# Reinforcement Learning
Final Project - 2023

The main goal of this mid semester project is to summarize the main topics that we have discussed in the course using some practice and theory, **and especially the second part of the course (Deep RL)**.

Project Definition

In this project you will solve a variation of the SOKOBAN environment.
The Environment: https://github.com/mpSchrader/gym-sokoban
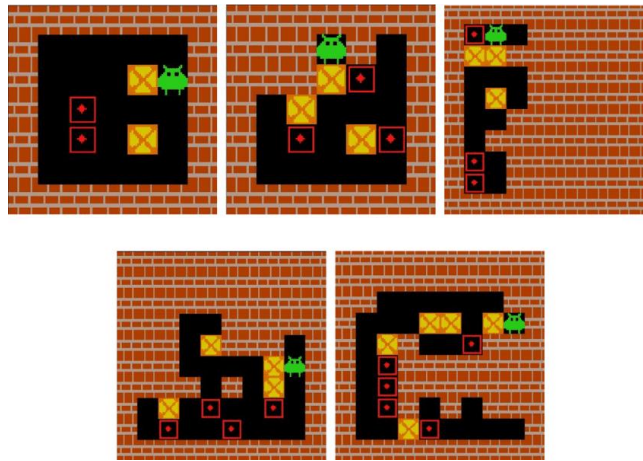
Sokoban (meaning 'warehouse keeper') is a Japanese puzzle video game genre in which the player pushes crates or boxes around in a warehouse, trying to get them to storage locations.

The game is a transportation puzzle, where the player has to push all boxes in the room on the storage locations/ targets. The possibility of making irreversible mistakes makes these puzzles so challenging especially for Reinforcement Learning algorithms, which mostly lack the ability to think ahead.

The room generation is random and therefore, will allow to train Deep Neural Networks without overfitting on a set of predefined rooms.
The game is played on a board of squares, where each square is a floor or a wall. Some floor squares contain boxes, and some floor squares are marked as storage locations.

The player is confined to the board and may move horizontally or vertically onto empty squares (never through walls or boxes). The player can move a box by walking up to it and push it to the square beyond. Boxes cannot be pulled, and they cannot be pushed to squares with walls or other boxes. The number of boxes equals the number of storage locations. The puzzle is solved when all boxes are placed at storage locations.

Room Elements:

Every room consists of five main elements: walls, floor, boxes, box targets, and a player. They might have different states whether they overlap with a box target or not.

Type	State	Graphic	TinyWorld
Wall	Static	▦	◼
Floor	Empty	◼	◻
Box Target	Empty	▣	◼
Box	Off Target	⊠	◼
Box	On Target	⊠	◼
Player	Off Target	▦	◼
Player	On Target	▦	◼

Actions:

The game provides 13 actions to interact with the environment. Push, Pull and Move actions into the directions Up, Down, Left and Right. The No Operation action is a void action, which does not change anything in the environment. The mapping of the action numbers to the actual actions looks as follows:

Action	ID		Action	ID
No Operation	0		Move Down	6
Push Up	1		Move Left	7
Push Down	2		Move Right	8
Push Left	3		Pull Up	9
Push Right	4		Pull Down	10
Move Up	5		Pull Left	11
			Pull Right	12

Rewards:

Finishing the game by pushing all on the targets gives a reward of 10 in the last step. Also pushing a box on or off a target gives a reward of 1 respectively of -1. In addition a reward of -0.1 is given for every step, this penalizes solutions with many steps.
This is a default choice of the environment. during your experiments, you are expected to try different rewards.

Programming Task

Solve the following environments:

EX1 - FIX SCENARIO - Easy:

One box, One Target, Grid size: 112 X 112 (RGB), i.e. Every episode you get the same configuration.

EX2 - "Push & Pull" - Medium:

RANDOM GENERATED SCENARIO - One box, One Target, Grid size: 112 X 112 (RGB).

You will find demonstration of how to use/render the game (random actions) in the template notebook:

https://colab.research.google.com/drive/1hMR2ZP2Y2PAUoKX8KqcesB1DRjA0m20T?usp=sharing

Explore the environment documentation carefully and make sure you understand what is the environment, the observation-space, the action-space, and how are the rewards defined. Also, make sure you understand what's the difference between the different versions.
After entering the notebook click on the "File" tab and then on "Save a copy into drive".
You can then work on the project within your personal drive environment.

The main goal for all the environments/exercises is to "solve" as fast as you can (less episodes) - **this is a competition.**

You will find a running example of each of the above environments in the template notebook.

Guidelines:

1. The main goal is to "solve" as fast as you can - this is a competition part.

2. You must use pixel space. You can use pre-process such as transformation to gray-scale, cropping etc.

3. In Ex2&Ex3 You are allowed to use Reward Shaping based on the locations of the Boxes/Targets/Player e.g. "sum of distances". The attached google colab demonstrates how to extract this info.

4. Even if you did not solve any of the environments - Supply a graph describing the average rewards on 100 episodes (after finishing the training phase). Note: stop an episode after 500 iterations (if it was not finished).

5. You are **not allowed to use an existing RL libraries**. Write the algorithm yourself.
   Remember: We want to see your understanding of RL and not your ability to use libraries and modules.

6. You are **not allowed to use** solutions or RL algorithms that you haven't learnt in the course (Refers only to network architectures and RL algorithms).
   But… If you want as an **extra**, you can add advanced methods and show other solutions.

7. You can of course use existing DL platforms such as Keras, Tensorflow, Pytorch.

8. **Experiments (Must)** - In your final report, compare different approaches, different parameters, different initialization values for the environment, different discount factors, epsilon-greedy etc. You can use a method that encourages exploration.
   Show graphs (In the notebook and report) comparing the different parameters and different methods. In addition, Show visualizations of Q, V of "interesting" states.

9. After working and analyzing with the learnt exploration-exploitation methods, you can consider using advanced methods from an external source.

10. For the most successful experiment, in each exercise, Show (In the notebook and report) the number of steps it took the agent to solve the environment, and a convergence graph (Rewards according to the number of episodes). In addition, show two video clips (In the notebook). The first video clip shows the agent in the middle of the training process and the second video clip shows the agent after the learning completed and converged, (for example, if it took you 40 episodes to converge, show how the agent tries to solve the environment after 20 episodes in the second video, show how the agent tries to solve the environment after 40 episodes).
    Even if you did not solve an environment - supply graphs describing the average rewards

on 100 episodes (after finishing the training phase).

11. All training outputs will be displayed in the notebook.

12. The test environment will be built according to the video we attached to the submission box. The test environment will load the trained weights/trained model (for each of the two different environments best solutions).
If you see fit, you can submit the test environment in a separate notebook. In this case, when submitting, you will submit a zip file that contains both notebooks (train and test).

13. Submit a final report in pdf format when your ID numbers are included in the file name.

14. You need to supply your code as Google Colab notebooks. Such that the course team can run it.

15. **Write a clean code**! Separate code cells and make extensive use of text cells. A sloppy notebook will result in a lower grade.

16. Do not change anything in notebook after the final submission date. A notebook that has been ran/changed after the submission date will be **automatically disqualified**.

Due Date:
Final Project submission is due **February 20th at 23:59**.

Submission:
Detailed report including graphs, source code links (to Google Colab), relevant images and a README file containing relevant explanations for running the notebook.
Your report should be in the style of a conference paper, including introduction, motivation, related work, etc.
Each part of the project must have experiments, if you failed in a particular experiment, explain why you think you failed and what you have done to improve from there.
You are expected to use very elaborate explanations and use visual means to show your results (plots, graphs, etc.). Your grade would be greatly affected by the report you write so write the report thoroughly!

The report will be written using an online collaborative LaTeX editor called overleaf (link below).

An example: https://www.overleaf.com/read/nmhycgyvdtxc

Everything in the report should be in your own words - all quotes must be clearly attributed.

The report will be submitted to the submission box. The file name of the report will contained your IDs as following: **report_ID1_ID2.pdf**

Be very clear about what code you've used from other sources, if any. Clear citations are essential. Failure to credit ideas and code from external sources is cheating and will result in immediate disqualification.
Make sure you evaluate both the good and bad points of your approach.
Even if you didn't accomplish your goal, evaluate what you did.

Do not forget to include the project title, your name and ID in this file.
**Max number of pages: 10 (but you don't have to use them all !!).**

The notebook will be presented in an orderly and clean manner, it will contain separate code cells and text cells that explain the actions performed.
** **Very important ** ** - when submitting, the notebooks will contain **all the outputs** relevant to the training results.

To the submission box you will also submit an **explainer.md** file that contains instructions and explanations on how to operate your notebook and other relevant details that need to be known to those who want to use your notebook.
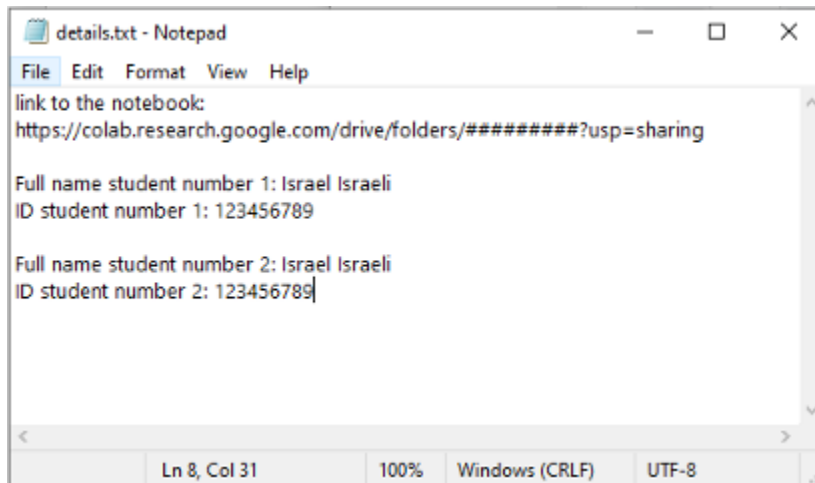
You will share the notebook from your "Google Drive" account, it can be shared with anyone who holds the link as follows:
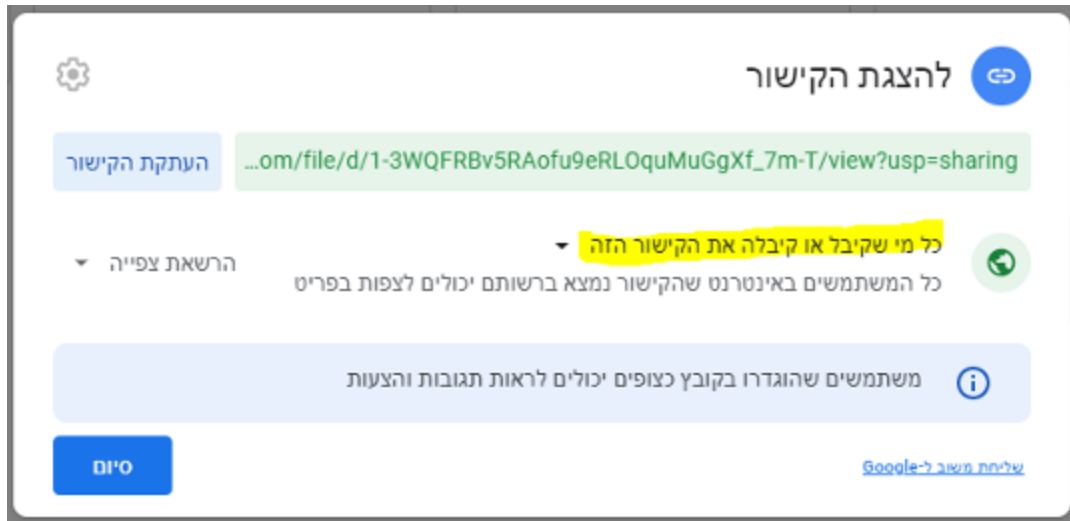
Submission will be done in pairs when only one of the partners submits the assignment to the submission box.

Enter your details in the text file named **submit.txt**, the address of your notebook, the names and ID of the two partners, as follows:

In addition to the links to the notebooks you submitted in submit.txt, you will also submit the notebooks themselves to the submission box. You can download the notebook from the Colab interface in the following way: File → Download → Download ipynb

To sum up, in the submission box, submit the following files:
1. submit.txt
2. notebook_name.ipynb
3. report_ID1_ID2.pdf
4. explainer.md

Team size
The project will be performed in groups of 2 students.

Academic Integrity
Team/Student may not copy code from other teams/students. Copying answers or code from other students for a project is a violation of the university's honor code and will be treated as such. All suspicious activity will be reported to the head of the department and the university authorities.

Giving code to another student is also considered a violation. Students are responsible for protecting their own work from copying.

If you build some of your code on existing work and utilize existing code (your own or code found on the web), you must give proper attribution to all existing work that you used and make it clear what you changed and contributed. Any unattributed or uncited work that you use will be considered a breach of academic honesty and dealt with according to the course policy in the syllabus.

In any matter related to the project, even administrative, you can contact Aviv German at: aviv.german@post.runi.ac.il