Final Report
SCC0251/5830 Image Processing

VOICESPLIT: TARGETED VOICE SEPARATION BY SPEAKER-CONDITIONED SPECTROGRAM

A PREPRINT

Edresson Casanova[1,1],Pedro Regattieri Rocha[1,2],

1 Institute of Mathematics and Computer Science, University of São Paulo (USP), Sao Carlos - SP, 13566-590, Brazil

1 **NºUSP: 11572715 Email: edresson at usp dot br**
2 **NºUSP: 8531702 Email: pedro dot regattieri dot rocha at usp dot br**

July 12, 2020

ABSTRACT

Keywords: voice separation, speaker embedding, neural network model comparison.

1 Introduction

VoiceSplit's goal is the development of a system that, given an audio input, is able to separate overlapping voices through the use of Spectrograms, based on the characteristics of each speaker's speech patterns. Therefore, the system must be able to separate overlapping voices, thus helping to increase the performance of automatic speech recognition systems in scenarios with overlapping voices. In addition, the system can be used in tasks that involve filtering overlapping voices.

VoiceSplit's neural architecture is based on VoiceFilter [Wang et al., 2018]. The VoiceFilter model was trained using the Librispeech [Panayotov et al., 2015] and VCTK [Christophe et al., 2016] datasets. On the other hand, we chose to train only on the LibriSpeech data set due to computational limitations. LibriSpeech is a data set that has 982 hours of speech, with over 2,000 speakers.

This work's contributions are as follows:

- Proposes improvements for the architecture of the VoiceFilter model;
- It is the first work that compares the performance of different loss functions in this scenario;
- All of our experiments are open source and can be used freely by the community.

2 Methodology

2.1 Spectrograms

Audio pre-processing has an important role in speech systems, be it in Automatic Speech Recognition [Amodei et al., 2016], Speaker Recognition [Bredin, 2017], Voice Synthesis [Wang et al., 2017, Shen et al., 2017, Ping et al., 2017, Arık et al., 2017, Tachibana et al., 2017, Arik et al., 2017] or the different applications that deal with speech. The Mel-Frequency Cepstral Coefficients (MFCCs) attribute extraction technique [Davis and Mermelstein, 1990] was very popular for a long time, however, recently, filter banks have become more popular [Fayek, 2018].

Filter banks and MFCCs are obtained with similar procedures, as when filter banks are computed with some extra steps it is possible to obtain the corresponding MFCCs. An audio signal passes through a pre-emphasis filter, being

then separated into overlapping pictures e a window function is applied to each picture. Then, a Short-time Fourier Transform is applied to each picture and the energy spectrum is calculated, and finally the filter banks are obtained from this energy spectrum. To obtain the MFCCs, a Discrete Cosine Transform (DCT) is applied to the filter banks maintaining the resulting coefficient number, while extras are discarded [Mohamed, 2014].

The first step is to apply a pre-emphasis filter on the signal to amplify high frequencies. The filter is used for a few different reason, first, it balances the frequency spectrum, as high frequencies generally have smaller magnitudes in comparison to lower frequencies. Secondly, the use of the filter may also improve the Signal-to-Noise Ratio (SNR) [Ribani et al., 2004]. Finally, it prevents numerical problems during the Fourier Transform, which shouldn't be a problem for modern implementations [Fayek, 2018].

After the pre-emphasis step, it is necessary to divide the signal into smaller time frames. This is necessary because speech is a non-stationary signal, which means its statistical properties are not constant as time passes. Instead of analysing the entire signal, it is necessary to extract data from a small window of speech data which can characterize a phone or specific character, and for which we can make the assumption that the signal is stationary [Quintanilha, 2017]. For this motive, in most cases, it does not make sense to apply the Fourier Transform to the entire signal, as it would lose the contours of the frequency of the signal over time. Therefore, the Fourier Transform is applied to these small time frames instead, as this allows for a good approximation of the frequency contours by concatenating adjacent frames [Gordillo, 2013].

The final step before the filter bank is obtained is the application of triangular filters, in a Mel scale to the energy specter to extract the frequency bands. The Mel scale has the objective of imitating the non-linear perception of the human ear, being more discerning for lower frequencies and less discerning for higher frequencies [Fayek, 2018]. After obtaining the energy specter filtered by the Mel scale, the logarithm of the result is calculated, which will in turn give us the associated Mel spectrogram [Quintanilha, 2017]. To obtain a linear spectrogram, the use of the Mel scale to filter the energy specter is skipped, which keeps the result of the energy specter linear,

2.2 VoiceFilter System

The figure 1 presents the general architecture of the model. VoiceFilter consists of two parts trained separately: the Speaker Encoder (in orange) and the VoiceFilter system (in blue), which uses the output of the Speaker Encoder as an additional input. The VoiceFilter model does not directly predict the desired spectrogram. Instead it predicts a mask that is multiplied by the spectrogram used as input (which possesses the voices of two different speakers) and this mask removes everything but the voice of the speaker used as reference by the speaker encoder from the sample, that is, any background noise and the second speaker.

2.2.1 Speaker Encoder

The experiments realized in this work, such as VoiceFilter, receive as input a spectrogram with two voices layered on top of one another, and an embedding of one of the speakers, and they must output an spectrogram containing only the voice of the referenced speaker. The embedding of the speaker is a compressed representation of their voice. To generate this representation we used a system of speaker verification called GE2E [Wan et al., 2018], trained with Mel spectrograms. In our experiments we used two versions of the model. The first one, denominated GE2E2k was trained with the Vox Celeb 2 data set[Chung et al., 2018] that possesses approximately two thousand hours of speech. The second one, GE2E3k was trained with the data sets Vox Celeb, Vox Celeb 2 and LibriSpeech, totalling about three thousand hours of speech. Google's VoiceFilter model was trained with the GE2E encoder using the Vox Celeb, Vox Celeb 2 and LibriSpeech data sets along with a private data set that possesses about 34 million samples of 138 thousand individuals. As such, while we used an identical topology to the speaker encoder used in VoiceFilter, our models were trained with a smaller sample size, as we did not have access to their final data set.

2.2.2 VoiceFilter Network

The VoiceFilter network is composed of eight convolution layers, one LSTM layer and two fully connected layers, each one with ReLU activation, except the final layer, which possesses sigmoid activity. The values of the parameters are as shown in Table 1. The speaker embedding is repeatedly merged with the output of the preceding convolution layer during each time slice. The resulting vector is then used as an entry for the LSTM layer. During the VoiceFilter model's training, the input audios are divided in segments of 3 seconds each and are converted, if necessary, into single channel entries with 16kHz sampling rate.

Table 1: VoiceFilter Parameters adpated from [Wang et al., 2018].

Layer	Width		Dilation		Filters / Nodes
	Time	Freq	Time	Freq	
CNN 1	1	7	1	1	64
CNN 2	7	1	1	1	64
CNN 3	5	5	1	1	64
CNN 4	5	5	2	1	64
CNN 5	5	5	4	1	64
CNN 6	5	5	8	1	64
CNN 7	5	5	16	1	64
CNN 8	1	1	1	1	8
LSTM	-	-	-	-	400
FC 1	-	-	-	-	600
FC 2	-	-	-	-	600

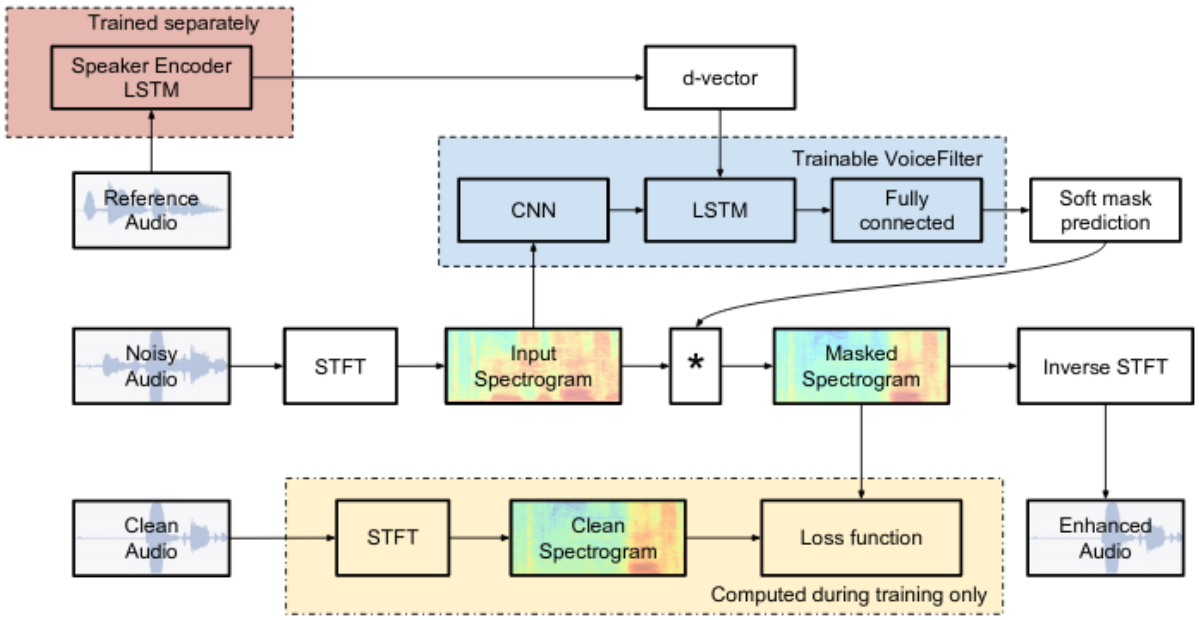

Figure 1: VoiceFilter architecture adapted from [Wang et al., 2018].

2.3 Dataset Preprocesing

We used the LibriSpeech data set to both train and test our models. The samples chosen were the same as the ones used in the work proposed by Wang et al. [2018].

The picture 2 presents the flow of the tasks to obtain the data for training of the VoiceFilter model. To create each training sample 3 audio files are needed, the audio target, that has the voice of only one of the speakers, the reference audio for embedding extraction, which is a voice sample of the speaker in the target audio and finally the interference audio, which is the sample that contains more speakers than the expected single speaker. All the audio files were converted into files with a single channel with 16kHz sampling rate, and were normalized using the library ffmpeg-normalize[1]. After that the initial and final silences were eliminated and only the first 3 seconds of each file were considered for the sample, except in the case of the embedding reference audio, which was considered in its entirety. The target audio and interference audios were overlapped in such a way that the resulting audio is a file that has both the voices of the target speaker and the interference speaker, and therefore is a mixed audio. After this step the spectrograms of the target, mixed and embedding files are extracted through the use of the audio reference sample.

[1]https://github.com/slhck/ffmpeg-normalizehttps://github.com/slhck/ffmpeg-normalize

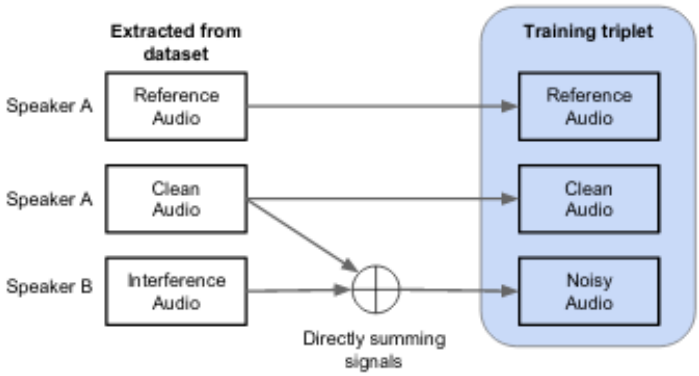

Figure 2: Data processing workflow adapted from [Wang et al., 2018].

For the extraction of the spectrograms and the training of the models the Librosa library [McFee et al., 2015] was used. For the extraction of the Mel spectrograms for the training of the speaker encoder, a 400ms window height and length was used for the Fast Fourier Transform(FFT), with 160 samples between successive frames, 400 FFT components e 40 Mel bands. For the extraction of the spectrograms used for the training of the VoiceFilter we used the same 400ms window sizes for the FFT, 160 samples between neighbouring frames and 1200 FFT components, three times the number used for the speaker encoder training.

2.4 Experiments

For all our experiments we used a DGX Station equipped with four NVIDIA V100 Tensor Core GPUs [2]. Our experiments were performed over a period of two weeks.

To compare our results with the results obtained in the VoiceFilter paper we used the demo archives available in the project's GitHub Page [3]. As VoiceFilter's formula for calculating the Source to Distortion Ratio (SDR) value was not clear, we could not replicate their results in our experiments, as even samples containing overlapping voices had positive SDR values (Their presentation does mention that samples containing large periods of silence will have higher SDR scores). We removed the initial and final silence of the audio samples for our experiments, so that each sample contains an audio signal throughout. Using only the available demonstration audios, we could not match the values obtained by VoiceFilter. For this reason we decided to use the Scale-Invariant Signal-to-Noise Ratio value (SI-SNR) [Le Roux et al., 2019], a measurement that has widespread use in current speech source separation works [Kadioglu et al., 2020, Luo and Mesgarani, 2019, Luo et al., 2020]. The higher the SI-SNR score, the better the voices in the sample are separated. As the calculation of the SI-SNRi score needs both voices to be separated, and VoiceFilter's demonstration page only has the reference embedding speaker's voice sample available, we had to subtract the predicted spectrogram from the mixed sample (i.e. the sample containing both voices) to obtain the second speaker's voice. This way, we were able to compute the SI-SNRi values for the original VoiceFilter paper. The final results obtained for the demonstration were an average SI-SNRi of 10.55729 and an average SDRi of 10.99677.

- **Experiment 1:** This experiment reproduces the VoiceFilter Network through the use of the Mean Squared Error (MSE) [Goodfellow et al., 2016] loss function and the GE2E2k speaker encoder.
- **Experiment 2:** This experiment replicates the voicefilter model, that is, it uses the Power-Law Compressed loss function [Wilson et al., 2018] as the original does, using the GE2E2k speaker encoder. For the λ in the Power-Law Compressed loss function we used the value 0.113, as it was the value that obtained the best results in the work proposed by Wilson et al. [2018]. This value was not reported in the original VoiceFilter paper, and we consider it a very important parameter for reproducibility.
- **Experiment 3:** This experiment replicates the voicefilter model, however it is trained using the SI-SNR loss function [Luo and Mesgarani, 2019] instead of Power-Law Compressed Loss function. Utterance-level permutation invariant training (uPIT) is applied during training to address the source permutation problem [Kolbæk et al., 2017]. This experiment uses the GE2E2k Speaker Encoder.

[2] https://www.nvidia.com/en-in/data-center/dgx-station/
[3] https://google.github.io/speaker-id/publications/VoiceFilter/

- **Experiment 4:** Similarly to Experiment 3 this experiment uses the SI-SNR loss function instead of the Power Law Compressed Loss function, however, we also replaced the ReLU activation function of the VoiceFilter model with the Mish [Misra, 2019] function. As with the previous experiments, this experiment also uses the GE2E2k speaker encoder.
- **Experiment 5:** Similarly to experiment 3, a VoiceFilter model is trained using the SI-SNR loss function instead of the Power Law Compressed Loss function, however, the GE2E3k speaker encoder is used instead of the GE2E2k. This experiment aims to verify the relationship between the increase in data in the training of the speaker encoder and the result of the VoiceFilter system.

3 Results and Discussions

All experiments we performed are freely accessible and are available in the VoiceSplit repository at Github[4]. The table 2 presents the SI-SNR of the original VoiceFilter paper and the results of each of the experiments proposed in the LibriSpeech test set, which are the same experiments used in the VoiceFilter paper. During all our experiments we used a part of the VCTK data set as a validation data set. This data set enabled us to choose the models that best suited each experiment.

Table 2: Results for Experiments.

Experiment	Avg SI-SNRi
VoiceFilter [Wang et al., 2018]	**10.55729**
1	6.02260
2	5.69875
3	5.66147
4	6.49116
5	**6.55238**

Experiments 1, 2 and 3 use the GE2E2k speaker encoder and the original VoiceFilter architecture. The worst result for these experiments was obtained in experiment 3 that uses the SI-SNR loss funcion and obtained an average SI-SNR of 5.66147, followed by experiment 2 which uses the Power-Law Compressed function and had a score of 5.69875. The best result for these three initial experiments was obtained in experiment 1 that uses the MSE loss function and obtained an average score of 6.02260. These results indicate that the MSE loss funtion is superior to the SI-SNR and Power-Law Compressed functions for the VoiceFilter model.

The purpose behind Experiment 4 was changing VoiceFilter's architecture, and so the experiment is similar to experiment 3, except the ReLU activation function is replaced with the Mish activation function [Misra, 2019], which showed improvements during the ImageNet challenge [Deng et al., 2009]. The main idea is to check if the Mish function can be used with success in tasks that involve speech, and, in this way, surpass results obtained with the ReLU function. The Mish function improved results and the average SI-SNR for experiment 4 was 6.49116, compared to experiment 3's average score of 5.66147. This was the second best result obtained in our experiments.

Experiment 5's objective is to verify whether or not the amount of data given as input to the speaker encoder affected VoiceFilter's results. To accomplish this we trained a new speaker encoder with one thousand extra hours of speech data and then we ran experiment 3 again, except this time we used the new speaker encoder. The new training data set increased the average SI-SNR from 5.66147 to 6.55238, which was the highest value obtained in our experiments.

The audios obtained after experiment 5 (Which had the best results) are available in the demonstration page: https://edresson.github.io/VoiceSplit/. Table 3 presents links to colabolatory containing a demo of each experiment, one through five.

Although we couldn't reach the SI-SNR obtained in the original VoiceFilter paper, our results indicated that the use of the MSE loss function instead of the Power-Law Compressed loss funtion using a λ de 0.113, may improve the SI-SNR of the VoiceFilter model by 0.32385. Furthermore, the replacement of the ReLU function by the Mish activation function may further improve the SI-SNR of the model by 0.79241, indicating that the Mish function may successfully replace ReLU in voice separation and also be promissing for other tasks that involve spectrograms, like automated speech recognition, voice synthesis and speaker identification.

We believe that the motives we couldn't reach the results of the original VoiceFilter paper was the amount of training data of their Speaker Encoder, as experiment 5 showed a 0.89091 improvement in the average SI-SRN over experiment

[4] https://github.com/Edresson/VoiceSplit/

Table 3: Experiment Demos.

Experiment	URL
Experiment 1	https://shorturl.at/eBX18
Experiment 2	https://shorturl.at/oyEJN
Experiment 3	https://shorturl.at/blnEW
Experiment 4	https://shorturl.at/qFJN8
Experiment 5	https://shorturl.at/kvAQ8

3, with an increase of one thousand hours in training audio duration. Additionally we trained the model only with the LibriSpeech data set while in the original the authors also used the VCTK data set. Furthermore, the VoiceFilter article did not report their parameters for the extraction of the spectrograms for VoiceFilter's training, they only did so for the extraction of the spectrograms related to the speaker encoder's training. Finally, as reported before, the value of the Power-Law compressed loss function's λ was not available.

The expected spectrograms and the ones obtained for some of the test set samples for experiment 5 may be visualized in the appendix A.

4 Conclusions and future work

In this work we replicated the VoiceFilter paper proposed in Wang et al. [2018], tested 3 different loss function and proposed improvements for the models. Although we could not completely simulate their results as we did not have access to their largest data set nor their SDR formula, we did find promising results with the usage of both the MSE loss function and the Mish activation function, which might be used to improve VoiceFilter.

This may lead to, for example, improvements to the quality of automatic speech recognition systems used in noisy environments and help with the separation of people having a conversation into different entries for the purpose of populating data sets.

In future works, we may test the Triplet Loss function by modifying model 4a in our data set with 80.000 audio files, similarly to experiment 9 (albeit without all combinations).

Acknowledgment

We would like to thank Coordination for the Improvement of Higher Education Personnel (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior — CAPES[5]) for financial support for this paper. We also appreciate the support of the NVIDIA corporation with the release of access to the GPUs used in the experiments presented in this research.

References

D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016.

S. O. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, J. Raiman, S. Sengupta, et al. Deep voice: Real-time neural text-to-speech. *arXiv preprint arXiv:1702.07825*, 2017.

S. O. Arık, G. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou. Deep voice 2: Multi-speaker neural text-to-speech. *arXiv preprint arXiv:1705.08947*, 2017.

H. Bredin. TRISTOUNET: TRIPLET LOSS FOR SPEAKER TURN EMBEDDING. (1), 2017. URL `https://arxiv.org/pdf/1609.04301.pdf`.

V. Christophe, Y. Junichi, and M. Kirsten. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit. *The Centre for Speech Technology Research (CSTR)*, 2016.

J. S. Chung, A. Nagrani, and A. Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In *Readings in speech recognition*, pages 65–74. Elsevier, 1990.

[5] https://www.capes.gov.br/

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

H. Fayek. Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between, 2018. URL http://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html.

I. Goodfellow, Y. Bengio, and A. Courville. Deep learning (adaptive computation and machine learning series). *Adaptive Computation and Machine Learning series*, page 800, 2016.

C. D. A. Gordillo. *Reconhecimento de Voz Contınua Combinando os Atributos MFCC e PNCC com Métodos de Robustez SS, WD, MAP e FRN*. PhD thesis, PUC-Rio, 2013.

B. Kadioglu, M. Horgan, X. Liu, J. Pons, D. Darcy, and V. Kumar. An empirical study of conv-tasnet. *arXiv preprint arXiv:2002.08688*, 2020.

M. Kolbæk, D. Yu, Z.-H. Tan, and J. Jensen. Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25 (10):1901–1913, 2017.

J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey. Sdr–half-baked or well done? In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 626–630. IEEE, 2019.

Y. Luo and N. Mesgarani. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 27(8):1256–1266, 2019.

Y. Luo, Z. Chen, and T. Yoshioka. Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 46–50. IEEE, 2020.

B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25, 2015.

D. Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 2019.

A.-r. Mohamed. *Deep neural network acoustic models for asr*. PhD thesis, 2014.

V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller. Deep voice 3: 2000-speaker neural text-to-speech. *arXiv preprint arXiv:1710.07654*, 2017.

I. M. Quintanilha. *End-to-End Speech Recognition Applied to Brazilian Portuguese Using Deep Learning*. PhD thesis, MSc dissertation, PEE/COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, 2017.

M. Ribani, C. B. G. Bottoli, C. H. Collins, I. C. S. F. Jardim, and L. F. C. Melo. Validation for chromatographic and electrophoretic methods. *Quimica Nova*, 27(5):771–780, 2004.

J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. *arXiv preprint arXiv:1712.05884*, 2017.

H. Tachibana, K. Uenoyama, and S. Aihara. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. *arXiv preprint arXiv:1710.08969*, 2017.

L. Wan, Q. Wang, A. Papir, and I. L. Moreno. Generalized end-to-end loss for speaker verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4879–4883. IEEE, 2018.

Q. Wang, H. Muckenhirn, K. Wilson, P. Sridhar, Z. Wu, J. Hershey, R. A. Saurous, R. J. Weiss, Y. Jia, and I. L. Moreno. Voicefilter: Targeted voice separation by speaker-conditioned spectrogram masking. *arXiv preprint arXiv:1810.04826*, 2018.

Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, et al. Tacotron: A fully end-to-end text-to-speech synthesis model. *arXiv preprint arXiv:1703.10135*, 2017.

K. Wilson, M. Chinen, J. Thorpe, B. Patton, J. Hershey, R. A. Saurous, J. Skoglund, and R. F. Lyon. Exploring tradeoffs in models for low-latency speech enhancement. In *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, pages 366–370. IEEE, 2018.

A Espectrogramas

In this appendix we present the expected, obtained and input spectrograms for a few experiments, obtained using our best model (Experiment 5's model), for some samples in the test set.

A.1 Model applied to audio with two overlapping voices

In this experiment the model must remove the voice of the the secondary speaker in the mixed audio.

Example 1:

Figure 3 represents the spectrogram of the mixed audio, which is generated by overlapping the clean audio of the reference speaker with an interference audio that features a secondary speaker.

Figure 4 represents the spectrogram of the expected audio, that is, the spectrogram of the audio file containing only de reference speaker's voice (the speaker in the clean audio).

Figure 5 represents the spectrogram predicted by the model that received as input both the mixed spectrogram and a sample audio embedding of the clean audio's speaker.

Example 2:

Figure 6 represents a mixed audio's spectrogram.

Figure 7 represents the target audio's spectrogram.

Figure 8 represents the predicted audio's spectrogram.

Example 3:

Figure 9 represents a mixed audio's spectrogram.

Figure 10 represents the target audio's spectrogram.

Figure 11 represents the predicted audio's spectrogram.

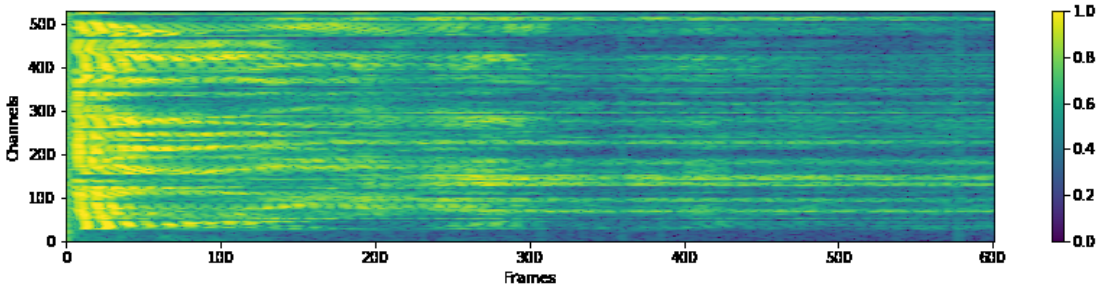

Figure 3: Spectrogram of mixed audio.

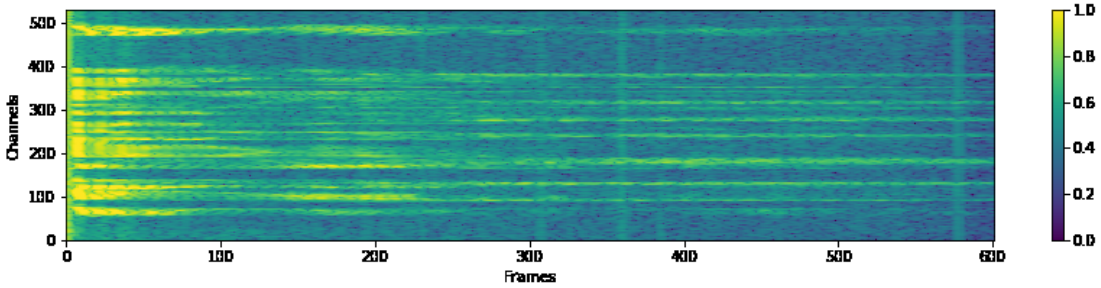

Figure 4: Spectrogram of target audio.

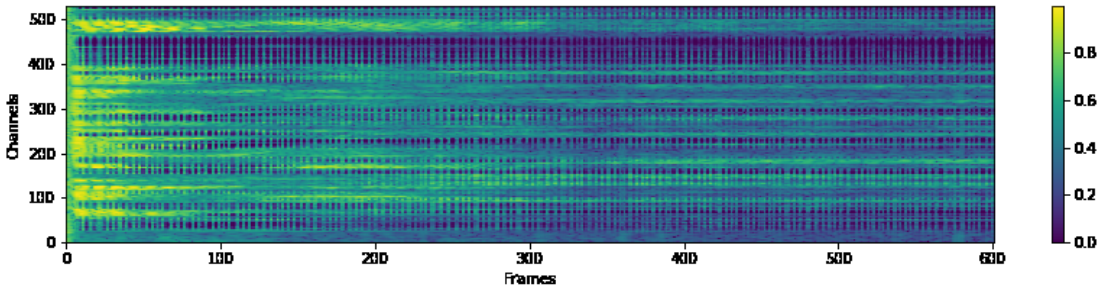

Figure 5: Spectrogram of estimated audio.

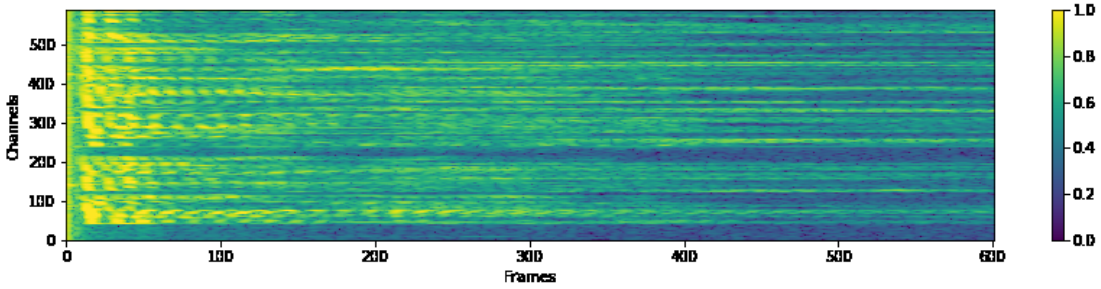

Figure 6: Spectrogram of mixed audio.

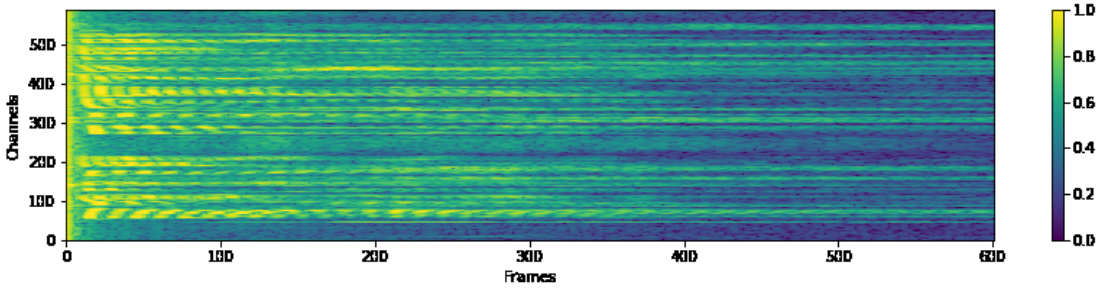

Figure 7: Spectrogram of target audio.

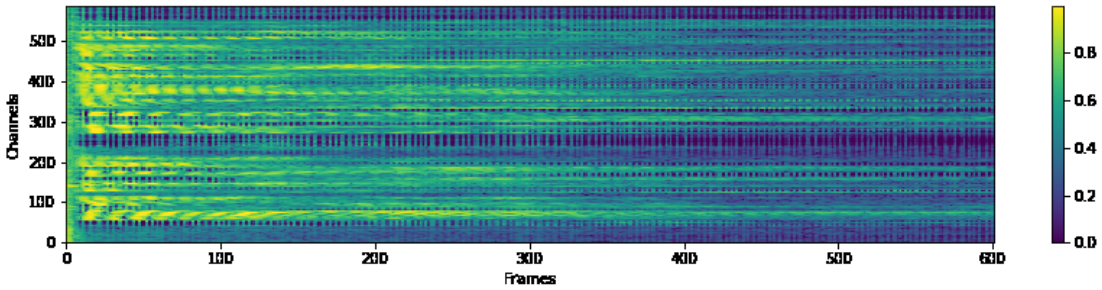

Figure 8: Spectrogram of estimated audio.

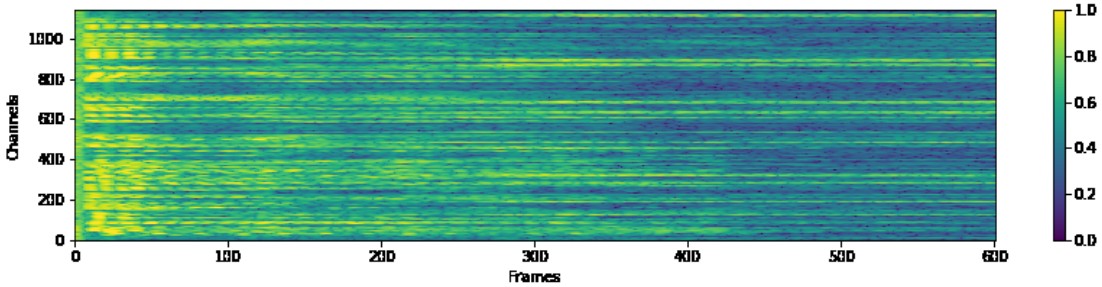

Figure 9: Spectrogram of mixed audio.

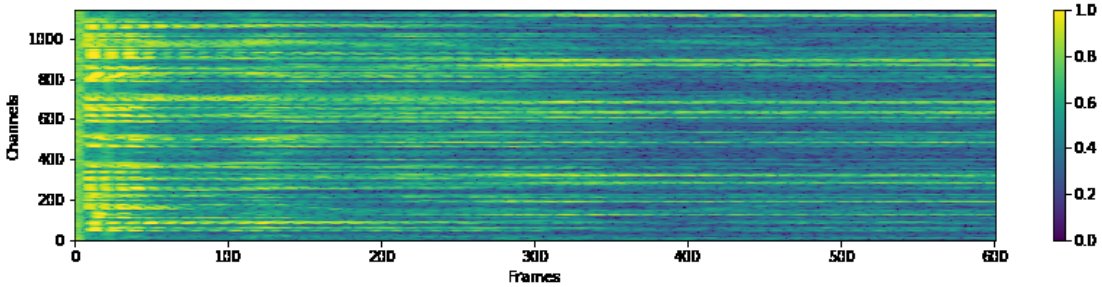

Figure 10: Spectrogram of target audio.

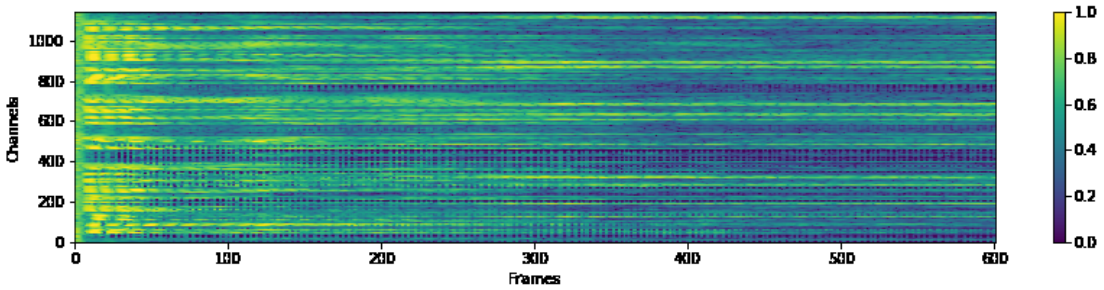

Figure 11: Spectrogram of estimated audio.

A.2 Model Applied to Clean Audio

In this experiment the model receives the expected audio as input and therefore should not change it, in other words, the output spectrogram should be as close as possible to the input one. This checks if the model is performing as intended, that is, it is only removing sounds that are not the desired speaker's voice.

Example 1:

Figure 12 represents the expected spectrogram, in this case the expected spectrogram is given as input to the model and is also considered the desired audio.

Figure 13 represents the spectrogram predicted by the model that received as input the clean audio, in which case the model should predict a spectrogram very similar to the input spectrogram.

Example 2:

Figure 14 represents the target spectrogram.

Figure 15 represents the predicted spectrogram.

Example 3:

Figure 16 represents the target spectrogram.

Figure 17 represents the predicted spectrogram.

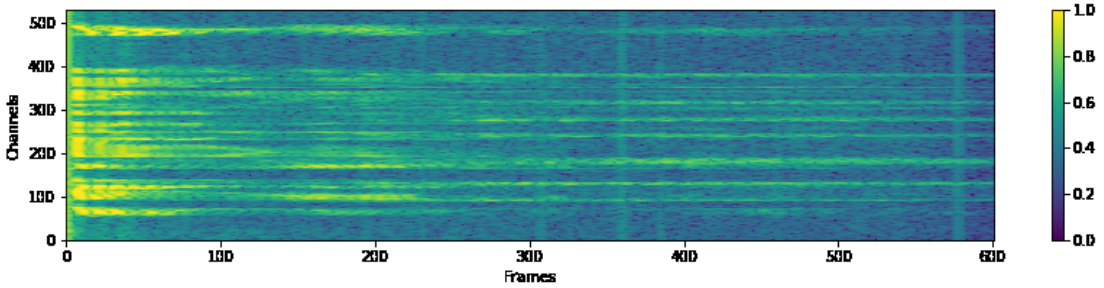

Figure 12: Spectrogram of target audio.

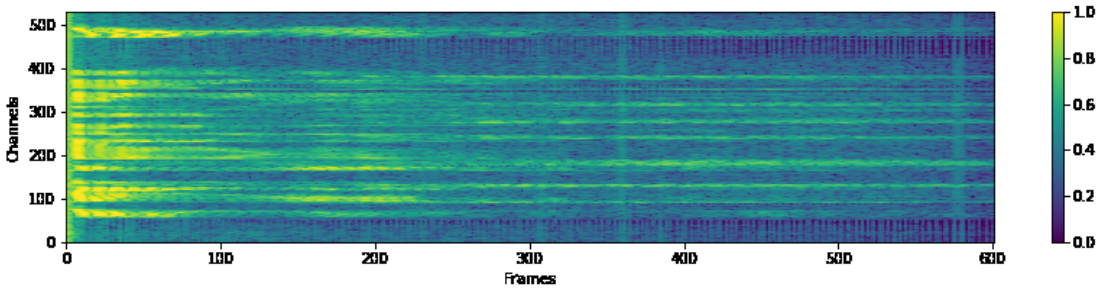

Figure 13: Spectrogram of estimated audio.

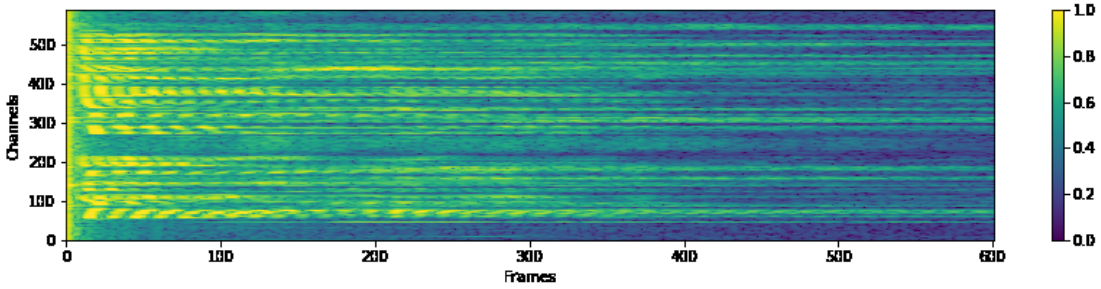

Figure 14: Spectrogram of target audio.

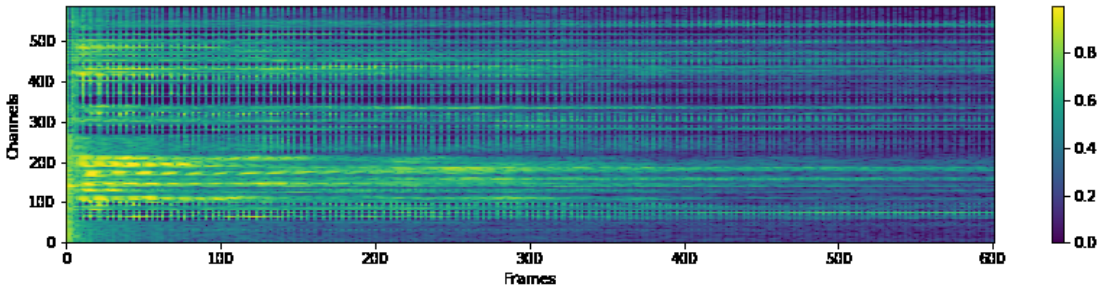

Figure 15: Spectrogram of estimated audio.

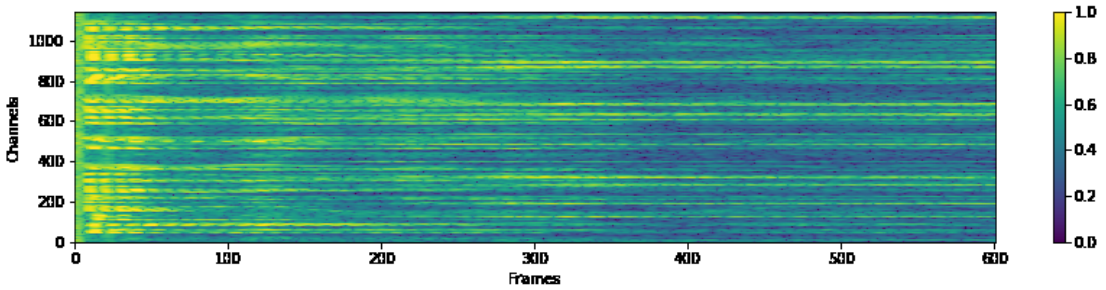

Figure 16: Spectrogram of target audio.

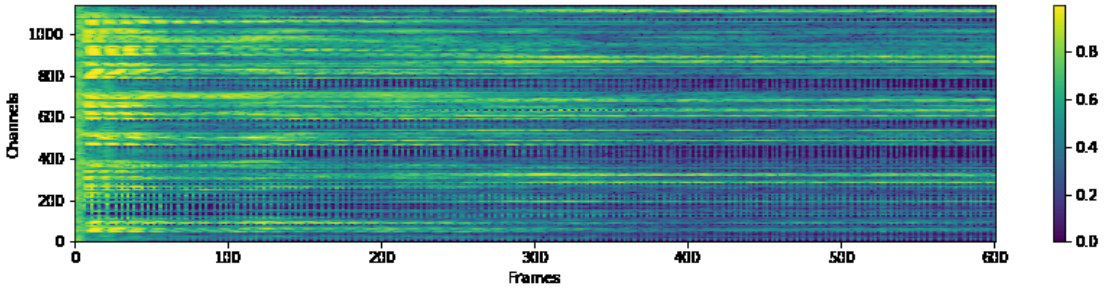

Figure 17: Spectrogram of estimated audio.