Speaker Classification with Deep Learning

Speech Technology Assignment 2023-24

Dr Andrea DeMarco

1 Introduction

Speaker Identification (SID) is the problem of identifying the identify of a speaker from a given speech sample, selected from a pool of known speakers. In order to build an SID system, speech data for different speakers is collected and use for training a model (or models) that represent the hypothetical speaker, or a model that learns to discriminate across speakers. SID systems have been built with many different approaches: generative, discriminative and neural. In this assignment you will be provided with speech data for a number of speakers (285 in total), which you are required to pre-process and then run through a deep learning classifier which can learn to determine speaker identify based on a voice sample from a speaker.

2 Dataset

The dataset you will work with is the Accents of the British Isles (ABI-1) Corpus. The corpus is split into 14 folders, each folder containing data from a single accent, as follows: BRM (Birmingham), CRN (Cornwall), EAN (East Anglia), EYK (East Yorkshire), GLA (Glasgow), ILO (Inner London), LAN (Lancashire), LVP (Liverpool), NCL (Newcastle), NWA (Northern Wales), ROI (Republic of Ireland), SHL (Scottish Highlands), SSE (Standard Southern English), ULS (Ulster).

Furthermore, within each accent folder, is data from approximately 20 speakers in separate male/female folders. Each speaker folder name serves as a unique ID for that speaker. For each speaker, you will find a number of .WAV files, coupled with transcriptions in .TRS or .TXT files. You will not need to make use of transcriptions for this assignment. You will also only need to utilize the .WAV files denoted by "shortpassage" in their filename. These .WAV files are of considerably long duration (around 40-60 seconds each), with the speaker reading a passage.

Hint: Feel free to clean up the files, leaving only the required .WAV files needed. Alternatively you can filter files in Python, but this is not necessary.

3 Task Overview

You are required to build an SID system for all speakers, with some very important conditions:

- Utterance independence - when splitting your dataset into training/validation/testing sets, you have to guarantee that utterance segments of a particular speaker are not present in more than one of these sets. If a speech segment from a particular speaker is being used for training, then that same segment cannot be in either the validation or testing sets, etc. This is because the intertwined text-dependency will bias the model to learn to track similar (or exact) looking sound segments. We want to test the model for SID performance, not Speaker/Segment-ID performance.
- Following from the above, after deciding on a training/validation/testing split ratio, this ratio has to be used to split data equally for each of the individual speakers. A global train/test split risks having less/more data from one speaker over another. A suggested split is to use 20% of all the data for testing. From the remaining 80% a further 20% can be used for validation, with the remaining bulk of the data being used for testing.
- Each utterance being used is quite long. You need to segment this data into exact 3-second chunks. Recall how you can know the number of samples in a .WAV file via the sampling rate, and therefore deduce the number of samples per 3-second chunk. All training, validation, testing will therefore be done on models trained on equal-length chunks of audio.

4 Deep Learning Model

You will then need to implement a deep learning classifier that may make use of a combination of convolutional blocks as well as LSTM-type RNN blocks. The sense of the architecture is more important than whether you use CNNs alone or CNNs combined with RNNs. However, given the nature of speech (a sequence of sounds), it is suggested to make use of both types of blocks. The CNNs responsibility will be to extract features (and compress the represenation) in light of the classification task. The LSTM will look at these features from the point of view of sequential information. The architecture is of course, up to you.

Hint: It is best to make use of a computer which has a GPU that can be used for Keras/Tensorflow/PyTorch for the various runs/tests you will make. You may make use of Google's Colab if you do not have access to a GPU.

Hint: Start off your model small and simple, and work your way up, avoiding over-fitting - perhaps even working on a smaller subset of speakers e.g. 10, at first. How will you know when over-fitting is happening? Performance between training/validation/testing should be similar. If they are not, then you have over-fit. Another way of noticing over-fitting is when your validation measure e.g. accuracy increases (as expected) over a number of epochs, and then starts to dip.

5 Report and Performance Analysis

Following your implementation, you are required to write a 3-page report in the style of a conference paper. The LaTeX template for this report is attached to the assignment pack.

Your report must include the following:

- A very short introduction - no need for any literature review. Jump straight to the task.
- A description and explanation of the architecture for your SID system. Justify the choice of optimizers, metrics, validation measures, classification layer outputs etc. Use figures if these help your explanation.
- All architectural and hyperparameter choices you make need to be explained in your report. Justification of these architectures can refer to the performance of the CNN and RNN blocks, and how they changed across main architectures you have tried out before reaching your final model. In order to demonstrate this, loss and accuracy curves should be utilized, and the discussion should circle around convergence properties of the architecture.
- An evaluation of the training process of each model. What did the loss curves for training/validation look like along the way? How many epochs and why? And overfitting problems? Discuss this in detail.
- An F-score based evaluation of the model. What improvements did you observe as your architecture got better?
- A confusion matrix of the system testing phase. Was the accuracy roughly the same across all speakers, or where some speakers harder to classify than others? Try and provide a qualitative assessment of this result, perhaps even from a linguistic analysis perspective.

Use any literature references you need to build your discussion, and cite your sources well in the bibliography. Do not include any code snippets in your write up.

There is not single solution to the problem, and your performances may vary. But you are expected to tweak your architecture to obtain the best possible performance. An example of a particular architecture on this dataset is shown in Figure 1. Something similar is expected: losses that go down over time, along with accuracy that goes up. Validation curves should follow training curves as much as possible.

6 Final Remarks and Submission

Do not leave this assignment till the very end. It needs time, both for the learning process that you will go through, as well as the time needed to develop and train/run the models on your machines.

Very importantly, your code should do what it purports to do, be organised into functions, be well-commented and descriptive (no cryptic variable and function names). You can submit your code as a Jupyter notebook if you wish. But the report itself has to be a PDF in the template provided. Make sure to ask questions when stuck, and after giving it a reasonable try.

The marking breakdown for this assignment is as follows:

- Pre-processing of data, chunking, dataset splitting etc. - **35%**

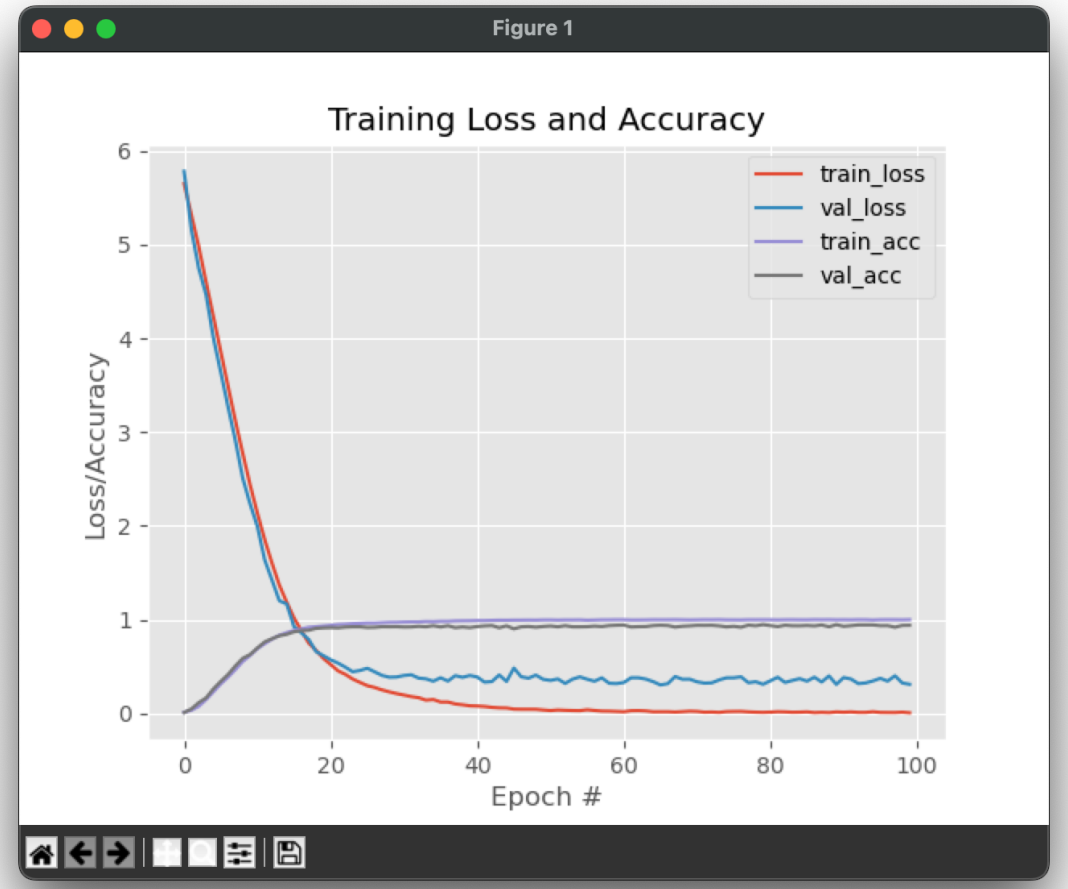

Figure 1: Loss plot example

- SID model design and implementation - **35%**
- Report - **30%**

Finally, you should submit your assignment by not later than **Sunday 21st January** at 17:00. **An attempt to all parts of this assignment needs to be submitted. Failure to do so will result in no mark being given.** You do not need to submit any hard copies, please zip your report, code and any other files as a single zip file and upload this to VLE. Please remember to also submit a signed plagiarism form[1].

Late submissions (by more than a few hours for some reasonable excuse) will not be marked.

[1]You can download a copy from here: `https://www.um.edu.mt/__data/assets/pdf_file/0006/398913/Plagiarismform.pdf`

7 First Aid: Getting Started

7.1 Obtaining a list of speaker labels in a corpus path

The following function will scan a corpus datapath, and return the path to individual speaker folders (one per speaker).

```
def get_speaker_roots_in_data_path(datapath='./corpus'):
    speaker_list = []
    accent_subfolders = [f.path for f in os.scandir(datapath) if f.is_dir()]
    for accent in accent_subfolders:
        for gender in ['female', 'male']:
            speaker_folders = os.listdir(os.path.join(accent, gender))
            for speaker in speaker_folders:
                if not speaker.startswith('.'):
                    speaker_list.append(os.path.join(accent,gender,speaker))
    return speaker_list
```

7.2 Obtaining a list of wav files in a directory

The following function will return a list of file paths of Wav files from a particular directory. This is not a recursive search and will not scan subfolders.

```
def get_wav_files_in_path(datapath):
    files = os.listdir(datapath)
    files_wav = [i for i in files if i.endswith('.wav')]
    return files_wav
```

7.3 Obtaining a Mel-spectrum from audio

Refer to demo from Lecture 2!