CS 753: Assignment #2

Instructor: Preethi Jyothi Email: pjyothi@cse.iitb.ac.in

March 7, 2021

Instructions: This assignment is due on or before 11.59 pm on March 21st, 2021. No grace period will be granted for this assignment. The submission portal on Moodle will be closed after 11.59 pm on March 21.

- This is an individual assignment. You will be building end-to-end ASR systems using the PyTorch library for isolated-word recognition of speech commands.
- Click here for a detailed structure of your final submission directory. It is **very important that you do not deviate from the specified structure.** Deviations from the specified structure will be penalized. All the files that need to be submitted are highlighed in red below; all the submitted files will be within the parent directory `submission/`. Compress your submission directory using the command: `tar -cvzf submission.tgz submission` and upload `submission.tgz` to Moodle.

Speech Command Recognition using End-to-End ASR

Speech Commands is an audio dataset consisting of isolated-word commands spoken by different speakers. (All the audio files are about 1 sec long.)

Task 0: Set up a baseline system [2 points]

Follow the PyTorch tutorial on audio command recognition that takes you through the entire recognition pipeline: Format the dataset, create train/test splits, build a convolutional neural network to train the model and finally evaluate it on held-out examples.

This tutorial can be run on Google's Colab. These runs will significantly benefit from access to GPUs. The second paragraph in the tutorial describes how to enable GPU access in your Colab account.

In the first cell under "Importing the Dataset", you will find two lists named `train_set` and `test_set`. Confirm that the length of `train_set` and `test_set` are 84843 and 11005, respectively. Next, select elements from these lists based on the indices in the following two files to construct modified training and test data sets, respectively: https://www.cse.iitb.ac.in/~pjyothi/cs753/train_list.txt and https://www.cse.iitb.ac.in/~pjyothi/cs753/test_list.txt. Make sure that the resulting `train_set` and `test_set` lengths are 12000 and 4000, respectively. (The audio from the first training file will be someone saying "backward" and the audio from the last training file will be someone saying "zero".) Set the number of training epochs `n_epoch` to 20. The entire training process should take less than a minute and the final test accuracies will be in the range $66\% - 71\%$.

What to submit: Submit a text file `task0/accuracy.txt` that contains the test accuracy on `test_set`. Submit a file `task0/notebook.txt` with a link to your Colab notebook that we can check and run end-to-end to reproduce your results. `task0/accuracy.txt` and `task0/notebook.txt` should each contain only a single line with the test accuracy (up to two decimal points, no $\%$ symbol) and a link to your notebook, respectively. All subsequent tasks that require such `.txt` files should follow the same structure. The very last line in your notebook should print the final test accuracy.

Task 1: LSTM-based ASR [10 points]

Question 1

Instead of the convolutional M5 network you used in Task 0 that directly processes raw audio data, design an LSTM-based recurrent neural network that unrolls across the length of the input audio and uses **Mel Frequency Cepstral Coefficients** (MFCC) features as inputs at each timeframe. Here, as in Task 0, you have a categorical classification problem and the loss will be a negative log-likelihood loss.

1. You should use the same `train_set` and `test_set` from Task 0.
2. You cannot use more than three layers in your LSTM network and each layer cannot have more than 400 hidden units. Use the Adam optimizer with a learning rate and a learning rate schedule of your own choosing.
3. You may refer to the following notebook to set up an LSTM network for speech command recognition. As mentioned in this notebook, use 12 MFCC features for your input.

Play around with the network architecture and various hyperparameters to derive the best possible performance on `test_set`.

What to submit: Submit a text file `task1/accuracy.txt` with your best test accuracy using your tuned hyperparameters and design choices. Submit a link to your notebook in `task1/notebook.txt` that we can run on Colab to reproduce your results. As with Task 0, this should run end-to-end to produce the final test results. (All the hyperparameters should be set to the best values.) The very last line in your notebook should print the final test accuracy.

Task 2: CTC-based model [15 points]

Question 2

In this task, you will implement an end-to-end ASR model that uses the Connectionist Temporal Classification (CTC) loss function. (The CTC loss function is built into Python.)

Your model architecture should consist of 1 or more convolutional layers (similar to M5 from Task 0), followed by 1 or more LSTM-based recurrent layers. This will be followed by a softmax output layer that gives a probability distribution over characters for each timeframe. (Note that your output space is now characters, unlike complete words as in Tasks 0 and 1.) Use Mel spectrograms as your input features for each timeframe. (Use all the default parameters specified in `torchaudio.transforms.MelSpectrogram`; please set the `sample_rate` correctly.) You can use any optimizer and learning rate scheduler of your choice.

Use a greedy decoder that greedily chooses the label with highest probability at each timestep. (Blank labels are removed from the final prediction.)

What to submit: Compute both character error rate (CER) and word error rate (WER) on the `test_set`. Submit two text files `task2/cer.txt` and `task2/wer.txt` with the best CER and WER, respectively, achieved by your model. Submit a link to your Colab notebook in `task2/notebook.txt` that we can run to reproduce your results. Mention *one advantage* and *one disadvantage* of the CTC-based system compared to either of the systems from Task 0 or Task 1. Submit your answer as a text file `task2/answer.txt`. The last two lines in your notebook should print the test CER and WER, respectively.

Task 3: CTC outputs [5 points]

Question 3

Instead of greedy decoding for the CTC-based model in Task 2, implement beam search decoding. You can use the ctcdecode library for this purpose.

What to submit: Submit two text files `task3/cer.txt` and `task3/wer.txt` with using the beam search CTC decoder. Also, submit a link to your Colab notebook in `task3/notebook.txt`. The last two lines in your notebook should print the test CER and WER, respectively.

Extra credit: Use an LM with beam search decoding to avoid misspelled words in your predictions and to guide the predictions towards words that appear in your vocabulary. The LM could be trained on the complete training set of the speech commands dataset. What is the resulting WER after using such an external LM? Submit this as a text file `task3/extra-wer.txt`. If you have attempted this part, you can add it to `task3/notebook.txt`. And, the last three lines in your notebook should print the test CER, WER (without the use of an LM) and WER (with the use of an external LM), respectively.

Task 4: Performance on blind test set [3 points]

Question 4

How well do your systems perform on unseen utterances? Within `task4/choice.txt`, choose one of the following labels for the type of trained system you would like to use to evaluate on unseen utterances: {M5,LSTM,CTC,CTC-beam,CTC-beam-lm}. Depending on your choice, we will use your corresponding notebook to evaluate a blind test set consisting of completely new utterances. A leaderboard with the top-scoring N roll numbers and the corresponding scores will be posted on Moodle. You will receive full points for this question if your blind test accuracy is higher than what we get with the baseline recipe in Task 0. The N top-scoring performers on the leaderboard will gain extra credit points.