Speech recognition using class-based neural networks (Arabic dialect classification)

Hangfelismerés és osztályozas neurális hálók segítségével (Arab nyelvjárási osztályozás)

Ferfar Mohamed Chakib

Advisors: Pásztor Adél, Lukács András.

Abstract

This project seeks to explore the potential of Convolutional Neural Networks (CNNs) in recognizing and classifying Arabic dialects, it aims to develop a reliable and accurate method for classifying Arabic dialects.

This will involve taking in a dataset of Arabic dialects as an input, pre-processing the data, and training the CNNs with the dataset.

The effectiveness of different CNN architectures and the impact of the hyperparameters on the accuracy of the classifier will be assessed. The goal is to create a model with maximum efficiency.

The results of this project will provide valuable insights into the performance of CNNs in this domain and contribute to the development of better methods for recognizing and classifying Arabic dialects.

Motivation

Arabic dialects are the various spoken varieties of the Arabic language used throughout the Arab world. These dialects originated from the classical Arabic language but have evolved over time and are now spoken in different regions and countries. Arabic dialects are characterized by their distinct phonological, morphological, syntactic, and lexical features. They are also highly influenced by local languages, such as Turkish and Persian, as well as by foreign languages, such as English and French.

The most widely spoken Arabic dialects are Egyptian, Lebanese, Iraqi, and Gulf. Each of these dialects is spoken by different populations in different regions of the Middle East. There is also a great deal of diversity within each dialect, as well as between them.

The study of Arabic dialects is complex and ongoing, and it is essential for anyone learning or studying the language. Speakers of different dialects may not be able to understand one another, and it is important to learn the various dialects to ensure effective communication in the region.

Furthermore, dialects are an important part of the culture and identity of the Arab world and understanding them is essential for appreciating the culture.

Introduction

Speech recognition and classification using class-based neural networks is an important research field in the field of artificial intelligence and machine learning.

By leveraging neural networks, researchers can build systems that can identify patterns in speech and classify them into different categories.

The key to speech recognition and classification is to learn how to identify meaningful patterns in speech. Neural networks are a powerful tool for this task as they can recognize patterns from large amounts of data.

The neural networks can then be trained to recognize speech patterns and classify them into different classes.

Class-based neural networks use supervised learning to train the network on labeled data. This means that the network is given a set of data that is labeled according to its class.

The network then learns to identify patterns in the data and classify them into the correct classes. Once the network has been trained, it can then be used to recognize speech from new data and classify it.

This is an important step in speech recognition and classification, as it allows the system to identify different kinds of speech and distinguish them from one another.

Overall, speech recognition and classification using class-based neural networks is a powerful tool for AI and machine learning researchers.

What's a CNN?

A convolutional neural network (CNN) is a type of neural network that uses mathematical functions called convolutions to process data. Convolutions are a type of mathematical operation where two functions overlap and interact with each other. The convolution operation is commonly used for image processing, where a small region of an image (known as a kernel) is overlaid on the original image and the output of the convolution operation is a new image, convolutions are used to extract features from the input data, allowing the network to learn patterns and identify objects in the data.
A convolutional layer is used to apply a convolutional operation to an input layer.
This operation is a mathematical process that takes two inputs, such as an image, and produces an output. The convolution operation is defined as:

$$\text{Output}(i,j) = \sum k,l \text{ Input}(i-k,j-l) * \text{Kernel}(k,l)$$

Where:

- i and j are the coordinates of the output element
- k and l are the coordinates of the kernel element
- Input is the input layer Kernel is the filter or kernel used in the convolution operation.

The output of the convolution operation is a feature map which is a 2-dimensional matrix of numbers that represent the presence of features in the input layer. The filter or kernel is then moved across the input layer and the convolution operation is applied at each position. This process is repeated until the entire input layer has been processed and a feature map is produced.

The Dataset

The dataset used in this project was taken from arabicspeech.org and it is called the Fine-grained Arabic Dialect Identification (ADI) dataset.

The task of ADI is dialect identification of speech from YouTube to one of the 17 dialects (ADI17). The previous studies on Arabic dialect identification using audio signal are limited to 5 dialect classes by lack of speech corpus. To present a fine-grained analysis of the Arabic dialect speech, we collected the Arabic dialect from YouTube.

For Train set, about 3,000 hours of Arabic dialect speech data from 17 countries on the Arabic world was collected from YouTube. Since we collected the speech by considering the YouTube channels in a specific country, certain that the dataset might have some labeling errors. For this reason, we have two sub-tracks for the ADI task, supervised learning track and unsupervised track. Thus, the label of the train set can be either used or not and it completely depends on the choice of participants.

For the Dev and Test set, about 280 hours of speech data was collected from YouTube. After automatic speaker linking and dialect labeling by human annotators, we selected 57 hours of speech dataset to use as Dev and Test set for performance evaluation. The test dataset was considered to have three sub-categories by the segment duration to represent short (under 5 sec), medium (between 5 sec and 20 sec), long duration (over 20 sec) of the dialectal speech.

Methodology

To begin, we will take sound files and convert them into spectrograms. These spectrograms will then be fed into a CNN plus Linear Classifier model, which will generate predictions about the class to which the sound belongs.

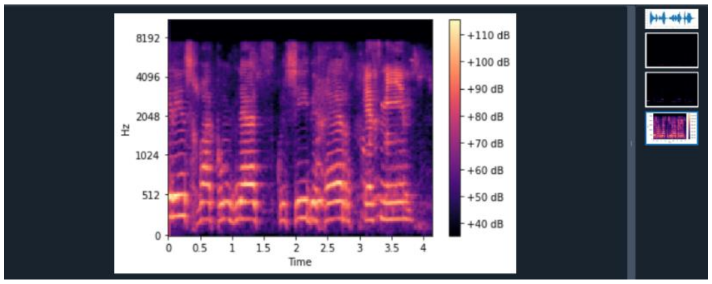


Figure 1 An augmented spectrogram of one of our dataset audio files

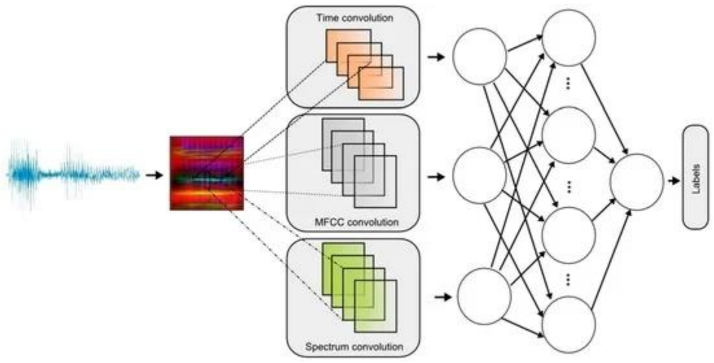


Figure 2 Graphical abstract by (https://www.mdpi.com/2073-8994/11/9/1185/htm)

We use spectrograms to represent the spectral content of a sound, which is typically computed using the Fourier Transform. The Fourier Transform is a mathematical technique that takes a signal in the time domain and converts it into the frequency domain which decomposes the sound into its constituent frequency components. A spectrogram is an image that shows the frequency content of a sound over time, which makes it easier to visualize and analyze. The magnitude of each frequency component is plotted on the y-axis of a spectrogram, while the time evolution of the frequencies is plotted on the x-axis. Because different sounds have different characteristic frequency spectra, the shape of the spectrogram of a sound can provide useful information for identifying and classifying the sound. For example, a sound with a strong fundamental frequency and several harmonics will have a distinct peak at the fundamental frequency on the spectrogram, while a sound with a more complex frequency spectrum may have multiple peaks at different frequencies. By analyzing the shape and characteristics of a sound's spectrogram, we can often determine what type of sound it is.

- Preparing the training data:

First, we must download our dataset and load it into our model, in this model, we will need a csv file containing all the relevant information about our dataset audio files for classification, such information can be the file name(path), the id label, the specific dialect and the class id.

Since our dataset does not have such a csv file, we must create one, by extracting the file paths from the dataset, input the result into an excel sheet, combine it with the txt file we downloaded containing the id labels, extracting the last 3 letters from each id label into a new column to create the dialect column, assign each dialect to a different number starting from 0 to 16 as we have 17 dialects in total in our dataset.

- Audio Pre-processing:

The provided training data containing audio file paths cannot be used directly as input for the model. Instead, the audio data from these files must be loaded and processed so that it is in a format that the model can accept as input. This may involve converting the audio data from the file into a different format, applying any necessary preprocessing steps, and ensuring that the data is structured in a way that is compatible with the model's expected input format. Once the audio data has been processed in this way, it can then be used as input for the model.

The audio preprocessing will be performed dynamically when the audio files are read and loaded at runtime. To avoid reading the entire dataset into memory all at once, we will only keep the names of the audio files in our training data. When the model needs to process an audio file, it will load the file by its name, apply the necessary preprocessing steps, and then use the processed data as input. This allows us to work with a large dataset without having to load all the data into memory at once.

At runtime, as we train the model, we will process one batch of audio data at a time by applying a series of transformations to it. This allows us to keep only the data for one batch in memory at a time. By loading and processing the data in this way, we can train the model efficiently and effectively.

The first step is to read and load the audio file in the "wav" format.

Some of the sound files may be mono (i.e., containing only one audio channel), while others are stereo (i.e., containing two audio channels). To ensure that all items have the same dimensions and can be processed by the model, we will convert mono files to stereo by duplicating the first channel to create a second channel.

Some of the sound files are recorded at a sampling rate of 48000 Hz, while others are recorded at a rate of 44100 Hz. This means that the number of audio samples per second will vary depending on the file, with some containing more samples than others. To ensure that all items have the same dimensions and can be processed by the model, we will convert all

audio to the same sampling rate. This allows us to uniformly process the data and train the model effectively.

We then adjust the length of all the audio samples to be identical by adding silence when needed, or by cutting off the end of the sample.

Next, we can enhance the raw audio signal by employing data augmentation through Time Shifting, which involves shifting the audio to the left or right by a random amount.

After augmenting the audio, we convert it to a Mel Spectrogram, which captures the most important aspects of the audio. This is often the most effective way to feed audio data into deep learning models.

- Data augmentation:

Using SpecAugment, we will now perform another round of augmentation on the Mel Spectrogram instead of the raw audio. This technique involves two methods: Frequency masking, which involves randomly obscuring consecutive frequencies on the spectrogram with horizontal bars, and Time masking, which involves randomly blocking out ranges of time on the spectrogram with vertical bars.

In the Figure 1 at the top right corner, we can see 3 spectrograms on top of each other getting augmented gradually from top to bottom.

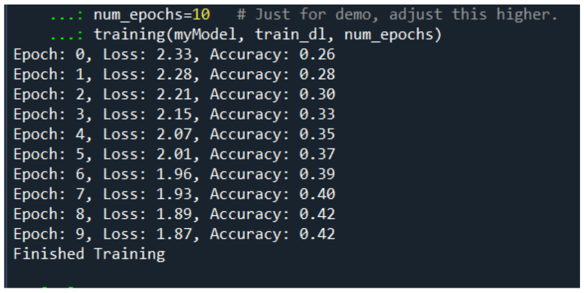
After we have established all the pre-processing transform functions, we will create a custom Pytorch Dataset object to enable us to feed our data to a model. This custom Dataset object will utilize the audio transforms to pre-process an audio file and generate one data item at a time. Furthermore, a Pytorch DataLoader object will be utilized to fetch individual data items and assemble them into a batch of data.

We randomly split our Pandas dataframe, containing Features and Labels, into a training set and a validation set in an 80:20 ratio. We then use our custom Dataset to create our training and validation Data Loaders from these sets.

- Training and validation:

After all the preprocessing steps, we just need to use our CNN model and training to get our first results.

Results



```
    ...: num_epochs=10   # Just for demo, adjust this higher.
    ...: training(myModel, train_dl, num_epochs)
Epoch: 0, Loss: 2.33, Accuracy: 0.26
Epoch: 1, Loss: 2.28, Accuracy: 0.28
Epoch: 2, Loss: 2.21, Accuracy: 0.30
Epoch: 3, Loss: 2.15, Accuracy: 0.33
Epoch: 4, Loss: 2.07, Accuracy: 0.35
Epoch: 5, Loss: 2.01, Accuracy: 0.37
Epoch: 6, Loss: 1.96, Accuracy: 0.39
Epoch: 7, Loss: 1.93, Accuracy: 0.40
Epoch: 8, Loss: 1.89, Accuracy: 0.42
Epoch: 9, Loss: 1.87, Accuracy: 0.42
Finished Training
```

Our CNN model approach for Arabic dialect detection had a below 50% accuracy and it was expected given the complexity of the task. Arabic dialects have a multitude of differences within and between them, making dialect detection difficult to achieve. Additionally, the data set used for the model was limited in size and scope and could not capture the full complexity of Arabic dialects. As a result, the current model was not able to achieve a high accuracy. In order to improve the accuracy of the model, a larger and more diverse data set should be used. This data set should include multiple dialects and should be varied in terms of region, gender, age, and other factors. Additionally, a more sophisticated model should be used, such as a deep learning-based model that can better capture the nuances between dialects. Ongoing research should also focus on improving the model's ability to correctly identify dialects with a low amount of training data. With these improvements, the accuracy of the Arabic dialect detection model could be improved significantly.

References:

1. Doshi, K. (2020, April 9). Audio Deep Learning Made Simple: Sound Classification Step-by-Step. Towards Data Science. Retrieved from https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5b

2. W. Wang, and C. Schuldt, "Spectral-temporal ConvNets for Audio Classification," arXiv, 2018.

3. J. Salamon, and J. P. Bello, "A Dataset and Taxonomy for Urban Sound Research," IEEE Transactions on Audio, Speech and Language Processing, vol. 24, no. 10, pp. 1690-1700, 2016.

4. M. Stöter, J. Schlüter, and R. Schlüter, "Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection in Real-Life Acoustic Scenes," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 26, no. 4, pp. 775-785, 2018.

5. Y. Huang, and P. Liu, "A Convolutional Neural Network with Attention Mechanism for Audio Classification," Applied Sciences, vol. 9, no. 5, pp. 990-1003, 2019.

6. S. W. Lee, and B. Raj, "A Convolutional Neural Network for Audio Classification Using Spectrograms