A Fine-tuned Wav2Vec2.0 For Speech Emotion Recognition

Dolev Abuhazira[1] and Dor Azaria[2]

[1]dolevictory@gmail.com
[2]dorazaria@gmail.com

Abstract

This Article written as part of the course "Deep Learning Methods for Language and Voice Analysis" by Dr. Or Anidger from Ariel University.
In this article we explore the fine-tuning on Wav2Vec2.0 [1] a pre-trained model for classifying different emotions that received by voice.
In addition, we will present the different ways we performed for handling the data, from constructing the sample space to turning them into input for the model using different libraries.

1. Introduction

Our emotions are an integral part of our lives, they usually reflect our opinions and our desires. It can be seen that there is a clear connection between emotions and human behavior. The ability to identify emotions provides many options in various aspects such as social media, market research, customer experience and more. As part of our work, we have fine-tuned a pre-trained model that trains on raw audio data and learns its contextual representation. With the help of fine-tuning, we can tweak this model to perform our task which is emotion recognition. In this article, we will describe the architecture of our model, and also, we will describe how we construct the dataset with elements of signal processing.

2. Data

2.1 Dataset Collection

To train our model for emotion recognition, we searched across the web for data that would represent the same classification problem. After a long search on Huggingface [2] and Kaggle [**?**], we've found the following:

- RAVDESS Dataset [3]
  Contains 1440 audio files from 24 professional actors in North American accent. Speech includes calm, happy, sad, angry, fearful, surprised, and disgusted expressions and the song contains calm, happy, sad, angry, and fearful emotions.
- TESS Dataset [4]
  Contains 2800 audio clips of 2 women expressing 7 different emotions anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral.
- URDU Dataset [5]
  The dataset contains 400 audio files from 38 speakers that took part in Urdu talk shows, speech includes angry, happy, and neutral.

2.2 Class Distribution

Since the amount of data we collected was insufficient, we thought of merging the various classes into three classification classes to increase the accuracy of the model.
Now our model will have to classify the voice into the following three classes:

1. **Positive** - a mixture of happy and surprise.
2. **Neutral** - a mixture of neutral and calm.
3. **Negative** - a mixture of anger, fear, sad, and disgust.

After collecting the dataset and merging the classes we got the following distribution:

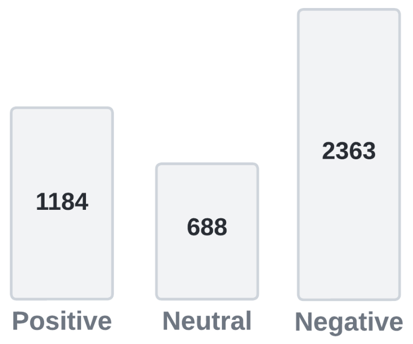


Figure 1 Class distribution

2.3 Data Pre-processing

The first stage of pre-processing was to collect datasets and merge all of them into one. For each audio file, we classified its label to one of the three classes (Positive, Neutral, Negative) and added it into the Panda's DataFrame by the following form: (file path, label).
For the second stage, we imported each WAV file from our DataFrame using ***touchaudio.load()***.
This ***load()*** function returns a waveform and sampling rate.
Before we're moving to the next stage, we need some theory about audio signal processing.
These audio samples are represented as time series, where the y-axis is the amplitude and the x-axis is the time series of the waveform. The amplitude's function measured the change in pressure around the receiver (a microphone).

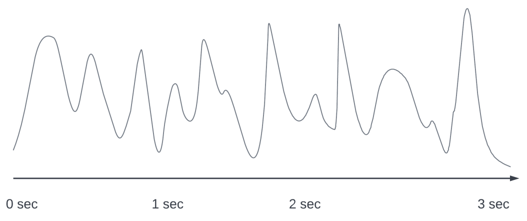


Figure 2 waveform of audio

One of the parameters of the ***touchaudio.load()*** function represents the number of frames to load. In this work, we trained the model with WAV audio files ranging from 2 to 3 seconds long and 16,000 samples per second. In signal processing, sampling is the reduction of a continuous signal into a series of discrete values. The sample rate is the number of samples taken per second. Each sample contains amplitude (loudness) information at that point in time. A sound is made by tens of thousands of frames that are played in sequence to produce frequencies. The WAV file contains a sequence of frames and multiple channels of data. Usually, a single microphone can make one channel of sound, and a single speaker can receive one channel of sound. For our model, we sampled mono-channels audios to 16kHz. Every audio is 3 seconds long, so for every audio, the number of audio signals is:

$$Seconds \times SampleRate \times Channels = 3 \times 16,000 \times 1 = 48,000$$

A high sampling frequency results in less data loss but higher computational expense, and low sampling frequencies have higher data loss but are fast and cheap to compute:

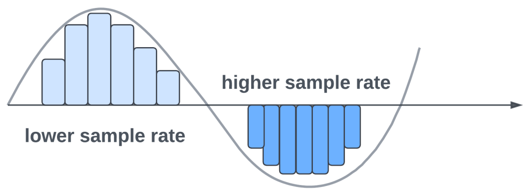


Figure 3 comparison between high and low sampling frequency

Since we fine-tune the Wav2Vec2.0 model, we must be sure that the size of the input is 48,000 samples, therefore we decided that each recording would contain exactly 48,000 samples (3 seconds long, 16k samples in a second). Since there are recordings shorter than 3 seconds, we had to fill in the missing samples by selecting random samples from the recording. This method fairly preserves the imagery of the voice.

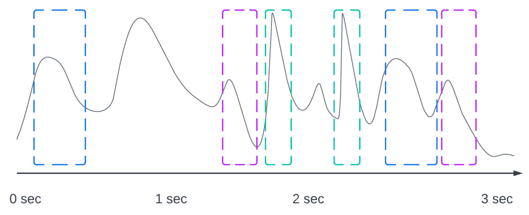


Figure 4 handling short recordings by padding samples randomly

2.4 Inference and fine-tuning

As mentioned above, we fine-tune a pre-trained model Wav2Vec2.0. The model was developed by researchers from Facebook.AI for the speech recognition tasks.
The architecture of the model is encoder-decoder. Encoding means converting data into a required format, and Decoding means converting a coded message into intelligible language.
The encoder of the model consists of several blocks containing a temporal convolution neural network to find a representation of the raw audio data and the output of the feature encoder are fed into transformer architecture which finds the contextual representation of the feature representation. the transformers do that by the self-attention layers. the self-attention mechanism allows the inputs to interact with each other and find out to who they should pay more attention to. The model was pre-trained and fine-tuned on 960 hours of Librispeech (a collection of approximately 1,000 hours of audiobooks) on a 16k sample rate speech audio.

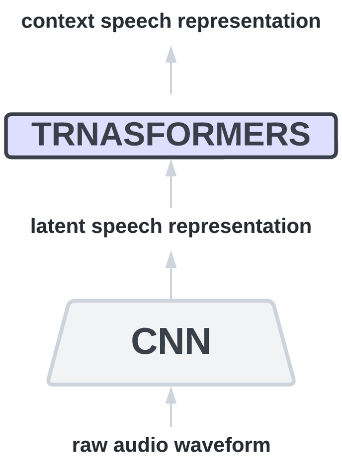


Figure 5 A very basic look at the architecture of Wav2Vec2.0

So far, we have collected 4235 raw voice data. After briefly explaining the pre-trained model, We rebuilt our data in such a way that our raw data is now contextually described by the Wav2Vec2.0 model.
Given the input to the model, the final output contains 12 vectors that describe a partial context from the context of the entire audio, for our purpose we using a matrix containing these 12 vectors.
this matrix describes the context of the audio received as input to the model.
Now, each of our samples from the raw data set will be passed as input to the Wav2Vec2.0 model and thus we will get a data set so that each example in it describes the context which is predicted by the model.

for each embedding matrix we make normalization feature scaling to bound the values between $[-1, 1]$. Let max be the greatest value and min be the lowest value from a set of features F. each feature value $1 \leq i \leq |F|$ is bounded between $[-1, 1]$ using the following formula:

$$F[i] = \frac{2 \times (F[i] - max)}{(max - min)} + 1$$

Finally, our new dataset contains 4235 normalized contextual representations for our old dataset, the size of each record - tensor in the dataset will be:

[1,149,32]

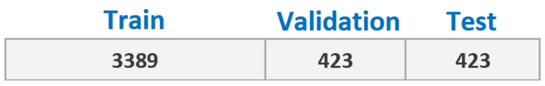
3. Architecture

Since the output of the pre-trained model is an embedding matrix that represents the context of the raw audio data, we chose to use a convolutional neural network whose purpose is to find specific patterns that will describe the emotion in the sound data. In each convolution layer, there are neurons so each neuron is a matrix that describes the different relationships between the data obtained from the input.
Our model consists of eight convolutional layers, and four pools layers whose purpose is to reduce the size of the matrices obtained from the convolution layers, one of the reasons for using these layers is the ability to optimize the calculation method and save processing time.

In each such layer, we performed batch normalization since it contributes to the increase in model training speed. The activation function in each layer is $ReLu$ and we chose to use it because its derivative is faster to compute in the back-propagation process.
In the last layer of convolution, we have implemented a regularization method called dropout whose purpose is to prevent overfitting during learning.
Also, after the data passes through the convolution layers they are fed to a linear neural network with one hidden layer.
The output layer of the neural network contains three neurons that specify the classes to which we want to classify the data.
Since we want to know what the probability is for each of the emotions we want to classify, the activation function in the last layer is $logSoftMax$, and our loss function is $CrossEntropyLoss$.

4. Training

After we made an inference and saved the new data set in a way that it would contain the context of each sound file, to train the model we divided the data as follows :



Train	Validation	Test
3389	423	423

Figure 6 splitting data into train, test, and validation sets

Since the division of the classes is not uniform, we have added weights to the classes which causes the model to penalize the misclassification made by the minority classes by setting a higher class weight and at the same time reducing the weight for the majority classes.
the weights that we defined are :

$$Positive : 1.99, Neutral : 3.0, Negative : 1.0$$

We first set the model to execute about 20 Epochs, so that in each Epoch we fed the model with a batch of size 32 , and the model train on learning rate with size 0.0001.
Because we used to validate our model performance during training with the validation set, We saw that there was overfitting in the eight epoch, so we decided to stop training in a method called Early Stopping, hence the amount of epochs performed in training is seven.

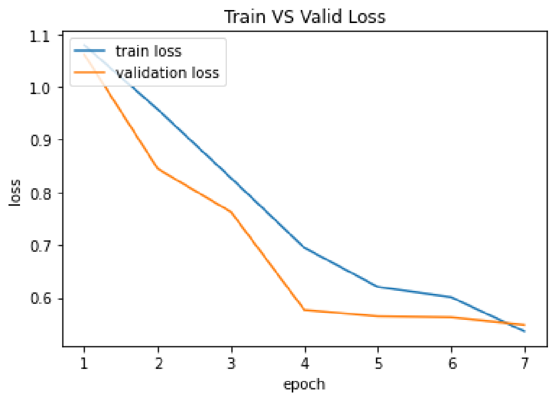


Figure 7 Early stopping method that perform on the seventh epoch

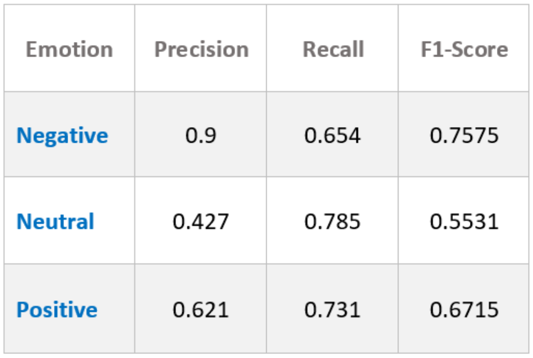
5. Results

After training the model, the moment has come to verify whether, given new examples of the model from the test group, he will be able to classify the emotion of the voices inputs. After performing the test, we received that the accuracy of our model stands at 70.43%. percent. And the accuracy of the model for each classes:

1. **Neutral** - 81.481%.
2. **Positive** - 72.881%.
3. **Negative** - 66.80%.

5.1 Performance Evaluations

To test our models, we use three different measurements to evaluate the performance: Recall , Precision, F-1 Score, and confusion matrix.



Emotion	Precision	Recall	F1-Score
Negative	0.9	0.654	0.7575
Neutral	0.427	0.785	0.5531
Positive	0.621	0.731	0.6715

Figure 8 The various measurements for model training

The measurements that we present above, are based on the confusion matrix which is a two-dimensional matrix that allows visualization of the model performance. It is a summary of the results of predictions on our classification problem task.

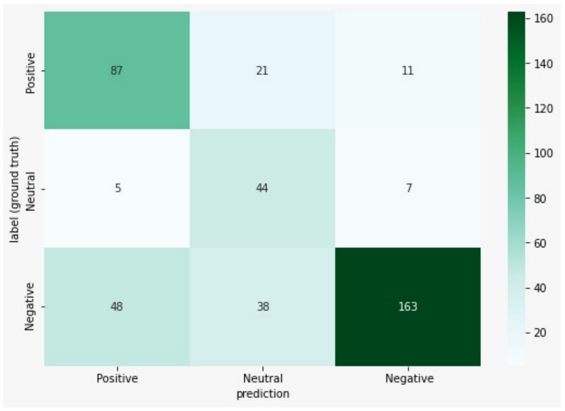


Figure 9 Confusion Matrix measurement

6. Conclusions

We have developed the advanced deep learning model for recognizing emotions in speech. We applied this learning model and demonstrated their results in the data set of RAVDESS, TESS and URDU, and got pretty good results regarding the fact that we had difficulty collecting a lot of data that representing the same classification problem.

In addition, during our work we came across many examples of sound files that were tagged differently from each other even though they sound pretty much the same.
We believe that if we had access to a wider and larger data set as well as a more balanced one, we would have achieved much better results.
As part of the course we learned to process raw audio and the principles behind signal processing, in addition, we learned an important principle which is Transfer learning which means taking the relevant parts of a pre-trained machine learning model and applying them to a new but similar problem.

7. Access to our project

All project components are at the following link :
Voice Emotion Recognition

Our project consists of various files: model architecture, training notebook, embedded data processing notebook, and Pth. file indicating the model we saved after training and the one that gave the best results for us. In addition, the main file through which you can run the program.

How to run?

1. Clone the voice emotion recognition repository
2. Enter the following command into your terminal :
   python3 Main.py

7.1 Required tools

Python3, Numpy, Pandas , sounddevice, Pytorch , turchaudio , matplotlib and more [6] .

References

[1] Wav2Vec2 Base 960H .
[2] Hugging Face .
[3] RAVDESS Dataset .
[4] TESS Dataset .
[5] URDU Dataset .
[6] Emotion Recognition in Greek Speech Using Wav2Vec 2.0 .