Deep Learning Introduction

Deep learning is a machine learning technique that is able to solve complex problems that require discovering hidden patterns in the data and/or a deep understanding of intricate relationships between a large number of interdependent variables. Deep learning algorithms are able to learn hidden patterns from the data by themselves, combine them together, and build much more efficient decision rules.

Deep learning really shines when it comes to complex tasks, which often require dealing with lots of unstructured data, such as image classification, natural language processing, or speech recognition, among others. However, for simpler tasks that involve more straightforward feature engineering and don't require processing unstructured data, classical machine learning may be a better option.

But deep learning has some drawbacks as well. Model interpretability is one of deep learning's biggest challenges. As the model handles very complex problems with high non-linearity and interactions between inputs, they are nearly impossible to interpret. Moreover, deep learning models are very slow to train and require a lot of computational power, which makes them very time- and resource-intensive.

Deep learning are mainly used when:

- Very high accuracy is a priority (and primes over straightforward interpretability and explainability)
- Large amounts of precisely labeled data is available
- Complex feature engineering is required
- Powerful compute resources available (GPU acceleration)
- Augmentation and other transformations of the initial dataset will be necessary

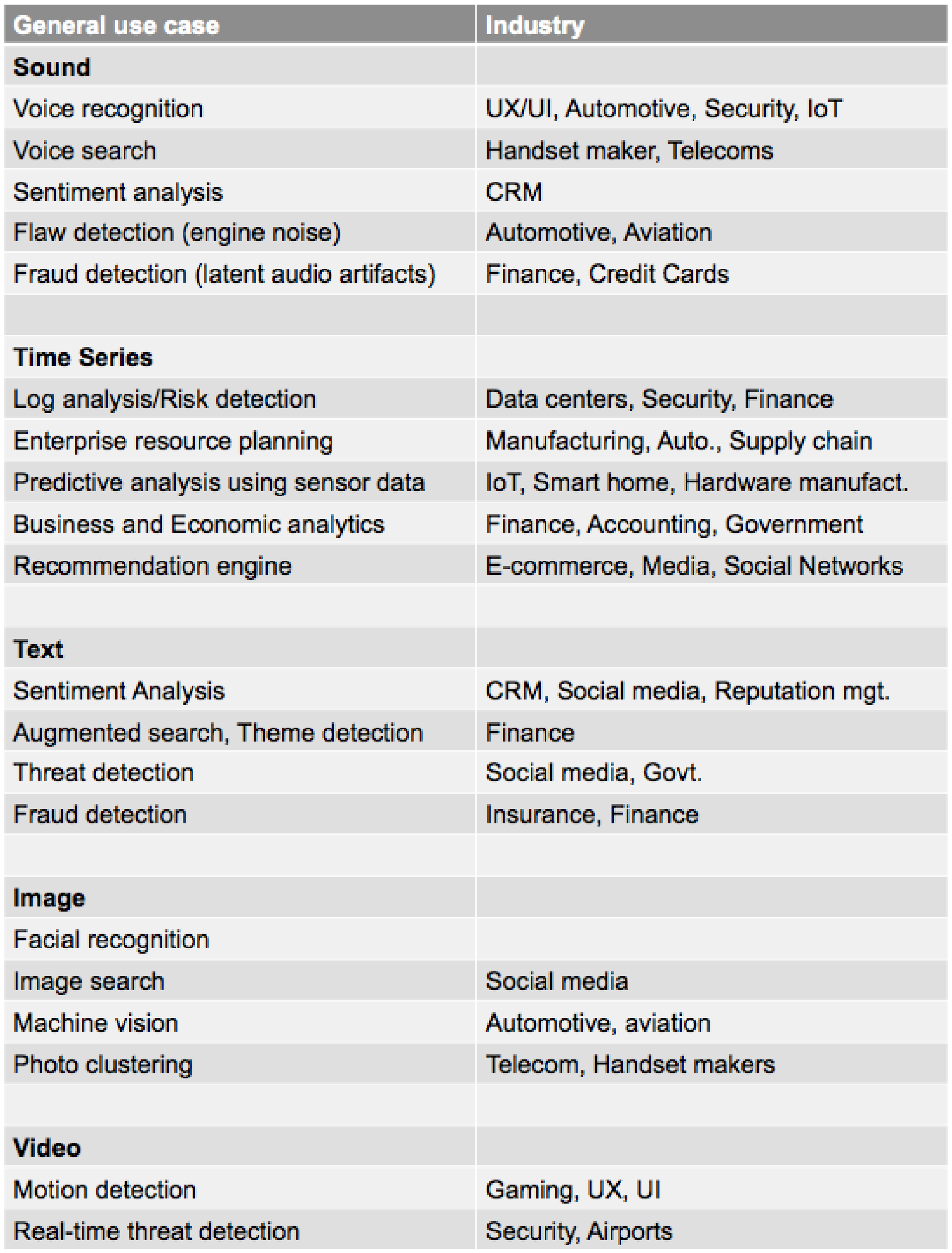
Neural Network

The basic idea behind a neural network is to simulate (copy in a simplified but reasonably faithful way) lots of densely interconnected brain cells inside a computer so you can get it to learn things, recognize patterns, and make decisions in a humanlike way.

A typical neural network has anything from a few dozen to hundreds, thousands, or even millions of artificial neurons called units arranged in a series of layers, each of which connects to the layers on either side. Some of them, known as input units, are designed to receive various forms of information from the outside world that the network will attempt to learn about, recognize, or otherwise process. Other units sit on the opposite side of the network and signal how it responds to the information it's learned; those are known as output units. In between the input units and output units are one or more layers of hidden units, which, together, form the majority of the artificial brain. Most neural networks are fully connected, which means each hidden unit and each output unit is connected to every unit in the layers either side. The connections between one unit and another are represented by a number called a weight, which can be either positive (if one unit excites another) or negative (if one unit suppresses or inhibits another). The higher the weight, the more influence one unit has on another. (This corresponds to the way actual brain cells trigger one another across tiny gaps called synapses.

Use cases of neural network and deep learning

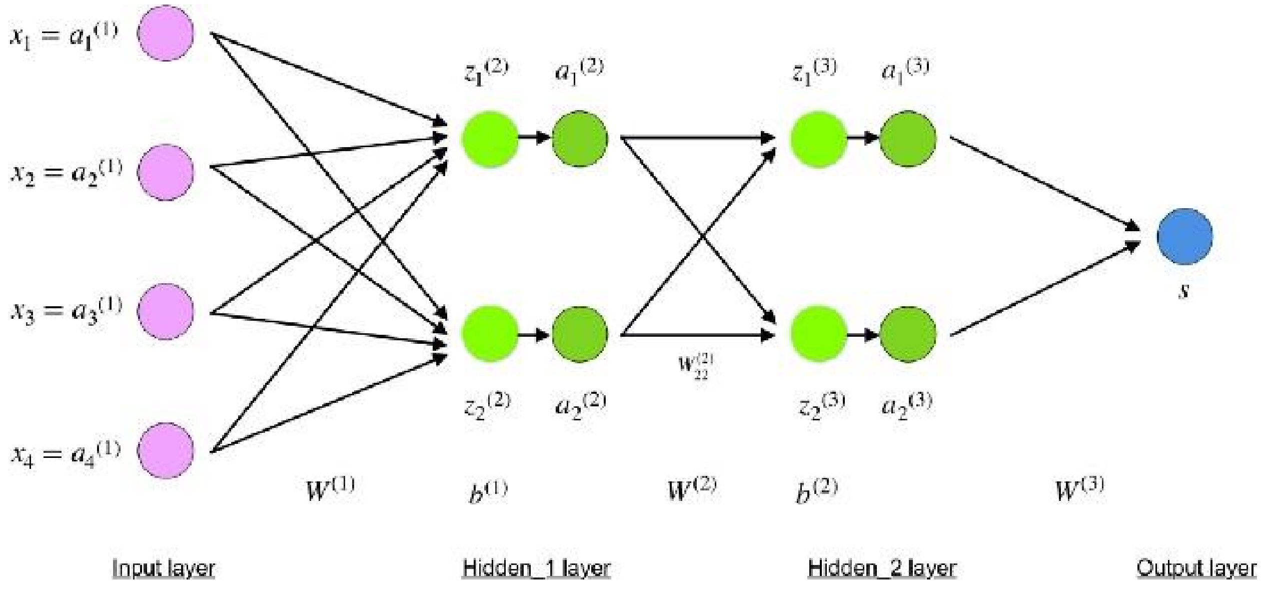
- Automatic speech recognition
- Image recognition
- Natural language processing
- Drug discovery and toxicology
- Recommendation systems
- Bioinformatics



General use case	Industry
Sound	
Voice recognition	UX/UI, Automotive, Security, IoT
Voice search	Handset maker, Telecoms
Sentiment analysis	CRM
Flaw detection (engine noise)	Automotive, Aviation
Fraud detection (latent audio artifacts)	Finance, Credit Cards
Time Series	
Log analysis/Risk detection	Data centers, Security, Finance
Enterprise resource planning	Manufacturing, Auto., Supply chain
Predictive analysis using sensor data	IoT, Smart home, Hardware manufact.
Business and Economic analytics	Finance, Accounting, Government
Recommendation engine	E-commerce, Media, Social Networks
Text	
Sentiment Analysis	CRM, Social media, Reputation mgt.
Augmented search, Theme detection	Finance
Threat detection	Social media, Govt.
Fraud detection	Insurance, Finance
Image	
Facial recognition	
Image search	Social media
Machine vision	Automotive, aviation
Photo clustering	Telecom, Handset makers
Video	
Motion detection	Gaming, UX, UI
Real-time threat detection	Security, Airports

Forward and Backward pass in a Neural Network

Let us consider a neural network having input, output and 2 hidden layers. Both hidden layers have two neurons and the size of input is 4 and output is one.



Input layer is given as

$$x_i = a_i^{(1)}, i \in 1,2,3,4$$

For first hidden layer output is calculated as

$$z^{(2)} = W^{(1)}x + b^{(1)}$$

$$a^{(2)} = f(z^{(2)})$$

For second hidden layer output is calculated as

$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$

$$a^{(3)} = f(z^{(3)})$$

W^2 and W^3 are the weights in layer 2 and 3 while b^2 and b^3 are the biases in those layers.
Activations a^2 and a^3 are computed using an activation function f. Typically, this function f is non-linear (e.g. sigmoid, ReLU, tanh)

Final output is calculated as

$$s = W^{(3)}a^{(3)}$$

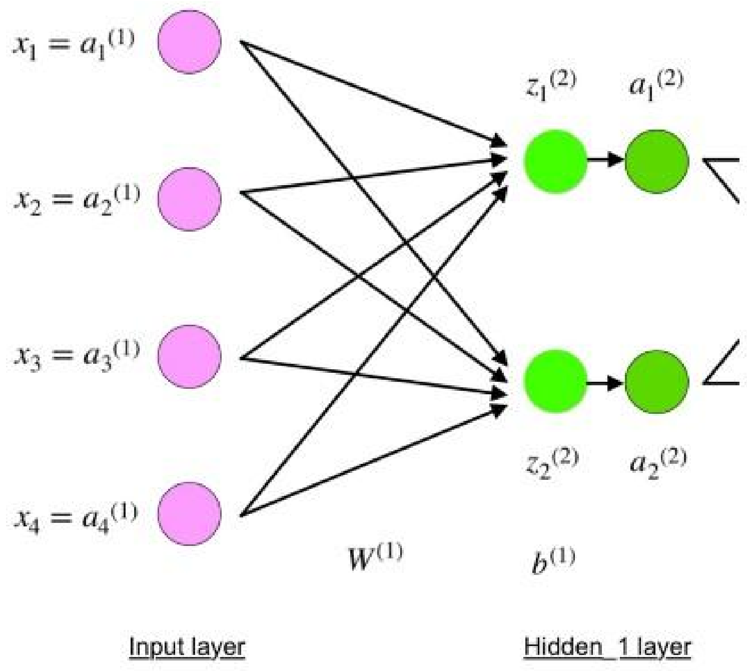
The weight matrix for one layer is given as

$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} & W_{14}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} & W_{24}^{(1)} \end{bmatrix}$$

First row represents weights for the first layer (weights of edges connecting input layer with first hidden layer) .
First row first column weight is for edge connecting first neuron for hidden layer1 and first neuron of input layer

Equation for output of layer 2 (first hidden layer)

$$z^{(2)} = \begin{bmatrix} W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + W_{14}^{(1)}x_4 \\ W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + W_{24}^{(1)}x_4 \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}$$



$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \end{bmatrix}$$

Formula for forward propagation

$x = a^{(1)}$ *Input layer*

$z^{(2)} = W^{(1)}x + b^{(1)}$ *neuron value at Hidden$_1$ layer*

$a^{(2)} = f(z^{(2)})$ *activation value at Hidden$_1$ layer*

$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$ *neuron value at Hidden$_2$ layer*

$a^{(3)} = f(z^{(3)})$ *activation value at Hidden$_2$ layer*

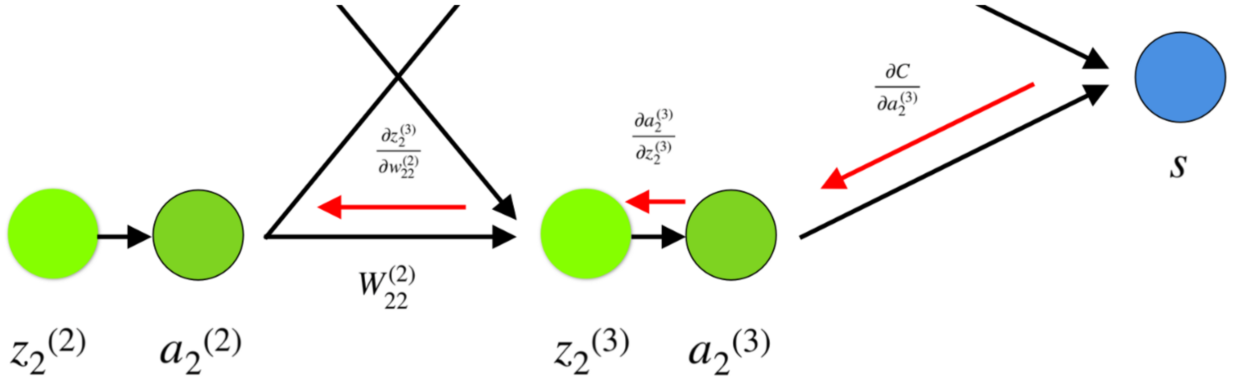
$s = W^{(3)}a^{(3)}$ *Output layer*

Cost function is calculated as

$$C = cost(s, y)$$

where y is the actual output and s is predicted output

For Backpropagation we calculate gradient for cost function using chain rule of derivatives



$\frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}}$
$\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}}$
$\frac{\partial C}{\partial a_2^{(3)}}$
s
$W_{22}^{(2)}$
$z_2^{(2)}$
$a_2^{(2)}$
$z_2^{(3)}$
$a_2^{(3)}$

$$\frac{\partial C}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial z_2^{(3)}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial a_2^{(3)}} \cdot \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \, . \, a_2^{(2)} = \frac{\partial C}{\partial a_2^{(3)}} \, . \, f'(z_2^{(3)}) \, . \, a_2^{(2)}$$

We calculate the derivative of the cost function wrt to every weight in the neural network.
The gradient shows how much the parameter W needs to change (in positive or negative direction) to minimize C.

The weights and baisas is each optimization steps are updated as

```
while (termination condition not met)
    $w := w - \epsilon \frac{\partial C}{\partial w}$
    $b := b - \epsilon \frac{\partial C}{\partial b}$
end
```