Using Deep Learning to Classify Countries of Origin based on English Accents

Dirk Jelle Schaap (s2745135) Jorge Anton Garcia (s4126890)
Ritten Roothaert (s2929244) Mario Cavero Vidal (s4113152)

March 19, 2023
University of Groningen, Netherlands

Abstract

Regardless of all the research done in accent identification, spectral and temporal features can make it difficult to properly classify the origin of an English speaker. In this paper we attempt to tackle the learning problem with our personal modifications to architectures inspired by the literature. After running experiments and making some adaptations to network architectures, such as LeNet and AlexNet, we managed to accomplish promising results with an accuracy near 80%.

1 Introduction

Oral communication is a daily routine in human life. Regardless of different regions and languages, humans have learned how to communicate with each other. Because of these geographical differences, when speaking the same language the pronunciation of words can be different. The tone, the accent and even sounds vary from one region to another.

Accents may have an emotional impact on people [3], causing stereotypes with regard to a specific social and moral image. Depending on the accent in question, it can be more suitable for a role in a film or a video game. Even today, assistants like Siri and Alexa have some problems classifying the intent of the user, because of the accent they have [5]. Being able to adopt a more suitable model for speech recognition depending on the accent could mean an increase in performance.

The accent classification problem has already been studied extensively. One of the problems of English enunciation is that non-native speakers have difficulty pronouncing English phonemes [1]. In speech recognition, data pre-processing is a crucial factor for a classifier to perform well. At the time this project was started, there were several pre-processing techniques that could perform well which will be explain later on.

In previous work two kinds of training are proposed, due to the characteristics of accent differentiation: long-term and short-term training [4]. A Deep Neural Network (DNN) was used to train on long-term statistical features whereas a Recurrent Neural Networks (RNN) was used to train on short-term statistical features. Subsequently, the results were fused together.

Spectral features and temporal features are said to vary within an accent. Gaussian Mixture Models and Hidden Markov Models were commonly used in this kind of problems, but nowadays DNNs and RNNs are used most commonly. Other work reduced the dimension of the data by using the Librosa library inorder to get the mel-frequency spectral coefficients [2]. The results of this Convolutional Neural Network (CNN), where a 3-layer network was found to be optimal, were quite accurate with a training accuracy of 86.3%.

2 Problem Description

The goal of this project is to take some audio file and try to classify the speaker's country of origin, based on their accent in English. The audio file will contain the voice of the speaker, who reads a

predetermined piece of English text out loud. The form of these audio samples is described in more detail in the next section. In order to fulfil this goal, we can break the problem down in several parts:

- **Create a well-balanced subset of data given the original dataset**
  The dataset we received from Kaggle is unbalanced and not immediately suitable for our project. Therefore, we will have to make a selection of data points and reduce the number of classes to make the problem more approachable.
- **Augment this new, smaller dataset to create more useful data points**
  To make our "new" dataset balanced, we had to reduce the number of data points of all classes to the size of the smallest class. This resulted in our subset we created to be relatively small. We will have to explore and deploy some form(s) of data augmentation to allow the use of more original and augmented data points and provide the neural networks with enough data to generalize.
- **Generate image representations in the form of spectrograms given the audio files**
  The neural networks cannot interpret audio files directly, of course. So, we must find a way to represent an audio sample in another way. The most common method is to create a spectrogram of the audio file, so that it will be accessible in the form of an image. Images are more easily fed as input to neural networks.
- **Define DDNs architectures to try different approaches**
  As mentioned in the Introduction, previous studies have used several different network architectures. In this project we want to try at least two different ones in order to compare them and review which one performs best.
- **Train the networks to classify the audio samples based on the speakers' accents**
  We need models that are able to receive the spectrograms of the data points as input and produce the native country of the speaker. These models should generalize well and ideally perform equally well among all classes.
- **Process the results and review all of the previous**
  To conclude, some metric measures of performance of the neural networks need to be defined. This allows us to review the performance of the different techniques, draw conclusions and try to identify interesting aspects for future research.

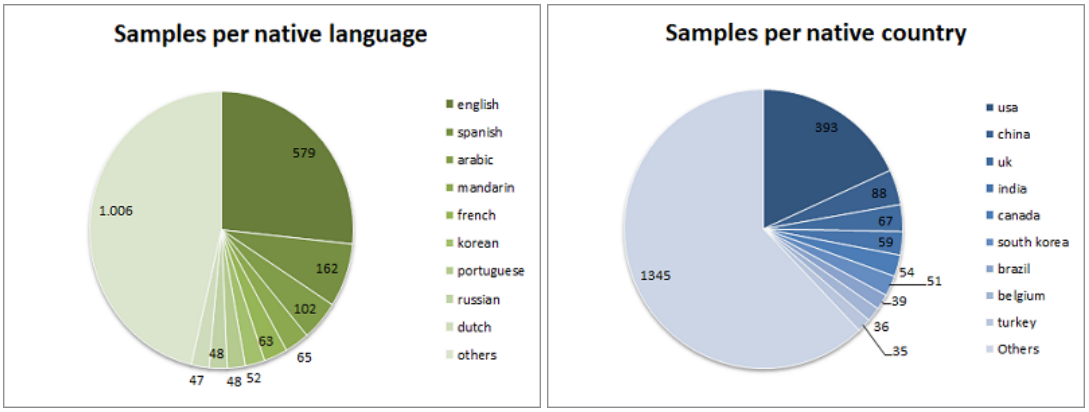
3 Dataset

> *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.*

The dataset was found and downloaded from Kaggle and is called the Speech Accent Archive [6]. It contains 2140 speech samples in the `MP3` audio format. Each sample is a recording of a speaker reading the same passage (shown in the box above) out loud in English. The subjects/speakers come from 177 countries and are said to have 214 different native languages. In this way, all speakers repeat the same script in the same language, but pronounced differently by every subject.

Sadly, this dataset is far from perfectly balanced. The distributions of samples per native language and samples per native country are shown in the form of pie charts in Figure 1. In Figure 1a we immediately see that the majority of the subjects have some form of English as their native language. Together with Spanish, these two languages represent almost 75% of the data. In Figure 1b we see a similar situation, where almost two-thirds of the subject were born in the United States of America.

To create a more balanced dataset, we had to select a subset of the data that will be used in this project. We decided to simplify the problem to just three different countries of origin. One important requirement we had for the three languages was that they had to be sufficiently different from one



(a) Samples taken by native language.

(b) Samples taken by native country.

Figure 1: Analysis of the samples.

another. For example, we felt like Spanish and Portuguese could be problematic to differentiate properly. There is no numerical metric for this, so we judged this subjectively.

English speakers were well represented also in the nationality analysis, so they were selected as the first native language for the experiment. Even though Spanish was the second language according to the number of samples registered, we did not include it. The main problem is that there were subjects from many different native Spanish-speaking countries. So, these possible differences within the accents led us to discard Spanish as a possibility. The same procedure was done for Arabic and French, being the factor why both of them were discarded. The second language selected was Mandarin which is spoken in China. We chose this language because China is the second largest country by samples registered in the dataset. The last language picked is Korean which is spoken in South-Korea. Here, the reason also was the fact that one of the countries with a greater number of samples.

4 Methods

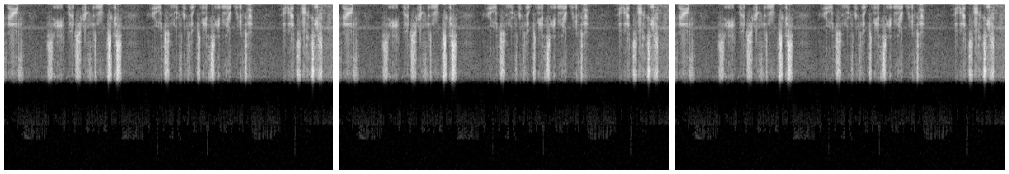
4.1 Data Augmentation

While the portion of English speakers is well represented in the Speech Accent Archive, Mandarin and Korean samples are not. Therefore we decided to augment our data in order to create enough samples for all three languages. The main issue that had to be overcome was the fact that the augmented samples needed to resemble actual human speech.

Two ways of augmenting audio samples are pitch manipulation and changing the speed at which the sample is played. We've explored both options. However, we quickly came to the conclusion that pitch modification is not a feasible method for creating enough augmented audio samples. Most samples produced by this method did not resemble regular human speech. Creating samples that did resemble human speech required a fair amount of parameter tuning which would needed to be repeated for each individual sample. Therefore we decided to stick to modifying the speed at which the audio sample is played. This method was quick and gave human-like speech samples which were sufficiently different from the original audio sample. We settled on multiplying the original speed by a factor of both 0.9 and 1.1, creating two augmented samples for each original sample.

4.2 Spectrogram Generation

In order to make convert the `MP3` files into a viable format for the neural networks, we will transform them into images. As we learned from previous studies, the most common method is to make use of spectrograms. We used the *PyTorch Audio* library to achieve this. It includes several ways of encoding audio files into a spectrogram. We settled on using the so-called *kaldi* spectrograms. since it provided the greatest contrast between the possible frequencies and amplitudes that occur in the audio files.



(a) Slowed down by 10% (b) No modifications (c) Sped up by 10%

Figure 2: Examples of spectrograms produced given one audio file.

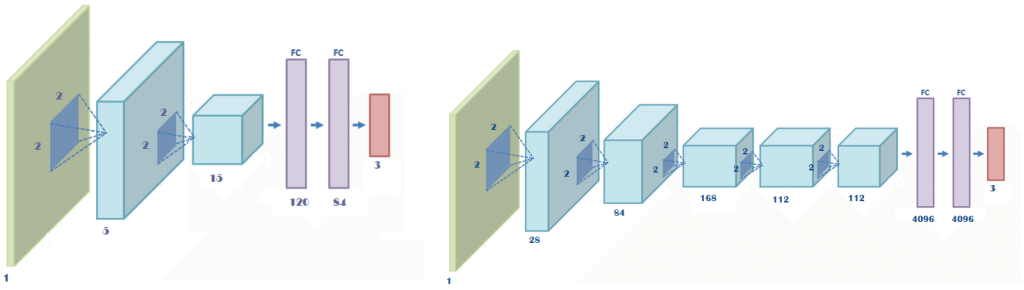
It is important to mention that not all audio data points are equal in length, and that posed a problem when creating these images. For example, Speaker A takes 25 seconds to read the text, while Speaker B needs 30 seconds to complete it. Therefore, we cut the spectrograms to take use the length just below the shortest audio file. To use the same example, we would take the audio files from Speaker A and Speaker B and take the first 24 seconds of both files to create the spectrograms. This causes a significant loss of data in some data points, since often the last seconds will be cut. However, we argue that the speed at which a subject speaks in English can be seen as a feature and will still provide useful information.

To give an idea of what the converted audio files would look like in an image format, we can refer to Figure 2. It is important to keep in mind that all three spectrograms are based on one single audio file. In Figure 2b we see the spectrogram that is based on the original file. Using the data augmentation, as explained in the previous section, we can get two augmented audio files from this original file by modifying their playback speeds. These augmented audio files are also converted into images and can be seen in Figures 2a and 2c. It is easy to see that this resulted in three very similar spectrograms.

4.3 Network Architectures

In this project we settled on trying two different network architectures. The first the based on the LeNet architecture and contains relatively few layers. The second is based on the AlexNet architecture and contains significantly more layers than the previous one.

- **LeNet adaptation:** For this adaptation of the architecture, we used 2 convolutional layers with max pooling and ReLU activation connected to three linear fully connected layers (see Figure 3a).
- **AlexNet adaptation:** this second architecture is more complex than the previous one, being made up of 5 convolutional layers with max pooling and ReLU activation, followed by an average pool and three fully connected layers (see Figure 3b).



(a) LeNet architecture. (b) AlexNet architecture.

Figure 3: Visualizations of the two used network architectures.

4.3.1 Experimental configurations

The LeNet network will be tested using several optimizers. The includes the infamous Stochatic Gradient Descent (SGD) optimizer in various forms, Adam and RMSProp. The AlexNet network will tested solely with the SGD optimizer with a momentum of 0.8. In all cases, the loss is calculated using the

cross-entropy loss function. Tthe experimental setups are presented in Table 1. Additional information that should be noted is that in #5 the Adam optimizer is set to a weight decay of 0.002 and betas of $(0.9, 0.999)$. Moreover, in #6 the RMSProp is set to an alpha of 0.99

Experiment	Network	Optimizer	Learning rate	Momentum
#1	AlexNet	SGD	0.03	0.8
#2	LeNet	SGD	0.03	-
#3	LeNet	SGD	0.03	0.4
#4	LeNet	SGD	0.03	0.8
#5	LeNet	Adam	0.001	-
#6	LeNet	RMSProp	0.001	-

Table 1: A summary of the experimental setups.

4.4 Executing the experiment

4.4.1 Data point selection

At this point, our reduced dataset contains roughly 373 original English samples, 50 Korean samples and 52 Mandarin samples. After the data augmentation this is increased to $(3 * 373 =)$ 1119 English samples, 150 Korean samples and 156 Mandarin samples. To keep the dataset balanced, we can take up to 150 samples from each language. However, if we do that without caution, we could end up with a dataset that only contains original files for the English language, since there are 373 available. Therefore, we sample original and augmented datapoints from each language equally. In this way, no class is privileged by having more "higher quality/original" data points.

4.4.2 Training/Testing the networks

The LeNet network is trained using 10 epochs, where each epoch present 80 samples from each native language $((3 * 80 =)$ $240)$ to the network. The AlexNet network experiences a similar setup, however this one is presented with 50 epochs. This process is repeated five times over for all experimental setups, as described in Section 4.3.1, to get a more reliable averaging of the accuracy and loss scores.

After the training of each epoch, a test of the network will be performed with 20 samples from each native language $((3 * 20 =)$ $60)$ to obtain the accuracy of the network. Once again, the samples were randomly selected from the dataset.

5 Results

In order to achieve a fair comparison between the experimental setups, we had to define some metrics:

- **Loss:** This variable associates with how well the network performs with the training data, which tend to decrease during the simulation.
- **Accuracy:** This gives an indication of the performance of the network for different data from the training one. This is the way to be able to predict how well the model will work with new and unlabeled data.

After running the simulations of the AlexNet approach, 5 different accuracy scores were obtained. Three of them successfully finish the classification problem at achieved a accuracy of over 70%, while the first and the fifth simulations encountered an error and dropped their accuracy to random guessing levels. All the LeNet approaches finished the simulations without any errors. A collection of all results in shown in two Tables containing the mean Loss/Accuracy for the simulations (see Tables 2 and 3).

Please note that for AlexNet, we exclude the two runs that had "crashed". It is also important to note that for the first setup show in Table 1 regarding AlexNet, the Epoch shown in the in Tables 2 and 3 represent the mean of an interval of 5 epochs, since AlexNet was trained using 50 Epochs instead of 10. Regarding the same Tables, the numbers in **bold** represent the earliest occurrences of the lowest loss/highest accuracy.

Epoch	AlexNet (#1)	LeNet (#2)	LeNet (#3)	LeNet (#4)	LeNet (#5)	LeNet (#6)
1	0.055	0.055	0.055	0.056	0.007	0.005
2	0.049	0.055	0.055	0.039	0.001	**0.000**
3	0.022	0.054	0.050	0.033	0.003	0.000
4	0.014	0.052	0.041	0.028	0.004	0.000
5	0.009	0.047	0.021	0.015	**0.000**	0.000
6	0.003	0.041	0.007	**0.000**	0.000	0.000
7	**0.000**	0.033	0.001	0.000	0.000	0.000
8	0.000	0.024	**0.000**	0.000	0.000	0.000
9	0.000	0.015	0.000	0.000	0.000	0.000
10	0.000	**0.007**	0.000	0.000	0.000	0.000

Table 2: Mean loss per epoch for all experimental setups.

Epoch	AlexNet (#1)	LeNet (#2)	LeNet (#3)	LeNet (#4)	LeNet (#5)	LeNet (#6)
1	33.42%	38.67%	41.33%	40.00%	69.67%	79.00%
2	38.68%	42.67%	44.00%	52.33%	73.00%	**80.33%**
3	61.61%	43.00%	45.67%	53.00%	74.33%	79.33%
4	64.94%	45.33%	51.67%	57.00%	75.67%	79.00%
5	71.77%	53.67%	67.33%	66.33%	76.33%	78.00%
6	69.16%	60.33%	77.00%	**74.67%**	74.33%	78.33%
7	71.00%	60.67%	77.00%	74.33%	76.33%	78.67%
8	71.98%	64.33%	78.67%	74.33%	76.00%	78.67%
9	**72.04%**	75.33%	**79.00%**	74.33%	**76.67%**	78.67%
10	70.80%	**76.00%**	79.00%	74.33%	75.67%	77.66%

Table 3: Mean accuracy per epoch for all experimental setups.

6 Discussion

To try and keep this section concise, we reduced the discussion to the most interesting discoveries:

- Even though the AlexNet approach performed errornously in 40% of the runs, it performed well with an accuracy of 72.04%. However, it did need at least 25 epochs to achieve a level of performance greater than 70.00%. This gives the impression that his network was too complex (and thus too time consuming) for the task.
- The LeNet approach equipped with the SGD optimizer achieved its highest performance level quicker when the momentum value of SGD was increased. With a value of 0.8 it is able to achieve an top accuracy of 74.67% after just 6 epochs.
- The LeNet approach paired with the Adam optimizer shows relatively high accuracy at the start of the training, but settles on relatively similar scores compared to the SGD optimizer(s).
- The LeNet approach with the RMSProp optimizer is able to achieve the overall highest accuracy of 80.33% after just 2 epochs. After this, it loses a little bit of performance over the remaining epochs, but still beats out all other approaches in the end.

Since this research was successful in some areas, but not in others, it opens up plenty of paths for future research. One aspect that requires more effort is the data augmentation. We had altered the playback speed of the files, hoping that it would result in more useful data. However, we found the spectrograms of the augmented files to be very similar. Also the modification of pitch seems like a useful method for augmentation. With regard to the network architectures, a RNN in the form of a Long short-term memory (LSTM) network has been shown to be promising in previous research. In this project a similar approach was attempted, but we were unable to progress enough to include it in the final paper.

References

[1] Morgan Bryant, Amanda Chow, and Sydney Li. Classification of accents of english speakers by native language. 2014. URL: http://cs229.stanford.edu/proj2014/Morgan%20Bryant,%20Amanda%20Chow,%20Sydney%20Li,%20Classification%20of%20Accents%20of%20English%20Speakers%20by%20Native%20Language.pdf.

[2] Kevin Chionh, Maoyuan Song, and Yue Yin. Application of convolutional neural networks in accent identification. 2018. URL: http://www.andrew.cmu.edu/user/yyin1/resume/cnn_project_report.pdf.

[3] Sergio Poo Hernandez, Vadim Bulitko, Shelby Carleton, Astrid Ensslin, and Tejasvi Goorimoorthee. Deep learning for classification of speech accents in video games. *AIIDE Workshops*, 2018. URL: http://ceur-ws.org/Vol-2282/EXAG_114.pdf.

[4] Yishan Jiao, Ming Tu, Visar Berisha, and Julie Liss. Accent identification by combining deep neural networks and recurrent neural networks trained on long and short term features. In *Interspeech 2016*, pages 2388–2392, 2016. URL: http://dx.doi.org/10.21437/Interspeech.2016-1148, doi:10.21437/Interspeech.2016-1148.

[5] Leon Mak An Sheng and Mok Wei Xiong Edmund. Deep learning approach to accent classification. *CS229*, 2017. URL: http://cs229.stanford.edu/proj2017/final-reports/5244230.pdf.

[6] Rachael Tatman. Speech Accent Archive: Parallel English speech samples from 177 countries, 2018. [Online; accessed 18-February-2020]. URL: https://www.kaggle.com/rtatman/speech-accent-archive/.