CIS 472/572, Spring 2022
Homework 4 (Programming): Deep Neural Networks
DUE: May 29th, 2022 at 11:59pm (via Canvas)

1 Overview

Sentiment Analysis (SA) is a task in Natural Language Processing (NLP). SA involves determining whether an input text is positive, negative, or neutral. Sentiment analysis is often used to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs.

In this project, you are going to train deep neural networks and evaluate it on a sentiment analysis dataset, namely **IMDB**. This dataset contains 50,000 data points, each data point consists of an IMDB comment and a label. The comment is a paragraph of several sentences derived from the IMDB website. A label is either **positive** or **negative**.

> **Positive**: *"A solid, if unremarkable film. Matthau, as Einstein, was wonderful. My favorite part, and the only thing that would make me go out of my way to see this again, was the wonderful scene with the physicists playing badmitton, I loved the sweaters and the conversation while they waited for Robbins to retrieve the birdie."*
>
> **Negative**: *"This picture started out with good intentions, Bacon the scientist out to test the theory of invisibility, and Shue is cute as usual in her role. It all falls apart after that, it's your typical Hollywood thriller now, filmed on a soundstage with special effects galore, minus any kind of humour, wit or soul. In other words, don't waste your time watching this."*

In this assignment, you are going to practice in a real deep learning project. Your activities include:

- Tuning hyper-parameters to reach the highest development accuracy.
- Visualizing the training and finetuning processes.
- Writing a technical report of your tuning experience.

2 Resources

You are provided:

- GLoVe embeddings (50D, 100D, 200D, 300D)
- A training data file of 25,000 samples.
- A development data file of 25,000 samples.
- A starter Jupiter notebook with a complete algorithm.

The GLoVe embeddings are downloadable at `http://nlp.uoregon.edu/download/embeddings/`

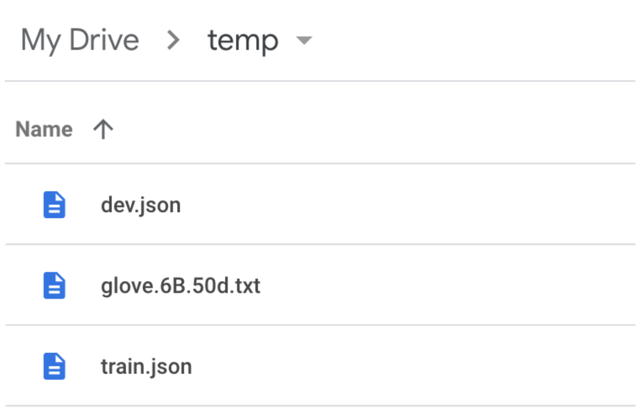
The training data, development data, and jupyter notebook are downloadable at `https://ix.cs.uoregon.edu/~vietl/cis472/hw4//`

3 Setting

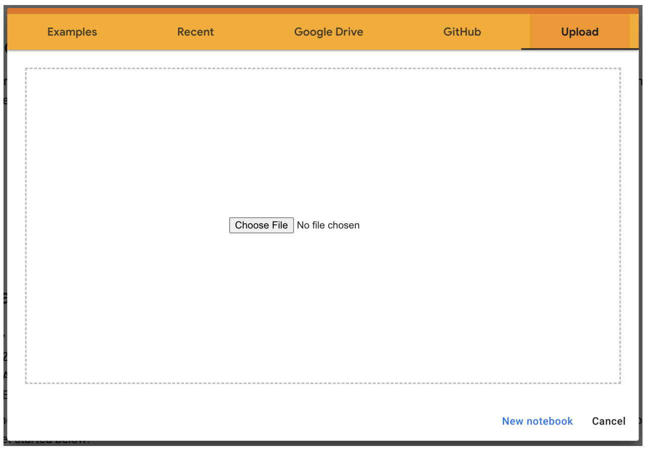
You will use Google Colab for this excercise. Google Colab provides free computer with accelerators such as GPU and TPU for training deep neural network. You will need a Google Drive account to do this assignment.

- Step 1: Create a folder named **temp** in your Google Drive. Upload the GLoVe embeddings and training/development data to the **temp** folder. See Figure 1a.

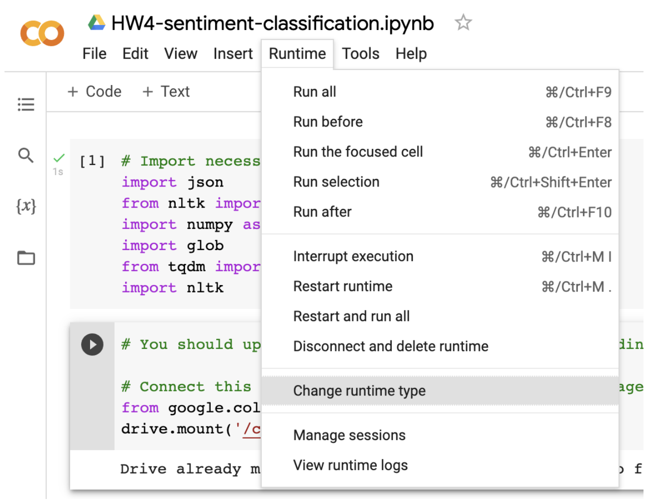
- Step 2: Upload the Jupyter Notebook to Google Colab (`https://colab.research.google.com/`). See Figure 1b and 1d. After this, Google Colab will open the starter code immediately. You should bookmark the link to the file for later access. The file is also automatically stored in the root of your Google Drive, so you can access the starter code through Google Drive too.
- Step 3: Configure runtime to use GPU instead of the default CPU. See Figure 1c and
- Step 4: Run the code. In the second cell of the starter code, you will need to mount your Google Drive to the runtime environment (aka. allow the Google Colab machine to read the files you just upload to Google Drive).



(a) Upload data to Google Drive.



(b) Upload starter code (Jupyter Notebook) to Google Colab


HW4-sentiment-classification.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
Run all ⌘/Ctrl+F9
Run before ⌘/Ctrl+F8
Run the focused cell ⌘/Ctrl+Enter
Run selection ⌘/Ctrl+Shift+Enter
Run after ⌘/Ctrl+F10
Interrupt execution ⌘/Ctrl+M I
Restart runtime ⌘/Ctrl+M .
Restart and run all
Disconnect and delete runtime
Change runtime type
Manage sessions
View runtime logs

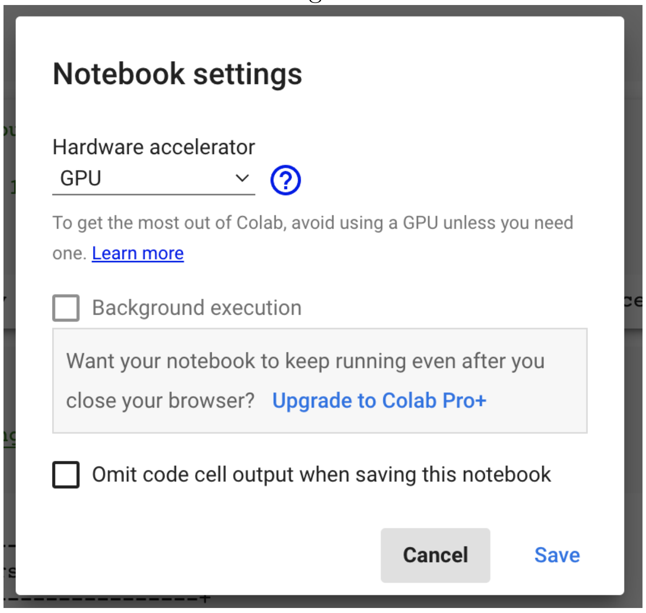
```
[1] # Import necess
    import json
    from nltk impor
    import numpy as
    import glob
    from tqdm impor
    import nltk

    # You should up

    # Connect this
    from google.col
    drive.mount('/c

    Drive already m
```

(c) Change runtime enviroment.


Notebook settings
Hardware accelerator
GPU
To get the most out of Colab, avoid using a GPU unless you need one. Learn more
Background execution
Want your notebook to keep running even after you close your browser? Upgrade to Colab Pro+
Omit code cell output when saving this notebook
Cancel
Save

(d) Select GPU.

Figure 1: How to upload and configure runtime environment.

4 Requirements

4.1 Finetuning

Hyperparameter tuning is a very important skill in machine learning. The starter code provides a working implementation with a set of initial hyperparameters. However, this set does not give the best development accuracy. You need to adjust these values and train the model to achieve your highest accuracy. You should finetune at least 3 hyper-parameters. Here are some hyper-parameters that you may choose.

- Number of training epochs
- Batch size
- Optimizer type
- Learning rate
- Maximum sentence length
- Dimension of hidden layers
- Activation function
- Regularization

Once you are done with tuning, for each hyper-parameter that you have tuned, you should present the values of the hyper-parameter that you have tried and the corresponding performance of the model on the development data (i.e., using a table or a graph). Please report the performance in accuracy percentage (2 digits precision).

Tips:

- Printing loss/accuracy during training will help you monitor the training process.
- Calibrating these parameters is not an easy work. Blindly adjusting might not lead to a good performance. You should have a strategy of which values you are going to try. Two most common strategies are grid search and hill-climbing. For simplicity, we suggest fine-tuning each hyper-parameter at a time. Once you finish the tuning of one hyper-parameter, you can fix that hyper-parameter with its best value (based on development data) and continue with the tuning of another hyper-parameter.
- Batch size and sentence length is relative to the amount of consumed GPU memory. You might run into an "Out of memory error" if these hyper-parameters are too high.
- Also, you can use any version of the GloVe embeddings (i.e., 50D, 100D, 200D, or 300D); however, 300D is often used in practice and might lead to best performance. In your report, please specify which version of GloVe you use in the experiments.
- The provided code is written using PyTorch library. The document of PyTorch is available here: `https://pytorch.org/docs/stable/index.html`.
- The provided model is a simplified version of the convolutional neural network for sentence classification by Yoon Kim. The paper is accessible at `https://arxiv.org/pdf/1408.5882.pdf`.

4.2 Visualization

Visualization is a very handy tool to examine the training progress. In some cases, visualization can help identifying severe faults in your code. Visualization is often done for typical performance mesaures, including training loss, training accuracy, development loss, development accuracy over the time (i.e.,. the number of parameter updates or epochs).

From the training session with the best hyper-parameter setting you found, you should:

- Provide a graph/table to capture training and development loss over time (i.e., learning curves for loss function over the number of parameter updates or epochs).
- Provide a graph/table to capture training and development accuracy over time (i.e., learning curves for accuracy over the number of parameter updates or epochs).
- Provide any findings/conclusions you have learnt in the tuning and visualization process.

5 Turn in

You should submit a single pdf file through Canvas with the following information (in order):

- Your name and your UO ID.
- A link to your Google Colab file. Make sure that the instructor and GE have access to your Google Colab by enabling document sharing.
- Credit: Full names of UO students who helped you (if any).
- Reference: Paper titles and URL(s) to the sources that you consult for this assignment (if any).
- Reports required in sections 4.1 and 4.2.

The PDF should be concise, no longer than **4 US-letter pages**, font size 12.

6 Rubric

Any error (intentionally or not) that leads to a mixture of training-development data/predictions/logs will result in 0 points for the finetuning part and partial credit for other related parts.

Task	Requirement	Points
Finetuning	Development performance evaluation (full score)	40 points
	$0 < acc \leq 70\%$	5
	$70\% < acc \leq 75\%$	15
	$75\% < acc \leq 77\%$	20
	$77\% < acc \leq 79\%$	25
	$79\% < acc \leq 81\%$	30
	$81\% < acc < 100\%$	40
Visualization & Report	Attempted values and corresponding development performance for three different hyper-parameters	30 points
	Learning curves for loss function on training and development data	15 points
	Learning curves for accuracy on training and development data	15 points