ECS759P Artificial Intelligence
Coursework 2

Due Date: 6th of December 2024 at 10AM

If you have questions, you should ask them and lead discussions via the Student Forum on the module QMplus page. Using the QMplus forum for discussion is preferable as it can help us broadcast essential clarifications to the entire group. You can also ask questions during your assigned lab session. Additionally, Dr. Iran R. Roman is open for inquiries during office hours (Wednesdays, 12pm-1pm, Peter Landin 4.16) and via email to i.roman@qmul.ac.uk.

In your final submission you MUST include the files:

- `XXXXXX_report.pdf`
- `XXXXXX_genetic.py`
- `XXXXXX_best_net.py`
- `XXXXXX_best_net.pth`
- `XXXXXX_cnn.py`
- `XXXXXX_best_cnn.pth`

You must substitute "XXXXXX" for your username ("eer043", for example).

Your submission will be automatically flagged as INCOMPLETE if you do not include ALL these files, properly named, and in the format specified.

General Instructions

The coursework is composed of three parts. The first two parts involve writing code, and also answering questions in a written form. The third part does not involve writing code. You should answer written questions with your choice of typesetting software or by hand on a physical piece of paper. It is preferred if you use a typesetting software (overleaf, for example). If you use pencil and paper, you will need to take pictures of all your work and convert them to a single pdf document that contains ALL your written answers to this coursework. Your written answers should be clearly labeled to match the corresponding question number in this coursework description.

The coursework parts are:

1. Genetic Algorithm: 40%
2. Convolutional Neural Network: 40%
3. Planning logic 20%

1 Genetic Algorithm

This part of the coursework is intended to help you understand how genetic algorithms (GAs) operate and to provide you with hands-on experience in implementing and applying them to a real-world problem: the optimization of feedforward neural network architectures for a challenging classification task.

You are provided with code to load and prepare a dataset of Farsi vowel audio (6 different vowel types in total) in the file called `genetic.py`. Start by downloading the data from `https://www.kaggle.com/sabermalek/pcvcspeech` and placing the `mat` files alongside the `genetic.py` file. After this you will be able to generate the dataloaders necessary to build and train neural network models using the code already given to you in `genetic.py`.

Your challenge is to evolve neural network architectures that can accurately classify these vowels. Each architecture's effectiveness will be assessed based on its accuracy in predicting the correct class for each data point.

If you do everything correctly, you should have no trouble generating Neural Network architectures with a validation-set accuracy of at least 35% (think: how good would that be compared to taking random guesses? how high can you reasonably expect your validation-set accuracy to be?).

1.1 Implement a Genetic Algorithm (in `genetic.py`)

Implement the methods and classes in `genetic.py` to arrive at your optimal neural network architecture. You will need to implement the methods and classes called `generate_initial_population`, `Net`, `selection`, `crossover`, and `mutate`. In your written report, specify the optimal neural network architecture that you got and the validation-set accuracy you observed with it.

After you complete the implementation and are able to run `genetic.py` from start to end, a file called `best_net.pth` will be saved. This file contains the parameters of the best model you found. Now define your best model (please only define one best model architecture) in the file called `best_net.py`. Upon running `best_net.py` you should observe the same validation-set accuracy you saw when you found the model running `genetic.py`

This question carries 25% of the marks for this coursework.

1.2 Elements of the algorithm (submit in your written report)

Briefly explain how specifically you chose to do the state encoding, selection, crossover, and mutation.

This question carries 5% of the marks for this coursework.

1.3 The number of reproductions (submit in your written report)

Report the number of reproductions you had to do to converge to the optimal neural network architecture. In other words, if you run `genetic.py` 10 times, how many times will you end up finding the same optimal architecture you reported in 1.1? What other "optimal" architectures are you likely to run into? Are they related to your absolute best model in any way?

This question carries 5% of the marks for this coursework.

1.4 Hyper-parameters (submit in your written report)

Explore the effect of at least one hyper-parameter on the performance of your algorithm. This could for instance be the choice of the mutation rate, or the selection/pairing scheme, the population number, etc. Provide a table or a plot showing how the performance of the best neural network found varies as a function of this parameter. Write a paragraph describing your observation.

This question carries 5% of the marks for this coursework.

2 Lost in the closet (Classification)

You are an artist who secluded yourself for years to come up with the perfect design for a new brand of clothes. However, your time off from civilisation was not so beneficial since you cannot distinguish a T-shirt from a dress or a sneaker from a sandal any more. In order to address that issue, you choose to train a Convolutional Neural Network (using PyTorch) that will help you identify each cloth to match the perfect design you created. In order to train it, you decide to rely on the dataset fashion MNIST (`https://github.com/zalandoresearch/fashion-mnist`). You can access the data using the following lines:

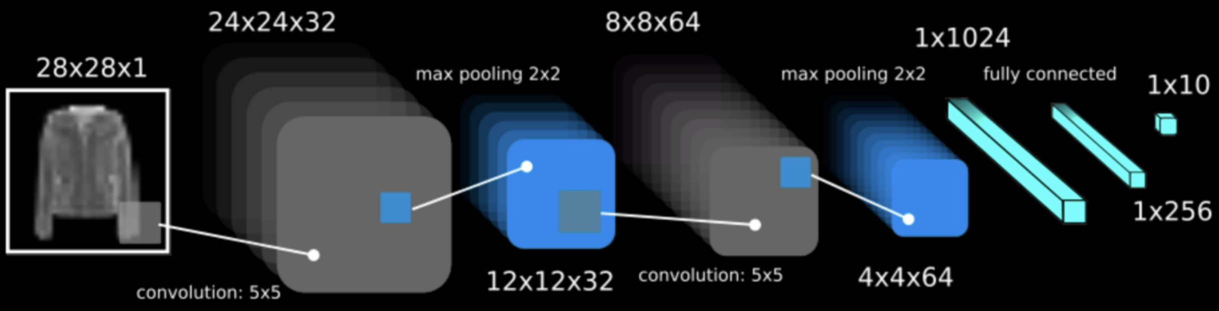
```
import torchvision
import torchvision.transforms as transforms
import torch
import torch.nn as nn

train_set = torchvision.datasets.FashionMNIST(root = ".", train=True,
                                              download=True,
                                              transform=transforms.ToTensor())
validation_set = torchvision.datasets.FashionMNIST(root = ".", train=False,
                                             download=True,
                                             transform=transforms.ToTensor())
train_loader = torch.utils.data.DataLoader(train_set, batch_size=32, shuffle=True)
validation_loader = torch.utils.data.DataLoader(validation_set, batch_size=32, shuffle=False)
# Fix the seed to be able to get the same randomness across runs and
# hence reproducible outcomes
torch.manual_seed(0)
```

1. Given the problem, what is the most appropriate loss function to use? Provide the name of the loss, its formula, its gradient w.r.t. the ”class scores” (a full derivation is not needed), and the interpretation of its mathematical properties (including the gradient with respect to the ”class scores”) in your written report.

   This question carries 5% of the mark for this coursework.

2. Create and train a Convolutional Neural Network corresponding to the following architecture (also see the diagram below):
   (a) Input image size: B x 28 × 28 × 1 (batch-size x height × width × number of channels). Your model should be able to handle pixels with a floating point value between 0 and 1.
   (b) First convolutional layer: Kernel size (5 × 5), Stride size (1 × 1) and 32 output channels. ReLU Activation function.
   (c) Max pooling layer: Kernel size (2 × 2) and Stride size (2 × 2).
   (d) Second convolutional layer: Kernel size (5×5), Stride size (1 × 1) and 64 output channels. ReLU Activation function.
   (e) Max pooling layer: Kernel size (2 × 2) and Stride size (2 × 2).
   (f) First fully-connected layer with input size being the output size of max pooling layer in (e). (flattened, i.e. 1024) and output size 1024. ReLU Activation function.
   (g) Second fully-connected layer with input size being the output size of fully connected layer in (f). (i.e. 1024) and output size 256. ReLU Activation function.
   (h) Output layer with input size being the output size of fully-connected layer in (g). (i.e. 256) and output size 10.



For training, initialise your weights using the Xavier Uniform initialisation, use ReLU as the activation function, a learning rate of 0.1 with the SGD optimiser. You will train your neural network for 30 epochs. In your report, provide the following: (a) final (train and validation) accuracy obtained; (b) plot of the accuracy on the training and validation sets per each epoch, comment on the speed of performance changes across epochs; (c) plot of the train and validation loss per epoch (total sum of per batch losses for each epoch) and comment on the speed of decrease.

You will submit a file called `cnn.py`, which should contain a class called `myCNN` defining your CNN and its training routine. You should also submit a file called `best_cnn.pth`, which will allow us to load your model and carry out inference with it. For reference, when we mark your work, we will re-build your CNN using code similar to the one in `best_net.py` (from section "1 Genetic Algorithm").

This question carries 35% of the marks for this coursework.

3 Planning Logic (include in your written report)

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 is thoroughly cleaned to remove any impurities or residues.
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, ensuring a healthy and tasty offering.

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.

3.1 Listing pre-conditions and effects associated with actions.

Using plain human language, state ALL the pre-conditions and effects associated with the following actions.

1. Take whole Apple (i.e. before it is peeled or chopped)
2. Peel Apple
3. Cut Apple
4. Put down Knife
5. Add chopped apple to the container (in order to serve to your guests).

This question carries 5% of the marks for this coursework.

3.2 Simple Statements with First-order logic

Now write the state of the world before each of the actions takes place using first-order logic statements.

This question carries 5% of the marks for this coursework.

3.3 FOL to CNF

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.

This question carries 4% of the marks for this coursework.

3.4 Linear Temporal Logic

Now use Linear Temporal Logic (LTL) to formalize the sequence of actions (strictly one action at a time) for preparing an apple salad, as described in the original recipe. Your goal is to create a logical representation of actions that captures the order and dependencies of events involved with the original recipe steps. Please use only the LTL operators 'X' (Next) and 'G' (Globally). You must formally define the initial conditions and actions within your LTL framework.

But first, an **example:** consider a traffic light sequence where the light transitions from Green to Yellow, and then to Red.

FOL Definitions:

- $\text{Green}(light)$: The traffic light is green.
- $\text{Yellow}(light)$: The traffic light is yellow.
- $\text{Red}(light)$: The traffic light is red.

LTL Formula:

$$\text{G}(\text{Green}(light) \rightarrow \text{X}(\text{Yellow}(light) \rightarrow \text{X}(\text{Red}(light))))$$

The 'G' operator specifies that the sequence should always start with the light being green, implying that this sequence must hold true in all scenarios where the system is active. The 'X' operator dictates that once the light turns green, it must next turn yellow, followed by red. This sequence repeats globally as long as the system is operational.

Now apply what you have learned from the traffic light example to formalize the sequence of actions to prepare an apple salad using the provided recipe steps. We suggest that you do the following:

1. Start by identifying the full and detailed sequence of actions using all the objects, predicates and verbs specified in the recipe PDDL domain definition.
2. Define each step with FOL that describes the transition from pre-conditions to effects.
3. Using LTL, write a logical sequence of actions that represents the recipe. Use ′X′ for sequential steps and ′G′ to assert what needs to be globally initialized in order to be able to carry out the sequence.

This question carries 6% of the marks for this coursework.

4 Submission

Submit your coursework to QMPlus as a single zip file (filename ending in <username>.zip).

Note that a human will be reading the code and your written report, and cases of plagiarism will be reported.

IMPORTANT NOTE ABOUT YOUR USE OF AI:

While the use of Generative AI (ChatGPT, Github Co-pilot, Claude, etc.) to produce text or code for your coursework is DISCOURAGED, it is NOT prohibited but MUST be acknowledged when used. If you insert AI-generated text or code in your report/code, you must include the name and version of the AI system you used as an in-line comment for each line you generate with AI. Additionally, you must specify the prompt(s) used to produce the content. We recommend using chain-of-thought prompting so that your prompt clearly describes how you critically analysed the problem and the behavior of the AI (also including the analysis of AI-generated mistakes and your subsequent refinements to the prompts).

Failing to acknowledge and thoroughly describe the use of AI-generated code and text will result in an automatic grade of zero. Remember that AI is not perfect and its outcomes can be fallible.

Note that cases of identical (or similar) code and/or text to your peers resulting from the usage of Generative AI (without acknowledgement or critical analysis of this usage) will be considered plagiarism and will be reported.