Research Assessment Outline

High Level Guidance for the assessment:

1. The code should be done in PyTorch
2. The goal is to replicate the experimental plots in the paper. **Note**: **The proposed major algorithm of the paper should be implemented.** The baseline algorithms are optional to implement.
3. To save the candidate's time, running the experiment on a small dataset is fine. The point of the assessment is to evaluate the candidate's capability of (1) understanding the paper, (2) Replicating the proposed major algorithm and (3) coding skills. Thus some proof-of-concept results are okay. (If the candidate's results are not as good as the result shown in the paper, which is fine, but it's a **bonus point** if the candidate can make their result better.)
4. If the candidate cannot find computing resources, they could try Google Colab to run the experiment
5. The candidate could prototype the project in google colab. However, the codebase for the final submission version is suggested to consist of multiple python code files, and looks like an established software.
6. The codebase should be bug free.
7. Code is strongly suggested to be written in an object oriented fashion. **Bonus point**: The candidate is encouraged to think about how to write good software.
8. The candidate could create a private repository in gitlab, and invite us by email. We will pull the code and run to test the code.
9. The standard timeline for this project is **2 weeks**. But it is okay to let us know if the candidate needs more time.
10. The project is not straightforward, and we want to use the project to test the candidate capability. However, we are here to help if the candidate needs help. Feel free to email your questions if you meet difficulties and cannot overcome them after critical thinking and trials and errors. We will reply to you to help.

Evaluation Criteria for Job Offer:

1. If the candidate can complete this research assessment, he/she will be very **likely** to get the offer.
2. We evaluate the candidate's codebase based on the following criteria:
    a. To what extent the codebase contains the complete/correct code for **the proposed major algorithm of the paper** (Don't be worried if you cannot fully complete coding the algorithm. We will also evaluate based on the part u finished.)
    b. If the codebase can output result plots which compare **the proposed major algorithm in the paper** vs baselines.
    c. If the codebase is bug free.
    d. The code review process with us in the interview for this assessment.

    e. The possible interaction between the candidate and us in the email for Q&A about the assessment. Don't hesitate to ask us questions! Asking questions won't hurt the evaluation, and instead, we want to see your critical thinking and technical communication skills!
    f. **Bonus point**: if **the proposed major algorithm** has competitive performance as what shown in the paper. (But don't feel frustrated if you can't replicate the paper's results. This is normal during a research process! We evaluate more on the process you try the assessment, but not on your results.)

Code Design Document:

11. We encourage the candidate to follow object oriented programming style. For example, the codebase could at least include the following skeleton classes, and each class should stay in a different python code file:
    a. Class for generating synthetic/simulated data
        i. Output: synthetic data for train and evaluation
    b. Class for real world data loading and preprocessing
        i. Input: raw data in the files for train and evaluation
        ii. Output: preprocessed real data for train and evaluation
    c. Class for training
        i. Contains a for loop for training, call algorithm each iteration
        ii. Input: a(i) or b(ii), i.e. synthetic data or preprocessed real data
        iii. Output: training metric files including training epoch index, loss and other training metrics
    d. Class for algorithms
        i. each type of algorithm should be defined in different class
    e. Class for evaluation
        i. Input: a(i) or b(ii), i.e. synthetic data or preprocessed real data
        ii. Output: evaluation metric files including metrics on evaluation dataset
    f. Class for plotting
        i. Input: training metric files or evaluation metric files
        ii. Output: plots
12. Feel free to add more classes according to the specific research problem.
13. We encourage the candidate to pay attention to prevent hard-coded issues when developing the software.
14. We encourage the candidate to use json as config files for defining parameters as input for each class.

The deliverables:

(1) slides that includes the technical points of the paper and the replicated results of the paper:
  (1.1) the methodology (the math),
  (1.2) related works
  (1.3) the algorithm
  (1.4) the experiment

(1.5) the result plots generated from the software built by the candidate, and the plots in the original paper
(1.6) results discussion
(1.7) **Suggestions**: It will be reader friendly if the candidate could make slides summarizing the important background, insights, graphs and results. Simply listing everything from the paper in the slides is not encouraged.
(2) the source code of a small software that implements the algorithms.
(3) a simple writeup in the similar style as the experiment section of the base paper
(3.1) clearly discuss the data used in the software.
(3.2) clearly discuss the algorithms implemented in the software. including: hyper parameters choice, training parameters, evaluation metrics etc.
(3.3) clearly discuss the plots. Including: x axis, y axis, legends.