Speech Processing
Exercise 4

Due: 20.6.2019 10:00PM (there will be no extensions!)

Guidelines

1. You are allowed to work in pairs.
2. In order to submit your solution please submit the following files:
   (a) `details.txt` - A text file with your full name (in the first line) and ID (in the second line).
   (b) `ex4.py` - The file that contains your main function (attach ANY additional files needed for your code to run).
   (c) `ex4_report.pdf` - A pdf file in which you describe your model and parameters.
   (d) `test_y` - your model's predictions on the given test set (see instructions below).

Follow the instructions and submit all files needed for you code to run.

Good Luck!

Ex4

In this exercise you are going to train your first end-to-end neural network for the task of Automatic Speech Recognition (ASR). To train your model you will use the Google Commands dataset (recall, this is the same dataset used in exercise 2).

You will optimize the Connectionist Temporal Classification (CTC) as your loss function. To implement your model (as well as the CTC loss) you are free to use PyTorch. You can read about PyTorch and see some examples in the following links:

1. installation - `https://pytorch.org`
2. tutorials - `https://pytorch.org/tutorials`
3. examples - `https://github.com/pytorch/examples`
4. CTC - `https://pytorch.org/docs/master/nn.html#torch.nn.CTCLoss`

Feel free to use all the techniques we have talked about during the course, e.g. DNN, CNN, RNN, Dropout, Batch Normalization, Data Augmentation, Optimization Methods, Etc.

Data. Each speech utterance is ∼1 second long. You are provided with a data loader called `gcommand_loader.py`. This data loader will load your data, create batches, randomly shuffle the data, etc. (An example of using this loader is provided in the `data_loader_tester.py` file). Notice, the provided dataloader extracts the STFT magnitudes, you may replace it with any features of your choosing (Mel-spectrogram, MFCC, etc.).

Decoding. After the model is trained, there are multiple methods to generate a prediction (this is called the decoding phase). In class, we have talked about greedy-decoding, beam-searching and ctc-beam-search. In this exercise, you will implement the decoding step using the greedy-decoding method (i.e., you should: (1) select the argmax of the model's output at each time step; (2) merge repetitions and; (3) remove blanks).

Instructions

1. Your goal is to train an end-to-end acoustic model. Your model should reach the best performance you can get on the validation set.

2. You should evaluate your model using the Character Error Rate (CER). You will find an implementation of this function in the file `cer.py` . Using the CER function is pretty straightforward:

   ```
   >>> from cer import cer
   >>> cer("hey, "yey")
   0.3333333333333333
   ```

3. You will receive the data already split to train, validation, and test sets. Each word category will be in a different folder.

4. You are provided with a file named: `gcommand_loader.py` to read the data and extract features from it. Additionally this file will create batches and shuffle the data for you.

5. After you have trained and validated your model, you should output you model's predictions on the examples in test folder to a file named `test_y`, where each row should contain:
   <filename>, <space> <prediction of your model>
   Example:
   `file0.wav, yes`
   `file1.wav, birde`

6. Describe your model's architecture and explain how you chose it and all hyper-parameters in `ex4_report.pdf`.

7. Submit **ALL** source code files along with your predictions file `test_y`. Note that you name it exactly as specified. Your grade will be based on your performance on the test set.