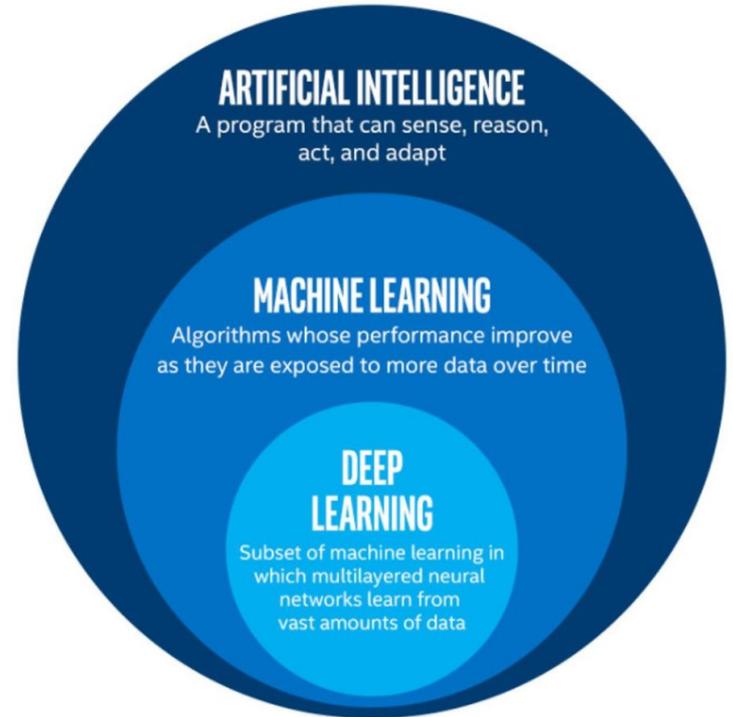


Deep learning introduction



Deep Learning, Machine Learning, and Artificial Intelligence

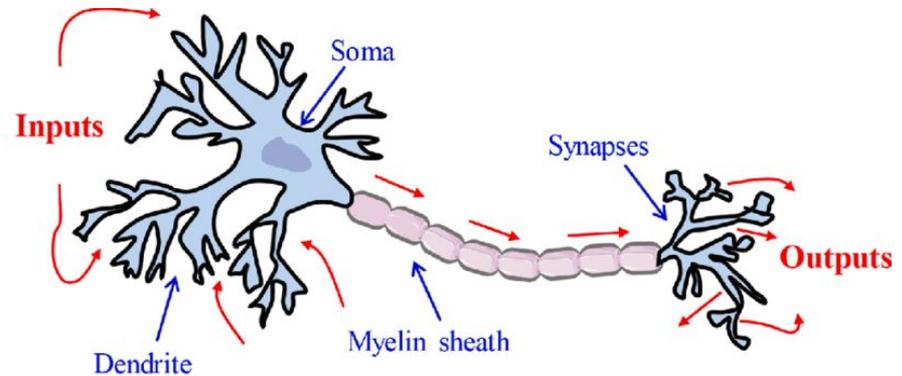
- **Artificial intelligence (AI):** programs mimicking the behaviour of intelligent biological systems
- **Machine learning (ML):** where a computer can “learn” patterns in data, usually by being shown numerous examples to “train” it
- **Deep learning (DL):** one of many techniques collectively known as machine learning





Artificial Neural Networks (ANNs) Are (Loosely) Inspired By The Brain

- **Multiple inputs:** A single neuron (brain cell) receives electrical inputs from many other neurons
- **Input strength:** Some inputs are stronger than others
- **If $\text{sum}(\text{input}) > \text{threshold}$:**
 - Output electrical signal to downstream neuron(s)



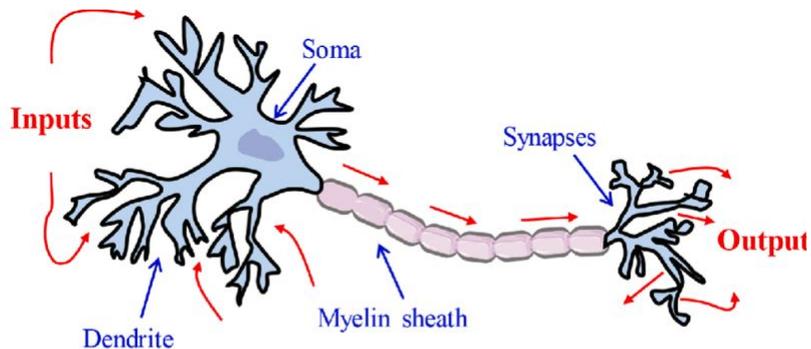
(a) Biological neuron

Image-source: https://www.researchgate.net/figure/Comparison-between-biological-neuron-and-artificial-neuron-40_fig4_351372032

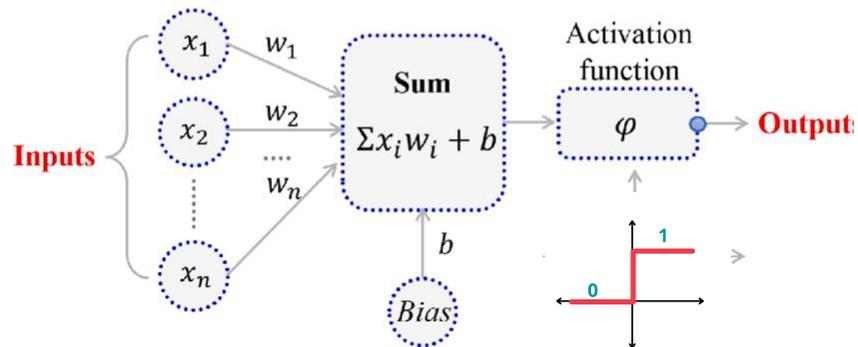


The Perceptron - Single Artificial Neuron

- **The perceptron:** 1st artificial neural network (Rosenblatt, 1959). Consists of one artificial neuron
- Inputs x_i are weighted by w_i — controlling strength of each input
- **Nonlinear activation function:** step fcn
- **Binary output:** - 1/+1 or 0/1
- **Bias term:** Allows us to shift the activation function along the x-axis – changing the threshold of the neuron



(a) Biological neuron



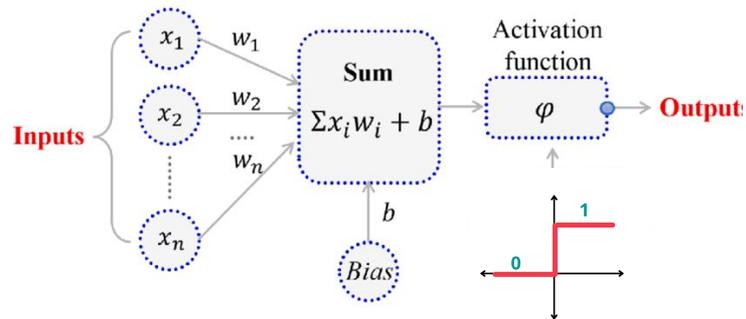
(b) Artificial neuron



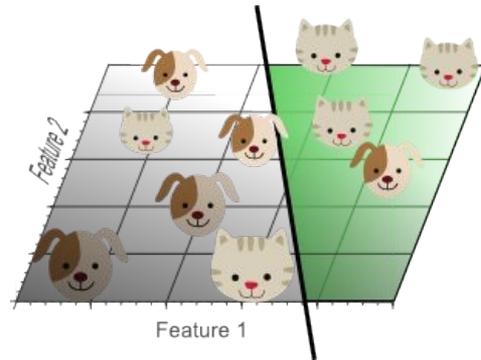
The Perceptron - Binary Classification

- **Inputs:** Inputs are *features* of a tabular dataset

Tail Length (in)	Weight (lbs)	Class
12.2	10.1	Cat (+1)
...
9.1	55.4	Dog (-1)



- **Goal:** Determine weights that allow us to separate two classes based on each class's pattern of input features
- **Method:** Train the model on many observations of cats and dogs, and adjust weights until classification accuracy stops improving





Exercise

 Calculate the output for one neuron

Suppose we have

- Input: $X = (0, 0.5, 1)$
- Weights: $W = (-1, -0.5, 0.5)$
- Bias: $b = 1$
- Activation function *relu*: $f(x) = \max(x, 0)$

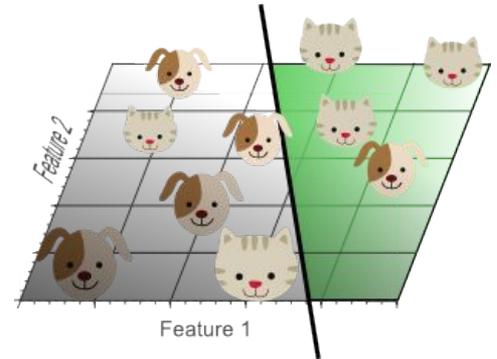
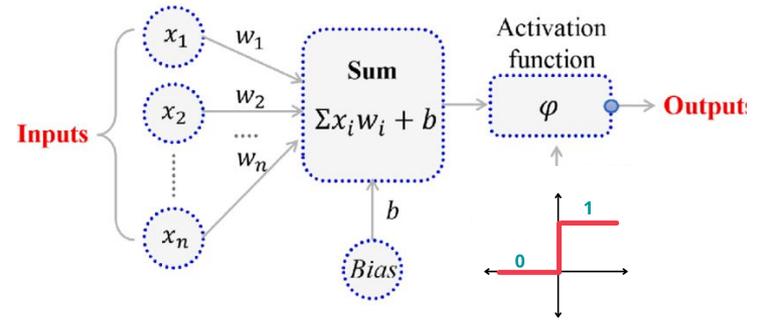
What is the output of the neuron?

Note: You can use whatever you like: brain only, pen&paper, Python, Excel...



Key Limitation of the Perceptron

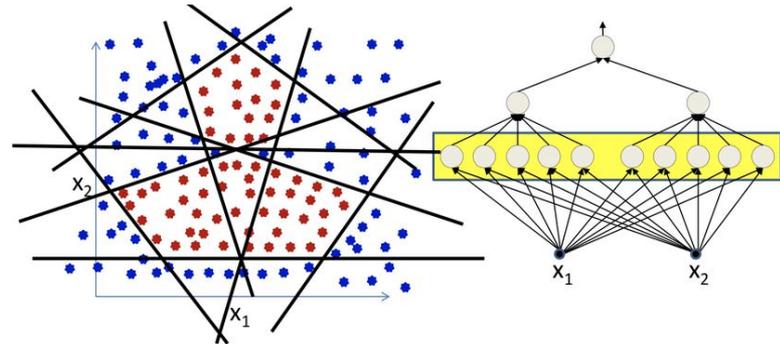
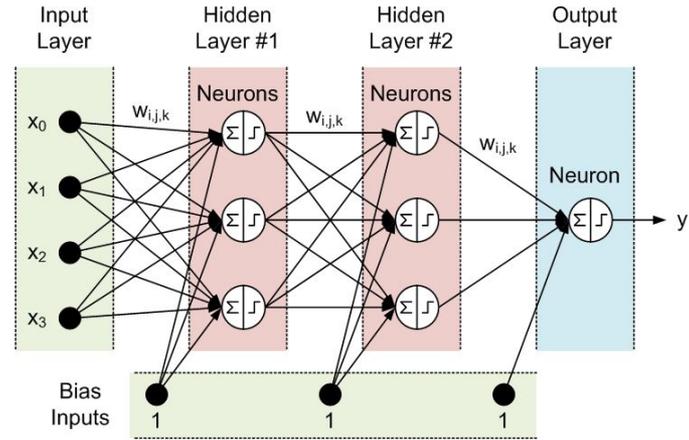
- Perceptrons can only produce **linear classification boundaries**
- A single observation (row in data table) belongs to 1 of 2 classes based on...
 1. weighted sum of inputs
 2. threshold of the activation function (bias weight)
- [Tensorflow playground - perceptron](#)





Multilayer Perceptrons (MLP) - More Advanced ANN

- Multiple neurons arranged in “hidden” layers connected together in a feedforward manner
- Combines multiple linear models together to get a nonlinear model
- **Universal approximation theorem:** An MLP with just one hidden layer of neurons, given enough neurons, is capable of approximating any arbitrary nonlinear function





Why bother with multiple layers?

- Each neuron acts as a pattern recognizer tuned to detect a specific pattern presented in the input data
- Neurons use the outputs of previous layers to build even more sophisticated pattern recognition capabilities (i.e., via pooling lower-level patterns detected)
- Altogether, the network forms what's called a **hierarchical feature representation** of the input data

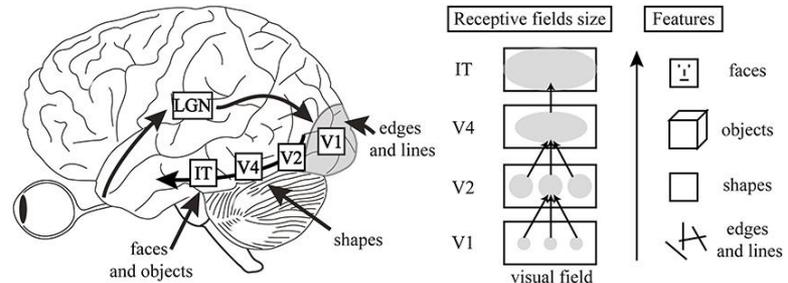
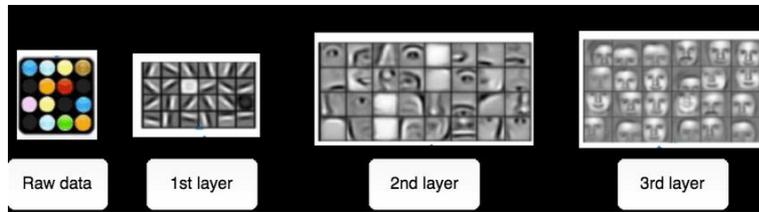


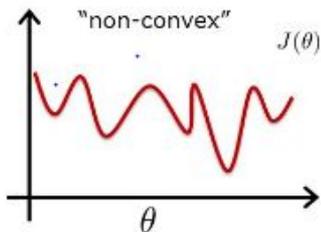
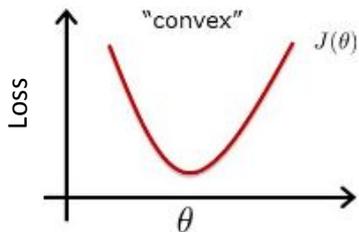
image from:

<https://www.frontiersin.org/articles/10.3389/fncom.2014.00135/full>



Training an MLP

- **Goal:** Learn a classification or regression function from labelled data
- **How:** Minimize the prediction error (loss)
 - Loss function is non-convex (has multiple local minima)



Iterative Optimization Terms

- **Epoch:** One complete pass through the entire training dataset — helps find the lowest loss possible
- **Batch size:** Number of samples used to measure loss before updating the weights



Training a Multilayer Perceptron

Procedure

1. Prepare data
2. Initialize weights randomly
3. Forward propagation: calculate model outputs of $N=batch_size$ training data samples
4. Measure average loss (prediction error) of samples in batch
5. Backpropagation & update weights
6. Repeat 2-4 for multiple epochs
7. Model evaluation using validation data
8. Hyperparameter tuning (adjust number of layers, neurons, learning rate, ect.)
9. Final testing using test data

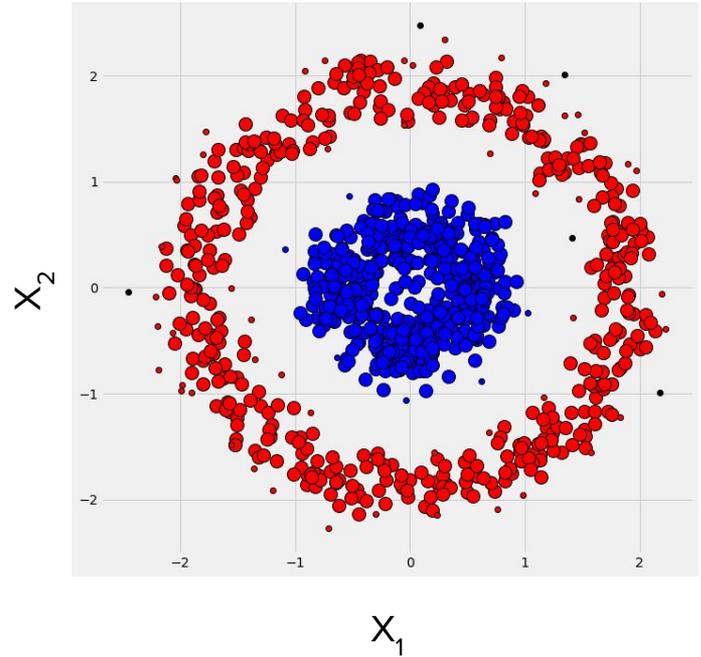
Key Terminology

- **Epoch:** One complete pass through the entire training dataset — helps find the lowest loss possible
- **Batch size:** Number of samples used to measure loss before updating the weights



Exercise

- Given an MLP with 1 hidden layer, how many neurons will be needed in the hidden layer to accurately classify the following data cloud
 - **Red** & **blue** represent two classes
- Test your theory using the [Tensorflow playground](#).





Deep Neural Network (DNN) Applications

- **2010:** improvements in computing power and the algorithms for training the networks made much larger and more powerful networks practical
- **Deep** = 3+ hidden layers (in most contexts). 10+ layers is more common
- Exceptional at approximating many functions

What sort of problems can deep learning solve?

- Pattern/object recognition



DOG



CAT

What sort of problems can deep learning solve?

- Segmenting images (or any data)



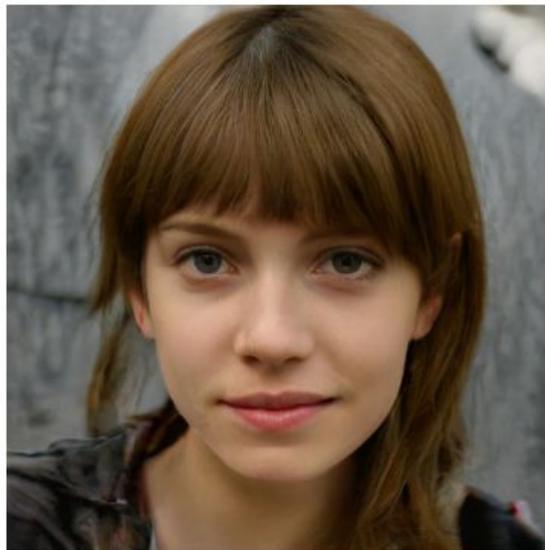
What sort of problems can deep learning solve?

- Translating between one set of data and another, for example natural language translation.



What sort of problems can deep learning solve?

- Generating new data that looks similar to the training data, often used to create synthetic datasets, art or even “deepfake” videos.





What are examples that you have of deep learning applied to research?



When to avoid deep learning

- Lack of abundant data (< 1000). Ideally $> 10,000$ samples.
Data requirements depend on...
 - Problem complexity
 - Data quality & diversity
 - Number of weights
 - Number of layers
- Tasks requiring an explanation of how the answer was arrived at.
- Cases already handled by traditional algorithms



Too little data?

1. Collect more data! ... not always feasible...
2. Find and use a **pretrained network** trained on a similar problem
3. Use **data augmentation** techniques to expand your dataset (e.g., mirroring images when classifying cats vs dogs)
4. Resort to other machine learning models which are still very useful!



Exercise

Which of the following would you apply Deep Learning to?

1. Recognizing whether or not a picture contains a bird.
2. Calculating the median and interquartile range of a dataset.
3. Identifying MRI images of a rare disease when only one or two example images available for training.
4. Identifying people in pictures after being trained only on cats and dogs.
5. Translating English into French



Deep Learning Workflow

1. **Formulate/ Outline the problem** → prediction or classification?
2. **Identify inputs and outputs** → e.g., pixel data and image-labels for image classification
3. **Prepare data** → read in, clean, and sometimes standardize data
4. **Choose a pre-trained model or build a new architecture from scratch**
5. **Choose a loss function and optimizer** → hyperparameter that controls how the model is trained
6. **Train or “fit” the model** → optimize the model’s weights (parameters) by giving it labelled data to learn from
7. **Perform a prediction/classification**
8. **Measure performance**
9. **Tune hyperparameters & Repeat steps 6-8**
10. **Share model**



Exercise

Think about a problem you'd like to use deep learning to solve.

1. What do you want a deep learning system to be able to tell you?
2. What data inputs and outputs will you have?
3. Do you think you'll need to train the network or will a pre-trained network be suitable?
4. What data do you have to train with? What preparation will your data need?
Consider both the data you are going to predict/classify from and the data you'll use to train the network.



Deep Learning Libraries (“Frameworks”)



TensorFlow

- Most popular choice overall, especially in industry & production environments
- Low-level API; tedious to use, but can customize models more
- Versatile and capable of more than deep learning.

 PyTorch

- **Most popular choice in research domains (e.g., NLP, Computer Vision)**
- Lower-level API but can customize models more
- Feels more “pythonic” (numpy w/ GPUs)



Keras



+

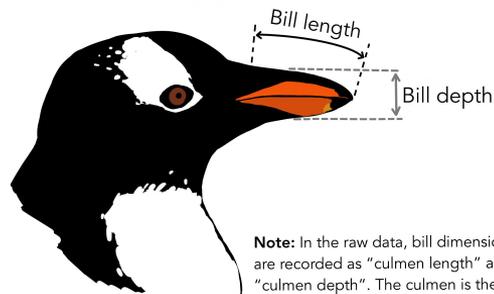
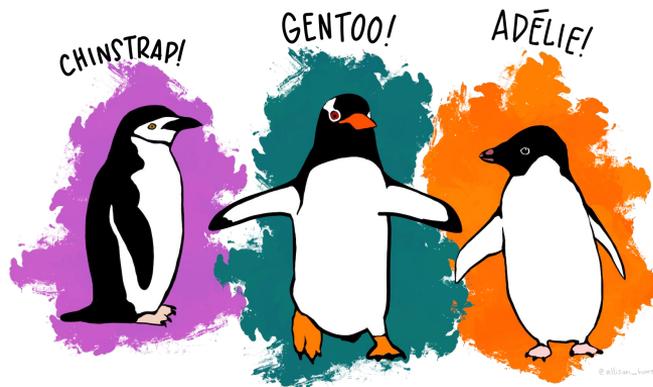


- Keras is not actually a framework on it's own.
- High-level API that sits on top of other supported deep learning frameworks (typically TensorFlow).
 - Currently does not support PyTorch.
- **Save time coding; quickly build models and run experiments**
- Slower to run than Tensorflow and PyTorch
- Difficult to build highly customized models with novel architectures (typically only a problem for ML researchers building new models)



Code Along: Training A Classifier

- Using penguin dataset — published in 2020 by Allison Horst
- Contains data on three different species of the penguins — **Chinstrap**, **Gentoo**, **Adélie**
- Use a DNN to classify penguin species based on their physical characteristics
 - bill length and depth
 - flipper length
 - body mass



Note: In the raw data, bill dimensions are recorded as "culmen length" and "culmen depth". The culmen is the dorsal ridge atop the bill.

Artworks by @allison_horst