ECS759P Artificial Intelligence
Coursework 2

This coursework contributes **50% of your final module grade**.

Deadline: 9th of January 2026, at 10AM, on QM+

This deadline will NOT BE CHANGED. Extensions are only granted through the official EC process. Late submissions will receive a grade of zero.

Communication and Support: If you have questions about this coursework, please post them on the **Student Forum on QMplus**. Using the forum allows all students to benefit from clarifications and official responses. You may also ask questions during your assigned lab sessions. For individual inquiries, you can contact **Dr. Iran R. Roman** during **office hours (Friday, 10–11am, Peter Landin 4.16)** or by email at i.roman@qmul.ac.uk.

Submission Instructions: Your final submission **must include the following files**:

- `report.pdf` — contains all written answers, explanations, derivations, and figures.
- `cw2.py` — contains all code implementations required for this coursework.
- `best_skipgram_523words.pth` — your Skip-gram model including the 523-word vocab.
- `best_cifar100_projection.pth` — your trained CIFAR-100 projection / embedding model.

If you reuse code from the lab sessions, you must also include the corresponding `.py` lab files (e.g. `lab1.py`, `lab2.py`, etc.) in your submission. IMPORTANT: do NOT submit jupyter notebooks (DO NOT submit files with extension `.ipynb`). We will not look at those at all. Failure to include all required files, correctly named and in the proper format, will result in your submission being automatically flagged as **INCOMPLETE** and an automatic score of ZERO.

Your complete submission should be within a single `.zip` file.

Coding Guidelines:

- You **must not use external libraries** beyond those used in the labs.
- Your code should be clear, well-documented, and modular.
- All required functions must be defined using the exact names and signatures specified in the starter code in `cw2.py`

Report Guidelines:

- Your `report.pdf` should include all required written answers, figures, and explanations.
- Answers must be clearly labeled to correspond to the correct question number.
- You may use any typesetting software (e.g. LaTeX, Overleaf, or Word). Using LaTeX is recommended. If you prefer to write answers by hand, take clear, legible photos of your work and combine them into a single PDF file named `report.pdf`.

Coursework structure: 6 main Tasks (total 100 marks):

1. **Strategic Game Analysis** — 10 marks

2. **Decision Tree Design with ID3** — 15 marks

3. **Neural Network Mathematics & Dynamics** — 5 marks

4. **Planning Logic** — 10 marks

5. **CIFAR-100 Semantic Expansion** — 30 marks

6. **Neuro-Symbolic AI: Multi-Modal Planning** — 30 marks

Assessment Criteria:

- **Correctness:** Does your code produce accurate and reliable results?
- **Design Quality:** Are your algorithms well-structured, efficient, and logically sound?
- **Clarity:** Are your explanations and justifications concise, coherent, and technically clear?
- **Presentation:** Is your report professionally formatted, well-organized, and easy to read?

Academic Integrity: All submitted work must be your own. Collaboration on conceptual understanding is encouraged, but direct sharing of code or written text is strictly prohibited. Any detected plagiarism or code similarity beyond what is expected for independent work will be reported and handled according to the **Queen Mary Academic Misconduct Policy**.

IMPORTANT NOTE ABOUT YOUR USE OF AI AND LLMs

The use of Generative Artificial Intelligence tools (such as **ChatGPT**, **Claude**, **GitHub Copilot**, **Gemini**, or other large language models, LLMs) is **permitted and encouraged** in this coursework, provided that it is used **critically, transparently, and responsibly**.

If you use AI tools in any part of your work (including code or report writing), you must:

- Mention FULL **name and version** of the tool (e.g., "ChatGPT, GPT-5, October 2025").
- Add a brief **note or comment** in your report or code indicating which parts were assisted or generated by AI.
- At the end of your report, summarize in less than 200 words **how you used the AI tool** (for example: to rephrase text, debug code, explore alternative ideas, your prompting style/technique).

You do **not** need to include every individual prompt or an extensive log of interactions. A short, honest statement of use is enough. The goal is **transparency**, not extra workload. **Full transparency** is required to receive credit for AI-assisted work. Using AI without clear attribution or explanation will be treated as academic misconduct and will result zero.

While the use of AI is allowed, you are still responsible for the **accuracy, originality, and quality** of all submitted content. The ability to critically evaluate, adapt, and improve AI-generated outputs will be considered a mark of strong performance.

1 Strategic Game Analysis (Total marks: 10)

You MUST show all work and provide detailed step-by-step calculations or reasoning. **Only complete and concise solutions with sound justification will earn marks.**

Part A: Utility Calculations for Strategic Market Entry [4 marks]

Three companies (A, B, and C) are considering entering a new market. They may launch independently, form coalitions, or abstain. Utilities and costs:

- Launching alone while others also launch: utility = 50.
- Launching alone while others abstain: utility = 80.
- Two-company coalition: combined utility = 120, split evenly.
- Three-company coalition: combined utility = 150, split evenly.
- Solo launch cost = 20. Coalition cost = 10 per company.

Calculate the expected utility for each company under all possible launching scenarios.

Part B: Nash Equilibria [2 marks]

Determine whether any Nash equilibria exist within this strategic setting.

Part C: Dominant Strategies [2 marks]

Identify whether any company possesses a dominant strategy.

Part D: Shapley Value for Grand Coalition [2 marks]

If all three companies form a grand coalition, compute the Shapley value for each firm.

2 Decision Tree Design with ID3 (Total marks: 15)

In this question you will design a **classification problem** that reflects a realistic scenario. First you will construct a small labelled and "entropically interesting" dataset. Then you will apply the ID3 algorithm to build a decision tree classifier.

Your answer must be **narrative, mathematical, and graphical**, at a level of detail comparable to the worked examples from the lectures (including full entropy and information gain calculations, and a carefully drawn coloured decision tree).

Part A: Scenario and Features [3 marks]

Choose a real-world inspired classification task. Describe the scenario in a short narrative clearly explaining what the input variables represent in the real world and what the target class label means. Define at least **four** input features, and one binary categorical "class" label. For each feature in the dataset, describe the type of values it can take. Do this using 200 words or less.

Finally, construct a table with at least **20 training examples** (and certainly less than 40), with each row showing the feature values and the corresponding class label.

Part B: Entropy and Root Information Gain [3 marks]

Let S denote your full training set.

Compute the class entropy $H(S)$ using base-2 logarithms, showing the class counts, the corresponding probabilities, the entropy formula, and the final numerical value rounded to 4 decimal places.

For each feature, do a full analysis of the conditional entropy of the split and the corresponding information gain, writing out all intermediate steps.

Conclude this part by clearly stating which feature has the highest information gain and therefore should be chosen as the **root node** of your decision tree.

Part C: Recursive Splitting [4 marks]

Starting from your chosen root feature, apply the ID3 procedure recursively and exhaustively. For each subset created by the root split, show:

- The remaining training examples in that subset
- The entropy calculation for that subset
- Information gain calculations for all remaining features
- The feature selected for the next split (or declaration of a leaf node)

Continue this process until all branches terminate in leaf nodes. Show all mathematical work.

Part D: Coloured Decision Tree Diagram [3 marks]

Create a clean, final diagram of your decision tree using either a drawing tool or a hand-drawn figure that you scan or photograph.

Use colours and shapes to distinguish between decision nodes and leaf nodes, similar in style to the coloured trees shown in the lecture examples.

Include this diagram in your `report.pdf` with a caption explaining what the colours and shapes represent. Do this using 150 words or less.

Part E: Narrative Explanation and Classification of Test Examples [2 marks]

Write a concise narrative explaining how your decision tree makes predictions in your chosen scenario, referring explicitly to the features and splits you designed (for example, "If the outlook is sunny and the temperature is below 18°C, then the tree predicts . . . ").

Provide at least **two** new test instances (feature vectors not present in your training table) and, for each one, trace the path from the root to a leaf, step by step, indicating at each node which branch is followed and what final class prediction is made. Comment on your model's performance on these two new test instances.

Briefly discuss two limitations or potential issues of your decision tree model in the context of your scenario.

Do all of Part E in 300 words or less.

3 Neural Network Mathematics & Dynamics (Total marks: 5)

Each of the following is worth **1 mark**. Provide a short, precise and complete answer for each.

Part 1 [1 mark]

You are training a 10-layer ReLU feedforward network (MLP), and you initialize all weights with a standard Gaussian distribution ($\mu = 0, \sigma = 1.0$). After one forward pass (before any training), you find the activations in the final layer are all exactly 0.0. Explain mathematically what happened.

Part 2 [1 mark]

You are training an MLP for a regression task where all target values y are negative (e.g., $y \in [-10, -1]$). You use an MSE loss and a **ReLU** activation on the final output neuron. What will the gradient of the loss be with respect to the network's weights? Show mathematically.

Part 3 [1 mark]

Examine the loss plot in Figure 1. Based on the principle of early stopping to prevent overfitting, at which epoch should you have stopped training the model, and why?

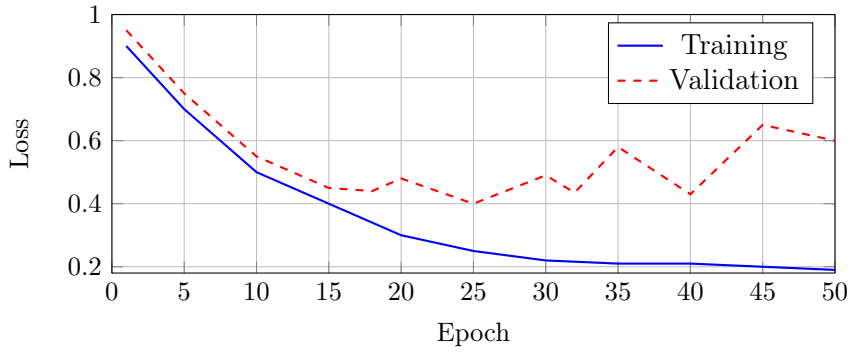

Figure 1: Training and validation loss curves for Part 3.

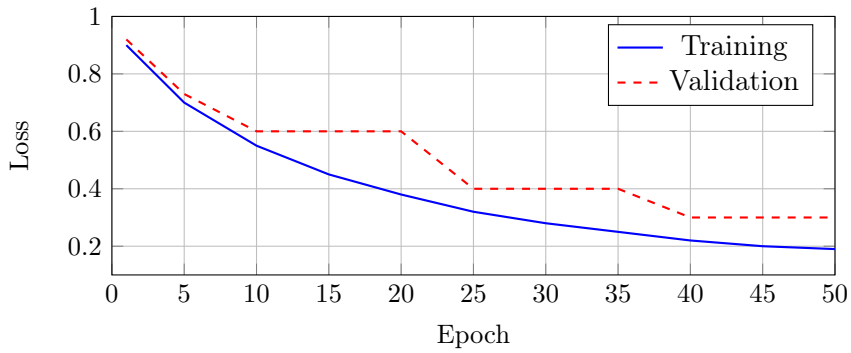

Figure 2: Training and validation loss curves for Part 4.

Part 4 **[1 mark]**

Examine the loss plot in Figure 2. The training loss (blue) decreases smoothly. The validation loss (red) is flat for long periods, then drops suddenly, then is flat again. Your dataset is clean and shuffled. What is the most likely cause of this "staircase" validation loss?

Part 5 **[1 mark]**

You build a 10-layer MLP (Dense) network but forget to add any non-linear activation functions (like ReLU) between the layers. What mathematical operation is this 10-layer stack of dense layers doing? Prove it mathematically.

4 Planning Logic (Total marks: 10)

You will examine and analyze the logical planning steps involved in a robotic assistant preparing a simple apple salad. Using the Planning Domain Definition Language (PDDL), you'll explore the basic actions a robot must execute, from handling tools and ingredients to preparing the food.

Refreshing Apple Salad

Prepare a delightful and refreshing apple salad that's perfect for sharing with your guests. Follow these steps to create a salad using a fresh apple, ensuring it is cleanly peeled, properly cut, and elegantly served.

Ingredients

- 1 Fresh apple

Equipment

- Knife
- Serving Container

Instructions

1. Ensure that the apple has been thoroughly cleaned.
2. Using a knife, peel the apple to remove its skin, revealing the fresh fruit beneath.
3. Proceed to cut the apple into even slices or cubes, according to the desired style for the salad.
4. Place the freshly cut apple pieces into a clean container.
5. The apple salad is now ready to be served to your guests.

Given this recipe, the PDDL domain definition you will work with is:

```
(define (domain apple-salad-making)
  (:objects
    apple
    container
    knife)
```

```
  (:predicates
    (is_clean ?x - food)            ; Indicates the food item is clean
    (is_cut ?x - food)              ; Indicates the food item has been cut
    (is_peeled ?x - food)           ; Indicates the food item is peeled
    (empty ?x - container)          ; Indicates the container is empty
    (contains ?y ?x)                ; Indicates something contains something else
    (in_hand ?x)                    ; Indicates something is in the robot's hand
    (available ?x))                 ; Indicates something is available

  (:action take
    :parameters (?x)
    :precondition (available ?x)
    :effect (and
              (in_hand ?x)
              (not (available ?x))))

  (:action put_down
    :parameters (?x)
    :precondition (in_hand ?x)
    :effect (and
              (available ?x)
              (not (in_hand ?x))))

  (:action peel
    :parameters (?x ?y)
    :precondition (and (is_clean ?x) (not (is_peeled ?x))
                       (in_hand ?x) (in_hand ?y))
    :effect (and
              (is_peeled ?x)))

  (:action cut
    :parameters (?x ?y)
    :precondition (and (is_peeled ?x) (not (is_cut ?x))
                       (in_hand ?x) (in_hand ?y))
    :effect (and
              (is_cut ?x)))

  (:action add_to
    :parameters (?x ?y)
    :precondition (and (in_hand ?x) (empty ?y))
    :effect (and
              (contains ?y ?x)
              (not (empty ?y))
              (not (in_hand ?x))))
)
```

NOTE: the action `clean` has been intentionally left out of this PDDL domain definition to make things simpler for you. You may assume and state that things are already "clean" as you find appropriate.

Action Preconditions and Effects in First-order Logic [5 marks]

For each action below, write *both* its preconditions and its effects directly in first-order logic (FOL).

1. Take whole Apple

2. Peel Apple

3. Cut Apple

4. Put down Knife

5. Add chopped apple to the container

Part B: FOL to CNF [5 marks]

Now describe each of the following actions using the "implies" connective to relate pre-conditions and post-conditions in a single first order logic (FOL) statement. After that, convert the statement to Conjunctive Normal Form (CNF). Show the steps you followed to convert from FOL to CNF.

1. The action of peeling an apple.

2. The action of cutting an apple.

5 CIFAR-100 Semantic Expansion (Total marks: 30)

Context: In Labs 6 and 7, you worked with the Visual Genome corpus to train Skip-Gram embeddings and explored Evolutionary Strategies to insert new words. You are now tasked with delivering a final embedding model that supports **all 100 classes** from the CIFAR-100 dataset.

The Visual Genome vocabulary ($V_{original} \approx 455$) contains some, but not all, of the CIFAR-100 classes. Your goal is to produce a single embedding matrix that contains the union of the lab6 vocabulary and the CIFAR-100 vocabulary.

You are advised **NOT** to use the weak "baseline" model provided in Lab 7. You should use your codebase from Lab 6 to train an optimal Skip-Gram model. You have full freedom to tune hyperparameters, including the **embedding dimension size**, window size, and negative sampling rate, to achieve the best possible semantic representation.

Part A: Implementation [20 marks]

You can choose one of two methods to generate the full set of embeddings:

- **Retrain from scratch:** Augment the corpus/vocabulary with the missing tokens and retrain the Skip-Gram model entirely (Lab 6 style).
- **Evolutionary Insertion:** Train a robust base model (Lab 6 style) and then use Genetic/Evolutionary Strategies to insert the missing words (Lab 7 style).

Other methods are NOT permitted as an option.

Regardless of the method chosen, you must implement the function `build_my_embeddings` in `cw2.py`. This function serves as the entry point for us to rebuild your final model.

Important Submission Note: You **MUST** include your best `.pth` PyTorch checkpoint file in your submission `.zip`. Call this file `best_skipgram_523words.pth`. Your code must load this file relative to the current directory. We will not run your model training/generation code; we will only load your optimal model and assess it.

Assessment Criteria (Part A):

- **Technical Correctness (5 marks):** Does the returned vocabulary contain all original Visual Genome words AND all 100 CIFAR classes? Does the embedding matrix shape ($N \times D$) perfectly align with the vocabulary length N?
- **Semantic Quality (15 marks):** Do all the embeddings in your model (including CIFAR words) make semantic sense?

Part B: Scientific Report [10 marks]

In your `report.pdf`, provide a scientific summary (max 500 words) detailing the development and validation of your final model.

You have freedom to structure this section as you see fit, but you must scientifically justify your decisions. A strong report will typically address:

- **Methodology & Rationale:** Explain why you selected your specific approach (Retraining vs. Evolution; perhaps you tried and compared different possibilities?) and how you determined your hyperparameters (e.g., embedding dimension).
- **Training Evidence:** Provide empirical proof of your training process. This can be plots (e.g., training/validation loss/fitness) or tables comparing different versions of your model.
- **Validation and Evaluation:** Demonstrate how you verified that your final model is the "best possible" version. This may include nearest-neighbor analysis or other semantic metric-based assessments.

6 Neuro-Symbolic AI: Multi-Modal Planning (Total marks: 30)

This final task represents the culmination of your coursework, combining Computer Vision, Natural Language Processing, and Symbolic Planning into a single **Neuro-Symbolic AI system**.

In Lab 8, you explored Contrastive Learning to align image encoders with word embeddings. You have also expanded your vocabulary to include all CIFAR-100 classes. Now, you will build a system that can "see" an object, identify it, and plan a course of action for it.

We have provided a PDDL domain file (`domain.pddl`). This domain models a world containing various CIFAR-100 objects and the logical rules that govern their interactions.

Your task is to implement an agent that receives an **image** OR the name of an object (you must support both types of inputs), a set of predicates describing this object's **starting state**, a set of predicates describing this object's **end state**, and a pddl domain file. It must then parse all the parameters to produce and return a valid plan.

In `cw2.py`, you must implement the function `plan_generator` to automate the following pipeline:

- **Model Loading**: Load the trained models from your checkpoint files:
  - Load the Skip-gram word embeddings from `best_skipgram_523words.pth`
  - Load the CIFAR-100 image projection model from `best_cifar100_projection.pth`
- **Input Processing**: Determine whether the input is an image or a text object name
- **Object Identification**:
  - If input is an image: Use the loaded CIFAR-100 projection model to encode the image, then match it to the correct word in the Skip-gram embeddings
  - If input is text: Validate that the object name exists in the 523-word vocabulary
- **PDDL Domain Parsing**: Parse the provided `domain.pddl` file to extract types, predicates, and actions (using techniques from Lab 9)
- **Problem Formulation**: Construct a PDDL problem instance using:
  - The identified object as the domain object
  - The provided starting state predicates as the initial state
  - The provided end state predicates as the goal state
- **Planning Graph Construction**: Build a planning graph from the initial state to find reachable states and applicable actions (Lab 9 techniques)
- **Plan Generation**: Search for a valid sequence of actions that transforms the initial state into the goal state
- **Plan Validation**: Verify that the generated plan is valid according to the PDDL domain specification (i.e., all action preconditions are met and the goal state is achieved)
- **Output**: Return the complete action sequence with the format used in Lab 9 as the final plan (or an appropriate failure message)

This question will be assessed based on the robustness of your integration. We will run your function against a **battery of test cases**. Each test case consists of a test image containing one of the objects in the CIFAR-100 dataset and a set of logic predicates. Robustness of your image-to-text embedding space mapping is key here. If your model is overfit, you will perform poorly and get very little or no points.

Marking Criteria:

- **System Integration & Automation (5 marks):** Does the system work end-to-end without manual intervention? This includes proper error handling and graceful failure modes.
- **Perception Accuracy (10 marks):** Does your system correctly identify the object from the image? Partial credit will be awarded if the identified object is semantically similar to the ground truth (e.g., identifying "pickup_truck" when the true label is "bus").
- **Plan Validity (15 marks):** Does the returned plan successfully satisfy the PDDL goal? Plans must be:
  - Syntactically correct (valid PDDL action sequence)
  - Logically sound (each action's preconditions are satisfied)
  - Goal-achieving (final state matches the specified goal predicates)

  Only plans that are valid and optimal in length (shorter is better) will receive full points.

Note: Test cases will include both image inputs and text inputs. Your system must handle both modalities correctly. You must also appropriately handle inputs that are outside the domain.