The lead TA for this assignment is Risako Owan (owan0002@umn.edu). Please communicate with the lead TA via Slack or office hours. All questions MUST be discussed in the homework channel (#HW1). Questions through emails, Direct Messages, and other channels will not be answered.

The goal of this assignment is to make sure you get used to machine learning (ML) programming with PyTorch and implement a simple neural network based text classifier. By following the steps below, you can train your own classifier from scratch. This assignment will be the basis of your next assignment and class project that require more advanced Python/PyTorch programming, analysis, and deep learning knowledge.

First, carefully read tutorial slides and notebooks presented by the TAs on Scikit-learn and Pytorch, and try to run the same scripts on your local machine with Jupyter Notebooks in Google Colab (NOTE: TA tutorials will be released during the week of Jan 27 and will be updated in this document as they are completed). In the future, we will be running jobs that will require Google Colab Pro. For now, this assignment is small enough such that the free tier of Google Colab should be sufficient. For future assignments, we will be switching to Colab Pro and providing documentation for you to set this up.

In the tutorials, we developed a multi-layer perceptron (MLP) based binary classifier for predicting whether a tweet is about a real disaster or not. Now, let's build a simple text classifier using PyTorch. In this homework, you will simply stack one more layer to your MLP and develop **a two-layer MLP text classifier** using Pytorch on **a new dataset**.

Step 1: Choose a dataset from TorchText.datasets

You can choose any dataset from PyTorch's torchtext. If you are using a laptop or local machine, choose a small dataset, such as IMDb (TRAIN/TEST: 25000/25000 samples) or SST2 (TRAIN/TEST: 67349/1821 samples). The TRAIN split is used to train your model, and TEST split is used to evaluate the trained model's performance. Your TEST set must not be used in any way during training. Below is an example script for loading the original IMDB dataset.

```
# import datasets
from torchtext.datasets import IMDB

train_iter = IMDB(split='train')

def tokenize(label, line):
    return line.split()

tokens = []
for label, line in train_iter:
    tokens += tokenize(label, line)
```

Step 2: Stack two layers of MLP

You implemented a one-layer MLP classifier in the tutorial. This homework simply requires adding one more layer to your one-layer MLP, as shown in the pseudo-code below. Note that you added a 100-dimensional intermediate layer between the embedding layer and the last class layer by setting num_layer = 100.

```
class MLP(nn.Module):
    def __init__(self, vocab_size, embed_dim, num_class):
        super().__init__()
        self.embedding = nn.EmbeddingBag(vocab_size, embed_dim, sparse=True)
        num_layer = 100
```

```
        # self.fc = nn.Linear(embed_dim, num_class)
        self.fc1 = nn.Linear(embed_dim, num_layer)
        self.fc2 = nn.Linear(num_layer, num_class)

        # initialize the weights
        self.init_weights()

    def init_weights(self):
        ..

    def forward(self, text, offsets):
        embedded = self.embedding(text, offsets)
        return self.fc2(self.fc1(embedded))
```

Step 3: Training and Analysis

Following the PyTorch tutorial, you can start training your model after loading a dataset and designing a classifier model. You can check out the following questions throughout different experiments:

- On the test set, what is the accuracy? Please draw a graph of test performance at each epoch of training.
- What is the performance (i.e., accuracy on the test set) of a two-layer MLP compared to a single-layer MLP?
- What happens in the performance if you increase the number of dimensions of the intermediate layer from 100 to 200?
- Do you have a look at the test set samples that are incorrectly predicted by the model? Why are the errors occurring?

Note: When using Google Colab, please avoid selecting A100 GPUs as they rapidly consume your allotted compute units. Instead, opt for L4 or T4 GPUs for a more efficient use of resources.

Deliverables

Your Jupyter notebook in Google Colab should be submitted. Your notebook file should contain all answers to the questions above, showing test accuracy and curves, performance changes in various setups (e.g., different number of intermediate layer sizes, number of layers), and incorrectly predicted samples with justification. Please submit your notebook file to Canvas by Feb 6, 11:59pm.

Rubric (12 points)

- Code looks good, i.e., each cell in the Jupyter Notebook runs without error and outputs intended results. (+3)
- Description of the task, dataset, and hardware used (+1)
- Test accuracy is reported (+1)
- Graph of test performance on each epoch (+1)
- Explain the performance comparison between two-layer MLP and single-layer MLP (+2)
- Shows and explains the results of changing the number of dimension of the intermediate layer from 100 to 200 (+2)
- Includes hyperparameters used in the experiment (+1)
- Error analyses on the test set samples (+1)