CSE641 Deep Learning
Assignment-3 [70 marks]
Deadline: 5th April 11:59 PM

General Instructions:

1. Each group member must do at least one of the following tasks. But all should know the working of all the tasks. (Recommended: Divide the sections among yourselves.)
2. For Plagiarism, institute policies will be followed strictly.
3. **Make sure to use Pickle or any other library to save all your trained models. There will not be enough time during the demo to retrain your model. This is a strict requirement.** You must upload your best models on Classroom to reproduce your results during the demo. If you cannot reproduce your results during the demo, no marks will be given. You are advised to prepare a well-documented code file.
4. You must submit Output.pdf, Code files (including both .py files and .ipynb files), and models dumped after training. Mention your sample outputs in the output.pdf. The Output.pdf should enlist all hyperparameters clearly for quick reference.
5. Submit code, models, and output files in ZIP format with the following name: A1_Member1_Member2_Member3.zip

TASK I (Classification) [25 marks]

USE OF TRANSFORMER BASED ARCHITECTURES FOR TASK 1 IS NOT PERMITTED. Teams employing them will be disqualified from step 5 of evaluation.

Dataset: A training dataset of sequential and permutated MNIST is uploaded on google drive : https://drive.google.com/drive/folders/1DnO7AN7L6ch0pJ4jr-EAVMTb--9-Icda?usp=sharing

The folder contains two pickle files : sequential and permuted MNIST. With the following format: {'train' : {'images' : train_images, 'labels' : train_labels}}. Test dataset will be provided to you during your demo.

Sequential and Permuted MNIST:
MNIST has 28 x 28 pixel images (of a handwritten digit) into one of ten digits (0 to 9). We first turn it into a stream of 784 (28x28) individual pixels, presented to the network one at a time. We then apply a fixed permutation to all of the pixel sequences. Overall the train dataset has an input 784 sequential (unordered) for a given image per label. Aim is to input this sequence and predict class labels.

Task: Develop a RNN-based model to classify the digits (**ONE EACH FOR SEQ AND PERMUTATED DATASET)**. Perform your own splitting of train data in training, validation sets with an 80:20 ratio (random stratified) and the **following setup for the 2 datasets**:

1. Visualize 3 random images with their corresponding sequence. **[3 marks]**
2. Propose and implement an RNN based architecture to predict the digit in the image. You can incorporate all the techniques you have studied so far of weight initialization, attention, loss functions, regularization etc as long as your base model is RNN based and your final layer is a classification head. No transformer based system allowed. **[10 marks]**
3. Show plots of validation and training loss vs. number of epochs **[1 mark]**
4. Report the class-wise F1 for your validation set. **[1 mark]**
5. Prepare an inference pipeline that during the demo will load your best model and the test dataset (we will provide) and run it live. Each team will be ranked based on the test set F1. The marks out of 10 will be based on your team's rank. We can ask you to run either of the systems (SEQ or PERMUTED). **[10 marks]**

Task 2 (Translation) [30 marks]

Dataset:
WNT16 (german-english) dataset. https://huggingface.co/datasets/wmt16/viewer/de-en
Loading the model using HuggingFace: load_dataset('wmt16', 'de-en')

Task: Machine translation from English to german.
Model A: LSTM
Model B: LSTM + GLOBAL ATTENTION
Model C: LSTM + LOCAL ATTENTION
Refer to this paper for global and local attention concepts: https://aclanthology.org/D15-1166/

Note : For dataset loading, students can use HuggingFace dataset for maintaining the train-test consistency. Also for comparison use the same set of embeddings for each model setup. The embedding layer has to be trainable in all 3 models.

1. Visualise the dataset using five different techniques (sequence length, frequency of words, mean token length, word cloud, etc) to show how the 2 languages differ. **[6 marks]**
2. Implement Model A with: one with non-contextualised initialised embeddings (or random embeddings) + LSTM based **[5 marks]**
3. Implement Model B with: one with non-contextualised initialised embeddings (or random embeddings) + LSTM + global attention based **[5 marks]**
4. Implement Model B with: one with non-contextualised initialised embeddings (or random embeddings) + LSTM + local attention based **[5 marks]**
5. Show plots of validation and training loss vs. number of epochs for Models A, B, C. **[3 marks]**
6. Report the overall Rouge-L score, Bleu-1,2 score for the test set. (metrics available in Huggingface metric) for Models A, B, C. **[4 marks]**
7. Out of the 3 models, which setup performed best and why? **[2 marks]**

Task 3 (Transformer) [15 marks]

Dataset:
WNT16 (german-english) dataset. https://huggingface.co/datasets/wmt16/viewer/de-en
Loading the model using huggingface: load_dataset('wmt16', 'de-en'). We use the same dataset as in Task 2, but this time we fine-tune transformer based model. Refer tutorial: https://huggingface.co/docs/transformers/tasks/translation

1. Atleast one layer of your fine-tuined model should be set to trainable. You are free to employ any model (even distilled ones.). **Huggingface models already tuned on WNT16 dataset cannot be used.** Special case should be taken for overfitting, include appropriate measures if overfitting is observed. **[10 marks]**
2. Show plots of validation and training loss vs. number of epochs for the model. **[1 marks]**
3. Report the overall Rouge-L score, Bleu-1,2 score for the test set. (metrics available in Huggingface metric) for the model. **[2 marks]**
4. What can be said about the quality of output generated by models in Task 2 vs Task 3. **[2 marks]**