Exercise 4: Basics of speaker recognition

1. Exercise instructions

- Implement and return files CNN.py and functions.py.
- Return your answers to MyCourses by 23:59 on Tuesday, October 6, 2020.

2. Introduction

The goal of this exercise is to develop a speaker recognition system using python. Thus, you will train a speaker recognition model using Voxceleb-1 dataset. Since this is a course project, we will use the test dataset from Voxceleb-1 and partition it into training and evaluation. We will use 4709 files to train the speaker recognition model and 20 files for evaluation.

Files description

Input
- training.lst contains the list of training files.
- trials.lst contains the list of trial files for evaluation

Output
- scores_VoxCeleb-1.lst is the output file. It shows the cosine distance scores of trials.

3. Packages you should install

numpy, scikit-learn, Keras, Tensorflow

4. Build the CNN model

4.1. Partition the dataset into training and evaluation. Allocate 90% of the data for training and 10% of it for evaluation. Use **train_test_split** which is a scikit-learn's function which helps us to split train and test audio data kept in the same folders.

4.2. Define the input shape. Use 350 by 80. We use 350 since we are taking only 3.5 seconds of the speech and 80 is the number of features.

4.3. Compile the model. Use categorical_crossentropy as a loss function.

Useful function (from keras)

Compile

4.4. Train the model for 50 epochs. Shuffle the data.

Useful function (from keras)

fit

4.5. Save the trained model as "Speaker_Recognition_Excercise.hf5"

Useful function (from keras)

save

5. Evaluate the model

5.1. Load the saved "Speaker_Recognition_Excercise.hf5" model.

Useful function (from keras)

load_model

5.2. When you load the model and do the test, you will get an output file that contains the cosine distance similarity between test files (i.e., scores_VoxCeleb-1.lst).

6. Compute EER using scores_VoxCeleb-1.lst. How much EER do you get?