CMP9137M Advanced Machine Learning – Workshop 7

Summary: In this workshop you will gain practical experience with Vision Language Models (VLMs) and their application to visual question answering. You are encouraged to apply VLMs to the workshop materials of the previous two workshops, and to discuss with your peer(s) and lecturer/demonstrator if you have any questions or if you want feedback on your ideas.

Task 1: **VLM-based visual question answering**

The program below illustrates the task of question answering in two different ways: (1) by generating a match label between a question and an answer, and (2) by comparing the similarity between a predicted answer and a ground truth answer.

Open Visual Code, as in previous workshops, and run the following:

- `python VisualQA-visual7w-example.py`

Once this program has been understood, you should choose one of those two ways of answering a visual question and attempt to integrate the corresponding code to that of workshop of week 5. By doing that, you could aim for a hybrid model including a trainable one and a VLM. This should motivate you to think about how to combine both into a composite model for question answering.

Task 2: **VLM-based autonomous game playing**

The following program illustrates that by asking questions to a VLM, their answers can be used as actions of a game agent—to play in particular the game of VizDoom. Run the following program:

- `python VisualQA-vizdoom-example.py`

After executing and analysing this program, you could extend one of the programs of the previous workshop. This can be the deep reinforcement learning (DRL) code or the supervised-based code, where their decisions could be shaped by a VLM. For example, the VLM's actions could be used in the first 10K (or so) training steps to bootstrap the DRL agent's behaviour instead of purely relying on trial and error. As another example but this time using the supervised agent, the probabilities of actions could be combined to find out whether they improve the purely supervised agent or not.

Task 3: **Implementation of Metrics for Visual Question Answering**

Write a program implementing the metrics Exact Match and Reciprocal Rank. Here their definitions:

a. Write a Python program to calculate Exact Match (EM) for question-answering. For each question-answer pair, if the word sequence of the model's predicted answer exactly matches the word sequence of the true answer, $EM = 1$, $EM = 0$ otherwise.

b. Write a Python program to calculate Reciprocal Rank (RR) as an alternative to the above. $RR = \frac{1}{rank_i}$, $rank_i$ is the position of the first correct answer in a ranked list. Given a ranking of predicted probabilities, assign 1 if the predicted answer is in the first place, 0.5 for second place, 1/3 and 0.25 for third and fourth places, respectively. Full example here.

Example:

What colour is the cat in this picture? Options A: White, B: Brown, C: Black (correct), D: Yellow

Probabilistic predictions: A=0.45, B=0.20, C=0.10, D=0.25
Sorted probabilistic predictions: A=0.45, D=0.25, B=0.20, C=0.10
Predicted Answer: A=White (0.45)
Exact Match (EM)=0
Reciprocal Rank (RR)=0.25

For all of the above, your program should output the average over all questions in the test set—resulting in Mean Exact Match (MEM) and Mean Reciprocal Rank (MRR).

This implementation can be useful to assess the performance of your assignment models for task 1 using multiple performance metrics—to potentially help you draw further conclusions.