SGN -24007

ADVANCED AUDIO PROCESSING

Singing Voice Separation using Deep Recurrent Neural Networks

Group members

Vishal Gaur (281683)
Vishal.gaur@tuni.fi
Ali Gohar (281668)
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 the provided research paper [1] 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 (1024), fft_size (1024), hop length (512) 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.

$$M_j = \left(\frac{\hat{v}_j^p}{\Sigma_k \hat{v}_k^p} \right)^v$$

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:

   a. Mean SDR = -11.17
   Mean SIR = -0.21
   Mean SAR = -7.31

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru_7 (GRU)                  (None, 9051, 16)          25440
_________________________________________________________________
gru_8 (GRU)                  (None, 9051, 16)          1584
_________________________________________________________________
gru_9 (GRU)                  (None, 9051, 16)          1584
_________________________________________________________________
time_distributed_4 (TimeDist (None, 9051, 513)         8721
_________________________________________________________________
dense_11 (Dense)             (None, 9051, 100)         51400
_________________________________________________________________
dense_12 (Dense)             (None, 9051, 513)         51813
=================================================================
Total params: 140,542
Trainable params: 140,542
Non-trainable params: 0


The model that does mask prediction had: Mean SDR: -11.17 | Mean SIR: -0.21 | Mean SAR: -7.31
********************************************************************************************
```

b. Mean SDR = -11.36
Mean SIR = -0.53
Mean SAR = -7.50

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru_5 (GRU)                  (None, 9051, 16)          25440
_________________________________________________________________
gru_6 (GRU)                  (None, 9051, 16)          1584
_________________________________________________________________
time_distributed_3 (TimeDist (None, 9051, 513)         8721
_________________________________________________________________
dense_8 (Dense)              (None, 9051, 100)         51400
_________________________________________________________________
dense_9 (Dense)              (None, 9051, 513)         51813
=================================================================
Total params: 138,958
Trainable params: 138,958
Non-trainable params: 0
```

```
The model that does mask prediction had: Mean SDR: -11.36 | Mean SIR: -0.53 | Mean SAR: -7.50
********************************************************************************************************
```

c. Mean SDR = -11.34
Mean SIR = -0.27
Mean SAR = -7.51

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru_10 (GRU)                 (None, 9051, 16)          25440
_________________________________________________________________
gru_11 (GRU)                 (None, 9051, 16)          1584
_________________________________________________________________
gru_12 (GRU)                 (None, 9051, 16)          1584
_________________________________________________________________
time_distributed_5 (TimeDist (None, 9051, 513)         8721
_________________________________________________________________
dense_14 (Dense)             (None, 9051, 70)          35980
_________________________________________________________________
dense_15 (Dense)             (None, 9051, 513)         36423
=================================================================
Total params: 109,732
Trainable params: 109,732
Non-trainable params: 0
```

```
The model that does mask prediction had: Mean SDR: -11.34 | Mean SIR: -0.27 | Mean SAR: -7.51
********************************************************************************************************
```

2. For multiple Audio Files

   As we already changed the hyperparameters for a single audio file and got different results for ratios and vocal files. We implemented the same algorithm for multiple audio files from the dataset. Instead of providing different audio files to the model, we concatenated the audio files into a single variable and passed it to the training model. The result was a better voice separated audio file which can be heard on the drop box link given below.
   https://drive.google.com/file/d/1E6GEl50r1NoUx7G33FLOVnQDg3H7eEWn/view?usp=sharing

   Here is the model we used to train for multiple audio files:-

```
Using TensorFlow backend.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru_1 (GRU)                  (None, 83465, 16)         25440
_________________________________________________________________
gru_2 (GRU)                  (None, 83465, 16)         1584
_________________________________________________________________
gru_3 (GRU)                  (None, 83465, 16)         1584
_________________________________________________________________
time_distributed_1 (TimeDist (None, 83465, 513)        8721
_________________________________________________________________
dense_2 (Dense)              (None, 83465, 100)        51400
_________________________________________________________________
dense_3 (Dense)              (None, 83465, 513)        51813
=================================================================
Total params: 140,542
Trainable params: 140,542
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
1/1 [==============================] - 108s 108s/step - loss: 3.2570
Epoch 2/50
```

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 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:

```
  File "C:\Anaconda\lib\site-packages\tensorflow\python\framework\errors_impl.py",
line 528, in __exit__
    c_api.TF_GetCode(self.status.status))

ResourceExhaustedError: OOM when allocating tensor with shape[237444,513] and type
float on /job:localhost/replica:0/task:0/device:CPU:0 by allocator cpu
     [[{{node time_distributed_2/MatMul}} = MatMul[T=DT_FLOAT,
_class=["loc:@training_1/Adam/gradients/time_distributed_2/MatMul_grad/MatMul"],
transpose_a=false, transpose_b=false, _device="/job:localhost/replica:0/task:0/
device:CPU:0"](time_distributed_2/Reshape, time_distributed_2/kernel/read)]]
Hint: If you want to see a list of allocated tensors when OOM happens, add
report_tensor_allocations_upon_oom to RunOptions for current allocation info.
```

Conclusion

The given **DSD100** dataset is a sterephonic dataset which means that the audio files have two different channels. During our project, we concluded that instead of processing both the channels separately, if we take the mean for the channels and process it through the proposed model, we will get a fine audio file with improved vocal source separation. The resulted output audio files are also uploaded on GitHub for a single audio file process. When we trained the model for multiple files, we were able to reach the outcome, more the number of files for training, better the source separated file becomes. Since we were bound by system constraints and we were not able to utilize the whole resources of the computer lab (TC303), we trained the model on 5 and 10 audio files. The reulting output was uploaded to the google drive and the link is given in the above sections. If we train the proposed model on more number of files, we will be able to improve the voice separation to a better quality.

REFERENCES

[1] Po-Sen Huang, Minje Kim, Mark Hasegawa-Jhonson, Paris Smargdis,"Singing Voice Separation from Monaural Recodings using Deep Recurrent Neural Networks", University of Illinois, Urbana-Champaign, USA, 2014.
[2] https://github.com/aligohar73/Singing-Voice-Separation