Capstone Project
Machine Learning Nano-Degree

Ahmed Hamdy
27/9/2019

Definition

Project Overview

Because of not dealing with speech recognition problems, I decided to strengthen my skills in such a domain and work on a speech classification example that covers my need for working in speech recognition tasks. In my project, I would like to build a radio website that plays a radio station and provide additional information below on the same page about the speaker that is currently speaking on the radio.

I decided to focus this project on building a demo project that covers a simpler problem than the final solution so I will just focus on building a speech classifier solution between two speakers and I will increase it's difficulty incrementally "Future Work".

The project's main library that I will be using is Librosa and here is the paper that explains the use of this library in audio analysis.

http://conference.scipy.org/proceedings/scipy2015/pdfs/brian_mcfee.pdf

Here is the link to the dataset that I will be working one:

https://github.com/Jakobovski/free-spoken-digit-dataset

Problem Statement

The problem that needed to be solved in my project was to classify who is currently speaking on a certain radio station in order to be able to display more data about the speaker such as name, age, previous work, photos, etc. I think that it's not an easy problem to classify a huge number of speech clips that could be possible in a radio

station, but the radio station that I am targeting is currently having a few numbers of speakers so I think the problem will not require much processing requirements in terms of hardware capabilities.

The scope that I am targeting and I will implement in the capstone project is just to classify simple conversations and assign each to their speaker so I will work on the best ways to classify speech data, the best libraries such as Libarosa library, and all the way towards classifying a speech.

Metrics

Accuracy is a common metric for binary classification problems. Accuracy uses both True Positives and True Negatives with equal weights.
Accuracy = (TP + TN) / Dataset Size.

Analysis

Data Exploration

The dataset that I will be using is the Free Spoken Digit Dataset (FSDD) that mentioned here: https://github.com/Jakobovski/free-spoken-digit-dataset and it simply some audio/speech dataset consisting of recordings of spoken digits in WAV files at 8kHz. The recordings are trimmed so that they have near minimal silence at the beginnings and ends.

FSDD is an open dataset, which means it will grow over time as data is contributed. In order to enable reproducibility and accurate citation, the dataset is versioned using Zenodo DOI as well as git tags.

The WAV files that are ready-made in the dataset above is perfect to be fed in the library that I choose for the project which is Librosa so there is no need for extra modification for the dataset other than possible scaling or something related to the classification task.

More information about the dataset:

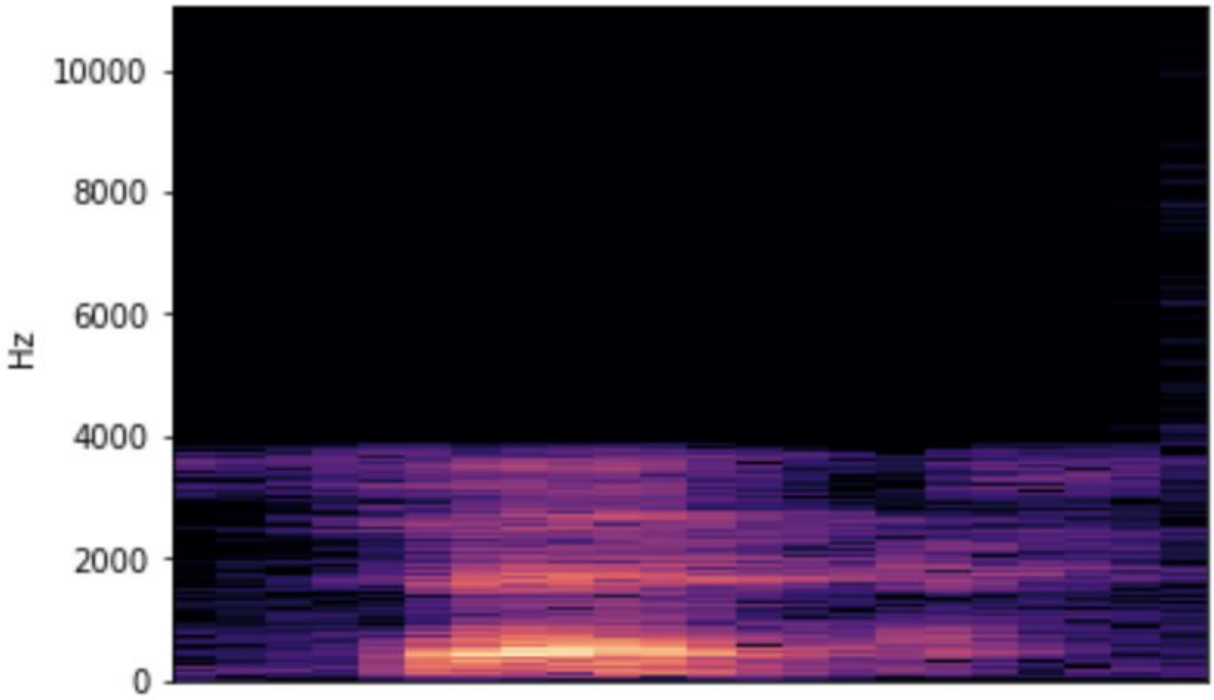
1- It has information about 4 speakers.
2- It has around 2,000 recordings (50 of each digit per speaker)
3- All recordings are in the English Language.

Exploratory Visualization

While trying to represent the data in my dataset, I draw a conclusion that I need a method that given a raw wave (a function of time), will return us the frequencies in it.

This is what Fourier transform is for achieving this. For more information about the Fourier transform method check this:
https://www.youtube.com/watch?v=spUNpyF58BY&t=3s

Human speech is not a static noise, it changes over time, So to correctly represent human speech, we'll break our recordings into small windows and compute what frequencies are used in each window. We can use the Short-time Fourier transform for this.



We can see some low-frequency sounds in the middle of the recording. This is typical of a male voice. This exactly what a spectrogram is! It shows us different frequencies in different parts of the voice recording.

I will use spectrograms instead of the row WAV files that are used in the benchmark model for building the final CNN solution.

Algorithms and Techniques

Great, now we are able to compute spectrograms for each file in our dataset and then use them to classify digits. What's really nice about spectrograms is that they are like 2D images so we can use image classification techniques on them, specifically Convolutional Neural Networks!

Using spectrograms has a strenth which made our problem much like a simple 2D image classification problem, so I will use CNN "Convolutional Neural Networks" algorithm for the final solution.
https://en.wikipedia.org/wiki/Convolutional_neural_network

A Convolutional Neural Network is a Deep Learning algorithm that can take in an input image “Our in our case Spectrograms”, assign importance (learnable weights and biases) to various aspects/objects in the inputs and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

In my model, I will use Conv2D with 4*4 kernel which means that the window that will loop over my data and try to extract patterns in it in 4*4 window size. I will also use the ‘RELU’ activation function with the ‘Same’ padding which is a method for dealing with the results of the curve from the scanning. All besides an input of 32 layers.

Benchmark

The complete solution will be a website that targets a sports radio station and display some information about the current speaker. The solution will simply make a wav file out of the currently speaking speaker and feed this to the classifier which should be able to classify the speaker based on it’s stored a dataset. “There is given info that the number of speakers on that certain radio station in limited so it will be easy to make a dataset of all the speakers in that radio station”.

After correctly classifying the speaker, I will display some information about him/her that I see that will be useful for the visitor of the website such as the name, age, previous work, etc.

The benchmark model is a simple MLP model that uses only one hidden layer and raw WAV files as inputs to for it’s classifier:

```
ip = Input(shape=(X[0].shape))
hidden = Dense(128, activation='relu')(ip)
op = Dense(10, activation='softmax')(hidden)
model = Model(input=ip, output=op)
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 30000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               3840128
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 3,841,418
Trainable params: 3,841,418
Non-trainable params: 0
```

The benchmark model shows good accuracy on training data, but it failed to achieve an acceptable accuracy over the validation data.

Methodology

Data Preprocessing

The preprocessing that I needed to implement in my final solution is to convert the raw WAV sound files to spectrograms to bb able to feed them as an input to a Convolutional Neural Network algorithm to easily classify the input to it's targeted labeled speakers.

I've defined the heading for this step under "Preparing the Test/Train Data" section of the notebook for your reference.

Implementation

The project implementation can be splitted into 3 parts:

1- Building the Conventional Neural Networks layers: In this stage, I implemented the CNN with it's inputs layers "32", Kernel Size "4*4", Activation Function "RELU", and the other needed parameters such as the Padding which I chose to go with the default "Same" setting.

2- The Compile and Fit stage: In this stage, I defined the compiling setting such as the Categorical Cross-entropy loss function, Adam optimizer, and the Accuracy metric.

The fit stage which I decided to train the model with 50 epochs and 32 batch-size for each.

3- The Final Stage: It's a simple graphing stage that shows the evolution of Train and Validation accuracy over the different epochs.

Refinement

The refinement that I had to add to the final solution is just converting the raw WAV files to standard spectrograms to be able to use them as an input to a simple CNN model.

I also added more layers in the final solution than just the 1 hidden layer that the benchmark model uses for classification.

Results

Model Evaluation and Validation

The final model is just the typical model that could be used for image classification tasks. It's a simple Convolutional Neural Network model that has a 32 layers input and Relu activation function.

What made the data suitable to be feed in the model is the transformation from the raw WAV files to the standard spectrograms which is the standard way for representing the sound files.

The model seemed to be good on both training and validation data which made it approach the expected accuracy score.

Justification

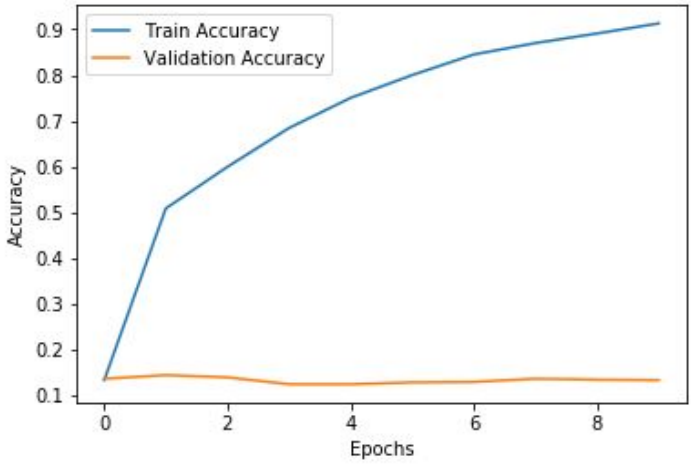
The final solution with the CNN model achieved more improved results than the simple model used in the benchmark model. Using more layers than the 1 layer that used in the benchmark model alongside modifying the dataset and turning the raw WAV files into spectrograms helped to reach accuracy over 60% on the validation data which is better than any score the benchmark model got.

The final CNN model with the spectrograms worked as expected and I think I can rely on for building a complete solution or on any speech recognition tasks.

Conclusion

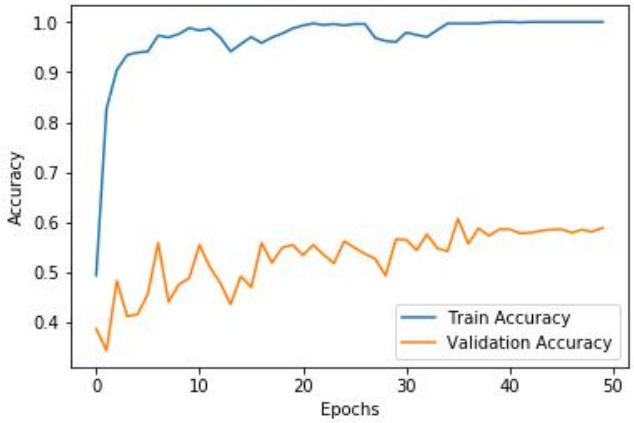
Free-Form Visualization

For this section, I will compare the 2 final graphs that show the performance of each model “The benchmark model and the final model” on both the training data and the validation data.
1- Benchmark model:
As the graph shows, the accuracy on the training data is faily good as it increases exponentially by increasing the number of epochs, but the accuracy on the validation data seems to be less than expected as the accuracy didn’t reach an accuracy over 20% regardless the number of epochs.




```
As we can see it acts good on training data, but poorly on the validation data.
```

2- The final model: As you can see from the graph below that shows the evolution of the accuracy over the epochs. It shows that the accuracy of the training data increased also exponentially over different epochs, but the difference than the benchmark model that the Validation Accuracy also increased on validation data and even reach levels that benchmark model couldn't able to achieve as it reached to 60%.




```
We can see that the accuracy on both training and validation data are significantly good.
```

Reflection

The trickiest part of my project was just working on sound files. Speech recognition tasks didn't cover much in the nano-degree, so I tried to explore the field and build a successful classification model.

The process took in the project start from getting a proper dataset, find a way to represent it to avoid using the raw WAV files in building the model. The way I found useful was spectrograms, so I decided to convert the raw WAV files to spectrograms. Then, I decided to use CNN to build the model as CNN proved accuracy on such type of tasks such as image classification tasks. After that, I built the model and the accuracy was as expected on both training and validation data.

Improvement

The improvement which I am looking forward to working on later is to try different models that proved accuracy on image classification tasks and try them out on my dataset after converting it to spectrograms.