English Accent Classification using MFCCs

Project report

for

The Artificial Intelligence module

5th semester

in

Undergraduate **DATA SCIENCE**

Department of **AL-KHWARIZMI**

By,

Ismail BACHCHAR

Professors

Dr. Ikram CHAIRI

Dr. Hasnae ZEROUAOUI

Table des matières

Abstract

Accent classification is the task of determining the native language of a person speaking.

The accent is a crucial factor in identifying the origin of the person, it carries information about his/her nation, locality, and even his/her social class.

The purpose of this work is to develop a classifier, that will be able to classify different accents of the English language.

Using only Mel Frequency Cepstral Coefficients (MFCC) features of audio and two different algorithms, the first is a Machine Learning (ML) approach using Support Vector Machine (SVM) to make a binary classifier for native or non-native speakers (English and non-English), and the second is a Neural Network (NN) approach using Convolution Neural Network (CNN) to make a 10-classes classifier for the Arabic, Dutch, English, French, Korean, Mandarin, Portuguese, Russian, Spanish, and Turkish accents.

Today's accent classifiers use innovative approaches and well-developed algorithms and a combination of different audio features such as Perceptual linear prediction (PLP) and MFCC which gets quite efficient results.

An efficient accent classifier would help in various areas when a speech recognition system is needed, knowing the accent of the spoken word will improve the accuracy of the system, for example, the system will be able to know that a spoken word is the same as it is being pronounced by different persons with different accents.

Introduction

Accent classification or identification, which provides information about a speaker's origins, is important in applications like crime investigation and airport security, an example of such a use case would be detecting a person trying to fake his/her identity. In these kinds of applications, it is crucial to recognize accents from short audio clips with high accuracy.

Even for humans, the task at hand seems to be hard, a person can only recognize, with high accuracy, the accent of people of the same origin as him/her, luckily, technologies such as Machine Learning and Neural Networks can offer robust solutions.

English is one of the most spoken languages in the world, this gives a combination of different accents such as American, British, French, Arabic, etc. It is also becoming increasingly the first chosen language for the Voice Command Device (VCD), which are devices controlled with a voice user interface, systems used in automobiles, computer and mobile operating systems, home appliances like washing machines, etc.

Also, due to the availability of English accent data on the internet, makes it an ideal language to choose for the classification task.

With the increase in the use of smart voice-controlled appliances interacting with the user based on speech commands, it has become increasingly important for the system to understand what a person is saying correctly and accurately. Different accents of the same command or word should be considered and interpreted the same way.

Furthermore, Basic knowledge about whether the speaker is native or not could dramatically increase the accuracy of such a system, a more complex task is knowing the accent the speaker is speaking, after that, identifying his/her native accent, the speech recognition system comes in hand and detect what the speaker was saying I efficient manner.

In this work, I tried to do the first two tasks, a binary classification for native and non-native speaker and the ten native languages classification, using respectively Support Vector Machine (SVM) and Convolution Neural Network (CNN)

Dataset and Features

Data

I used the Speech Accent Archive [1] as my main data source since it offers all the needed information for the classification task.

The data was collected from more than 200 different countries around the world, with more than 2100 different persons recording audio clips reading the following text

" Please call Stella. Ask her to bring these things with her from the store: Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station. ", which is chosen because it includes some specific words that are known to be sensitive to accent, hence difficult to pronounce. These recordings are in mp3 format with variation in the duration, ranging from a minimum of 15secs to a maximum of 1 minute and 45secs depending on the person is recording.

The data composed of a total of 3.94GB (zip compressed), one CSV file containing speakers related data, such as age, native-language, sex, and country, etc, and a folder containing all recordings.

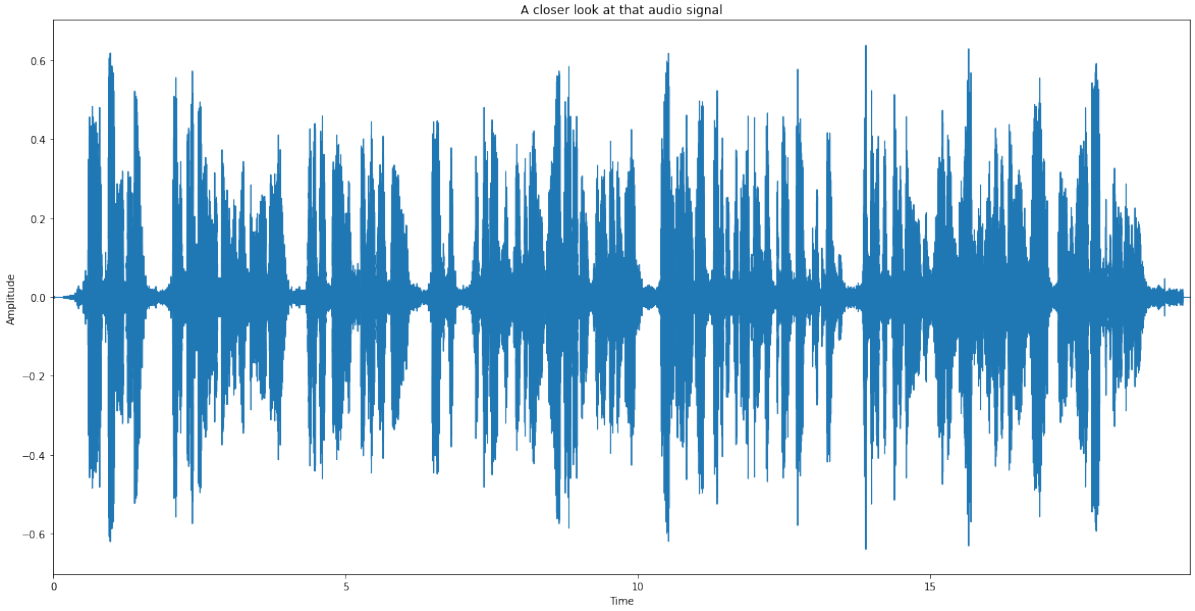
For the binary classification, I divided the data into native, whose native language is English, and non-native, the rest, resulting in imbalanced data, millions of audio clips for the second class.

I picked ten accents - Arabic, Dutch, English, French, Korean, Mandarin, Portuguese, Russian, Spanish, Turkish - to work on for the second task because it would take a lot more(more) time to run experiments on all classes, even if classifying speech into 10 classes is also algorithmically challenging, as it requires multi-class classifiers.

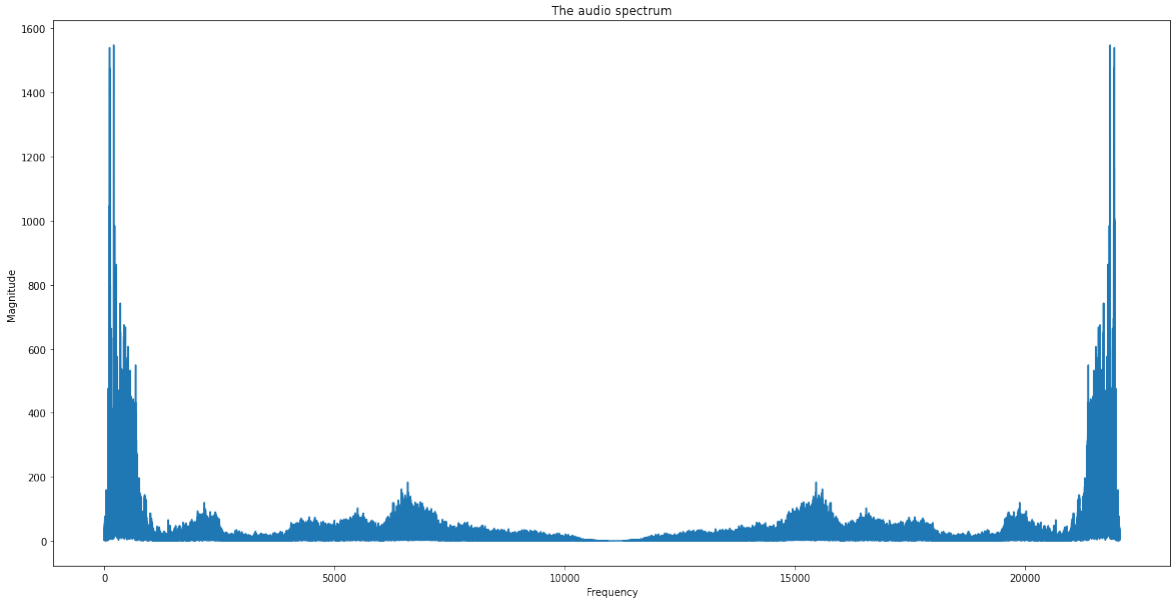
Features

The audio has multiple features that are suitable for different problems, but for the accent classification, after doing research, I found that Mel frequency cepstral coefficients (MFCCs) are the most used, so working with them sounds a good decision.

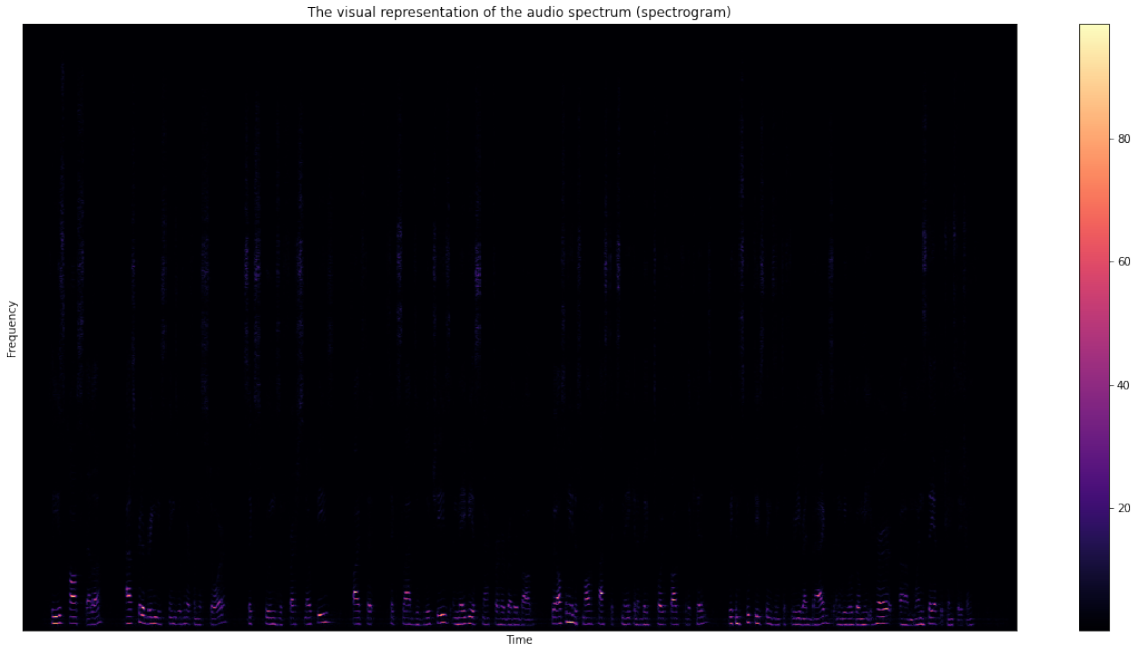
The audio consists of sample amplitudes varying over time, as shown in next image.



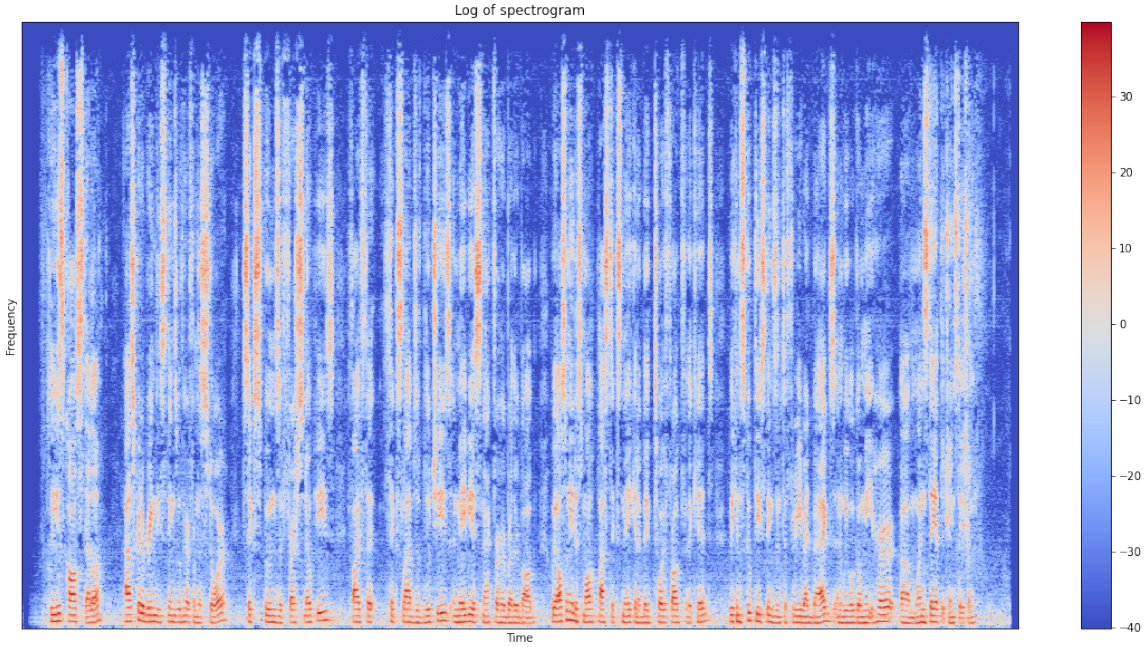
so, to map this to a more useful representation, I used the MFCCs features, which separate the audio samples, the signa, into small windows, called frames, and then apply the Fast Fourier Transformation on it to calculate its “spectrum” (the signal frequency domain representation).

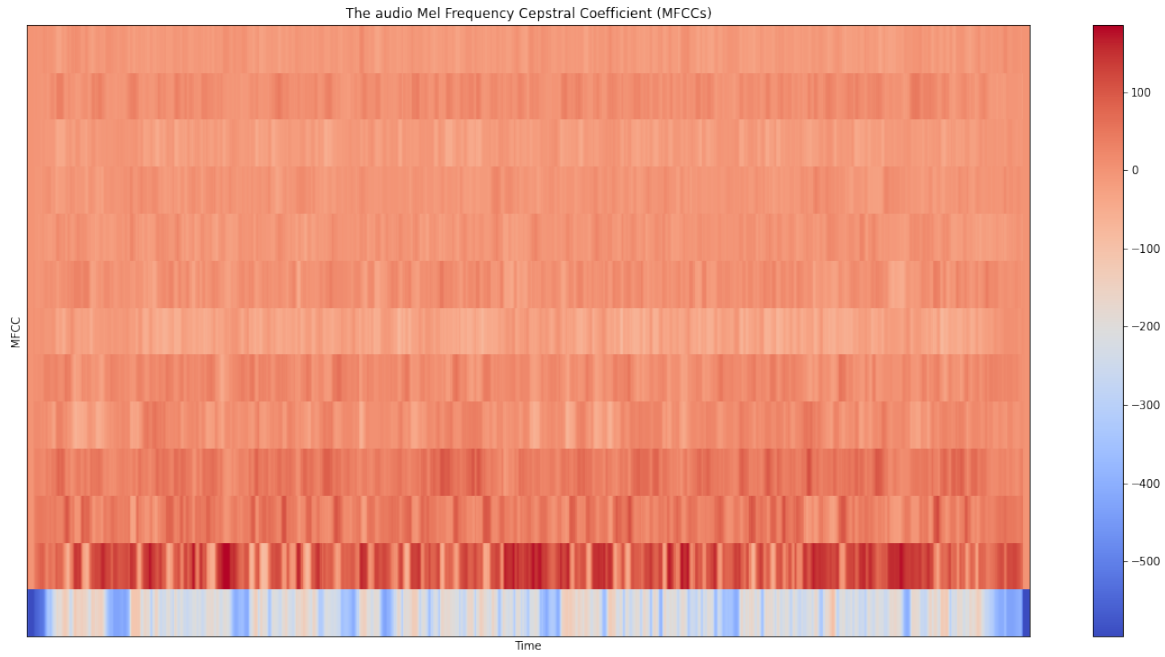


Also, the spectrum can be visualized by calculating the absolute value of a Short-time Fourier Transformation (STFT) applied on the audio signal; this results in what is called “spectrogram” (another representation of the spectrum itself).



Taking the Logarithm (log) of this spectrogram results in more clear representation.





Finally, we got a representation of the signal that combine the two most important aspects of the audio, time and frequency.

Because the MFCCs separates the audio samples into small windows (frames) and the audio clips differ in duration, I faced this problem when the number of extracted MFCCs per frame varies from audio to audio.

The solution was to segment each audio clip into several segments, of the same length, the number of segments is defined by:

$$Numberofsegments = \frac{AudioDuration(secs) * SampleRate}{165357}$$

Where **165357** is fixed length for all segments segment is calculated by the assumption that the minimum audio duration, which is 15secs, will be devided into two segments.
and the **sample rate** is default to 22050

$$Segmentlength = 165357(samples) = \frac{MinAudioDuration(secs)15 * SampleRate(samples/sec)22050}{2}$$

With this approach I extracted a total of:

6247 segments

323 windows or frames for each segment

13 MFCCs of each frame

The shape of input (X) is (6247, 323, 13) and output (y) is (6247,) each value in y corresponds to the native language of the segment in the speaker's audio clip.

Algorithms and Results

SVM with RBF kernel

To do the binary task, which is classifying to native and non-native classes, I used the SVM algorithm with the RBF kernel and the tuning parameter C in [0.1,100] (I could not do more because of computational costs).

Knowing that the SVM requires data to be in a particular shape, 2-d array, and that the classifier is based on windows, instead of the whole sound sample, along with 13 MFCCs features for each window, I had to reshape the inputs from

(1455, 323, 13) => (1455*323=469 965, 13) for audio windows of the English native speakers and (4792, 323, 13) => (4792*323=1 547 816, 13) for non-native speakers, the gap is reasonable since we have much more audio clips for the second class.

Instead of working with the whole over million samples of non-native class, leading to an imbalanced data problem, I sampled "randomly" -it is not guaranteed the data will be representative of all the samples- the necessary number of samples from them.

After that, I created the associated output vector, y, and concatenated the native and non-native data into input data, X, resulting in:

input X: (939930, 13)

output y: (939930,)

The luck computational resources forced me to do, again, the magic sampling and choose randomly 20% of the data, resulting in:

input X: (187986, 13)

output y: (187986,)

splitting the remaining data into 80% for training and 20% for testing give 131590 samples for training and 56396 for testing.

The result obtained by running the SVM-RBF (SVM with RBF kernel) model using two values, 0.1 and 100, to tune the C parameter with 5-folds cross validation.

	precision	recall	f1-score	support
0.0	0.51	0.52	0.52	93900
1.0	0.51	0.50	0.50	94086
accuracy			0.51	187986
macro avg	0.51	0.51	0.51	187986
weighted avg	0.51	0.51	0.51	187986

The model accuracy is not that interesting, about 51%

With an execution of 2 hours and 42 minutes.

The result I got, I was not expecting, because this binary classification seems to be an easy one compared to the multiclass classification, even though the SVM-RBF could not give a good accuracy, or maybe because of the “random” sampling I did, or the data needs some preprocessing first. The SVM method took too much time to execute for that reason I could not get more experiments with it.

SVM with POLY kernel

I tried to run SVM with a different kernel than the RBF previously used, there were two to choose from, linear and polynomial, I tried the linear model first, but it ends up not converging after about 6 hours of execution time.

The second choice left, I reduced the number of samples as for the SVM-RBF model, let the "degree" parameter takes the values 3 and 8 (again could not do more for computational reasons) with 5-fold cross validation, unfortunately, the model could not fit with the degree 8, the model took more than 7 hours stopping at it, so I had to stop it.

CNN without normalization

For this multiclass classification task, I decided, based on the samples of each class, to use these ten accents Arabic, Dutch, English, French, Korean, Mandarin, Portuguese, Russian, Spanish, and Turkish.

The CNN model does not require reshaping the data, so when I extracted the 10-classes data I got (3404, 323, 13) as the input (X), 3404 different audio segments each one of them is further divided into 323 windows which each has the 13 MFCCs, and (3404,) as the output (y) contains values between 0 and 10.

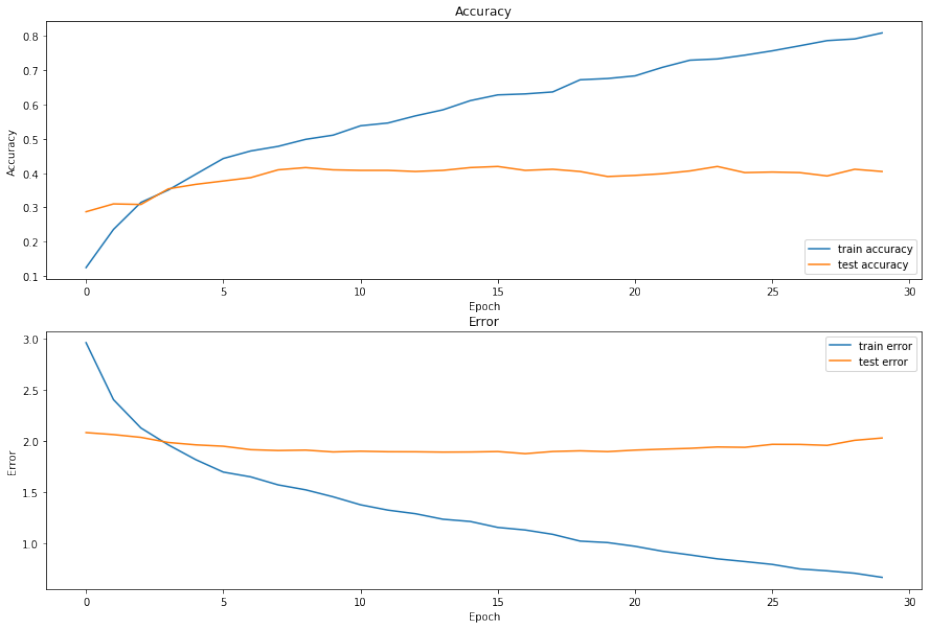
Using now, 10% of data for testing, 20% for validation, and 70% for training.

The "learning rate" parameter of the CNN model was tuned on a range from 1e-7 to 1.1 resulting in 8 different models, in about 20 minutes of execution time.

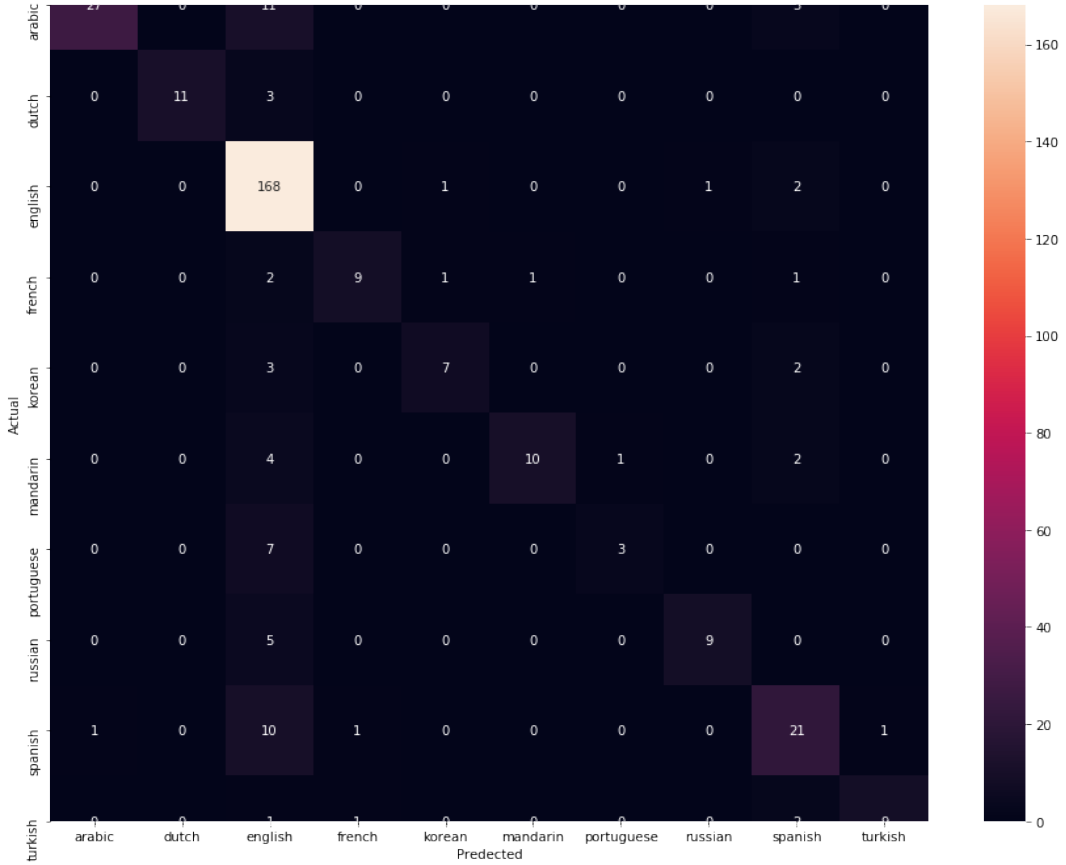
	Learning rate	Training Error	Training Accuracy	Test Error	Test Accuracy
0	1.100000e+00	388.691956	0.418544	1.918931	0.504399
1	1.000000e+00	4444.644531	0.418544	156199.687500	0.501466
2	1.000000e-02	1.538228	0.744042	1.456768	0.733138
3	1.000000e-03	1.153334	0.823702	1.032425	0.818182
4	1.000000e-04	0.910758	0.780607	0.862551	0.803519
5	1.000000e-05	1.818360	0.417564	1.768774	0.445748
6	1.000000e-06	2.879302	0.083252	2.823290	0.120235
7	1.000000e-07	3.085929	0.070846	3.149771	0.061584

The most performant model is 5th one with a learning rate of 1e-4 and training accuracy of 78% and test accuracy of 80%.

The 5th model (the best model) accuracy over epochs of training.



The confusion matrix of the best model



The perfectly classed accent is the English one, and that is because the is imbalanced, more samples for English samples than for other classes.

The model accuracy is good because there are not enough data points uniformly distributed across the all the 10 classes.

After doing some predictions with data not seen by the model, it seems that the model is biased towards English accent, all the well-spoken and/or fast sounds are classified as English.

The model could not extract any hidden features that will separate the accents.

CNN with normalization

Here the same approach used in the previous section is used, the same number of training and testing samples, the only difference here is that the data will be normalized before passed through the network.

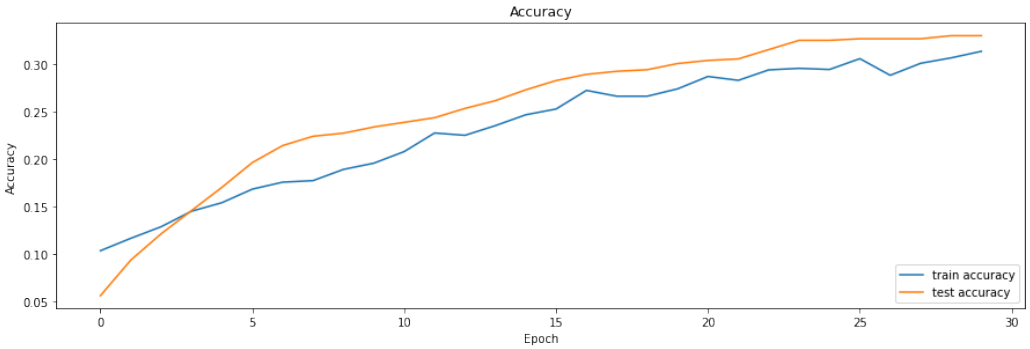
The execution time for this model is only 16 minutes.

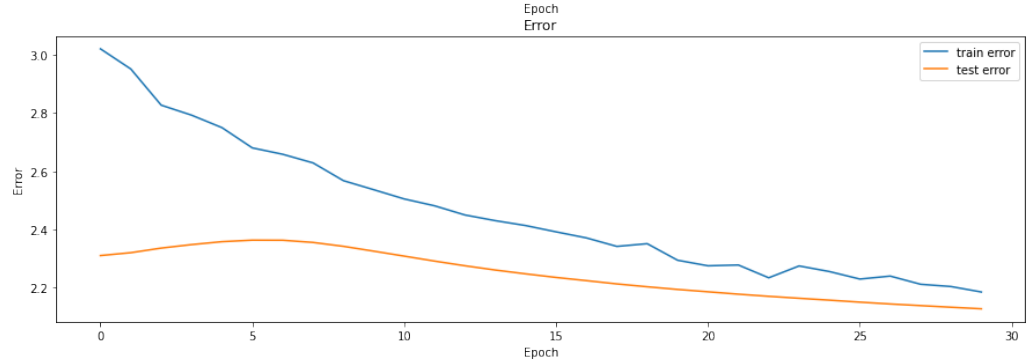
Different models corresponding to different learning rate values

	Learning rate	Training Error	Training Accuracy	Test Error	Test Accuracy
0	1.100000e+00	4.344861e+11	0.093046	4.021274e+11	0.108504
1	1.000000e+00	1.308650e+10	0.408423	1.451227e+10	0.419355
2	1.000000e-02	1.883932e+02	0.117858	1.787551e+02	0.134897
3	1.000000e-03	1.351062e+03	0.118838	1.293943e+03	0.131965
4	1.000000e-04	2.601154e+02	0.427685	2.563411e+02	0.425220
5	1.000000e-05	8.162880e+01	0.154097	7.231428e+01	0.152493
6	1.000000e-06	1.490352e+02	0.427685	1.484292e+02	0.425220
7	1.000000e-07	9.791326e+01	0.263794	9.821678e+01	0.269795

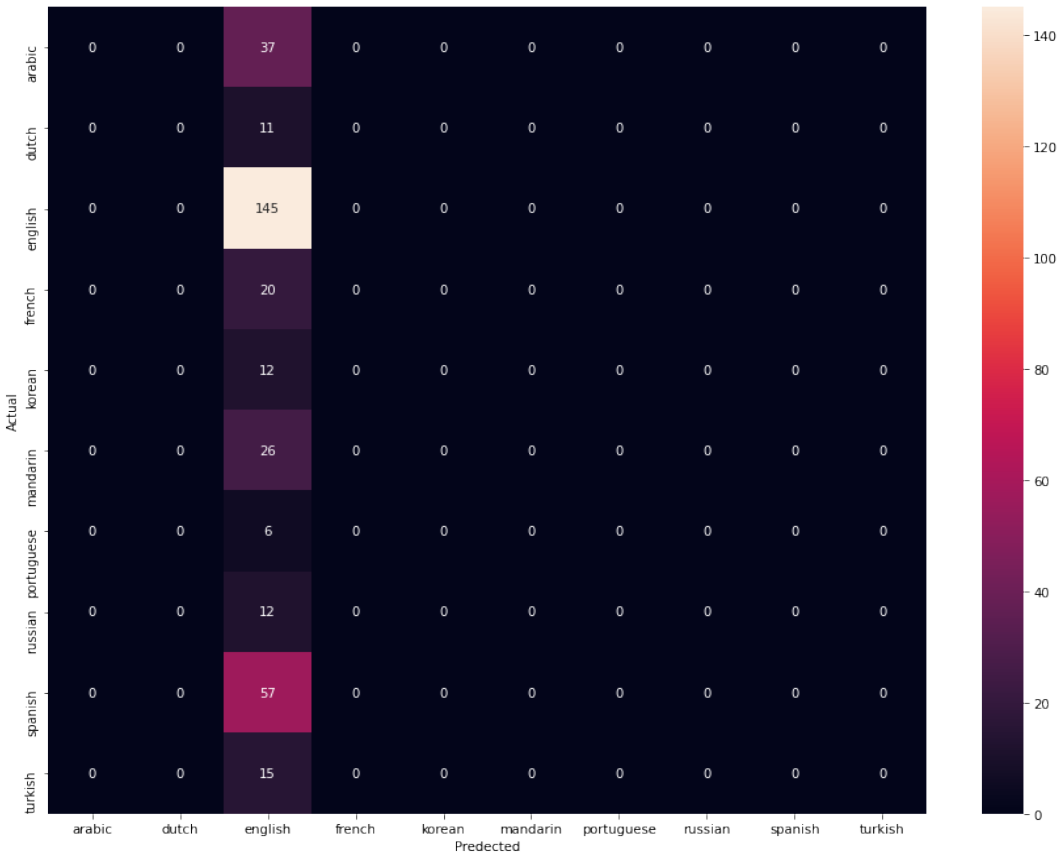
The 7^{th} model chosen to be the best one among the 8 models with an accuracy of 43%.

It gives a stable accuracy between train and test sets over all the epochs of training.





The confusion matrix of the best model



All data are classified as English accent, so the normalization has just added complexity to the first biased model.

Conclusion and Future Work

This classification task is so interesting any many levels from its importance to its complexity, making an algorithm that can determine the origin of a person from his/her audio data, needs a lot of experimentation and considering all possible audio features and their combinations, and also some well adapted ML and NN algorithms.

Future work

The most difficult thing I faced in this project is the limited computational power, which restricted me from trying more algorithms and tuning multiple parameter combinations, especially for the SVM model.

Also, the MFCCs alone are not enough to describe audio, while there are other features, the reasonable approach is to use all of them in different combinations, which may lead to some good results.

One other problem is that there a lot of pauses, repeated speech and noises on the audio clips, which gives at the end some windows (frames) that are not appropriate and do not help in the classification task, because these pauses or noises are the same for every accent.

One other thing to take into consideration is the speaker's gender because female voice’s pith is often very different from the male.

References and Bibliography

The Speech Accent Archive: https://accent.gmu.edu/

Speaker Identification using MFCC-Domain Support Vector Machine: S. M. Kamruzzaman, A. N. M. Rezaul Karim, Md. Saiful Islam, Md. Emdadul Haque: https://arxiv.org/abs/1009.4972

VFNet: A Convolutional Architecture for Accent Classification: Asad Ahmed, Pratham Tangri, Anirban Panda, Dhruv Ramani, Samarjit Karmakar: https://arxiv.org/abs/1910.06697