Programming Assignment # 2

Due: Friday, November 14, 2025, 11:59 p.m.
Total Points: 100

This assignment provides hands-on experience in building and experimenting with deep learning models for Voice Activity Detection (VAD) using PyTorch. Voice Activity Detection is the task of identifying speech segments in audio signals. In this assignment, you will train models on noisy speech data using spectrogram-based features. You will implement and compare different architectures for sequence modeling, including a:

- LSTM model
- Bi-directional LSTM model
- CNN + LSTM hybrid model

You must use Python and PyTorch for implementation. A PyTorch DNN example is provided in Canvas; you may implement it differently if preferred. Refer to the official PyTorch documentation for all functions: https://docs.pytorch.org/docs/stable/nn.html.

Read all instructions carefully and check MS Teams regularly for updates. Public questions on Teams must not include any part of your solution. For help with your solution, message the instructor or TA privately. General questions should be posted in the `programming2` channel.

This is an individual assignment. You must submit your own work and may not use external resources (e.g., websites, books, blogs, papers, or other people). **Submit your assignment by committing your code to your private GitHub repository within our organization before the deadline.** Create a folder named `pa2` and commit all related files there. Shell code is available in the Shell-Codes repo, which includes three python files and one notebook file. The notebook is for running the experiments and saving the results. You only need to work on the models.py and experiments.py code, i.e., **MUST NOT edit dataloader.py and run.ipynb**. **Run your notebook before committing so results are visible.** Unexecuted notebooks will receive a zero. Submit early and often to avoid last-minute issues.

Deliverables: Python files with code and solutions, including answers in comments or markdown cells. Submit to the `pa2` folder in your private class repository. Grading criteria include completeness, correctness, code efficiency (e.g., use of linear algebra), and readability (e.g., comments).

Assignments must be submitted on time for full credit. Late submissions incur a 10% penalty per calendar day, up to a maximum of 3 days. Extensions will not be granted for unexcused reasons.

Question 1. [100 POINTS]

Develop a three-layer long short-term memory (LSTM) neural network using PyTorch that performs voice activity detection on various audio signals. Voice activity detection seeks to distinguish active regions of speech from those of non-speech or silence. It is important for many speech problems, including automatic speech recognition, speaker recognition, and noise reduction. Prominent researcher Andrew Ng, mentioned how this is one of the major problems that needs to be addressed for communicating with chatbots[1].

Please adhere strictly to the provided shell code and notebook when implementing your solutions. The output format must remain unchanged, and no modifications to the code are permitted unless explicitly instructed to do so. You should only update the sections marked with "TODO." While independent experimentation is encouraged, any alterations not specifically requested must be removed prior to submission.

Tasks

Part 1: Data Preparation: refer to the provided dataloader.py file

This dataloader transforms continuous audio files into small, labeled Mel-spectrogram windows that the model can train on. **You do not need to modify this file**. More details are provided in the code, if you are curious. **A link to the data can be found in the Canvas assignment.**

Part 2: Model Implementations

Implement the following architectures in PyTorch:

1. **LSTM-based VAD model** Refer to the official PyTorch documentation for LSTM functions: https://docs.pytorch.org/docs/stable/generated/torch.nn.LSTM.html:
   - Two LSTM layers (e.g., 128 units each).
   - Fully connected output layer with **Softmax activation**.
   - Loss: Cross Entropy. see
     - https://docs.pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html
2. **Bi-directional LSTM-based VAD model**.
3. **CNN + LSTM hybrid model**:
   - CNN layers for feature extraction:
     - see https://docs.pytorch.org/docs/stable/generated/torch.nn.Conv2d.html
     - Conv2D layer with 16 filters, kernel size 3, stride 1, padding 1.
     - ReLU activation and MaxPooling.
     - Second Conv2D layer with 32 filters, kernel size 3.
   - Feed CNN output into LSTM or Bi-LSTM layers.
   - Fully connected layer with Softmax activation.

[1] https://x.com/AndrewYNg/status/1897776017873465635

Part 3: Experiments

For each architecture:

- Train and evaluate using sequence lengths of 10, 25, 50, and 100 frames.
- Experiment with:
  - Number of layers (1 vs. 2 vs. 3).
  - Number of hidden units (64 vs. 128 vs. 256).
  - Record learning curves and average test accuracy
- Record learning curves and average test accuracy.

The baseline system should use the following:

1. Xavier initialization based on uniform distribution (see https:// pytorch.org/ docs/ stable/ nn.init.html)
2. mini-batch gradient descent, using a (mini) batch size of 64
3. early stopping with cross-entropy loss function: Stop training when the validation loss does not improve for 3 consecutive epochs
4. A learning rate scheduler using ReduceLROnPlateau, factor=0.5, patience=2.

- see https://docs.pytorch.org/docs/stable/generated/torch.optim.lr$_{s}cheduler.ReduceLROnPlateau.html$

5. Initial learning rate of 1e-3
6. number of epochs: 20

After your basic programming is done, do the following:

1. Generate learning curves for the validation and training setes. Discuss what trends you observe in performance.
2. Evaluate the model on the testing set and report accuracy, along with precision and recall.

Analysis

Compare performance across architectures and hyperparameter settings. Discuss:

- Impact of sequence length.
- Effect of bidirectionality.
- Benefit of CNN feature extraction.
- Impact of hidden units and number of layers

Deliverables

- Jupyter Notebook and Python files with code, plots, and discussion.
- Submit to GitHub under `pa2`.