Homework 6 – Deep Neural Networks (CS/DS 541, Murai, Fall 2022)

You may complete this homework assignment in teams of 2-3 people.

1 Neural Machine Translation [35 points]

In this project, you will train a seq2seq model to translate from English to French (see `eng-fra.txt.zip` on Canvas) using 3 different strategies. You can use either TensorFlow (`https://www.tensorflow.org/text/tutorials/nmt_with_attention`) or PyTorch (`https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html`). In either case, you can download a GRU-based encoder-decoder jupyter notebook as starter code (which gives you the advantage that the text preprocessing steps are already implemented), but the pytorch starter is easier to follow since it uses less of the advanced features of the framework.

You will need to make the following changes to the existing code:

1. You must set up the training pairs so that the model can learn how to translate from English to French. The pytorch starter translates from French to English, while the TensorFlow starter is using a different dataset than the one on Canvas.

2. Randomly partition the data into 80% training, and 20% testing.

3. **Version 1: supervised training.** Train the network for at least 20k epochs.

4. **Version 2: pretraining as autoencoder.** Use unsupervised training to pretrain the encoder. Specifically, set the pairs so that you have two of the same phrase (I am test\tI am test), in order to train autoencoder. Try this:

   - Train as an autoencoder
   - Save only the Encoder network and freeze its parameters (they must not be updated during training)
   - Train a new Decoder for translation from there

5. **Version 3: using pretrained embeddings for English in the Encoder.** Now you will go back to Version 1 and replace the Embedding layer of the encoder by pretrained embeddings. To keep the training time small, use GloVe 1B with embedding size 100 (or 50) . These embeddings have been trained on a corpus much larger than the dataset you are using now, so you must either keep the embedding layer frozen or only fine tune them (in the latter case, apply a much smaller learning rate to those parameters).

   When you download pretrained embeddings, they include a dictionary that maps from words to indexes (see `stoi`). These indexes can be used to retrieve the vector embeddings (e.g., 'park' $\rightarrow$ 625 $\rightarrow$ $[0.1513, 0.2098, 0.4437, \ldots]$). However, it does not include a mapping for "unknown" words. Solution: create a new vocabulary (token-to-index mapping) based on the GloVe dictionary, but including the special token "$\langle$unk$\rangle$", either at the beginning or at the end of the index. If created at the beginning (resp. at the end), prepend (resp. append) a zero vector (with shape `(1,embedding_size)` to the pretrained embedding vectors.

   Hint: remember to use your new vocabulary to convert sentences to indexes only in the Encoder. For pytorch, check `torchtext.vocab.GloVe` and `torchtext.vocab.vocab`.

For each of the three versions, you need to either plot the testing loss or to report its values for epochs $T - 15k, T - 10k, T - 5k$ and T, where T is the total number of epochs.

In addition to your Python code (`homework6_WPIUSERNAME1.py` or `homework6_WPIUSERNAME1_WPIUSERNAME2.py` for teams), create a PDF file (`homework6_WPIUSERNAME1.pdf` or `homework6_WPIUSERNAME1_WPIUSERNAME2.pdf` for teams) containing the screenshots described above. **Please submit both the PDF and Python files in a single Zip file.** Also, please do **not** submit the `eng-fra.txt.zip` file!