Instructor: Alina Vereshchaka

Final Course Project

Multiple deadlines, please see below

Description

The goal of the Final Course Project is to explore advanced methods and/or applications in RL. You will be expected to prepare a proposal, checkpoint, final submission, and presentation. All projects should evaluate novel ideas that pertain to deep RL or its applications and must involve RL algorithms.

You are encouraged to use your ongoing research work as a project in this course, provided that this work relates to deep RL. Discuss the topic of your final project with course instructors by private message in Piazza, or during OH. If you are not sure about the topic, we encourage you to speak with us.

There are a few directions suggested, please check the end of the description for more details.

1. Register your team (Sept 25)

You may work individually or in a team of up to 3 people with both undergraduate and graduate students. The evaluation will be the same for a team of any size. You can team up with students from the online section or in-person section.

Register your team at UBLearns (**UBLearns > Groups**).

Note: Make sure that all members join the same team on UBLearns (e.g. Team 27)

2. Project Management Tool [5 points] (Sept 30)

Set up a project management (PM) tool. E.g. Github Project (free plan), Trello (free plan).

1. Create a board with at least three columns: **To do, In Progress, Done**
2. Divide the project into at least 20 steps/milestones (E.g., Explore the topics, create a proposal, implement a basic version, implement advanced algo, Submit Checkpoint, Submit Final, Prepare for the presentation, etc.).
3. Add course instructors as observers or as team members to your board @ub-rl for GitHub.
4. There should be tracked activities every week by each team member
5. **Include a few screenshots of your board activities and the link to your board as part of your Checkpoint and Final submissions. Submissions without the link won't receive points.**

3. Submit the proposal [10 points] (Sept 30)

The project proposal should be a one-two pages single-spaced extended abstract motivating and outlining the project you plan to complete. Your proposal should have the following structure:

1. **Title**
2. **Short summary**. What problem statement or research question that your project aims to address? Why is it interesting?
3. **Objectives**. Outline the goals you aim to achieve.
4. **Methodology**. What RL techniques, models, or algorithms you plan to use? How do you plan to improve or modify such implementations? You don't have to have an exact answer at this point, but you should have a general sense of how you will approach the problem you are working on.
5. **Evaluation**. How will you evaluate your results? Qualitatively, what kind of results do you expect (e.g. plots or figures)? Quantitatively, what kind of analysis will you use to evaluate and/or compare your results (e.g. what performance metrics or statistical tests)?

6. **Environment**. What environment(s) do you plan to use?
7. **References**. Include a list of the relevant literature and resources you plan to use.

- Name as **TEAMMATE#1_UBIT_TEAMMATE#2_UBIT_ proposal _project.pdf**
- Submit at **UBLearns > Assignments**

4. Submit the checkpoint [20 points] (Nov 4)

- Submit the initial results of your model (e.g., built baseline model and prepared a pipeline for training).
- You need to submit the code and a draft presentation/report with prior results.
- All project files need to have clear naming, e.g. **avereshc_nitinvis_checkpoint_project.ipynb** and **avereshc_nitinvis_checkpoint_project.pdf**
- All project files should be packed in a ZIP file named: **TEAMMATE#1_UBIT_TEAMMATE#2_UBIT_ checkpoint _project.zip** (e.g., **avereshc_nitinvis_checkpoint_project.zip**).
- Submit at **UBLearns > Assignments**

5. Submit the final results [50 points] (Dec 1)

- Submit at **UBLearns > Assignments**
- The code of your implementations should be written in Python. You can submit multiple files, but they all need to be labeled clearly.
- All project files need to have clear naming, e.g. **avereshc_nitinvis_final_project.ipynb** and **avereshc_nitinvis_final_project.pdf**
- All project files should be packed in a ZIP file named:

TEAMMATE#1_UBIT_TEAMMATE#2_UBIT_final_project.zip (e.g., **avereshc_nitinvis_final_project.zip**).

- Your Jupyter notebook should be saved with the results.
- A report can be submitted with the presentation. Thus, if you discuss all the technical implementation of your project and provide the results, a separate report will not be necessary. As part of your final submission, you can include a draft report version.
- Include all the references that have been used to complete the project.
- If you are working in a team, we expect equal contribution for the assignment. Each team member is expected to make a code-related contribution. Provide a contribution summary by each team member as a table below. If the contribution is highly skewed, then the scores of the team members may be scaled w.r.t the contribution.

Team Member	Project Part	Contribution (%)

6. Present your work [15 points] (Dec 1 – Dec 3)

- Present your work during the presentation day. Registration slots will be available prior to the date.
- The whole team should equally present the work.
- Your presentation should represent the work you will submit.
- Submit the final presentation by **Dec 3, 11:59pm**.

Presentation details

Length: 15 mins + follow-up questions

Suggested Templates: UB branded templates

Suggested presentation structure:

- Project Title / Team's Name / Course / Date [1 slide]
- Project Description [1 slide]

  Describe the problem you are working on, why it's important
- Background [max 2 slides]

  Discuss the related background and works that relates to your project. How is your approach similar or different from others?
- Environment [max 2 slide]

  Describe the environment(s) you are working with for your project.
- Methods [max 2 slides]

  Discuss your approach for solving the problem. Did you consider alternative approaches? Include figures, diagrams, or tables to describe your method or compare it with other methods.
- Results [max 3 pages]

  Discuss the experiments that you performed. The exact experiments will vary depending on the project, but you might compare with previously published methods, use visualization techniques to gain insight into how your model works. Include graphs, tables, or other figures to illustrate your experimental results.
- Demo (if available)
- Key Observations / Summary [1 slide]
- For Teams: Contribution Summary by Each Team Member [1 slide]
- Thank you Page [1 slide]

Extra Points [max +20 points]

Participate in Weekly Scrum Meetings [5 points]

Weekly Scrum is a regular group meeting led by Nitin Kulkarni (Tue/Thu, 5 pm) starting from September 30 where at least one team member gives updates about the project progress. This is not evaluated, and you can also consider these meetings as an opportunity to get feedback on your current results.

Your team must take part in a scrum meeting for at least 5 weeks to be eligible for the bonus. We encourage all team members to join scrum meetings.

Real-world RL application [5 points]

If you are formulating and solving a real-world problem using RL methods, your work is eligible for this bonus. In your report you must highlight the references to the real-world case that you are solving as well as the real-world data you use for your project.

CSE Demo Days [10 points]

If you get interesting results, we encourage you to share your project with the public in terms of participating in the CSE Demo Days. CSE Demo Days is a semester event, where you can highlight your project results.

Send us your prior results before November 20. **Selected teams** will have to prepare a poster and present it.

No requests to take part in CSE Demo Days will be accepted after November 20.

If you receive a winning place award for your project, **your bonus points will be +50!**

Important Information

This project can be done in a team of up to three people.

- All team members are responsible for the project files submission
- No collaboration, cheating, and plagiarism is allowed in assignments, quizzes, the midterms or final project.
- All the submissions will be checked using SafeAssign as well as other tools. SafeAssign is based on the past semesters' submitted works and current submissions. We can see all the sources, so you don't need to

worry if there is a high similarity with your Checkpoint submission.

- The submissions should include all the references. Kindly note that referencing the source does not mean you can copy/paste it fully and submit it as your original work. Updating the hyperparameters or modifying the existing code is subject to plagiarism. Your work must be original. If you have any questions, send a private post on piazza to confirm.
- Submitting material that has been previously submitted, in whole or in any part is not allowed.
- All group members and parties involved in any suspicious cases will be officially reported using the Academic Dishonesty Report form. What does that mean?
    - In most cases, the grade for the assignment/quiz/final project/midterm will be 0 and all bonus points will be subject to removal from the final evaluation for all students involved.
    - A grade reduction will be applied towards the final evaluation
    - Those found violating academic integrity more than once throughout their program will receive an immediate F in the course.
- Please refer to the Academic Integrity Policy for more details.

Late Submission Policy

For the final project everyone will be provided with 3 late days per team, no matter whether you are working individually or with teammates. These late days can be applied only to final project-related due dates. Be aware that some of the project components have hard deadlines.

These cannot be combined with your individual late days.

You do not have to inform the instructor, as the late submission will be tracked in UBLearns.

Final Project Grading

Project Management Tool [5 points]

- Graded during the final evaluation based on 0/5 points.
    - “5” is assigned, if:
        - At least 20 steps/milestones
        - There is a tracked activities every week by each team member
    - “0” is assigned for all other cases

Proposal submission:

- Graded based on 0/10 points. Evaluated within two weeks after the due date.
    - “10” is assigned, if the proposal is complete, realistic and includes all the details following the structure suggested (see 3. Submit the proposal)
    - “0” is assigned for all other cases

Checkpoint submission:

- Graded based on 0/20 points. Evaluated within two weeks after the due date.
    - “20” is assigned, if the checkpoint clearly shows progress towards the final submission following the proposal submitted.
    - “0” is assigned for all other cases
- **Note**: it is ok to slightly adjust your initial proposal, if your initial results are not as expected. In this case, please submit your updated proposal along with your checkpoint submission.

Final submission:

- Graded based on the X out of 50 points + bonus [if applicable]
- During the final evaluation, all the parts are evaluated, so please include a final version of all the parts of the assignment in your final submission.

Present your work:

- Graded based on the 0/15 points.
    - “15” is assigned, if:
        - whole team equally present the work
        - the presentation represents the work you will submit
        - the presentation follows a structure suggested
    - “0” is assigned for all other cases

Notes:

- Only files submitted on UBlearns are considered for evaluation.
- Files from local device/GitHub/Google colab/Google docs/other places are not considered for evaluation

- We strongly recommend submitting your work in-progress on UBlearns and always recheck the submitted files, e.g. download and open them, once submitted

Important Dates

Sept 25, Thu, 11:59pm -- Register your team (UBLearns > Groups)

Sept 30, Tue, 11:59pm -- Abstract is Due and setup PM tool

Nov 4, Tue, 11:59pm -- Checkpoint is Due

Dec 1, Mon, 11:59pm -- Final Submission Deadline

Dec 1 – Dec 3 -- Presentation [hard deadline, no late days can be applied]

Final Project Directions

Below is a list of possible directions for your final project, choose one.

Multi-agent RL

Build and solve multi-agent tasks, including but not limited to agent communications, transportation problems, multi-agent cooperation, etc. It might potentially lead to a research project.

STEPS INCLUDE:

1. Build a small multi-agent environment with at least two agents (this can be an extension of your work at Assignment 1).
2. Solve the environment using any tabular methods.

   [Steps 1 & 2 can be used for checkpoint submission]

3. Solve the environment using any deep RL methods (DQN, DDQN, AC, A2C, DDPG, TRPO, PPO, etc.) and compare the results.
4. Apply the algorithm from Part 3 to solve any of the existing MARL problems. E.g., Open AI Particle Environment, Hide & Seek

References:

- A collection of papers/books/other [github]
- PettingZoo: multi-agent version of Gymnasium [github]
- A collection of well-defined MA Gym-like env [github]
- Hide and Seek [blog post, github]

Suggestions:

1. Go with collaborative multi-agent systems, it is easier to scale and has more real-world reference. Possibly you can think about a mixed cooperative-competitive, e.g., Hide and Seek example.
2. While building your env, think about some real-world problems that you can solve using MARL settings. It is good if the problem you formulate is a socially important one (e.g., mitigating natural

disasters, transportation, rescuing people, etc.), so it can be later converted to the research project.

Applied RL

Consider a real-world problem that can be solved using RL methods. Deploy the model and show the results.

POSSIBLE DIRECTIONS:

- RL for A/B testing [video]
- Marketing -- real-time bidding with multi-agent RL [paper, paper]
- Dynamic medical treatment regimes
- Smart home energy management system [CityLearn env]
- Many others

Note:

For this direction you might need to build a web app or construct an environment that is based on real-world data. Deploy the RL model and show the results real-time.

Inverse Reinforcement Learning (IRL)

Implement IRL algorithms that aim to recover the underlying reward function from observed behavior data. Your project could focus on implementing and evaluating different IRL methods to learn interpretable and human-like reward functions in various applications, such as autonomous driving or human-robot interaction. Provide a comparison of the different IRL methods.

Possible IRL Methods:

1. Maximum Entropy Inverse Reinforcement Learning (paper)
2. Bayesian Inverse Reinforcement Learning (paper)
3. Generative Adversarial Imitation Learning (paper)

4. Meta-Inverse Reinforcement Learning (paper)

Transfer Learning in RL

Explore how transfer learning techniques from supervised learning and unsupervised learning can be adapted and applied to reinforcement learning problems. This could involve studying methods for transferring knowledge between tasks, domains, or even agents to improve learning efficiency and generalization.

Possible Directions:

1. Domain Adaptation in RL
2. Meta-RL with Transfer Learning

RL in Natural Language Processing

Apply reinforcement learning to various NLP tasks such as dialogue systems, machine translation, or text summarization. You can develop algorithms that learn to generate coherent responses, translate between languages, or summarize documents based on interaction with users or feedback from evaluation metrics. Provide a detailed comparison of your approach vs existing approaches.

Possible Directions:

1. Deep Q-Networks for Dialogue Systems
2. Policy Gradient Methods for Machine Translation
3. Attention Mechanisms for Text Summarization

Exploring Deep RL Algorithms [Safe Mode]

Research and apply recent advances in RL. This may include solving any of the following environments using deep RL algorithms.

Possible environments include:

- Robotics by Farama [details]
- MuJoCo by DeepMind [DeepMind article, documentation]
- Atari Environments [details]
- PyBullet [details] - a well-supported env for robotics simulation

Steps include:

1. Set up the environment
2. Explore the existing baseline methods applied to solve it & run it

   [Steps 1 & 2 can be used for checkpoint submission]

3. Apply a at least 3 other deep RL algorithms to improve the results. You can use your Assignments 2 & 3 implementations as a baseline code for comparison.

Some advanced directions to consider:

1. Random Network Distillation (RND) [blog post, paper]
2. Offline reinforcement learning [blog post, tutorial]

Propose you own topic

Any final project should have a major programming component. You may come to us with your topic proposal! We understand that it might end up being challenging, if you find out you are completely stacked, you are welcome to discuss your possible switching to any other directions.

Although you will follow your own direction, the general requirements for the first checkpoint include the following:

1. Build your own environment or use an existing one.
2. Explore the already existing solutions and replicate it, you will later use it as a baseline to compare your solution.
3. Propose your solution to solve the problem and compare the results

Please talk to the course instructors to ensure the project you have in mind is feasible.

GENERAL STRATEGY

- The main motivation of the final project is to explore novel ideas (either with the problem setup or the algorithm or both) or have a comparison of existing solutions. Start with a simpler problem and build on it going forward.
- If you have a challenging environment to solve, e.g., MARL, MUJOCO or RLBench, or you create your own, you can choose easy algorithms to work with such as Double DQN/ Advanced Actor-Critic etc., as there is considerable effort in designing and setting up the env
- If you choose to explore Deep RL algorithms, consider applying those algorithms on multiple env to have a better comparison and analysis, e.g., Atari Breakout, SpaceInvadors and FetchReach.
- You are encouraged to use your implementations in A1, A2 or A3 (Tabular Methods/DQN/Double DQN/Actor-Critic) as a baseline to compare against other approaches that you will use for your project.