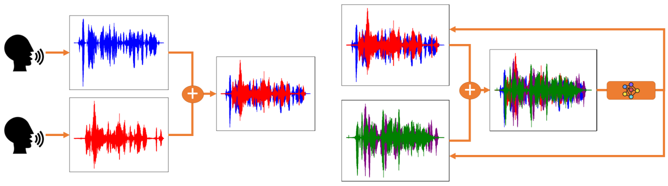
Excel@FIT 2021
http://excel.fit.vutbr.cz

Mixture of mixtures method for unsupervised speech separation

Ján Pavlus



Abstract
Speech classifier systems often fail on overlapped speech signals of more speakers as an input. For that reason, there are speech separation systems separating each speaker's signal from others to provide better input signals for further speech classification. In these separation systems, neural networks turn out to perform quite well. To train these networks it is necessary to have parallel mixtures and single speaker's signals as inputs and targets. Unfortunately, this criterion can not be frequently met for real mixtures. That also happens to be the reason why the training of the neural network is usually performed on artificial mixtures.
In this article, the mixture of mixtures method has been used to provide the required training on the unsupervised mixtures. This method was presented in the article *Unsupervised Sound Separation Using Mixture Invariant Training* [1]. This particular method mixes two existing mixtures into one called the mixture of mixtures and it is further being used as an input for the neural network. The original mixtures are used as training targets. Such a method enables training speech separation neural networks on full or partly unsupervised datasets. The unsupervised mixtures can be real recordings, which could lead to better separation results for real data during test time.
We combine the mixture of mixtures method with ConvTasnet and perform experiments on the fully unsupervised and semi-supervised datasets generated from the WSJ0-2mix dataset. In our experiments, this method fails on the fully unsupervised dataset and it also does not have any positive impact on the experiments with the semi-supervised datasets. We discuss the possible reasons for the failure and outline the future work.

Keywords: speech separation — mixture of mixtures — neural networks

Supplementary Material: Downloadable Code

*xpavlu10@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Speech separation systems are very useful for preparing speech signals for further speech recognition which often fail on signals with overlapped speakers. Speech separation systems can separate these overlapped speakers and therefore simplify the recognition process.

Nowadays the speech separation systems are mostly built with neural networks [2, 3]. The neural networks have to be trained on artificial mixtures mixed from known single speaker signals. There are a lot of these datasets and systems trained on those datasets that produce very good results for the matched data. Unfortunately, these systems mostly fail on real-life recordings of mixtures. This could be caused by the echoes of different premises as halls, public places, flats, conference rooms, etc., and other features of real recordings, which are impossible to simulate in the artificial mixtures.

The real-life mixtures are recorded without determining each speaker's signals, so they form unsupervised datasets. The benefit of these mixtures is the presence of the real-life features, as mentioned above, which offer better training of the system for future real deployment. However it is not easy to train the speech separation system on these unsupervised datasets because the conventional training requires known single speaker signals.

In the article *Unsupervised Sound Separation Using Mixture Invariant Training* [1] the authors present the Mixture of mixtures method with good results on the unsupervised and semi-supervised datasets. The mixture of mixtures method is using, as the name suggests, a mixture of two mixtures as the input for the training and these two mixtures as targets. The speech separation system implemented in the original article is based on *time-domain convolutional network*[4] which is very similar to *ConvTasNet*[5].

In our work, we run experiments on a speech separation system that tries to reproduce results from the mentioned article. Unlike the original Mixture of mixtures system, our system is using a *ConvTasNet*[5] neural network. Results obtained from our experiments do not match the original results, mainly for the fully unsupervised dataset. In the original article, there are very good results for that case, but we are not able to achieve them. The authors also mention the problem of over-separation, which could cause this poor result. This problem is further described in section 3. In the original article was not described neither solution of this problem nor the exact process which lead to successfully trained system on the fully unsupervised dataset. We do not meet this problem, but unfortunately we encounter malfunction of the used method itself as it occurs in our experiments.

2. Speech separation

The aim of the speech separation is to separate two or more signals from the given mixture, only with a minimal amount of information about the separated signals. We assume mixing model:

$$y_t = \sum_{n=1}^{N} x_{t,n} \tag{1}$$

where y_t is the mixture to be separated, $x_{t,n}$ is the source signal of speaker n, t is the time index and N is the number of sources. The speech separation target is to estimate all x_n from the given y.

For speech separation are mostly used neural networks. In this article, the system is built on the ConvTasNet architecture, which consists of encoder, decoder, and separation parts as it is shown in figure 1. The encoder is the convolutional block which gets a signal on the input and produces representation that resembles Short-time Fourier Transform (STFT) as an output. It will be further referred to this as a pseudo-STFT. This described behavior is learned during the training process.

The separation part consists of a series of consecutive convolutional blocks. Each convolutional block in the series is a filter, that is used on the bigger and bigger parts of the context. The number of these convolutional blocks in each series determines how much context will be taken into account. The separator takes a pseudo-STFT as input and by series of filters generates separation masks. These masks are then applied to the input mixture's pseudo STFTs given by the encoder. The result of this application are the estimated pseudo STFTs of the separated signals. Finally, there is the decoder part which is again a convolutional block trained to reverse pseudo-STFT. This block takes separated pseudo STFTs one by one and generates separated signals from them.

The Scale invariant Signal to noise ratio (SI-SNR) is used as a loss function. It is defined as[5]:

$$\vec{s}_{\text{target}} := \frac{\langle \hat{\vec{s}}, \vec{s} \rangle \vec{s}}{\|\vec{s}\|^2} \tag{2}$$

$$\vec{e}_{\text{noise}} := \hat{\vec{s}} - \vec{s}_{\text{target}} \tag{3}$$

$$\text{SI-SNR} := 10 \log_{10} \frac{\|\vec{s}_{\text{target}}\|^2}{\|\vec{e}_{\text{noise}}\|^2} \tag{4}$$

where $\hat{\vec{s}} \in \mathbb{R}^{1 \times T}$ is the estimated source. $\vec{s} \in \mathbb{R}^{1 \times T}$ is the original source signal used as the target and $\|\vec{s}\|^2 = \langle \vec{s}, \vec{s} \rangle$ denotes the signal power.

The neural network separates the signals from the mixture into outputs, but there is no pre-defined order in which the speakers should appear on the output. So loss function between the estimated outputs and ground-truth must be computed on all of the estimated outputs permutations. The permutation with the best result of the loss function is the one where the estimated outputs are attached to the targets in the correct order. This result is then used for training. In figure 2 there is the example of using PIT for two outputs and two targets. In this case, the loss function is computed between the $Output_1$ and $Target_1$, $Output_2$ and $Target_2$. Then the loss function is computed also between the $Output_2$ and $Target_1$, $Output_1$ and $Target_2$.

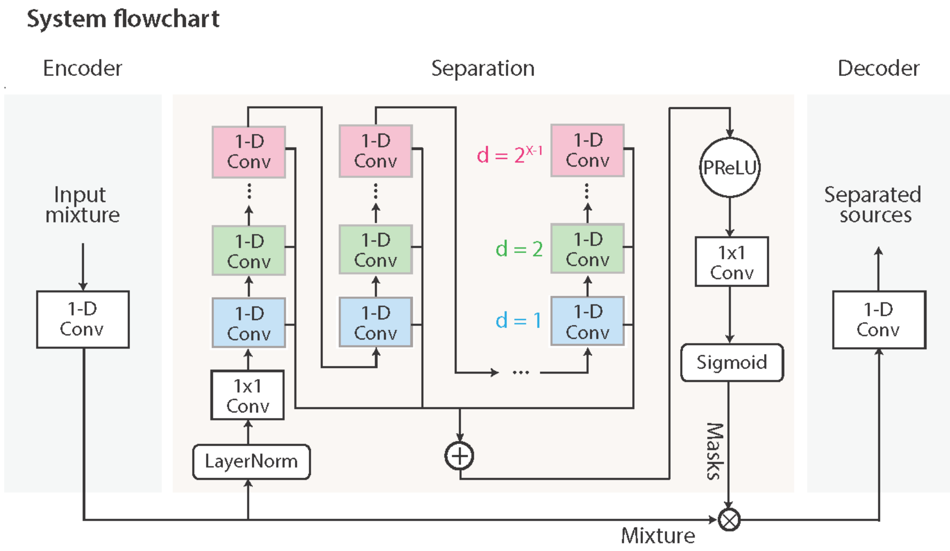


Figure 1. ConvTasnet flowchart [5]

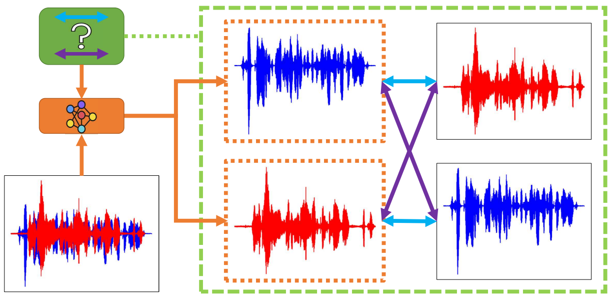


Figure 2. Example of permutation invariant training (PIT) with two estimated outputs by neural network in the orange dotted boxes and two ground truth targets on the right from them. PIT compares estimations and ground truth in all permutations and chooses the permutation with the best result.

The better result from these permutations is used as the result of the loss function.

3. Using mixture of mixtures

The Mixture of mixtures method enables training on unsupervised mixtures. This method takes two unsupervised mixtures and mixes them together to create the **mixture of mixtures**. This mixture of mixtures is used as the input and the original unsupervised mixtures are used as the training targets. It is also necessary to estimate n outputs, where n must be equal or greater than the number of estimated speakers signals, to train the neural network to separate a mixture of mixtures to the single speaker's signals and not only to original mixtures. For example, the neural network should have four or more outputs when the mixture of mixtures is mixed from two mixtures, that each contains two speaker's signals.

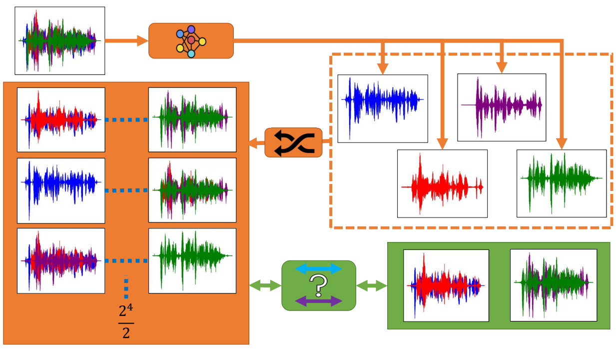


Figure 3. Mixture of mixtures method example for four outputs from the neural network. These outputs are mixed in various combinations as in orange box and compared to the ground truths in the green box using PIT method.

To compute the loss, the estimated outputs are mixed into two mixtures. There are 2^4 different ways how to assign four outputs to these two mixtures. Two examples of such assignments are:

1. first and second output are mixed together and the third and fourth output are mixed together and the loss function is computed between these mixtures and the target ones,
2. first output is not mixed with anything and the

other three outputs are mixed together and the loss function is computed again on these mixtures.

This process is defined as[1]:

$$\mathscr{L}_{\mathrm{MixIT}}(x_1, x_2, \hat{\mathbf{s}}) = \min_{\mathbf{A}} \sum_{i=1}^{2} \mathscr{L}(x_i, [\mathbf{A}\hat{\mathbf{s}}]_i), \tag{5}$$

where $\mathscr{L}$ is the SI-SNR loss and the *mixing matrix* $\mathbf{A} \in \mathbb{B}^{2\times M}$ is constrained to the set of $2 \times M$ binary matrices where each column sums to 1, i.e. the set of matrices which assign each source $\hat{s}_m$ to either x_1 or x_2.

Every two created mixtures are compared to the target mixtures and for each comparison loss is computed. The best result of the loss function is used for the training.

Using the Mixture of mixtures method on a fully unsupervised dataset can lead to an over-separating problem. This means that the expected speaker's signal is separated not only to one of estimated outputs but to more of them. For example, some parts of this speaker's signal can be separated to the output number one and the other to output number three. The over-separation is caused by not having any penalty in the Mixture of the mixtures loss function when the network over-separates the mixture of mixtures. These over-separated signals are mixed together and they create the target mixture. The loss value is thus the same as for good separation.

There is a simple solution, that can be used to prevent the over-separating problem. It is possible to use some percentage of the supervised mixtures in training, which leads to the semi-supervised dataset. Unlike of the mixture of mixtures, the supervised mixture has two target speaker's signals, so the classic PIT loss can be computed between estimated outputs and these target signals. This informs the separating system that the speaker's signals are what is desired as a result.

Although fully unsupervised datasets are in literature often stated as those being used for experiments, in reality, semi-supervised datasets would be more common. The over-separation problem is thus not a big issue.

4. Experiments

In our work we use Wall Street Journal mix dataset (WSJ) [6], which is publicly available. It consists of three parts which contain training, cross-validation, and testing data. The dataset contains both mixtures and parallel single-speaker recordings. Speakers are randomly mixed at random locations in synthetic rooms with anechoic conditions with various signal-to-noise ratios (SNR) between 0 dB and 10 dB. For training there are 20000 mixtures which means 30 hours, for cross-validation, there are 5000 mixtures so 10 hours, and 3000 mixtures, 5 hours, for testing.

In the experiments the percentage of supervised mixtures in the dataset is set to the various values from zero to one hundred, so the semi-supervised dataset is used. It is generated randomly at the start of the training, from the supervised WSJ dataset, which is described above. So therefore there is a high possibility that the two runs with the same parameters give us different results. To measure the possible difference between these runs and to get some verifiable results, there are 5 training runs of each percentage of supervised data in the dataset. Experiments use percentage values ten units apart, so 10%, 20%...

To prove that the Mixture of mixtures method works we also set up experiments that use only some part of the original dataset i.e. 10%, 50% and 80% of the mixtures. In these experiments only supervised mixtures are used to train the system. To prove that the method works these experiments should have worse results than the experiments with the semi-supervised dataset.

In contrast with the original article where TDCN++[4] is used, we use the ConvTasnet[5] as the neural network architecture. There are four outputs of the neural network, where the speaker's signals are estimated. This number is selected because there are originally present four speakers in the unsupervised mixtures. These mixtures are mixed from two mixtures, which both contain two speakers. The neural network also uses supervised mixtures, which contain only two speaker's signals. The used optimizer is the Adam with the exponential learning rate scheduler, where the initial learning rate is set to 0.001.

Trained systems are evaluated on the testing set of the original supervised dataset, which consists of artificial mixtures. These mixtures are separated by the trained system and then the evaluation starts. To deal with the higher number of estimated outputs than the expected speakers, we use white noise as additional targets. Then loss function is computed on all permutations of the estimated outputs. From these permutation with the best result, there are selected the outputs that match the original targets. The loss function is computed on these selected outputs and known targets and the result is used as system success metrics.

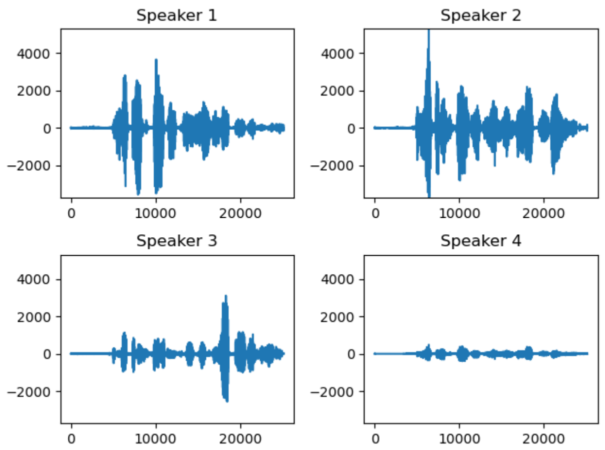


Figure 4. Example of not well estimated separation outputs by the system trained on the fully unsupervised dataset.

At first we run experiments with semi-supervised datasets created by the different percentage values of the supervised mixtures contained. These experiments should show how the system results depend on different amounts of supervised and unsupervised mixtures. Results of these experiments are shown in Figure 5. Systems trained without using any of the supervised mixtures achieve poor results around 2.4 dB of SI-SNR. Some of the runs get the results slightly higher than 3 dB of SI-SNR, but from listening to the results, there is not a bigger change compared with others. Such results differ from the results presented in the original article, where the results of the fully unsupervised training are around 11 dB which is very similar to the results of the semi-supervised ones.

For 5% of supervised mixtures, the average result is 8.4 dB of SI-SNR, which is about 6 dB better than for a completely unsupervised dataset. Adding as little as 5% of supervised data can thus lead to significant differences in the separation quality. However, the subjective quality of the single speaker's signals separated from the mixtures are still far from the clear speaker's signal and the recognition systems would probably still struggle with them. Note that with only 3% of supervised mixtures, the results are still comparable to 0%.

From the 10% of supervised mixtures in the dataset, the results come to the values around 10 dB of SI-SNR, and with the bigger and bigger amount of the supervised mixtures are the results gradually improving. The 10% and more experiments can be grouped into three groups. The first group with the results around the 10 dB of SI-SNR, contains the systems using 10% to 30% of supervised mixtures. The second group with the results around the 12 dB of SI-SNR, contains systems using 40% to 60%. Systems with a higher percentage of the supervised mixtures are getting the results about the 12.5% and they are in the third group. It is expected that the best results come from the systems trained on the fully supervised dataset, but some other trained systems from the third group reach a little bit better results.

On the other hand, the result on the fully unsupervised dataset is very low for an unknown reason. To test whether the mixture of mixtures method works and whether this problem appears only on the fully unsupervised dataset, we perform experiments without the unsupervised mixtures but only on parts of the WSJ dataset of various sizes. If the mixture of mixtures method works, worse results are expected in this case. However the results of these experiments as it is shown in Figure 6 are similar and sometimes slightly better. This shows that the mixture of mixtures doesn't boost the quality of the training results, maybe it makes it a little bit worse. These results show that the mixture of mixtures method does not seem to work in this process and further testing is required.

There are several differences between our setup and the one from the original paper [1]. As mentioned above, we use a slightly different type of neural network, ConvTasnet instead of TDCNN++. We also use the SI-SNR loss function in contrast with the original article, where the negative thresholded SNR loss function is used, which is defined as:

$$SNR = -10\log_{10}\frac{||\vec{s}||^2}{||\vec{s}-\hat{\vec{s}}||^2+\tau||\vec{s}||^2} \tag{6}$$

where $\vec{s}$ is the reference, $\hat{\vec{s}}$ is the estimation from a model and $\tau = 10^{-SNR_{max}/10}$, where SNR_{max} is set to the 30 dB [1].

Those differences may have some impact on the obtained results, but the difference between our result on the fully unsupervised dataset and the result from the original article is big. The mentioned differences could boost our results a little bit, but probably not enough to be similar to the original ones. However, in future work we will aim to remove these differences to get results strictly comparable to the original paper and experiment with the proposed method further.

5. Conclusions

Our work aims to reproduce the results from the article *Unsupervised Sound Separation Using Mixture Invariant Training* [1], where the Mixture of mixtures method was introduced. This method allows training the neural network on the unsupervised mixtures. The real-world mixtures are almost always unsupervised, so it could be possible to train the system on them.

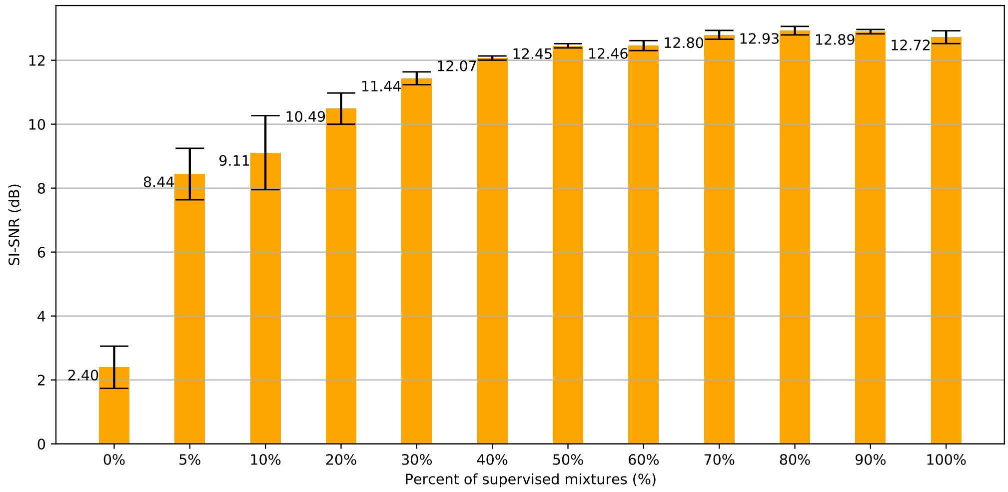


Figure 5. Results from three training runs for each percentage value of the semi-supervised dataset. The bars values are mean computed from these runs and there is also variance showed.

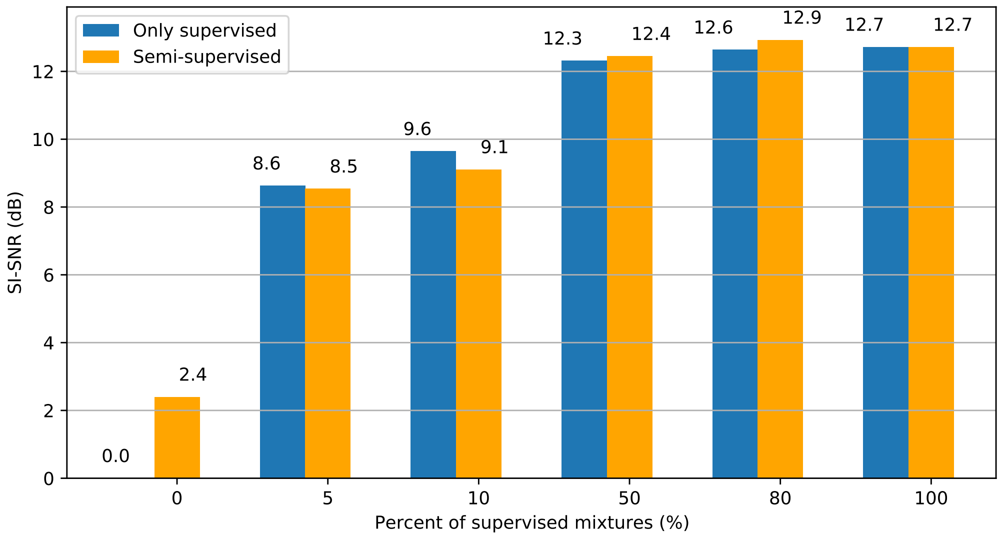


Figure 6. Results from the additional experiments. The orange bars show results for experiments on semi-supervised datasets. The blue bars shows results for experiments with supervised mixtures from the part of the dataset determined by a percentage value on the x axis.

This means the separation system will be able to train with the real-world features and could have better separation results on the real-world mixtures. We try to reproduce results on a slightly different neural network than the original one. However, these two networks are built on very similar architecture and they give very similar results.

In contrast with the original paper, in our experiments the mixture of mixtures method does not perform well. However experiments with the semi-supervised datasets containing the unsupervised mixtures for which the mixture of mixtures method is used, show better results that are similar to original ones. We provide other additional experiments on the only supervised part of the semi-supervised dataset. These experiments show that the unsupervised mixtures do not have any positive impact on the results.

This leads to the conclusion that this method does not work as it was expected. The next step will be to run further experiments with the same neural network as in the original article and with the same loss function. We also want to try to add mixture consistency projection layer to the outputs of the neural network and try to use different packs of combinations in the mixture of mixtures loss function. These experiments could prove that the method really does not work or we could find an error in our implementation. If the method happens to work despite the current results the next step will be to run experiments on the dataset containing the real-world unsupervised mixtures. These experiments could prove that the network trained on the real-world mixtures can lead to better results on the real data, than the one trained on only artificial mixtures. It would be useful to evaluate these experiments by a speech recognition system and compare the results of this system with results for the systems trained on the real-world and artificial training datasets.

Acknowledgements

This paper and the research behind it would not have been possible without the exceptional support of my supervisor Ing. Kateřina Žmolíková. I would like to thank her for her patience, ideas and the other necessary help and guidelines during the writing of this work. I also want to thank my dearest friend for the language and stylistic correction and mainly for her time.

References

[1] Scott Wisdom, Efthymios Tzinis, Hakan Erdogan, Ron J Weiss, Kevin Wilson, and John R Hershey. Unsupervised sound separation using mixtures of mixtures. *arXiv preprint arXiv:2006.12701*, 2020.

[2] DeLiang Wang and Jitong Chen. Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1702–1726, 2018.

[3] Yan-min Qian, Chao Weng, Xuan-kai Chang, Shuai Wang, and Dong Yu. Past review, current progress, and challenges ahead on the cocktail party problem. *Frontiers of Information Technology & Electronic Engineering*, 19(1):40–63, 2018.

[4] Ilya Kavalerov, Scott Wisdom, Hakan Erdogan, Brian Patton, Kevin Wilson, Jonathan Le Roux, and John R Hershey. Universal sound separation. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 175–179. IEEE, 2019.

[5] Yi Luo and Nima 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.

[6] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35, 2016.