A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

18.1 Artificial Intelligence (AI)

Name: _____ Class: _____ Date: _____ **Total: 17 marks**

Objective

Build the skills to answer exam questions on **Artificial Intelligence** 人工智能—graphs in AI, neural networks, and the types of machine learning.

You must be able to:

- explain how **graphs** help AI and how an **artificial neural network** works
- compare **supervised**, **unsupervised** and **reinforcement** learning
- explain **deep learning** and **backpropagation**

1 Worked examples

Study these first. Each one shows the method for a question type used later —follow the steps and you can do the Practice and Exam-style questions yourself.

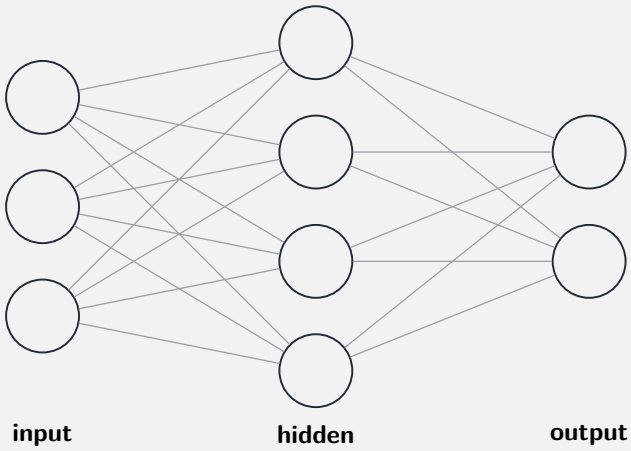
■ AI, ML and graphs

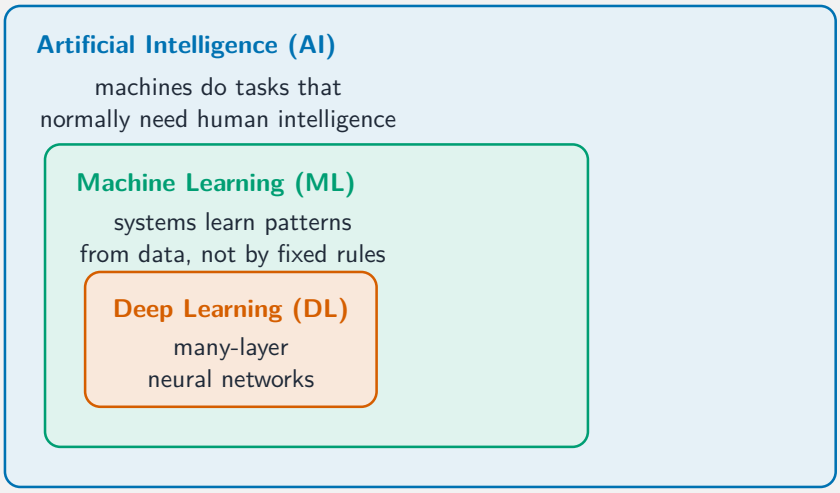
Machine learning (ML) learns patterns **from data** instead of fixed rules; **deep learning (DL)** is ML using deep neural networks. Many AI problems sit on a **graph** —**nodes** (states/places) joined by **edges** (moves, each with a cost). Two algorithms find the **shortest path**:

- **Dijkstra's algorithm** explores outward from the start, always extending the lowest-cost node, until it reaches the goal.
- **A*** does the same but adds a **heuristic** estimate of the distance **still to go**, which steers the search **toward the goal** —so it usually checks **fewer nodes** and is faster.

■ Neural networks

A network has an **input** layer, one or more **hidden** layers, and an **output** layer. Each connection has a **weight**; a neuron sums its weighted inputs (plus a bias), applies an **activation function**, and passes the result on:





How AI, machine learning and deep learning relate

Types of machine learning

- **supervised** —data has **labels**; learns input → label (classification, regression).
- **unsupervised** —**no labels**; finds structure (e.g. **clusters**).
- **reinforcement** —an **agent** acts, gets a **reward**, and learns a **policy** that maximises total reward by trial and error.

Training adjusts the weights by **backpropagation**: a forward pass computes the output, a **loss function** measures the error, and the error is propagated **backwards** to update each weight (gradient descent). Deep learning needs **lots of data and compute**.

2 Practice

Now apply the methods above.

2.1 State what **machine learning** is. [1]

2.2 Name the **three** kinds of layer in a neural network. [2]

2.3 In a graph used for **pathfinding**, state what an **edge** represents. [1]

2.4 State what a **weight** is in an artificial neuron. [1]

3 Exam-style questions

3.1 Describe the difference between **supervised** and **unsupervised** learning. [2]

3.2 Explain how **reinforcement learning** works, using the terms **agent**, **reward** and **policy**. [3]

3.3 Explain why **deep learning** needs a large amount of **data** and **computing power**. [2]

3.4 State the role of **backpropagation** in training a neural network. [2]

3.5 State the purpose of **Dijkstra's algorithm** and **A***, and outline **one** difference between them. [3]

4 Go further

You are now ready for the real exam questions on this subtopic. Open the **18.1 Artificial Intelligence** past-paper sheet in the Library, or try these in **Practice mode**:

- 9618/32 J25 —Q6 (types of machine learning)
- 9618/31 N25 —Q7 (graphs and pathfinding)
- 9618/33 J25 —Q10 (neural networks and deep learning)
- 9618/12 J25 —Q3 (AI use cases)

Solutions

2.1 Algorithms that **learn patterns from data** (and improve with experience) instead of being programmed with explicit rules.

2.2 Input layer, **hidden** layer(s), **output** layer.

2.3 A **move/connection** between states (e.g. a road between two places), often with a **cost/weight**.

2.4 A value each input is **multiplied by**, setting how strongly that input affects the neuron's output.

3.1 Supervised learning trains on **labelled** data (each input has a known correct output); **unsupervised** learning uses **unlabelled** data and finds structure/clusters itself.

3.2 An **agent** takes an **action** in an environment, which changes the state and returns a **reward**; over many trials it learns a **policy** (strategy) that **maximises total reward** by trial and error.

3.3 Deep networks have **very many weights** to learn, so they need large datasets to learn good patterns and heavy compute (GPUs) to train in reasonable time.

3.4 It propagates the output **error backwards** through the layers to **adjust each weight** so the network's output gets closer to the target (gradient descent).

3.5 Both find the **shortest path** between two nodes in a weighted graph. Difference: **A*** uses a **heuristic** to estimate the remaining distance and head toward the goal, so it usually explores **fewer nodes / is faster**; **Dijkstra's** has no heuristic and explores outward uniformly.