ECE 689 Spring 2025 - HW3

The Jupyter Notebook template for this homework is available on Canvas. Please upload your Jupyter Notebook file (.ipynb file) with all outputs and necessary derivations exported as a PDF or HTML to Canvas. All necessary coding documentation can be found in `https://pytorch.org/docs/stable/index.html`. **We highly encourage you to submit your derivation in LaTeX. If you choose to submit a hand-written derivation, please make sure your derivation is legible. If you have hand-written solutions, please scan/take a photo and insert them into their designated space.** You can follow this tutorial to insert images to .ipynb files: `https://www.geeksforgeeks.org/insert-image-in-a-jupyter-notebook/`

1 Problem 1: Transformer for Translation (50 pts)

Here, we implement transformers for neural machine translation (NMT), such as turning "Hello world" to "Salut le monde". You are going to follow the following steps:

1. **Loading and Preparing the Data**. In this problem, we consider the English-French dataset provided in the "en-ft.txt". This data can also be found at: `https://github.com/jeffprosise/Applied-Machine-Learning/tree/main/Chapter%2013/Data`. In this dataset, each line contains: an English phrase, the equivalent French phrase, and an attribution identifying where the translation came from.

2. **Building and Training the Language Model**. Implement a transformer **from scratch** in Pytorch. Do not use any pre-trained transformer model. Train the model with the provided English-French dataset. Hint: `https://github.com/gordicaleksa/pytorch-original-transformer`.

3. **Testing the Language Model**. Using the model to translate English sentences to French.

Additionally, for deliverables, please plot your training and validation accuracy. The x-axis should be epoch, the y-axis should be your translation accuracy. You can find the implementation example of neural machine translation here: `https://github.com/jeffprosise/Applied-Machine-Learning/blob/main/Chapter%2013/Neural%20Machine%20Translation%20(Transformer).ipynb`. This model achieves 85% accuracy after 14 epochs. Note that you do not have to achieve the same performance to get full marks.

2 Problem 2: BERT for Sentiment Analysis (50 pts)

For the last problem, we are going to learn how to use the Hugging Face library to train a simple BERT classifier for sentiment analysis. Particularly, we will use the IMDB dataset. You can download the dataset via `https://datasets.imdbws.com/`. You can also find the dataset from Hugging Face using the following commands:

```python
# Import IMDB Dataset
from datasets import load_dataset
imdb = load_dataset("imdb")
```

In this problem, you can use pre-trained BERT model. To access the pre-trained model, you can use the following commands:

```python
# Load pre-trained BERT
from transformers import BertForSequenceClassification
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels =
    len(label_dict), output_attentions = False, output_hidden_states = False)
```

The pre-trained model is used to extracted meaningful features from the data. These features are then used to train a classifier. After obtaining the pre-trained model, you need to design a classifier layer for the task of sentiment analysis. The classifier should have a minimum of 4 layers. To reduce training complexity, you have to freeze the weight of the pre-trained BERT model and only train the classifier using the provided data. Additionally, you can find examples of sentiment analysis here:

- `https://huggingface.co/blog/sentiment-analysis-python`
- `https://github.com/baotramduong/Twitter-Sentiment-Analysis-with-Deep-Learning-using-BERT/blob/main/Notebook.ipynb`