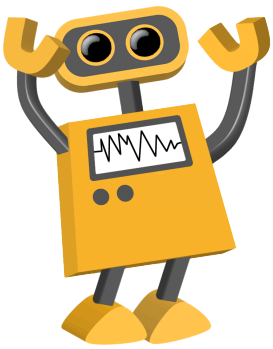
Reinforcement Learning (Second Half) Coursework

Dr. Edward Johns

Tuesday 10th November 2020

This is the coursework for the second half of Reinforcement Learning. There are two parts. Part 1 is based on the DQN tutorial and has 4 sections, where each section asks you to create a figure and answer two short questions. Part 2 is more open-ended and allows you to develop your own algorithm to solve a deep reinforcement learning problem, which we will then run in a "competition". At the end of the coursework, you will submit a single PDF to CATE called **report.pdf**, which contains your answers for both Part 1 and Part 2, together with a Python script called **agent.py**, which contains your implementation for Part 2. At the end of this document is an example of how the submitted document should be formatted.

All material for the coursework is at www.robot-learning.uk/teaching.



1 Coursework Part 1

1.1 Online Learning and Mini-Batch Learning

For this question, complete Sections 1 - 6 of the tutorial only.

Figures 1 (a) and (b) Plot two graphs of *Loss* (y-axis) vs *Episodes* (x-axis), as in Section 5 of the Tutorial. The y-axes should have a logarithmic scale. For both graphs, the agent should be run for 100 episodes. *Loss* is the Q-network's average loss across all steps in an episode, and *Episodes* is the total number of episodes the agent has taken in the environment so far. Figure 1 (a) should be for online learning (Section 5 of the Tutorial), and Figure 1 (b) should be when using the experience replay buffer (Section 6 of the Tutorial). So for online learning, the loss for each step is the loss for a single transition, whereas with the experience replay buffer, the loss for each step is the average loss over the mini-batch. And so far, the loss is based on just predicting the immediate reward. Remember to label the graphs and their axes.

Question 1

(i) Inspect the variance of the loss in each graph. State which of these two methods results in the most stable training, and explain the reason for this.

(ii) State which method is more efficient at improving the Q-network's predictive accuracy per episode of interaction, and explain the reason for this.

1.2 Visualisation of Q-values and Policy

For this question, complete up to (and including) Section 7 of the tutorial only.

Figures 2 (a) and (b) Show two images visualising what the agent has learned after 100 episodes of interaction and training, as in Section 7 of the Tutorial. Figure 2 (a) should visualise the Q-values, and Figure 2 (b) should visualise the greedy policy.

Question 2

(i) Inspect the Q-value visualisations for the top-right and bottom-right of the environment. Is it likely that one of these two regions will have more accurate Q-value predictions than the other region? Explain your reasoning.

(ii) Inspect the visualisation of the greedy policy. If the agent were to execute the greedy policy, state whether or not it would reach the goal, and explain why this is.

1.3 The Bellman Equation and Target Network

For this question, complete the entire tutorial.

Figures 3 (a) and (b) Similarly to Section 1.1 of this coursework, plot two graphs of *Loss* (y-axis) vs *Episodes* (x-axis), but now when the full Bellman equation and a target network are used, as in Section 8 of the Tutorial. The y-axes should have a logarithmic scale. For both graphs, the agent should be run for 100 episodes. Figure 3 (a) should be for when there is no target network (i.e. the Q-network is used in the Bellman equation), and Figure 3 (b) should be for when a target network is used. When the target network is used, it should be updated by the Q-network every 10 episodes. Remember to label both graphs and their axes.

Question 3

(i) Now that you have introduced the Bellman equation, explain what the agent is now able to learn, that it was not able to learn before.

(ii) Inspect the shape of the loss curves. Explain why the shape of the loss curves is different in the two graphs.

1.4 Exploration vs Exploitation

For this question, there is no tutorial section. It is up to you to decide how it should be implemented.

So far, the agent has only been executing random actions. Now, implement ϵ-greedy exploration, to allow the agent to begin to exploit what it has learned. Try to find a decay rate for ϵ which eventually enables the agent to consistently reach the goal on every episode. You may also want to change some other parameters in the overall code, such as the episode length, the number of episodes of training, the network learning rate, and the mini-batch size, as well as others.

Figure 4 (a) and (b) Similar to Section 1.2 of this coursework, show two images visualising what the agent has learned, after it has been trained using the algorithm you designed above. Figure 4 (a) should be the Q-values, and Figure 4 (b) should be the greedy policy. The greedy policy should be plotted for an episode length of 20, even if the episode length was longer during training.

Question 4

(i) If ϵ was always 0.0, is it possible that the agent could ever reach the goal? Explain your answer.

(ii) After a period of training, but before the Q-network has converged, imagine that the agent executes the greedy policy, which results in the agent reaching the goal. But after reaching the goal, the agent continuous on in a straight line, and hits the wall on the left of the environment. Why might this happen?

2 Coursework Part 2

Files

Take a look at the Python 3 files **random_environment.py**, **agent.py**, and **train_and_test.py**. The relevant parts of the code are well commented, so you may be able to understand these files by reading through them. Below is a short description of each file.

random_environment.py
This is similar to the **environment.py** file in Part 1 of the coursework. The difference now is that the environment it creates is more complex, and it creates a random environment every time the **Environment** class is instantiated.

agent.py
This is the only file which you should edit. There are currently five functions, which are used by **train_and_test.py** to allow the agent to interact with the environment. You should ensure that the names of these functions, and their argument lists, do not change. However, you are free to edit the code within each of these functions. And you are free to create new functions as you wish.

train_and_test.py
This is the script which will be used to train, and then test, your agent. It will create a new random environment, train your agent for 10 minutes in this environment, and then test your agent's greedy policy in this environment with an episode of 100 steps.

If you run **python3 train_and_test.py** from the command line, you should see the environment being displayed. The red circle is the agent's current state, and the blue circle is the goal. The light region is free space, which the agent can move through, and the dark region is an obstacle, which the agent cannot move through. If the agent tries to move into the obstacle, the agent will remain in its current state. The agent must navigate the "maze" and reach the goal as quickly as possible.

Implementation

Your task is to create your own deep DQN implementation in the file **agent.py**. Your implementation may be similar to the code you have written for Part 1 of the coursework, with some optimised parameters (e.g. episode length, mini-batch size, network architecture, reward function, learning rate, epsilon decay rate, etc.). But additionally, you may wish to implement any of the advanced methods introduced in Lecture 2 (e.g. Prioritised Experience Replay and Double Q Learning). Furthermore, you may wish to introduce some of your own ideas, although there is a limit on what is allowed (discussed below).

I suggest proceeding in the following order:

1. Take the code you have developed in Part 1, and rewrite it so that it fits into the structure of the new Agent class.

2. Study the effect of various parameters in your code, and find a set of parameters which work well for a range of different environments.
3. Introduce some of the Deep Q-learning extensions from Lecture 2, and see if they improve the agent's performance.
4. Introduce any other ideas you may have.

Remember that for this course, 50% of the grades are awarded for your courseworks. This is larger than most courses, and so you are expected to spend more time on coursework in this course, than in other typical courses. Part 2 of this coursework is advanced and is intended to challenge even the top students. If you are having difficulty, just focus on inserting in the basic DQN implementation from the tutorial, and you can still be awarded a decent grade.

Rules

Below are some rules which your solution must abide by:

- The magnitude of your action vector must not exceed 0.02. Otherwise, the agent will stay still. You can see this imposed in line 104 of **random_environment.py**.
- Your **agent.py** may import any Python 3 modules from the Python 3 standard library. However, it may only import the following additional modules: numpy and torch.
- Your implementation must be a genuine Deep Q-learning solution. You may not use tabular reinforcement learning. You may not create a hand-crafted controller (e.g. keep moving right, if the agent hits a wall, randomly move up or down until it can move right again, repeat ...). To check: your **Agent.get_greedy_action()** must return the action with the highest Q-value, based on the output of a Q-network. It cannot return an action based on a heuristic you have developed.
- You are not allowed any form of memory, other than by use of an experience replay buffer. For example, you cannot save a network if it is working well, and then load it later. You cannot create a variable which stores the highest reward so far. And you cannot create a variable which stores what the agent did in the previous timestep.
- Whilst you may use an experience replay buffer, this cannot be used for anything other than training the Q-network. For example, you cannot achieve any of the memory above by querying data in the replay buffer.
- You may not import anything from the **environment** module into the **agent** module. For example, you might be tempted to import the goal state, or the locations of the obstacles. This is not allowed; the agent must learn only from the **distance_to_goal** value which is returned from the environment. Importing the environment will not work anyway, since the **environment.py** file we test your code with is different to the one you have been provided.

- You may not copy and paste entire existing implementations you have found elsewhere. Any implementations must be your own, and we will be checking your code for plagiarism. Feel free to use other implementations for inspiration, but do not copy and paste.
- If you are not sure whether your idea is allowed, ask on Piazza!

3 Submission

You should submit two files to CATE. First, is a single document named **report.pdf**. This should contain your answers to Part 1, and also a one-page description of your implementation in Part 2. Second, is your **agent.py** file, and you should keep the name **agent.py**.

In your one-page description for Part 2, you are free to write what you like, but it should clearly and concisely explain what you have implemented. When describing your implementation, use line numbers (from your **agent.py** file) to help you describe what you have implemented. During marking of the document, methods you describe will be verified in your **agent.py** file. Therefore, if you do not include line numbers to make the relevant code sections easy to find, you may lose marks for a lack of clarity. You do not need to describe every single line of your code, but you should be able to write down the line number ranges for each section which you are describing in the document. Even if your implementation is effectively the same implementation as Part 1 of the coursework, you should still describe the main components, and in which lines of your code they appear.

Your **agent.py** will contain your Deep Q-learning implementation. When we receive this file, we will train and test your agent using the script **train_and_test.py**. Therefore, you need to make sure that your **agent.py** code is compatible with **train_and_test.py**. For example, if you run "**python3 train_and_test.py**" with the original **agent.py** in the same directory, you will see that the agent moves randomly around the environment during training, and will then move to the right during testing. After this, the result of the test will be printed out. You need to make sure that your **agent.py** file will also allow **train_and_test.py** to run the training and testing in this way. So, be sure that you have run **train_and_test.py** fully, both for training and testing, to ensure that your **agent.py** file will run as expected.

If there are any bugs in your **agent.py**, such that running **train_and_test.py** cannot train or test your agent, then you will have marks deducted; we will not spend time fixing your code for you. **Important:** It is better to have a simple solution that runs fully, rather than a complex solution which contains a minor bug, causing the training or testing to not run fully. Please also ensure that you have tested your code using Python 3.

4 Grading

This coursework will be graded using the following scheme. Each of the four sections in Part 1 is worth 15%, for a total of 60% for Part 1. The description of your implementation in Part 2 is worth 10%. The evaluation of your code in Part 2 is worth 30%.

For Part 1, the questions will be graded against a specific marking scheme we have. The answers do not have to be very long. Around three clear sentences for each question will be sufficient. For Part 2, the description of your implementation will typically be awarded a full 10% as long as you have described your implementation to a satisfactory level, and that your description aligns with the code you submitted.

For evaluation of your code in Part 2, we will be training and testing your code ourselves. **train_and_test.py** will be used to train your agent for 10 minutes, and then test your agent using its greedy policy. This evaluation will be done on an Intel Core i7-7820X CPU. You will receive a grade based on how close your agent is to the goal after an episode of 100 steps with the greedy policy. If your agent reaches a distance of less than 0.03 to the goal at any point, then you will receive bonus grades, based on how many testing steps it took the agent to reach the goal. Your code will be evaluated on three different, random environments, with grades averaged across the three. Therefore, you should ensure that your method is able to train the agent well across a range of different environments.

The remaining pages of the document show what the format of the **report.pdf** file should be. For Part 1, you are not allowed to write more than 2 pages (excluding the figures). For Part 2, you are not allowed to write more than 1 page.

Good Luck, and Have Fun!

Student Name

CID: 00123456

Reinforcement Learning (Second Half) Coursework

*This is an example of how to format your **report.pdf** file

Question 1 (i) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Question 1 (ii) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Question 2 (i) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Question 2 (ii) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Question 3 (i) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum

dolore eu fugiat nulla pariatur.

Question 3 (ii) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Question 4 (i) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Question 4 (ii) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Question 5 (i) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Question 5 (ii) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

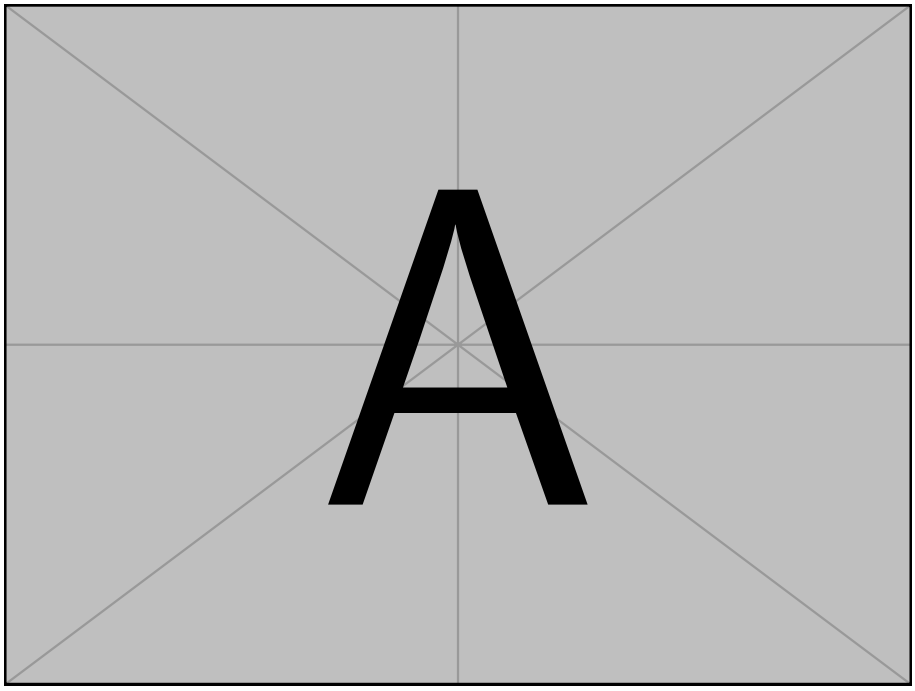


Figure 1 (a)

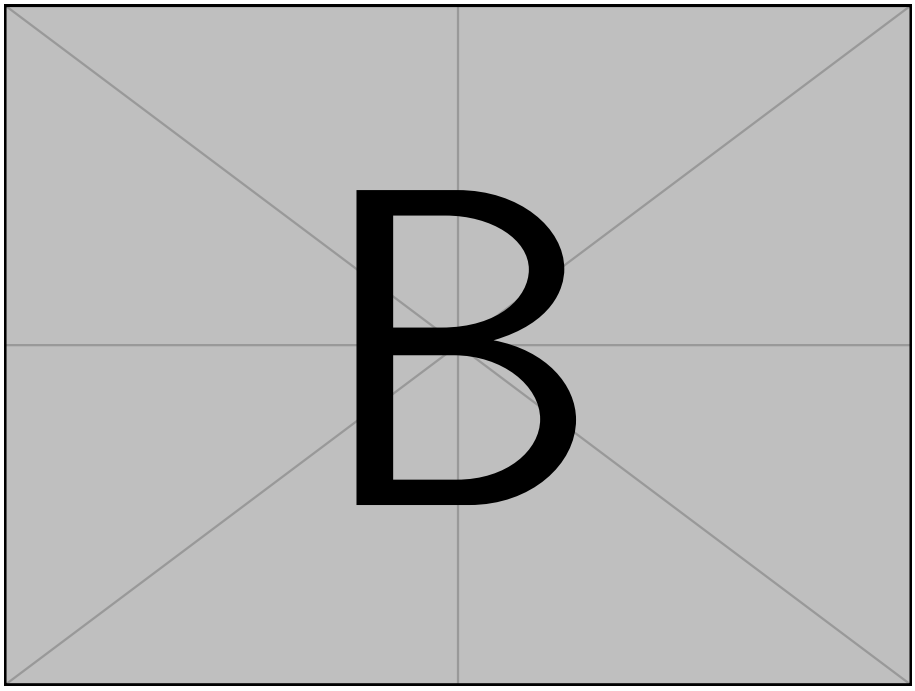


Figure 1 (b)

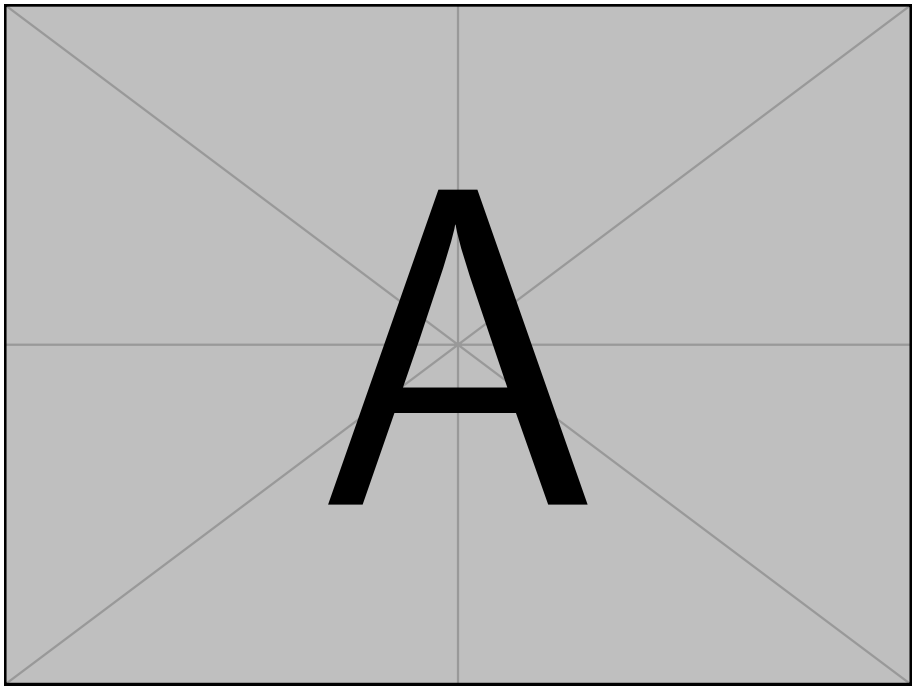


Figure 2 (a)

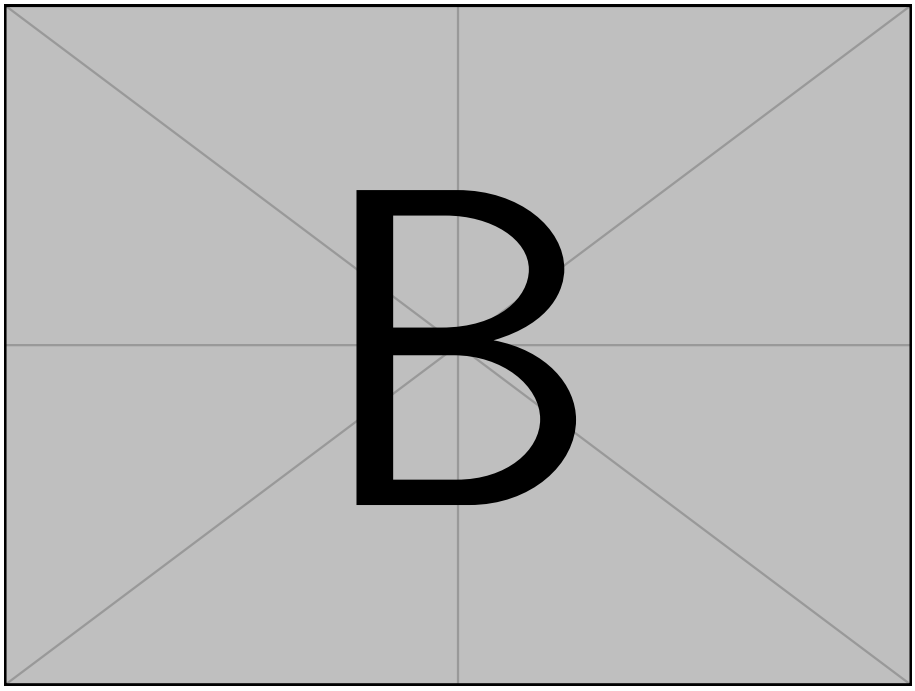


Figure 2 (b)

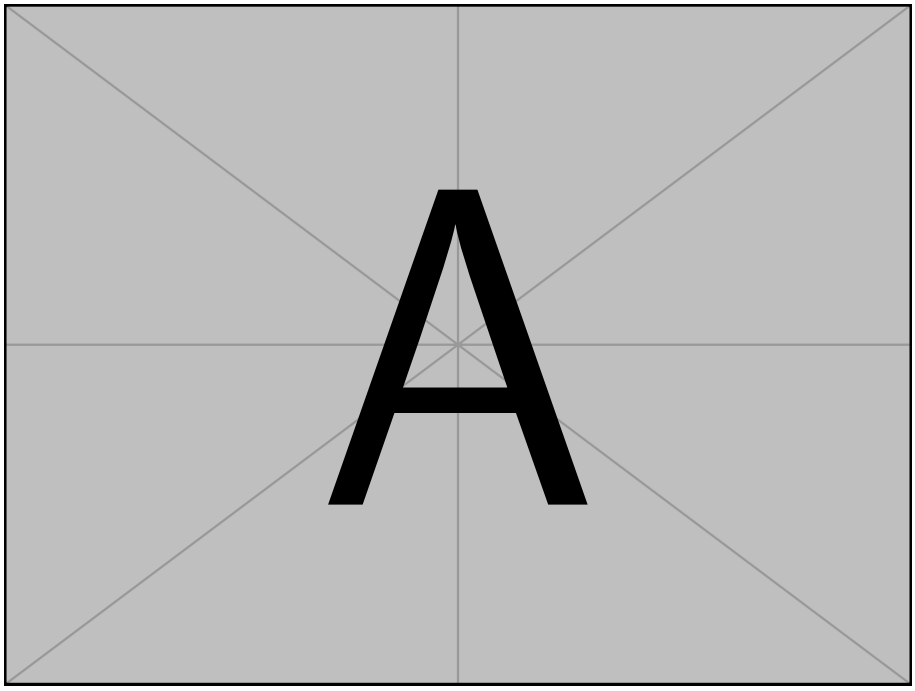


Figure 3 (a)

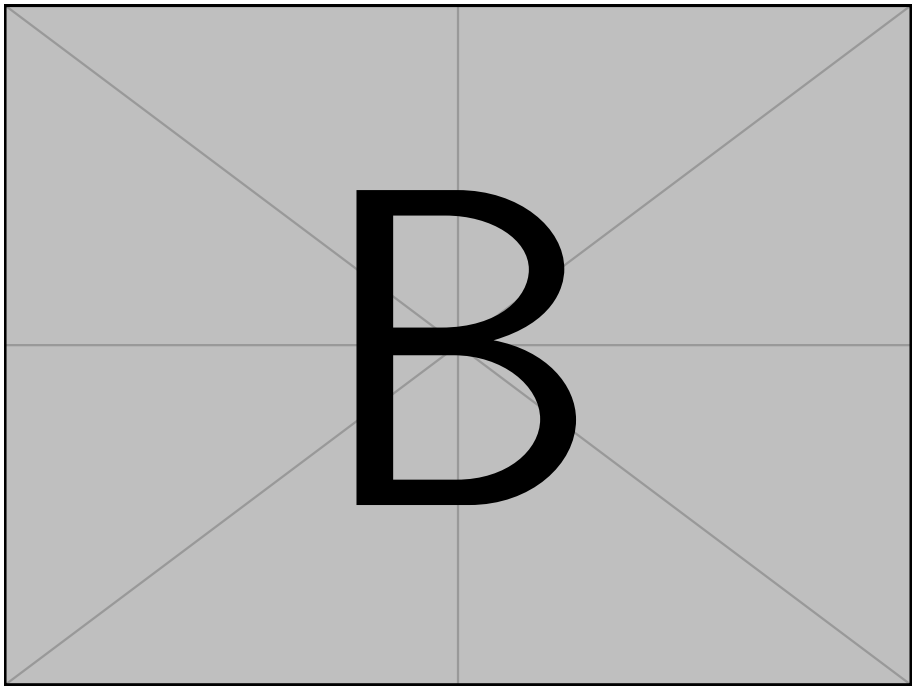


Figure 3 (b)

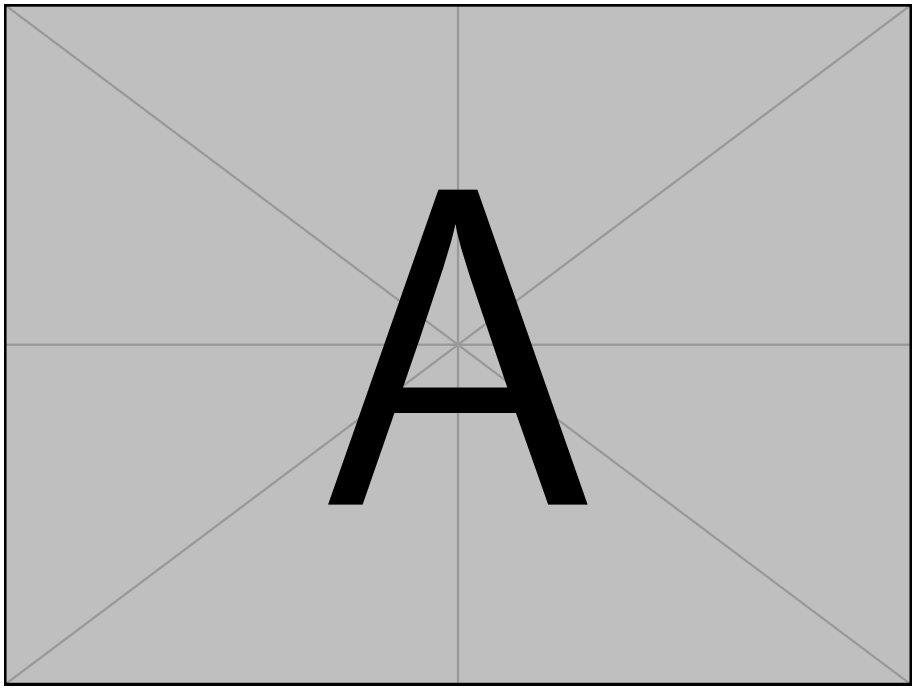


Figure 4 (a)

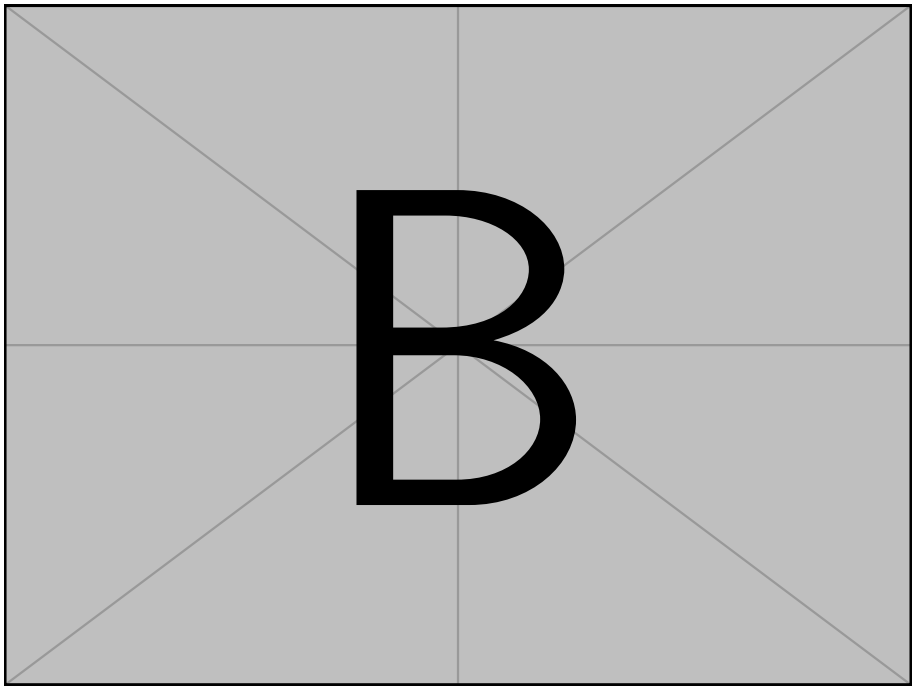


Figure 4 (b)

Description of Implementation for Part 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.