Machine Learning W11 Tutorial

COMP30027 | Sandy Luo

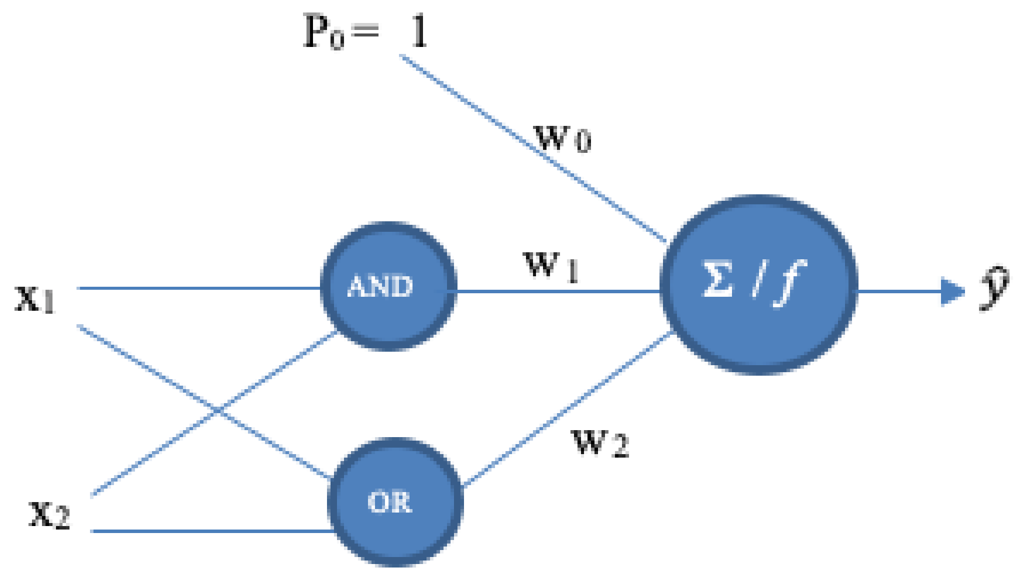
Overview

Neural Networks

Concept, code

Q1:

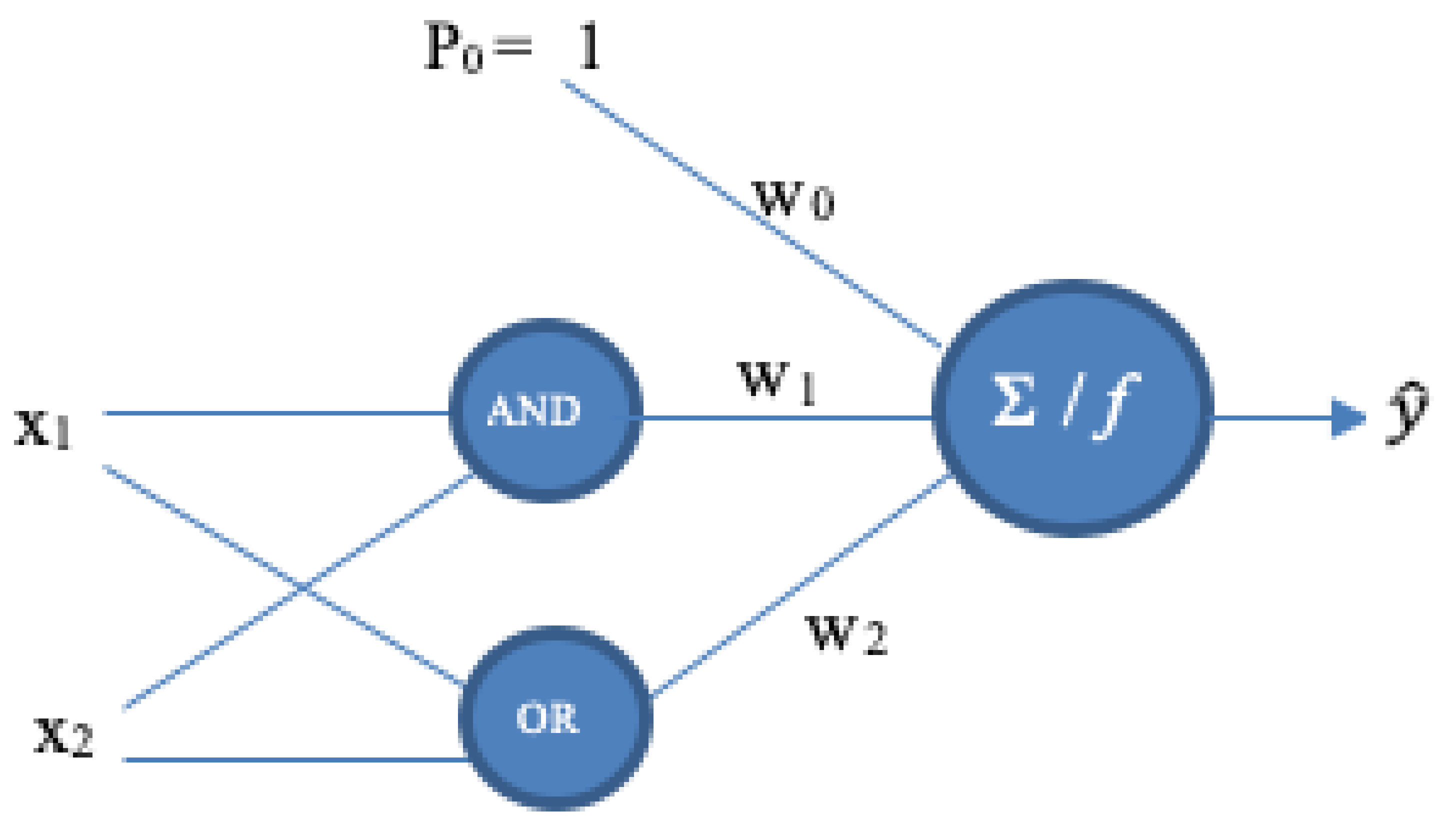
This two layers deep network is composed of three perceptrons. The two perceptrons of the first layer implement the AND and OR functions, respectively.



Determine the weights w_1, w_2 and bias (w_0) for the layer 2 perceptron such that the network implements the XOR function.

Q1:

Determine the weights w_1, w_2 and bias (w_0) for the layer 2 perceptron such that the network implements the XOR function.

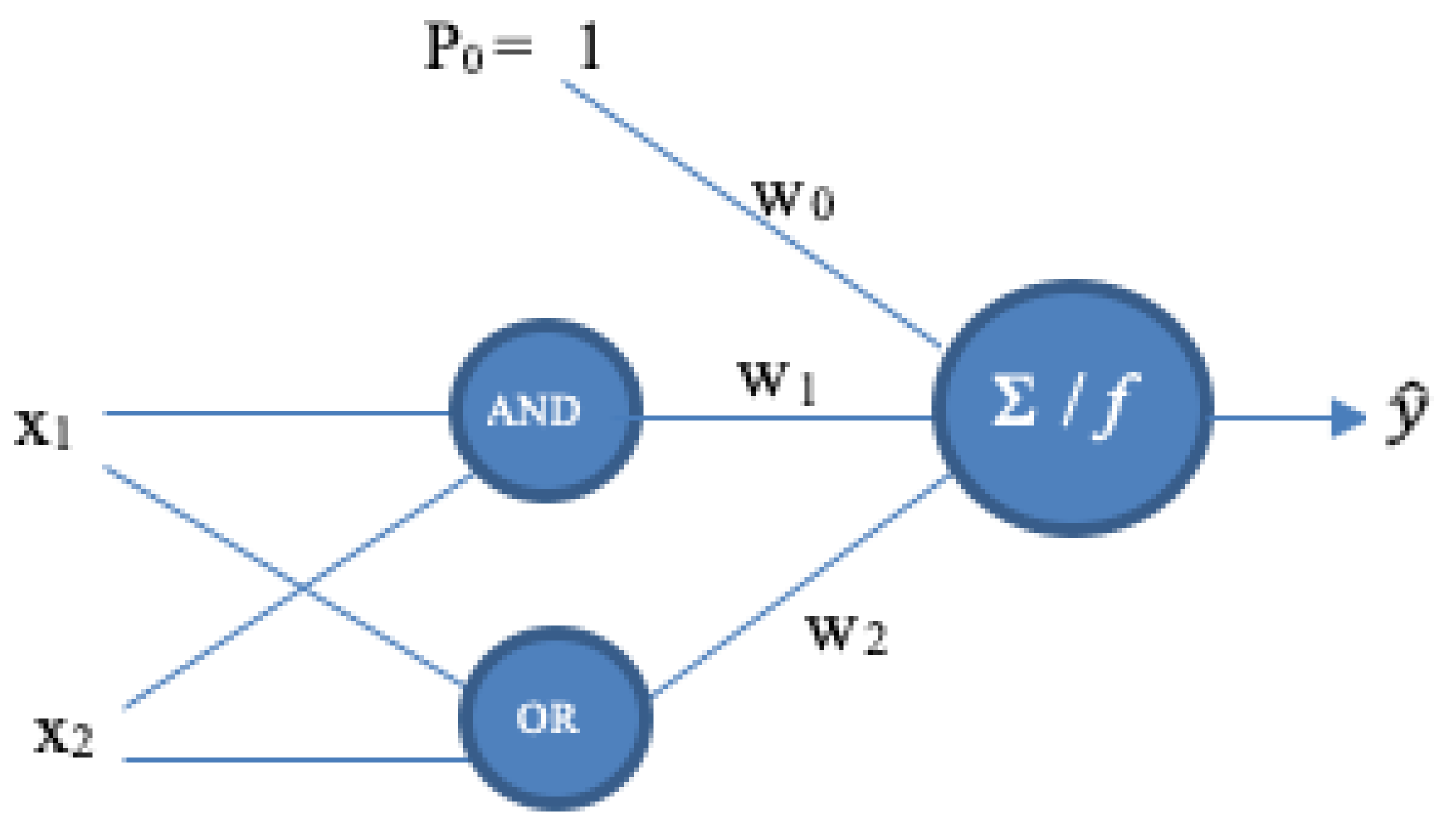


- The input function for the perceptron on layer 2 = weighted sum of its input.
- The activation function *f* for the perceptron on layer 2 is a step function.
- Assume the weights for the layer 1 perceptrons are given.

Q1:

- Input: P = <1, p1, p2>, W = <w0, w1, w2>

x_1	x_2	p_1 $f_{AND}(x)$	p_2 $f_{OR}(x)$	y $x_1\ XOR\ x_2$
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0
0	0	0	0	0



ep	*P* $<p_0, p_1, p_2>$	z = w . *P* $w_0 \times p_0 + w_1 \times p_1 + w_2 \times p_2$	$\hat{y}$ *f(z)*	*y*	Δw_{1i} $\lambda\,(y - \hat{y})\,p_i$

Q2(a):

Neural networks are used for representation learning; the representations (or "embeddings") learned by neural networks trained on one task are often useful for a variety of other tasks.

- **What are the features of neural networks that make them particularly suitable for representation learning?**

Q2(a):

What are the features of neural networks that make them particularly suitable for representation learning?

- **Hierarchical Representation Learning**
- **Non-Linearity**: Non-linear activation functions
- **Automatic Feature Extraction**
- **End-to-End Learning**: Can directly map input data to output predictions without relying on explicit intermediate steps.
- **Scalability**

Q2(b):

Neural networks are used for representation learning; the representations (or "embeddings") learned by neural networks trained on one task are often useful for a variety of other tasks.

- **What is a "neural network embedding" and how is it useful in machine learning?**

Q2(b):

<u>What is a "neural network embedding" and how is it useful in machine learning?</u>

- Learned, dense, low(er) dimensional, representation of data created by NNs
  - Makes features high-level patterns! → Dimensionality reduction
  - Captures similarity
- Input for downstream tasks
  - Allows for transfer learning

Q3:

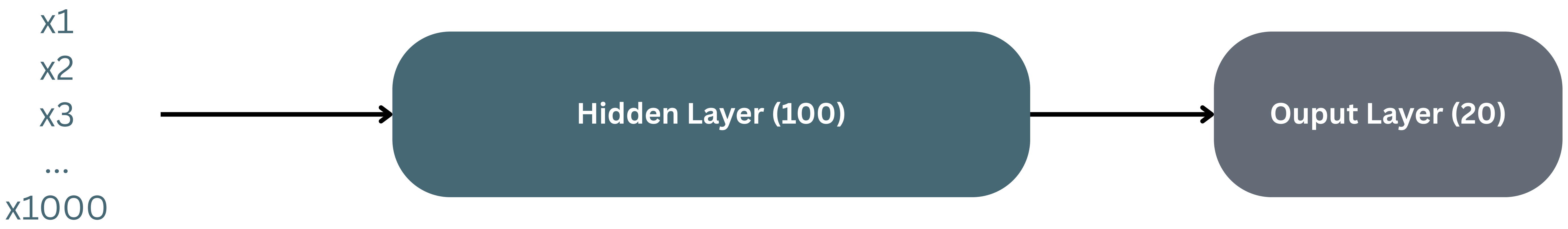
Calculate the number of trainable parameters in a multilayer perceptron with one hidden layer. Assume:

- the input size is 1000
- the hidden layer size is 100
- the output layer size is 20

Q3:

Calculate the number of trainable parameters in a multilayer perceptron with one hidden layer. Assume:

- the input size is 1000
- the hidden layer size is 100
- the output layer size is 20

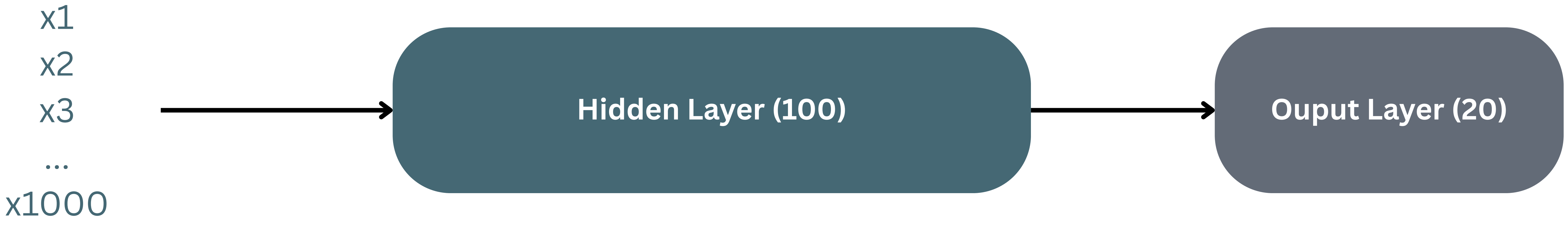


Q3:

Calculate the number of trainable parameters in a multilayer perceptron with one hidden layer. Assume:

- the input size is 1000
- the hidden layer size is 100
- the output layer size is 20

(1000 weights + 1 bias) * 100 layers +
(100 weights + 1 bias) * 20 layers =
100100 + 2020 = 102120



(1000 weights + 1 bias) * 100 neurons

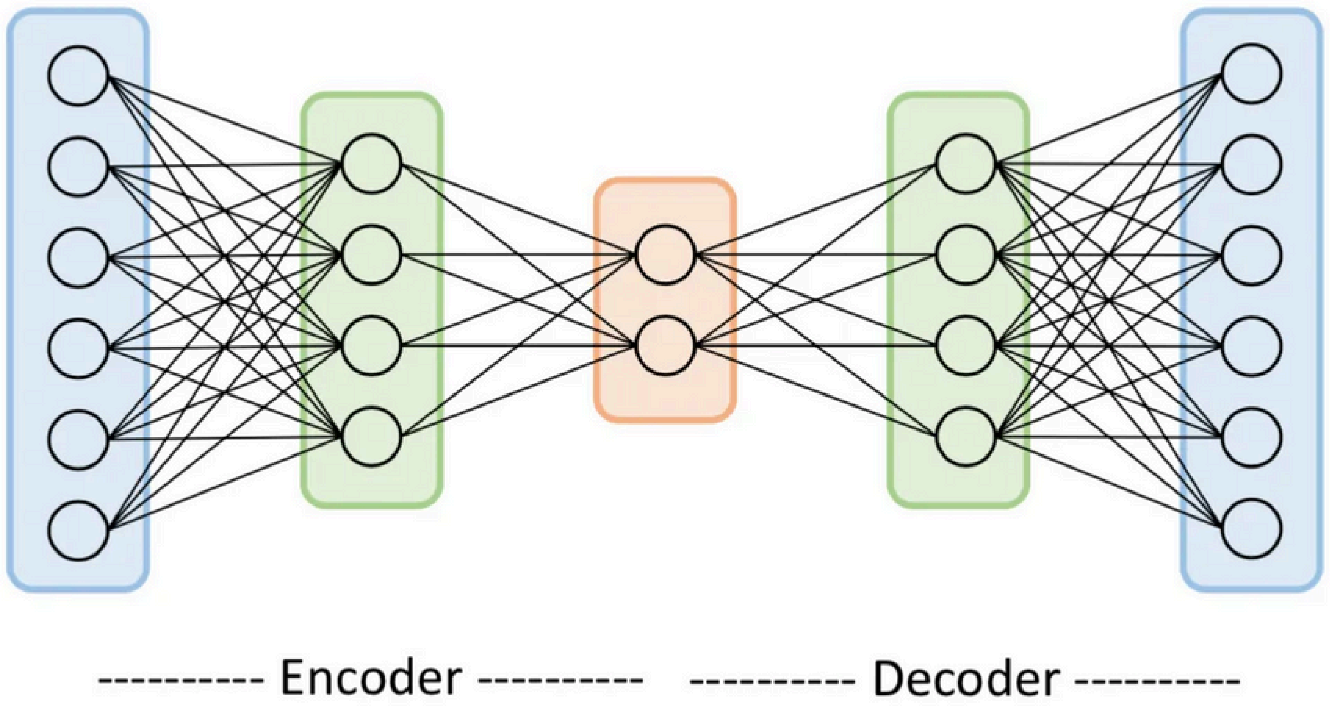
(100 weights + 1 bias) * 20 neurons

Q4:

What is an **autoencoder** and how is it trained?

Q4:

What is an **autoencoder** and how is it trained?



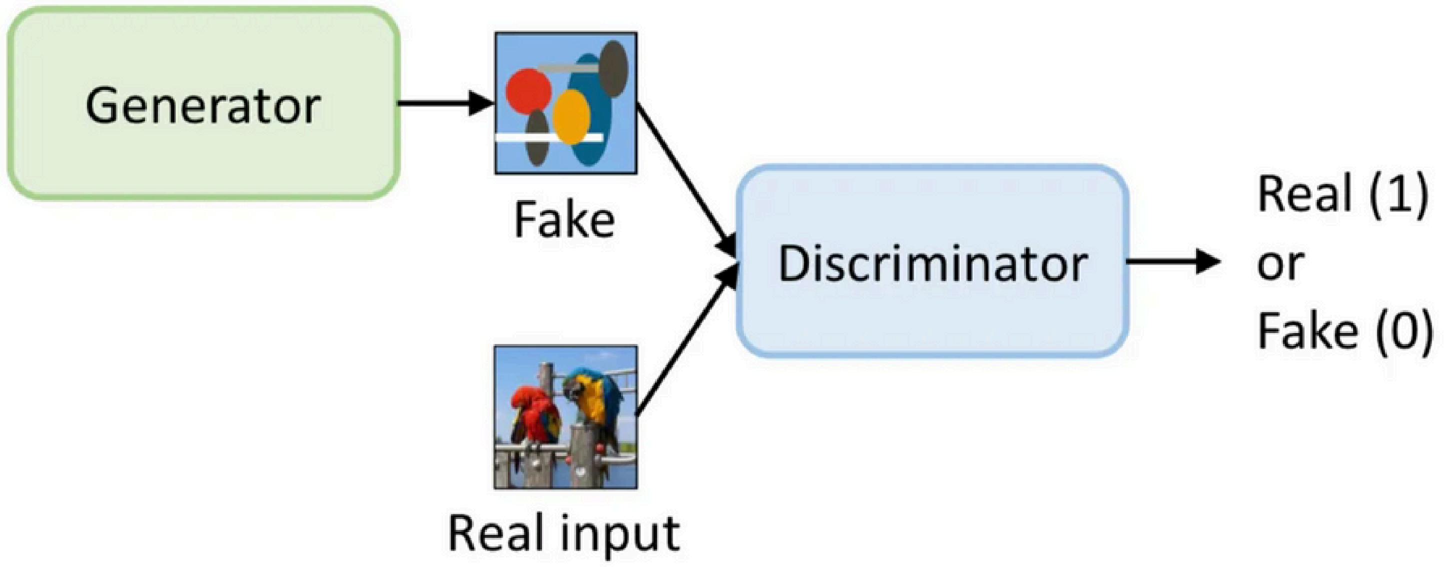
- NN for unsupervised learning
- Ouput in the same form as the input → e.g. image
- Hidden layer → lower-dimensional (latent) representation of input
- Training Objective:
  - Minimise the difference between the input and its reconstruction

Q5:

How does the training process of a **Generative Adversarial Network** (GAN) differ from traditional supervised learning?

Q5:

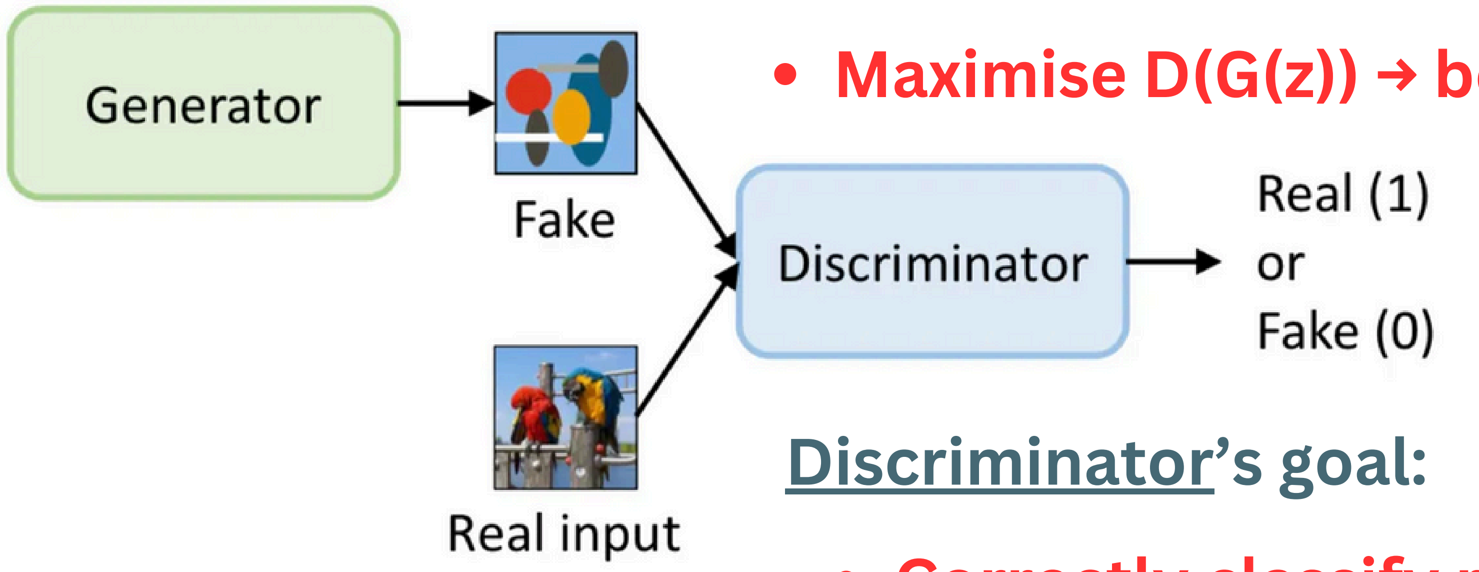
How does the training process of a **Generative Adversarial Network** (GAN) differ from traditional supervised learning?



- **Consists of two neural networks:**
  - **Generator**
  - **Discriminator**
- **Training → competition b/w the two networks**
  -

Q5:

How does the training process of a **Generative Adversarial Network** (GAN) differ from traditional supervised learning?



Generator's goal:

- **Fool the discriminator**
- **Maximise D(G(z)) → better fakes**

Discriminator's goal:

- **Correctly classify real vs fake**
- **Maximise D(x) and minimise D(G(z)) → distance b/w real/fake**

Can treat as a zero-sum game w/ goal of finding equilibrium b/w G & D