**SGN -24007**

**ADVANCED AUDIO PROCESSING**

**Singing Voice Separation   
using   
Deep Recurrent Neural Networks**

**Group members**

**Vishal Gaur (281683)**[Vishal.gaur@tuni.fi](mailto:Vishal.gaur@tuni.fi)  
**Ali Gohar (281668)**  
[Ali.gohar@tuni.fi](mailto:Ali.gohar@tuni.fi)

**Vladimir Vashchenko(281802)**Vladimir.vashchenko@tuni.fi

Introduction

Source separation from audio signals is an important real-world problem. For instance, better source separation will improve the accuracy in various speech recognition algorithms. During the recent years, there has been a drastic jump in the techniques used for source separation which has elevated the accuracy to a different level. But still research continues and day after day we get new techniques to figure out better, faster and efficient models to provide us with accurate results.   
For our current audio processing project, we created a GRU model to separate the sources from the **DSD100** dataset. The **DSD100** dataset is a mixture of stereophonic signals which are encoded at 44.1 kHz. It contains 100 full length music tracks of different styles along with their isolated drums, vocals, bass and other stems.   
The dataset also contains the audio files in two different folders, namely “Train” which contains 50 songs for training and “Test” which contains the remaining 50 songs for testing the model.   
To begin with we started by getting the clues from ***Research Paper name*** where we figured the transforms and deep recurrent neural network to use. After this we had the clues for source separation from the weekly exercises that we had already implemented during our course. And, finally we were able to separate the **Vocals** into a separate **.wav** file. Although there are different parameters that define the distortion rate and amount of interference in the separated signal, we were able to isolate the **Vocals** from other sounds along with the **SDR**, **SAR** and **SIR** ratios.

Implementation

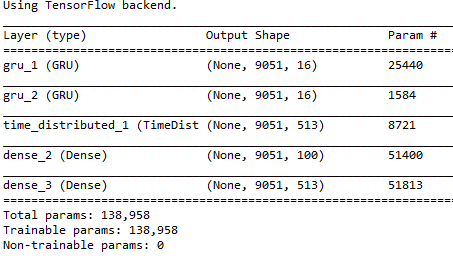
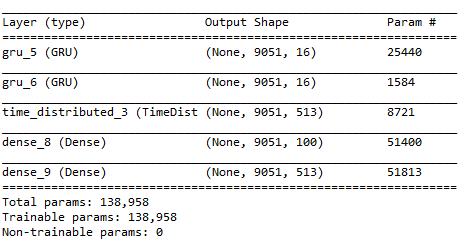
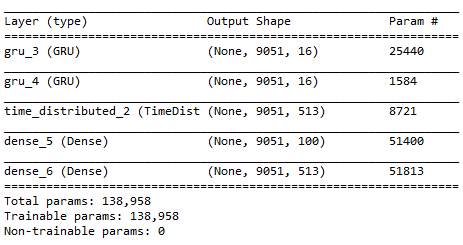
To begin with the problem, we followed the following steps -

1. Separated the Vocals and Other sounds (bass, drums) in two different variables.
2. Because the given audio signals are stereophonic, to avoid complications we took the mean of the channels and converted them into one.
3. Calculate the STFT of the input signals. (Music, Mixtures)
4. Estimate the mask using the IRM equation.
5. Split the data into testing and training parts by 50 percent.
6. Apply the mask on the given signals and store in the variables.
7. Train the GRU model created and predict the new vocals.

Formulaes Used

1. Claculating the STFT of input signals using the functions:-   
     
   For calculating the short term fourier transform of the given input audio signals we created the user defined functions which had the parameters as windowsize, fft\_size and the type of window used. For our specific project work we used Hamming Window.   
   Similarly, there are multiple user defined functions for Inverse STFT, DFT, Inverse DFT in a separate python script which are then imported in the main file to call the functions are get the values required.
2. Claculating the IRM mask for the given Audio Signal :-   
     
     
   For calculating the IRM or Ideal Ratio Mask, the formula used was alreaddy dicussed during our lecture session.   
     
     
      
   where the variables p and v can be modified to change the shape of the mask.

Evaluation and Visualization

1. For single audio file   
     
   We evaluated the mask prediction for one audio file by changing the hyperparameters for the GRU model and getting the mean values for Signal to Distortion Ratio (SDR), Signal to Interference Ratio (SIR) and Signal to Artifact Ratio (SAR) . The results obtained were:
   1. Mean SDR = -11.51   
      Mean SIR = -0.57   
      Mean SAR = -7.56   
        
        
        
        
        
        
      
   2. Mean SDR = -11.36   
      Mean SIR = -0.53   
      Mean SAR = -7.50   
         
        
        
        
      
   3. Mean SDR = -11.41   
      Mean SIR = -0.65   
      Mean SAR = -7.52   
        
        
        
        
        
      

Workload distribution

First, we studied the paper individually and then arranged a meeting for discussing the main idea of this project and confusions we had during the reading session. Later, we tried to implement the algorithm together using the branches on GitHub. We regularly updated the results on separate branches and decided to merge them at last. For the final report, we arranged a single day to write all the necessary information and edit it together.

 Limitations

As we understood, the research paper was almost 3 years old which makes this technique of sound separation almost obsolete. There have been new techniques which are, quite frankly faster and more efficient than the method we implemented in our project. Moreover, the training time for this algorithm is slower as compared to many state of the art methods present. (As concluded from practical implementation).   
Another limitation that we came across while training the model was the system configuration. Using this algorithm, when we try to run a single audio file it is quite slower than other methods. Whereas if we try to run the whole dataset, it will provide us with memory error. This is a huge limitation in this algorithm that we need to have a system with high GPU memory and high storage for larger datasets.   
One error that we came across is shown here:

A screenshot of a cell phone

Description automatically generated

Separation quality measurement and assessment

The quality of signal can be calculated by calculating the error signal between the original signal and the signal after addition of harmonic and percussive signals. The error is calculated in such a way that the signal x(t) contains the monophonic mixture of instruments and y(t) is the mixture of separated harmonic components. We calculated the error using the formula:   
 error = original signal in time domain - percussive components – harmonic components   
 err = ft – p [: length(ft)] – h [: length(ft)]   
The error came out to be -1.52 \* 10-6 which is almost zero.   
Ideally, the error should be close to zero and the sound to noise ratio should be high.   
  
While playing the original audio along side the harmonic+percussive audio, we could spot a slight noise variation, but most of the sound signal remained unaltered.   
Whereas while playing both the harmonic and percussive signals separately we could imitate the sound from the examples provided.

Conclusion  
  
From the results obtained, it is seen that the harmonic and percussive part, when played separately, are clearly perceived by human ear. Once they are combined, there is a slight variation in original and the processed audio signal. The spectrograms also show the horizontal lines in the harmonic spectrogram and vertical lines in Percussive spectrogram. The error came out to be almost zero and the SNR ratio is comparatively high.

REFERENCES

[1] K. M. J. L. R. H. K. a. S. S. Nobutaka Dno, «Separation of a monaural audio signal into

harmonic/percussive components by complementary diffusion on spectrogram,» в 16th

European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, 2008.   
[2] <http://www-users.york.ac.uk/~jes1/Audio_Signal_Processing_Research.html>