BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

LEARNING SPEECH SEPARATION USING SPATIAL CUES

UČENÍ SEPARACE ŘEČNÍKŮ POMOCÍ PROSTOROVÉ INFORMACE

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **JÁN PAVLUS**
AUTOR PRÁCE

SUPERVISOR **Ing. KATEŘINA ŽMOLÍKOVÁ**
VEDOUCÍ PRÁCE

BRNO 2020

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2019/2020

Bachelor's Thesis Specification

Student: **Pavlus Ján**
Programme: Information Technology
Title: **Learning Speech Separation Using Spatial Cues**
Category: Speech and Natural Language Processing

Assignment:

1. Get acquainted with the problem of speech separation using neural networks and Deep clustering method.
2. Get acquainted with methods of obtaining spatial information from stereo recordings.
3. Train neural network for speech separation using spatial information as supervision.
4. Evaluate the method and compare with published results.
5. Suggest and discuss ways to improve results or extend the method.

Recommended literature:

- Seetharaman, Prem, et al. "Bootstrapping single-channel source separation via unsupervised spatial clustering on stereo mixtures." *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.
- Hershey, John R., et al. "Deep clustering: Discriminative embeddings for segmentation and separation." *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016.

Requirements for the first semester:

- Items 1 to 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Žmolíková Kateřina, Ing.**
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2019
Submission deadline: May 28, 2020
Approval date: November 5, 2019

Abstract

This thesis discusses the idea of using spatial cues in speech separation for estimating target masks, that is stated in article *Bootstrapping single-channel source separation via unsupervised spatial clustering on stereo mixtures*. This idea may make it possible to use real-world mixtures for the training of speech separation systems, which use neural networks. In the thesis two training methods, permutation invariant training and deep clustering method are mentioned and used for experiments with training neural networks using target masks estimated by spatial cues. The result of the work is a comparison of the results of these experiments with the results of the above-mentioned article. This comparison showed that the use of estimated masks with the help of spatial information can lead to a quality training of the speaker separation system.

Abstrakt

Tahle práce pojednává o možnosti použití prostorových informací pro odhadnutí masek pro cíle, které je uvedeno v článku *Bootstrapping single-channel source separation via unsupervised spatial clustering on stereo mixtures*. Tahle myšlenka umožňuje použití neumělých náhrávek směsice signálů pro trénování systémů separace řečníků, které používají neuronové sítě. V práci jsou zmíněny dvě trénovací metotody a to permutačně invariantní trénování a dále pak metoda deep clustering. Tyto metody jsou použity pro experimenty s trénováním neuronových sítí s použítím masek cílů, které jsou odhadnuty pomocí prostorové informace. Výsledkem práce je porovnání výsledků těchto experimentů s výsledky výše zmíněného článku. Tohle porovnání ukázalo, že použití odhadnutých masek za pomoci prostorových informací, může vést ke kvalitnímu natrénování systému separace řečníků.

Keywords

Speech separation, deep clustering, spatial cues, machine learning, neural networks, long short term memory

Klíčová slova

Separace řečníků, deep clustering, prostorová informace, strojové učení, neuronové sítě, long short term memory

Reference

PAVLUS, Ján. *Learning Speech Separation Using Spatial Cues*. Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Kateřina Žmolíková

Rozšířený abstrakt

Systémy separace řečníků umožnují z nahrávek, obsahujících směsici více hovořících řečníků, separovat jednotlivé signály. Separace řečníků je využívána pro zlepšení kvality výsledků systémů pro rozpoznávání hlasů. Tyto systémy se používají například v chytrých domácnostech nebo bezpečnostních zařízeních umístěných například na letištích.

Tahle práce se zabývá trénováním těchto systémů používajících neuronové sítě trénované metodou „deep clustering“. Především však tahle práce rozebírá řešení problému potřeby znalosti původních signálů nahrávky směsi pro trénování neuronové sítě. Tento problém je řešen za pomoci použití prostorové informace, z více kanálové nahrávky, pro odhadnutí cílových masek původních signálů. Tyto masky jsou pak použity pro trénování neuronové sítě namísto neznámých původních signálů jednotlivých řečníků.

Práce popisuje architektury neuronových sítí používaných pro separaci řečníků a podrobně popisuje LSTM bloky, které pracují s kontextem nahrávek. Sítě používající tyto bloky vykazují lepší výsledky trénování právě díky práci s kontextem nahrávky. Pro implementaci neuronových sítí, potřebných pro experimenty, byla použita knihovna Pytorch. Implementace práce probíhala v jazyce Python. Vstupem neuronových sítí jsou krátké Fourierovy transformace nahrávek směsi řečníků z otevřené multikanálové datové sady „Wall street journal“. Výstupem sítí jsou cílové masky pro jednotlivé řečníky dané směsi. Model sítě tvoří čtyři vrstvy LSTM bloků zapojených ve dvojím směru, tedy tzv. BLSTM bloky. Výstupní vrstva sítě se liší podle použité metody trénování.

Kromě trénovací metody „deep clustering“ je v práci také popsána permutačně invariantní metoda trénování. Obě metody jsou zároveň implementovány a použity při experimentech. Dále je také implementován algoritmus pro odhadování cílových masek řečníků pomocí prostorové informace. Tento algoritmus byl otestován a nevydával kvalitní výsledky. V práci byl tedy použit jiný algoritmus používající prostorovou informaci. Tento algoritmus pracuje s „cACGMM“ modelem, jehož výsledek zarovnává dle frekvencí pomocí „permutation aligment“ algoritmu. Testování jím odhadnutých masek vykazovalo dobré výsledky. Algoritmus byl tedy použit při experimentování.

Provedené experimenty ukázaly lehce nižší výsledky pro neuronovou síť trénovanou „deep clustering“ metodou, při použití masek odhadnutých za pomocí prostorové informace, než ve zpracovávaném článku. Výsledky s použitím permutačně invariantní metody trénování byly ale poněkud lepší než uvedené. Porovnáním výsledků referenčních experimentů s experimenty používajících prostorových informací, bylo zjištěno, že systémy používající prostorovou informaci dávají poněkud lepší výsledky. Dané chování se vysvětlilo možností jiného typu práce s přípravou dat z použité datové sady při jednotlivých metodách trénování.

Lepších výsledků při použití prostorových informací by mohlo jít dosáhnout použitím míry důvěryhodnosti. Tato míra byla zmíněna v článku a slouží k stanovení hodnoty kvality odhadnutých cílových masek za pomoci prostorové informace. Dle takto změřených by bylo možné vyřadit z trénovací sady data, pro která byly masky nekvalitně odhadnuty. Zároveň by bylo zajímavé otestovat sadu neuměle smíchaných nahrávek natrénovaných systémech, pro zjištění jejich funkčnosti v reálnem prostředí.

Learning Speech Separation Using Spatial Cues

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Kateřina Žmolíková. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Ján Pavlus
May 28, 2020

Acknowledgements

I wish to express my deepest gratitude to my supervisor Ing. Kateřina Žmolíková, who guided to me to the needed information and helped me with problems that showed during the writing of thesis and for infinite patience. Without her persistent help, the goal of this project would not have been realized.

Contents

Chapter 1

Introduction

Speech separation tries to solve problem of separating two or more speakers talking at the same time. In fact it is a system that gets mixed recording called mixture on input and estimate separated signals of each speaker as an output. Speech separation is in these days an expanding field, which can be useful for creating better hearing aids, building smart homes with voice assistants that can be also used in cars or other vehicles and public places. We can also fight against crime using these systems as we place them on airports, train stations, public places as historical monuments, shopping centers and government buildings, where they can detect criminals, terrorists and other potentially dangerous people. For all of these applications speech recognition systems are used. But they do not work well when speakers overlap. This cause significant degradation of the accuracy of that systems. That is why speech separation systems are so necessary to solve the overlapping problem and allow speech recognition system to work better.

System for speech separation can use simple machine learning algorithms as principal component analysis or independent component analysis. But nowadays separations systems are mostly based on deep learning and they are the most successful ones. These systems use many different neural network architectures and different training methods, but they all have one thing in common. They are mostly trained on artificial mixtures. The source signals of these mixtures are known, which mean that they are available to be used as a targets for training of the neural network, used for separation. But artificial mixtures are not same as real-world mixtures. Systems can therefore fail when real-world mixtures are separated. To solve that it would be better to use the real-world mixtures for training. But for real world mixtures, there is problem to obtain source signals of the speakers, to obtain targets for training. For stereo or multi channel records there is a possibility to estimate them by using a spatial information of speaker position. There are several algorithms using the spatial cues information for estimating these targets. In this work two of them will be described in chapter 2. First used algorithm was introduced in article called *Bootstrapping single-channel source separation via unsupervised spatial clustering on stereo mixtures* [18], second was introduced and described in article *Complex angular central Gaussian mixture model for directional statistics in mask-based microphone array signal processing* [9]. These algorithms do not directly estimate source signals, but only masks, that can be applied on given mixture. Result of this application give us the desired source signals.

Estimating source signals using the spatial cues information can be applied on artificial mixtures and also on real-world mixtures. This may solve the problem of missing source signals from real-world mixtures. It may be possible to train the neural network on large database of real-world mixtures, using estimated signals of that mixtures as an training

targets. These datasets can be created from any recorded stereo mixtures. But the question is if masks estimated this way will be of sufficient quality, to be used as an training targets and if the trained neural network will give the same or better results then that trained on artificial mixtures, with known source signals.

There are more methods used for training of neural network speech separation systems. In this work two will be used as described in chapter 4. Permutation invariant training method, which is generally known and commonly used simple method for supervised training of speech separation systems. Second method nowadays also commonly used is deep clustering method. This method provides supervised training of the neural network speech separation system. It promises good results of training by using embedding vectors of characteristics for separating speakers from mixture. In this work, speech separation system is trained with each of these two methods and then it is tested for multichannel data with known targets and also for data with estimated targets using spatial cues. Results of these tests are compared between these methods and conclusion is given.

The thesis is structured as follows. In chapter 2 we define speech separation problem in more detail, then describe some commonly used evaluation methods and define two methods of estimating target masks using spatial cues. chapter 3 will describe the basics of neural networks, benefits of using recurrent neural networks and describe step by step how long-short time memory blocks work. Using two different methods of neural network training, permutation invariant and deep clustering training will be described in chapter 4. Also estimating of target masks and testing spatial cues methods will be described in this chapter. System implementation will be shown in chapter 5. In chapter 6 it is described how the dataset for experiments looks like and how it was prepared. Also results of experiments and architecture of used neural network will be described there and chapter 7 will give final results and reflection on executed experiments.

Chapter 2

Speech separation

In speech separation, we have a mixture of two or more original signals. Original signals are either of individual speakers or different noises. The aim is to separate these signals from given mixture as best as possible, with very little information about original signals. Let us define it by:

$$y_{t,m} = \sum_{n=1}^{N} x_{t,m,n} \tag{2.1}$$

where $y_{t,m}$ is mixture to be separated, $x_{t,m,n}$ is speech signal of single speaker or noise, m is index of microphone, t is time index and N is number of sources.

In most systems the short-time Fourier transformation (STFT) is performed on signals. For source signal it is $X_{t,f,m}$ and for mixture it is defined as:

$$Y_{r,f,m} = \sum_{n=1}^{N} X_{r,f,m,n} \tag{2.2}$$

where r is frame index and f is frequency. Goal of speech separation is to recover each $X_{r,f,m}$ from mixture $Y_{r,f,m}$.

This problem can be also presented in the example of the 'Cocktail party problem', where there is a group of people talking over each other at a party. The listener tries to concentrate on one of the present discussions and separate it from others, to be able to communicate with these people. The human ear and brain are able to solve this problem. For computers, solving of this problem is very complex discipline.

Speech separation tries to solve this problem by classic machine learning methods as independent component analysis. But systems built on these algorithms work well only when the problem is greatly simplified. If silent blocks, echoes, and delays are present in mixtures these systems mostly fail and give poor quality results.

Nowadays most commonly used methods for speech separation are based on neural networks. These networks use known source signals of mixtures for training. For artificial mixtures source signals are known, but these are difficult to obtain for real-world recording, so it is impossible to use them for system training. This knowledge limits the range of mixtures that can be learned to be separated. But if the mixture is stereo or multi-channel recorded, then spatial cues can be used for estimating target masks, which can be applied to mixtures to obtain estimated source signals needed for training. So the neural network can be trained on a much bigger range of mixtures including real-world ones. How to use spatial cues for estimating targets is described in section 2.2.

2.1 Evaluation methods

To measure the quality of separation systems and allow their comparison, it is necessary to evaluate results given by the separation system, which can be a trained neural network or some kind of classification model as f.e. Gaussian mixture model (GMM). Evaluation methods are also very useful for generating datasets of mixtures with a different measure of complexity. In this work there are two types of units used. Signal to noise ratio, which is useful for creating artificial mixtures with specified ratio to other speakers and also for measuring the quality of the trained system for speech separation. Signal to noise ratio (SNR) measures ratio of powers of desired signal to the background signal f.e noise. It is defined as:

$$SNR = \frac{P_{signal^{(0)}}}{P_{signal^{(1)}}} \tag{2.3}$$

SNR can be also showed in decibels as:

$$SNR_{db} = 10 \log_{10}(SNR) \tag{2.4}$$

The second metric is signal to distortion ratio, which is used to measure the quality of the trained neural network and also to compare different types of systems used in this thesis. This unit is also counted in decibels (dB) and it is defined as [11]:

$$SDR_{db} = 10 \log_{10} \frac{||s_{target}||^2}{||e_{noise}||^2} \tag{2.5}$$

where e_{noise} is defined as:

$$e_{noise} = \hat{s} - s_{target} \tag{2.6}$$

where $\hat{s}$ is estimated target and s_{target} is original target defined as:

$$s_{target} := \frac{< \tilde{s}, s > s}{||s||_2} \tag{2.7}$$

The higher value of SDR of estimated target compared to original target, the better separation system.

2.2 Spatial cues

Spatial cues can be used for speech separation of stereo mixtures. The main idea is that time-frequency bins with similar spatial information belong to the same speakers because they probably came from the same direction, where speaker is. In contrast different spatial information means different speaker. This can be disrupted, if speakers stand too close in one direction from microphones, for example. There will be very similar spatial information for both speakers and spatial source separation will fail. However it works very well for most situations.

There are more algorithms for blind spatial source separation. In an article called *Bootstrapping single-channel source separation via unsupervised spatial clustering on stereo mixtures* [18] there is spatial source separation provided by specifying interchannel phase

difference of short-time Fourier transform (STFT) spectrograms of the mixture from each of two channels, as described in equitation 2.3.

$$\theta_{t,f} = \angle(\mathbf{X}_{t,f}^{(0)}\overline{\mathbf{X}_{t,f}^{(1)}}) \tag{2.8}$$

where $\mathbf{X}_{t,f}^{(i)}$ is STFT spectrogram computed on channel i of mixture. $\theta_{t,f}$ is obtained inter-channel phase difference.

Then there is sine and cosine computed from $\theta_{t,f}$, that is stacked and projected to a single dimension by principal component analysis (PCA) algorithm. The single-dimensional result is then clustered by the Gaussian mixture model (GMM) using the full covariance matrix and using the expectation-maximization (EM) algorithm. In the article it is also mentioned to use the threshold to cluster only time-frequency bins with significant energy. In this work, the threshold mask (TM) is defined as:

$$\mathbf{TM}_{t,f} = \mathbf{X}_{t,f}^{(0)} > max(|\mathbf{X}_{t,f}^{(0)}|) * \tau \tag{2.9}$$

where $\mathbf{X}_{t,f}^{(i)}$ is STFT spectrogram computed on channel i of mixture and τ is threshold constant which was manually set to 0.001.

The second method to obtain target masks used in this work was introduced in article *Complex angular central Gaussian mixture model for directional statistics in mask-based microphone array signal processing* [9] and also mentioned in article *Unsupervised training of a deep clustering model for multichannel blind source separation* [3], using complex angular central Gaussian (cACG) observation model, which is defined as:

$$cACG(\tilde{\mathbf{y}}_{tf}, \mathbf{B}_{kf}) = \frac{(D-1)!}{2\pi^D det\mathbf{B}_{kf}} \frac{1}{(\tilde{\mathbf{y}}_{tf}^H \mathbf{B}_{kf}^{-1} \tilde{\mathbf{y}}_{tf})^D} \tag{2.10}$$

where D is number of channels, $\mathbf{B}_{kf}$ is distribution parameter, $\tilde{\mathbf{y}}_{tf}$ is normalized complex-valued observation vector, defined as $\tilde{\mathbf{y}}_{tf} = \mathbf{y}_{tf}/||\mathbf{y}_{tf}||$. The model for vectors $\tilde{\mathbf{y}}_{tf}$ is defined as:

$$p(\tilde{\mathbf{y}}_{tf}; \theta) = \sum_k \pi_{kf} cACG(\tilde{\mathbf{y}}_{tf}, \mathbf{B}_{kf}) \tag{2.11}$$

where θ represents model parameters $\theta = \{\pi_{kf}, \mathbf{B}_{kf} \forall k, f\}$.

This model is first trained to fit over given mixture to provide the clustering of spatial information in all time-frequency bins. Then posterior probabilities for each speaker in each time-frequency bin are predicted. cACG mixture model does not reflect frequency dependencies. This problem is known as the frequency permutation problem described in article *Measuring dependence of bin-wise separated signals for permutation alignment in frequency-domain BSS* [16] and the solution for it is the permutation alignment algorithm presented in *pb_bss* package [3].

Chapter 3

Neural Networks

Neural networks (NNs) [6] are models that can learn to map inputs to outputs. They are composed of a set of neurons formed into layers. Neurons are based on the biological neuron. As it is shown on Figure 3.1 their mathematical model is defined as:

$$y = f(\mathbf{w}^\top \mathbf{x}) \tag{3.1}$$

where $\mathbf{w}$ is vector of input weights, $\mathbf{x}$ is vector of input values, $f(\cdot)$ is activation function and y is the output value.

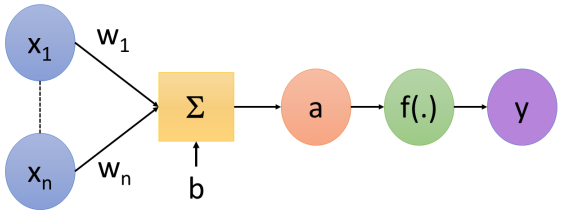

Figure 3.1: Mathematical model of neuron.

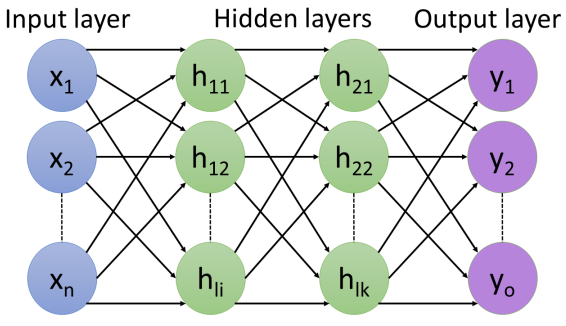

Figure 3.2: Example of neural network with two hidden layers.

As it shown on Figure 3.2 neurons form layers. Input layer, output layer together with hidden layers forms whole neural network. That can be defined as sorted triple [10]:

$$(N, R, w) \tag{3.2}$$

where N is set of neurons. R is set of connections between neuron i and neuron j defined as $\{(i,j)|i,j \in N\}$. Function $w : R \to \mathbb{R}$ defines weights for connection (i,j) written as $w_{i,j}$. If w has zero value, it means that there is no connection between neuron i and j.

3.1 Training Neural Networks

Training procedure consists of two repeating steps: training and validating. In the training step, NN computes output for each mixture input data. There are no raw data used in computation, but the STFT computed on them. Outputs are processed by objective functions and compared to given targets. Difference or how many mistakes NN made is represented by loss value. This value is used in backpropagation algorithm, which modifies weights in NN, in the other words, trains it. After training step, system continues by cross-validating the actual system state, to see whether it generalizes to unseen data. Cross-validation is very similar as training, but backpropagation algorithm is not called after computing losses. This step is also processed on different data then the training is. Training of NN ends when it reaches manually specified number of epochs or accomplishes another specified rule e.g. there is no progress in results of loss function in the last five epochs.

To speed up and refine the training process, the system makes batches. Batches are groups of input data sequences, that will be processed in a parallel way. Batches are constructed from randomly selected data with the same shapes on the start of the training procedure and the order of input data sequences is changing in time, which improves the training results. If batches are used, loss functions are computed on each data in batch, and results are averaged, to give a value for the backpropagation algorithm.

Mean square error

Mean square error loss function measures mean squared error between each element in the input x and target y, which is used for regression It is defined as [14]:

$$l(x,y) = L = \{l_1, ..., l_N\}^T \tag{3.3}$$

$$l_n = (x_n - y_n)^2 \tag{3.4}$$

where N is the batch size.

Binary cross entropy

Binary cross entropy loss function measures cross entropy error between each element in the input x and target y, which is used for binary classification. It is defined as [14]:

$$l(x,y) = L = \{l_1, ..., l_N\}^T \tag{3.5}$$

$$l_n = w_n[y_n * \log x_n + (1 * y_n) * \log(1 - x_n)] \tag{3.6}$$

where $\mathbf{w}$ is weight vector of size N which is batch size.

3.2 Recurrent neural network

The main feature of the recurrent neural network (RNNs) is remembering the past. This means that for the same input RNNs can produce different outputs depending on previous inputs in the series [19]. That is caused by the fact that outputs are influenced not only by weights applied to inputs but also by a hidden state vector representing the context of the previous decision. For example, the traditional architecture of RNN looks like this [2]:

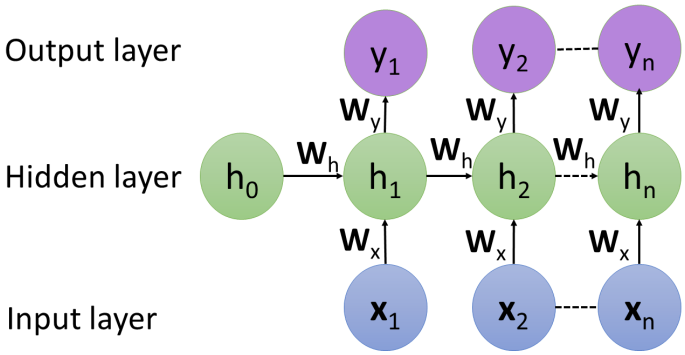

Figure 3.3: Example of recurrent neural network, where x_t is feature vector for time t.

For each timestep t, starting from one, the activation n_t and the output y_t is expressed as follows:

$$n_t = g_1(\mathbf{W}_h n_{t-1} + \mathbf{W}_x x_t + b_n) \tag{3.7}$$

$$y_t = g_2(\mathbf{W}_y n_t + b_y) \tag{3.8}$$

where x_t is feature vector for time t, $\mathbf{W}_h$ is weight coefficient matrix for hidden state, $\mathbf{W}_x$ is weight coefficient for input, $\mathbf{W}_y$ is weight coefficient for output, b_n, b_y are biases and g_1, g_2 are activation functions.

3.3 Long Short Term Memory networks

The problem of RNNs is in learning from a distant past. In theory they can do it, but it has been explained in the article *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies* [7]. Long short term memory networks (LSTMs)[8] are designed to remember long-term dependencies.

LSTMs[13] have a chain-like structure similar to RNNs. The difference is in the repeating module, which has four neural network layers instead of one. Unlike RNNs, LSTMs have a cell state, which is a value that is forwarded through each time step and it is changing based on actual hidden state.

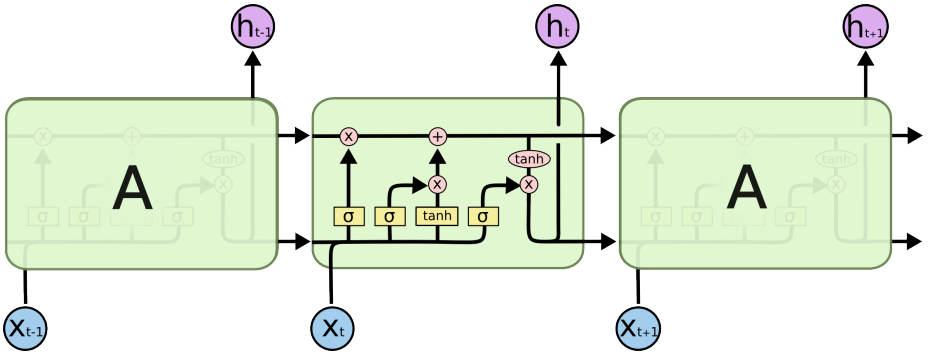

Figure 3.4: LSTM cells chain[13].

LSTMs do the computation in four steps. As shown in Figure 3.5 first step is deciding which information we are going to remove from the cell state. This is provided by the „forget gate layer“, which consists of one sigmoid layer. It makes decision by using the formula:

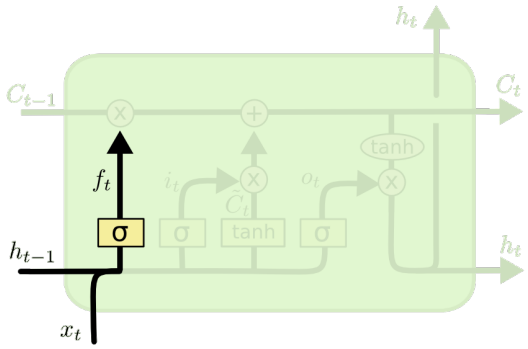

Figure 3.5: Forget gate layer[13].

$$\mathbf{f}_t = \sigma(\mathbf{W}_f * [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \tag{3.9}$$

In the second step LSTMs decide what new information it is going to add into the cell state. This consists of a hyperbolic tangent layer that creates a vector of new candidate values that could be added to the state and „input gate layer“ that decides how important will be each of these candidate values.

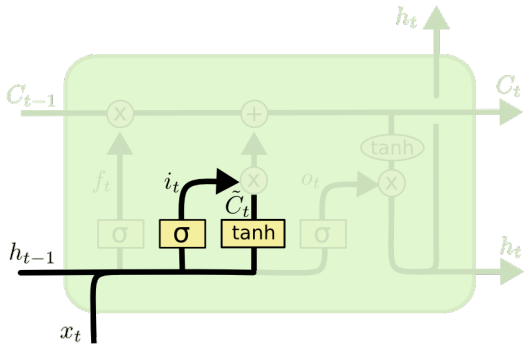

Figure 3.6: Input gate layer[13].

$$\mathbf{i}_t = \sigma(\mathbf{W}_i * [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \tag{3.10}$$

$$\tilde{\mathbf{C}}_t = tanh(\mathbf{W}_C * [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C) \tag{3.11}$$

Now LSTMs will forget things that it decided in the first step and add $i_t * C_t$, which add new candidate values multiplied by importance for each of them.

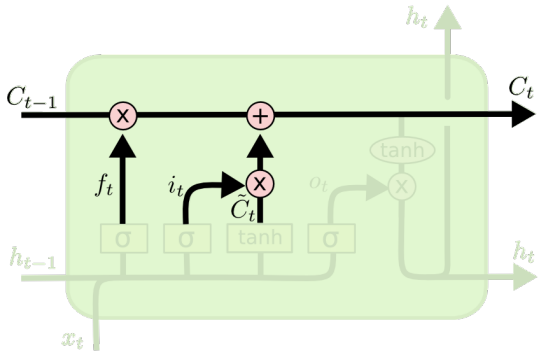

Figure 3.7: Third step that applies changes of previous steps to the cell state[13].

$$\mathbf{C}_t = \mathbf{f}_t * \mathbf{C}_{t-1} + \mathbf{i}_t * \tilde{\mathbf{C}}_t \tag{3.12}$$

In the fourth step. LSTMs decide what will be generated on output. The output will be based on the current cell state and will be also influenced by information from hidden and input state vectors. The sigmoid layer is used to decide what parts of the cell state going to output. In the end, LSTMs put the cell state through hyperbolic tangent, which normalizes cell state values into the interval between -1 and 1 finally multiply these values by the output of the sigmoid layer.

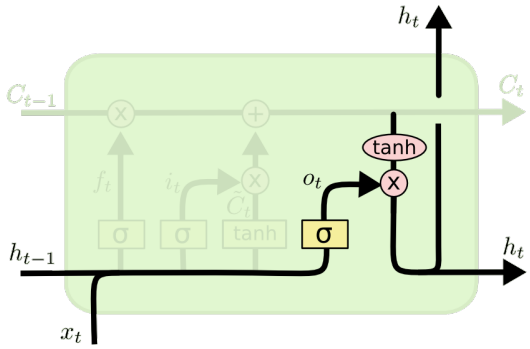

Figure 3.8: Fourth step, that forms a final output[13].

$$\mathbf{o}_t = \sigma(\mathbf{W}_o * [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \tag{3.13}$$

$$\mathbf{h}_t = \mathbf{o}_t * tanh(\mathbf{C}_t) \tag{3.14}$$

3.3.1 Bidirectional Long Short Term Memory networks

Bidirectional Long short term memory networks (BLSTMs) [20] are based on LSTMs, that besides the forward layer of LSTM cells, also have a backward layer of these cells. This means that there are two different independent LSTMs networks, where one of them provide cell state and hidden state in the forward direction in time and the second one in backward. Decisions from these two networks are concatenated.

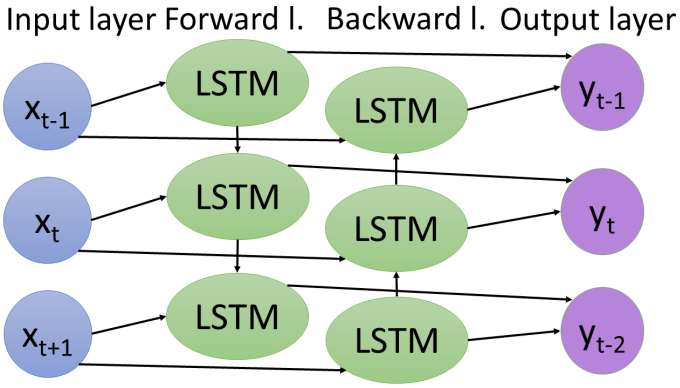

Figure 3.9: Example of BLSTM.

Chapter 4

Speech separation using neural networks

Speech separation using neural networks is commonly used these days and many types of the neural network architecture are invented to solve the separation problem. These neural networks commonly use BLSTM blocks as is described in subsection 3.3.1, which provide better training by learning from a distant past, in the other words using context. There are two commonly used ways of training these systems, permutation invariant training section 4.1, and Deep clustering method section 4.2. One of the disadvantages of these methods is the need of original speaker signals to provide targets for training. This can be solved by estimating speakers signals from stereo mixtures using spatial cues clustering section 2.2.

4.1 Permutation invariant training

Permutation invariant training (PIT) [21] is supervised method of NN training. It is commonly used in speech separation systems. In PIT there are two linear output layers each of the output size of source mixture at the end of NN. With these layers the NN estimates separated signals of speaker A and speaker B, from mixture given on the input. But NN does not know in which order speakers should appear on the output layers. So it can not be assumed that the speaker A will be always separated to the first linear layer and the speaker B to the second one. If the NN will train with this assumption, objective function will give as an result bigger loss values then expected. Otherwise when the NN assumes that speaker A is in the second layer and speaker B in the first one, objective function will give bigger loss values in different cases, but the result will be the same. NN will not be properly trained and it will not learn, where objective function will give her big loss value on right separated mixtures.

This problem has to be solved by computing objective function results for both permutation of speakers in training procedure. This means to compute objective function in the permutation, where the first linear layer output will be compared with target of speaker A and the second linear layer output with speaker B. Then system has to compute objective functions on the first linear layer output compared with the speaker B and the second linear layer output compared with the speaker A. Loss values are computed in both permutation has to be compared and lower one is the right one. So the lower value is sent to NN and it is used in backpropagation algorithm to train the NN. It is also necessary to count on this

behavior in the testing phase, where it is also necessary to count with this problem. SDR values between outputs of the linear layers of the NN has to be computed also with both permutation of target speaker signals. The higher values has to be used as the result.

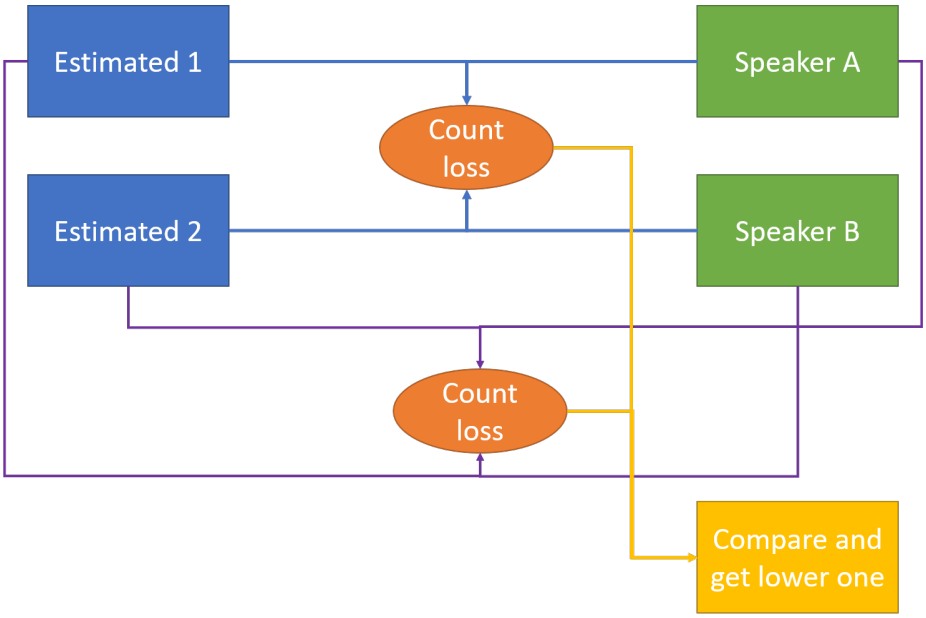

Figure 4.1: Permutation invariant training procedure.

4.2 Deep clustering

In Deep clustering (DC) [5] method embedding vectors for each time-frequency bin are estimated. Embedding is the mapping of time-frequency bin discrete value to the vector of characteristics. These characteristics are specific for each speaker in the mixture, so similar embedding vectors belong to the same speaker. The loss function is computed using two one-hot dimension matrices, for each speakers. One-hot dimension for the speaker contains zeros for bins, which do not belong to this speaker and ones for that which do. These matrices are estimated as an oracle mask from known target speakers and thresholded, to avoid computing loss on low energy bins. Loss is computed as:

$$loss = ||\hat{\mathbf{A}} - \mathbf{A}||_F^2 \tag{4.1}$$

where $\hat{\mathbf{A}} = \mathbf{Y}\mathbf{Y}_T$ which is an ideal afinity matrix and $\mathbf{A} = \mathbf{V}\mathbf{V}_T$ is an estimated affinity matrix. The neural network is led to learn to estimate the characteristics values of each time-frequency bin to be very similar in bins that belong to the same speaker. If the bin does not belong to the same speaker NN is enforced to give values that are further apart than the similar ones as it is showed on Figure 4.2, where the first matrix is three dimensional matrix of spectrogram with deep clustering embedding. The second one is reshaped first matrix to two dimensional space by move frequency and time shape to the same dimension. Last matrix in figure showed the example of values that the neural network trains to estimate for separating two speakers.

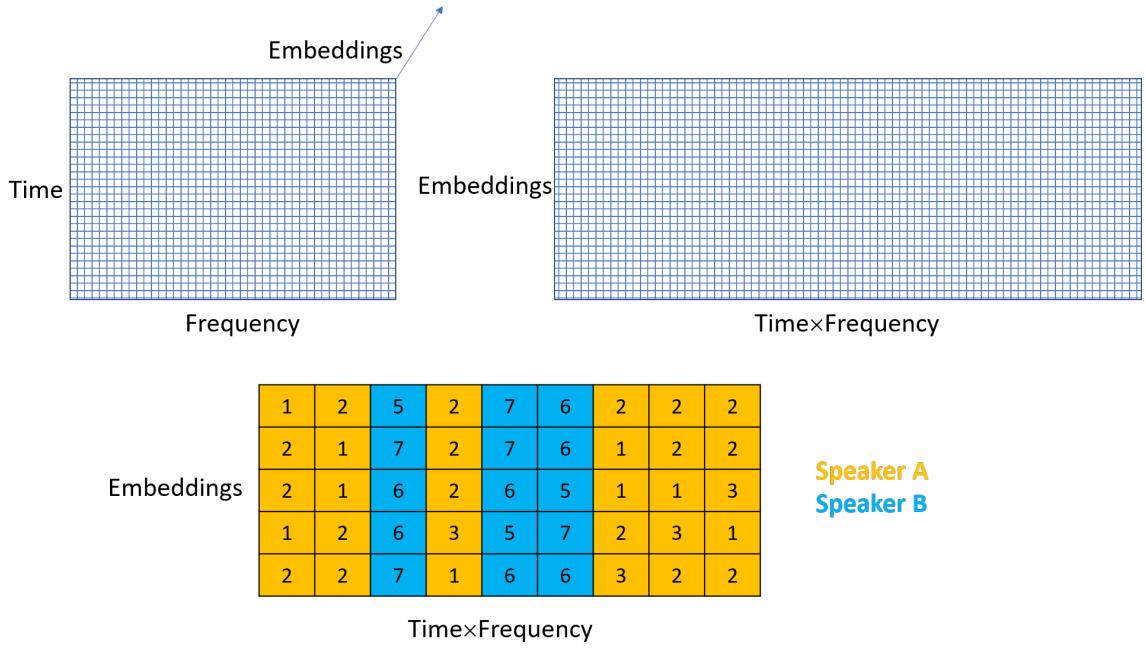

Figure 4.2: Deep clustering matrices

To estimate target masks to predict separated speakers, it is necessary to use some clustering algorithm such as K-Means or GMM. The output from the neural network is sent to one of these clustering algorithms, which try to fit on the given embedding vectors and cluster them into two components as it shown on Figure 4.3. Then the same output of NN is given to this algorithm to predict which bin belongs to which speakers. It is better to apply a threshold mask to NN output, to avoid the low energy bins to be clustered, because if they will, the result of the whole system can be much worse then estimated. Then predicted masks can be applied to the original mixture and SDR can be computed between the estimated result and real target.

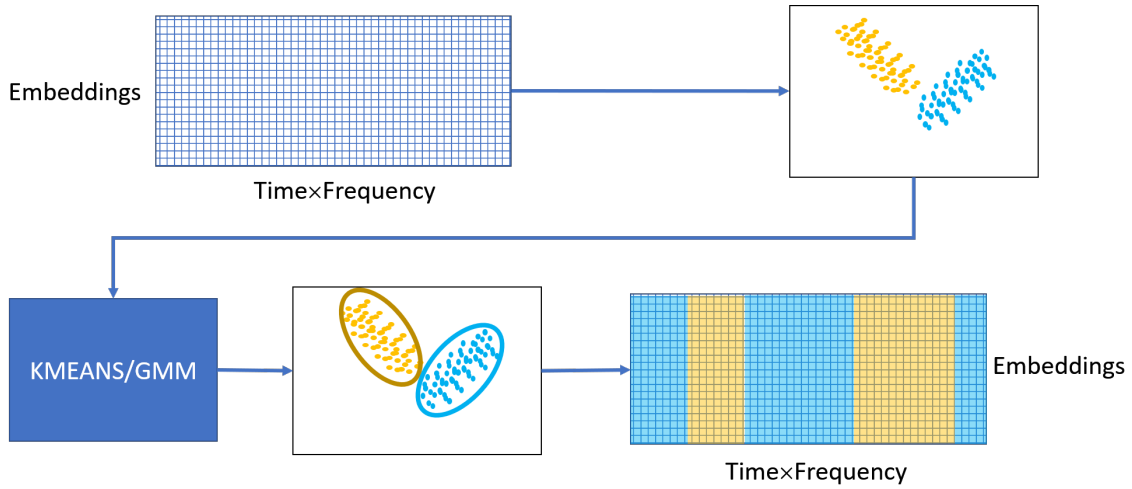

Figure 4.3: Deep clustering signal estimating procedure.

4.3 Bootstrapping speech separation with spatial cues

If the target signals are not aviable for training, it is possible to estimate them by using spatial cues algorithms described in section 2.2. To obtain target masks for speech separation there are two randomly selected channels from the first 4 of 8 mixture channels used. In the first method, there are computed phase differences from these two channels, then PCA is executed and in the end GMM is trained and masks are predicted. In this method permutation alignment algorithm is not pronounced, and it supplied by dividing IPD by a vector of linear spaced numbers in the range from zero to sample rate of the given mixture. These method does not work really good, so the second method was used.

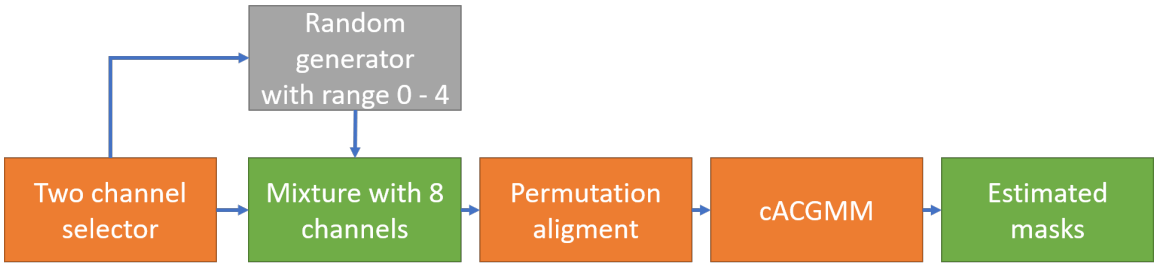

Figure 4.4: Procedure of estimating masks from mixture.

In the second method two randomly selected channels are also used. On these channels the permutation alignment algorithm is executed. Then the cACG mixture model is trained and masks are predicted on it as shown on Figure 4.5, where oracle masks derivated from known targets are also shown for comparison. There is no big difference between oracle and spatial cues predicted masks. The quality of obtained target masks was also tested on the cross-validation set by direct application masks on mixtures.

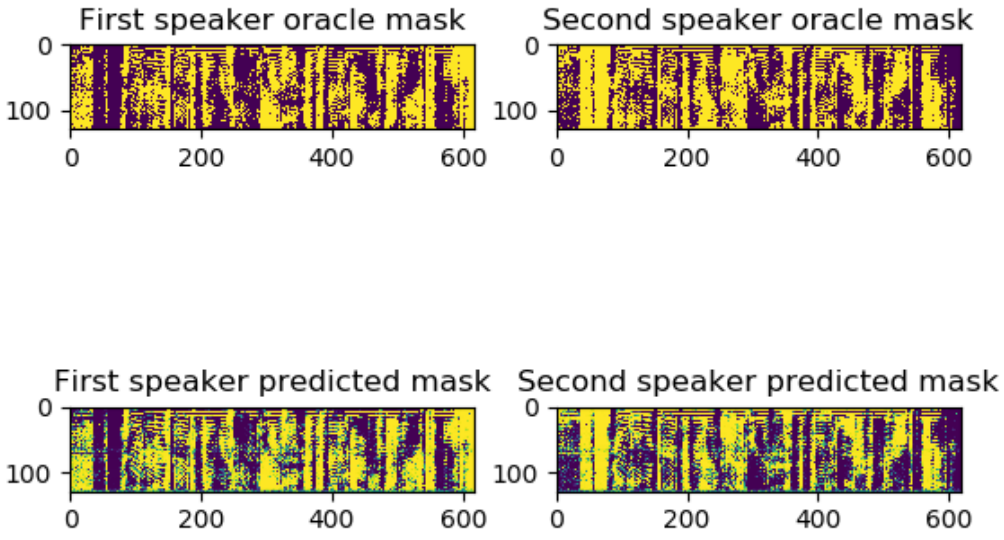

Figure 4.5: Estimated masks by spatial cues clustering by second method.

Objective functions are also affected by using spatial cues. So for PIT loss function using spatial cues means multiplying original mixture by target masks from spatial cues, defined as:

$$\mathbf{IS}^s = \mathbf{E}^s * \mathbf{M} \tag{4.2}$$

where $\mathbf{IS}^s$ is the ideal target signal, $\mathbf{E}^s$ is the estimated mask by spatial cues, $\mathbf{M}$ is the mixture and s is the speaker index. Then the NN outputs are applied on the original mixture:

$$\mathbf{ES}^s = \mathbf{O}^s * \mathbf{M} \tag{4.3}$$

where $\mathbf{ES}^s$ is the estimated signal, $\mathbf{O}^s$ is the masks estimated by the NN, $\mathbf{M}$ is the mixture and s is the speaker index. These results are used as the target original signals to count mean square error between them.

$$loss = lossPIT(\mathbf{ES}^s, \mathbf{IS}^s) \tag{4.4}$$

For the deep clustering method it means to use spatial cues estimated target masks as one hot dimension matrices, this can be obtained by comparing masks to each other.

$$\mathbf{IS}^0 = \mathbf{E}^0 > \mathbf{E}^1 \tag{4.5}$$

$$\mathbf{IS}^1 = \mathbf{E}^1 > \mathbf{E}^0 \tag{4.6}$$

Chapter 5

Draft and implementation

The whole system is designed as a structure of different objects, that are interconnected. There are two systems used in this work. First shown on Figure 5.1 is used for training with known target signals. There are three data loaders used in this system for getting the mixture and target signals for the speaker A and the speaker B from dataset. The second system presented on Figure 5.2 is used for training with estimated target masks from spatial cues algorithm. These masks are estimated by `preparer` method in spatial cues data loader, which is used for both speakers together. Each object of whole system provides one part of the functionality, that will be described in sections below. This chapter also contains information about the implementation of system parts and which Python packages are used.

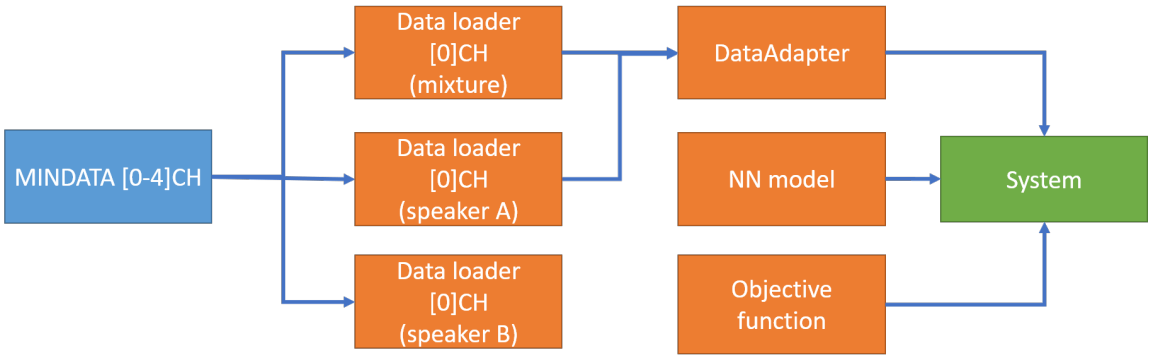

Figure 5.1: System structure for training using known target signals.

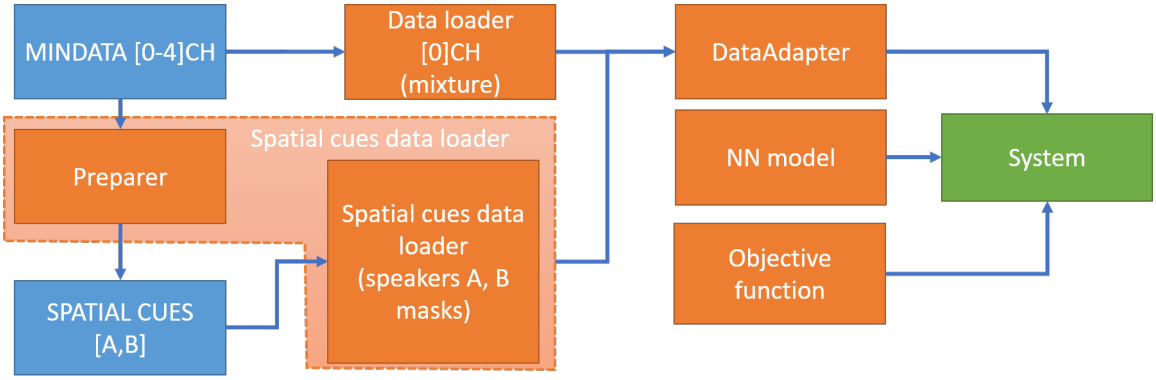

Figure 5.2: System structure for training using spatial cues.

5.1 Used packages

In thesis following packages were used:

- *Soundfile*[1] package is used for loading mixtures from sound files in dataset folders and for writing outputs of testing procedure,
- *Scipy*[2] package is used for counting short-time Fourier transform,
- *Numpy*[3] package is used for math,
- *PyTorch*[14] module is a Python based framework for programming neural networks, that contains implementation of linear, LSTM, BLSTM, convolutional layers, back-propagation algorithms and loss functions,
- *Mir_eval*[15] package is used for computing SDR score for predicted signals of separated speakers,
- *Tensorboard*[4] module, which is part of *Tensorflow* package [1], is used for watching state of the NN during training procedure, and
- *Matplotlib*[5] for visualizing spectrograms, masks etc.

5.2 Data loader

The data loader is the base module of the whole system. This module provides loading files from the dataset which is stored in the folder structure as it is showed on Figure 6.1. One instance of a data loader provides loading files from one data set's folder. Pytorch's data loader provides packing these files to batches. This module needs to get files for mixture and targets together, in this work there is a data adapter object used to group data loaders for mixtures and targets together. This module also counts mean and standard deviation for providing normalizing of mixtures. Normalization can be turned off by parameter on

[1] https://github.com/bastibe/SoundFile
[2] https://www.scipy.org/
[3] https://numpy.org/
[4] https://www.tensorflow.org/tensorboard
[5] https://matplotlib.org/

initialization. Batches are created by selecting random indexes from the data adapter, which request them from data loaders. Files in the batch have to be the same shape. So it is necessary to cut original files to the specified length. This is provided by algorithm 1 showed below. This algorithm makes zeros padding to files shorter than the specified length and longer files are cut to a segments of the same size. It is also possible to specify length parameter value to -1, which forces data loader to use uncutted files, when requested. This is used in a testing phase and for generating target masks using spatial cues.

Algorithm 1: Data preparing algorithm

```
N = loadedDataLength, idx = 0, l = givenLength, fIdx = currentFileIndex;
// while segment size is less then the rest of the data size
while idx+l < N-idx do
    // append start index of next segment to the cutteList
    cuttedList.append(idx);
    idx += l;
end
// if next segment overflows size of processed file
if idx - (( idx + length ) - ( N - idx )) >= 0 then
    // move idx back to align with the file size
    // and append it to the cuttedList
    cuttedList.append(idx-((idx+length)-(N-idx));
else
    // else append idx directly to the cuttedList
    cuttedList.append(idx);
end
```

This algorithm is repeated to each file found in a given folder in alphabetic order and uses *cuttedList* as a list of prepared indexes for all of them. Prepared indexes are saved to file to avoid recounting them every time the system starts. If the system asks the data loader for data on the specified index, module loads the required file and gives the system part of data in a specified length. Files, that are shorter than the specified length, are padded by zeros to the length. In the case of the multichannel dataset it is necessary to specify the channel number that we want to obtain by data loader.

5.2.1 Spatial cues data loader

Spatial cues data loader prepares target masks from the dataset using spatial cues blind separation method described in section 4.3. This module uses data loaders to get data from different channels of the mixture to provide mask preparation. The preparation method firstly creates two directories, one for generated masks and second for visualization of them. Masks are saved to files in batches of 10 sets of two masks for speaker A and speaker B, then masks are given to the system when are requested.

5.3 System, module, statistics and objective function

Let us describe objects `system`, `module`, `statistic` and objective functions showed on Figure 5.1 and Figure 5.2.

- `Module` is object that is inherited from Pytorch's `nn.Module` and contains definition of the specific NN. The NN is defined by Pytorch's layer objects from `nn` package,

that contains for example`nn.Linear`, `nn.LSTM` etc. Layers defined by these Pytorch's classes are connected together in `forward` function, which define how the input given to the NN will be processed. Module object also contains `apply` function for applying output of neural network to the mixture, if it necessary f.e. if neural network predicts masks that has to be applied to estimate target signals,

- `statistics` object is used by system to keeps statistics information from each epoch in training and writes them out to Tensorboard module. It is possible to define statistics array by name in this object. Append new statistic value to the end of array specified by name, and write that array out to the standard output or to Tensorboard summary writer,

- `objective` functions are used for training, it is represented as loss function and it is given by parameter to the system on initialization. In this module PIT and DC objective function are defined for training the multichannel data with known target signals. There are also defined the same two functions, modified for using with the multichannel data with estimated target masks and

- `system` is object that connects all the previously mentioned objects together. It contains training procedure and function for predicting results of the trained NN. System can save the NN trained state into the specified path and also load them back, to provide testing.

5.4 Training procedure

Training procedure, as showed on Figure 5.3, calls two methods `training`, where NN is trained and `cross validating`, where the neural network is tested to see whether it generalizes. NN is trained in epochs over batches, where batches are packs of randomly selected mixtures and target masks from dataset folders section 5.2. Before resolving, the short Fourier transform (STFT) is counted on a given mixture. STFT of the mixture is then normalized by mean and standard deviation counted on the whole dataset. Then loss function is computed between results of NN applied on denormalized STFT of the mixture and estimated targets by target masks. For permutation invariant training in this work there is a mean square error loss function used [14]. For the deep clustering training method there is a loss function from Onssen library used [12]. The average of results of loss function computed on the whole batch is used in the backpropagation algorithm section 3.1. The second step, cross-validation, is similar to the first step, but it does not call the backpropagation algorithm after computing loss function. Cross-validation is also executed on different data than the training step. After these two steps system saves the currently trained model and continues with the next epoch.

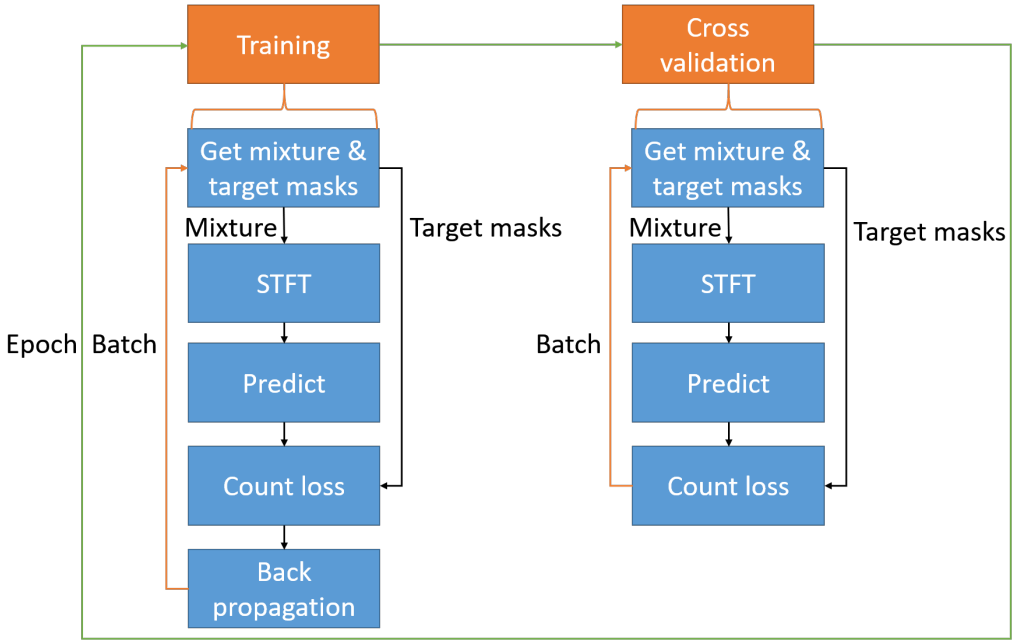

Figure 5.3: Training procedure.

5.5 Testing procedure

Testing procedure tests already trained system. As it is shown on Figure 5.4 procedure loads uncutted mixtures and original targets from part of dataset designed for testing. Firstly trained NN is loaded from the specified path and in the specified epoch. Then for each mixture in testing dataset STFT is computed and normalized. The result of normalization is sent to the system and processed. The system processes the given STFT of the mixture by sending to NN and then applying the output of NN to denormalized STFT of the given mixture. That gives two STFTs for each speaker, this is transformed back to normal signal using the mixture phase. Estimated speakers are then evaluated in mir_eval bss_eval_sources [15] function that counts SDR value between both estimated speakers and original target speakers. These values are summarized and in the end the average is counted.

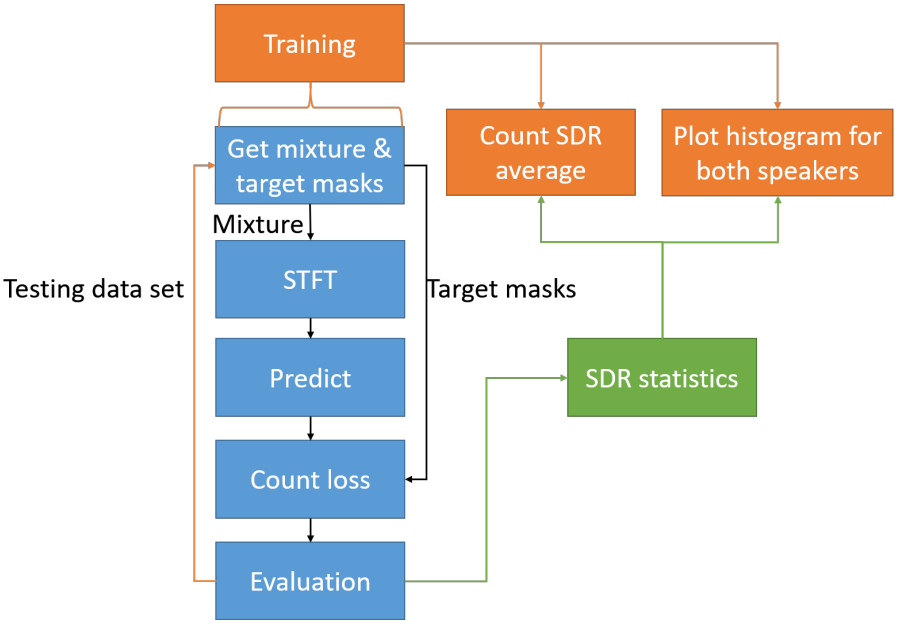

Figure 5.4: Testing procedure.

Chapter 6

Experiments

This chapter contains results of executed experiments. Describes properties of dataset used in this work. Also NN architecture is described in this chapter and other parameters used by system when executing different types of experiments.

6.1 Dataset

Dataset used in this work is Wall Street Journal mix dataset [4] in the spatialized version with two speakers, that is publicly available. On Figure 6.1 directory structure of the dataset is showed. 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. All mixtures consist of 8-channels two speakers mixtures as is showed on Figure 6.2, each channel has a different phase and magnitudes. Speakers in the testing dataset are different from these in training and cross-validation datasets. [17].

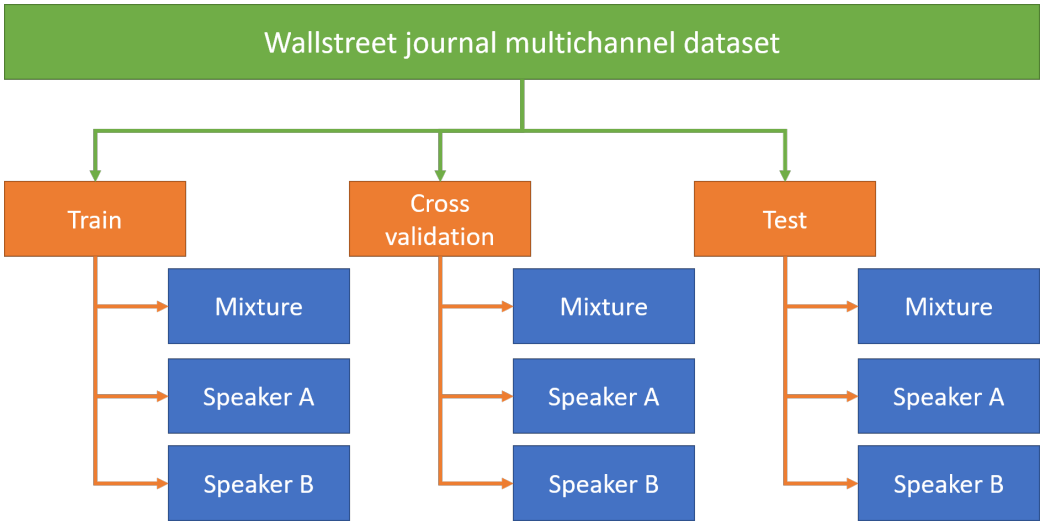

Figure 6.1: Dataset directory structure.

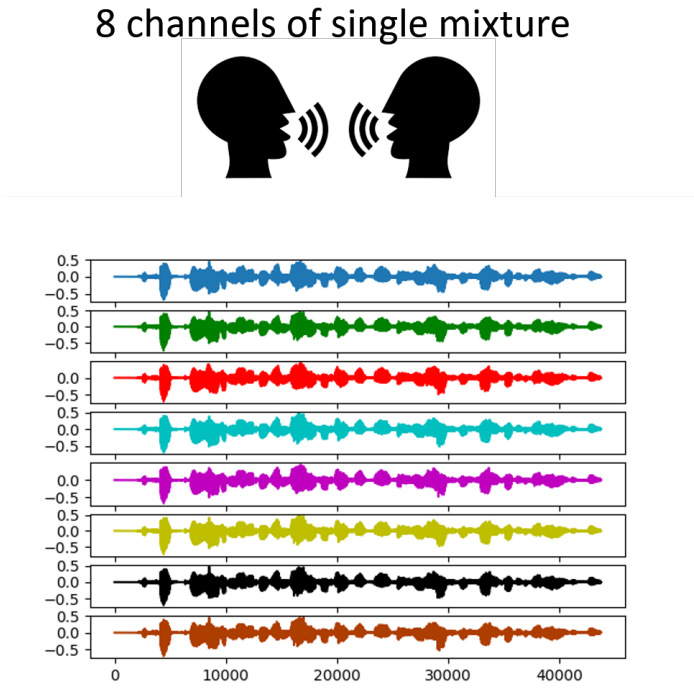

Figure 6.2: Single mixture with 8 channels from Wallstreet journal multi channel dataset.

6.2 Neural network architecture

The neural network that is used by the system in this work for permutation invariant training consists of four BLSTM blocks described in subsection 3.3.1 as it shown on Figure 6.3. The input dimension size of BLSTM is the frequency shape of STFT computed on the mixture, it is 129. The output dimension of BLSTM blocks is twice as long as the specified hidden dimension, which is 300, so the output of BLSTM blocks is the size of 600. This output is then connected to two linear layers, each of output dimension size of 129. These linear layers give target masks for both speakers. These masks are then applied to denormalized STFT of the original mixture to obtain the wanted result. For the deep clustering method very similar neural network architecture is used. But instead of two linear layers at the end, there is only one linear layer with an input size of 600 and output size of STFT frequency shape multiplied by embedding size, which is 30, so $129 * 30 = 3870$.

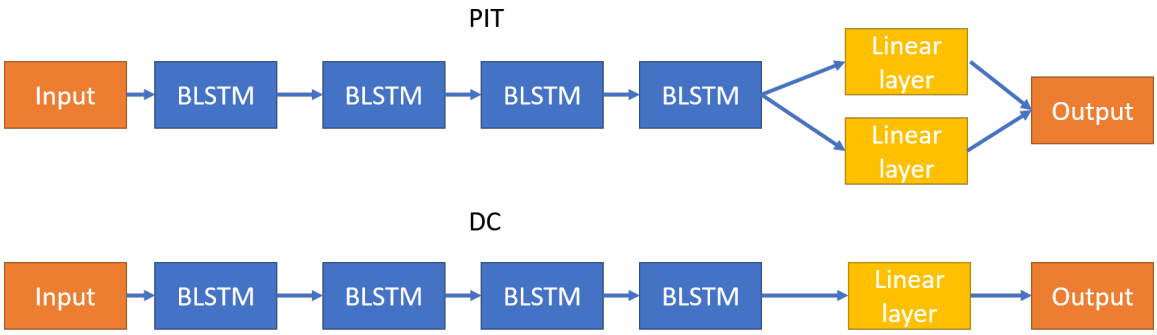

Figure 6.3: The NN architecture for training with PIT and DC method.

6.3 Parameters settings

STFT on data from the dataset is computed with parameters 256 samples in the segment and with 192 shift. Data are cut in data loader for $4000ms$ in PIT and DC systems that use known target speakers. For the PIT and the DC system that uses obtained target masks, data are not cut in the data loader, but STFTs computed on full-length data and obtained target masks are cut in data adapter to the size of 501 on the time shape. Two NNs used in experiments are specified in section 6.2.

6.4 Experiments and results

Experiment is ran for estimated target masks to obtain a SDR value result, that measures the estimation quality of the spatial cues algorithm. Several experiments are executed on a different kind of implemented system, to get reference SDR result values, that can be compared with and experimental systems that use target masks obtained by spatial cues clustering algorithm. All systems are trained and tested on the dataset described in section 6.1.

6.4.1 Target masks estimating

Let us look at the estimation of target masks for speakers using spatial cues information. Estimating algorithm for these masks were implemented as it is described in the article *Bootstrapping single-channel source separation via unsupervised spatial clustering on stereo mixtures* [18]. But after some testing it was obvious that estimated masks by this algorithm are not of good quality. So the implemented algorithm was investigated and it was found that there could be a problem with missing PA section 4.3. It was founded how the author of the article solves the problem in his code, and this solution was applied to the implementation. Unfortunately even then estimated masks by this algorithm give bad results on testing by applying them directly to the mixtures from which they were estimated.

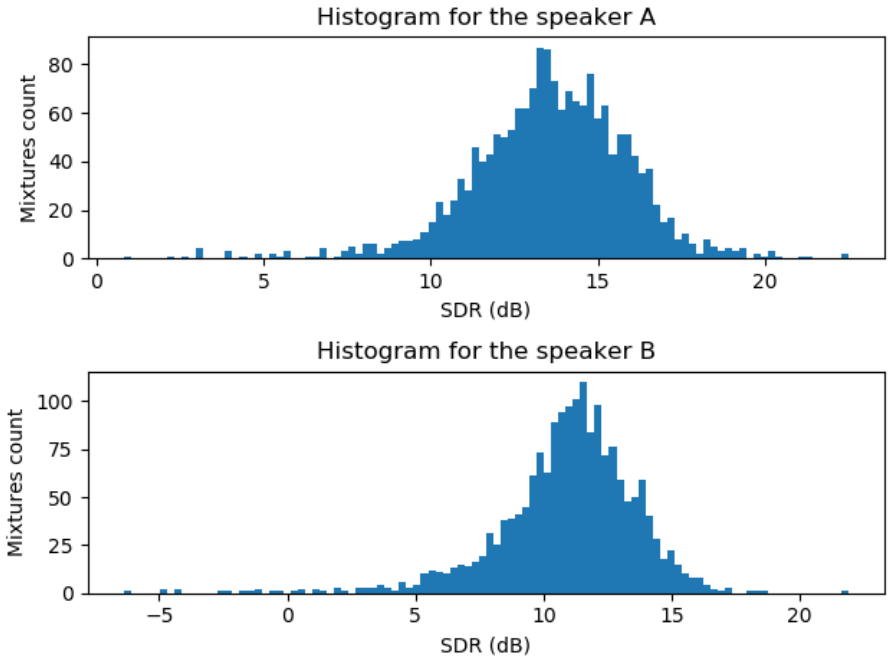

Figure 6.4: Histogram of estimated masks test results, where the speaker A is always dominant.

This problem was solved by using of another algorithm presented in *pb_bss* package [3]. In this package there was the cACGMM model used for masks estimation, and also there was the PA algorithm presented. Results given by tesing of this algorithm were $12.55dB$ SDR. This value is average of all 3000 tested masks, which are presented on Figure 6.4 separately for the speaker A and for the speaker B. The example of estimated results is shown on Figure 6.5, where on the second line are tested estimated masks. On the third line there are estimated target signals using mentioned masks. Finally on the third line there are spectrograms of the known target signals, which are compared withe estimated masks. Result of this test shows that estimated masks are in a good quality, so this algorithm was finally used.

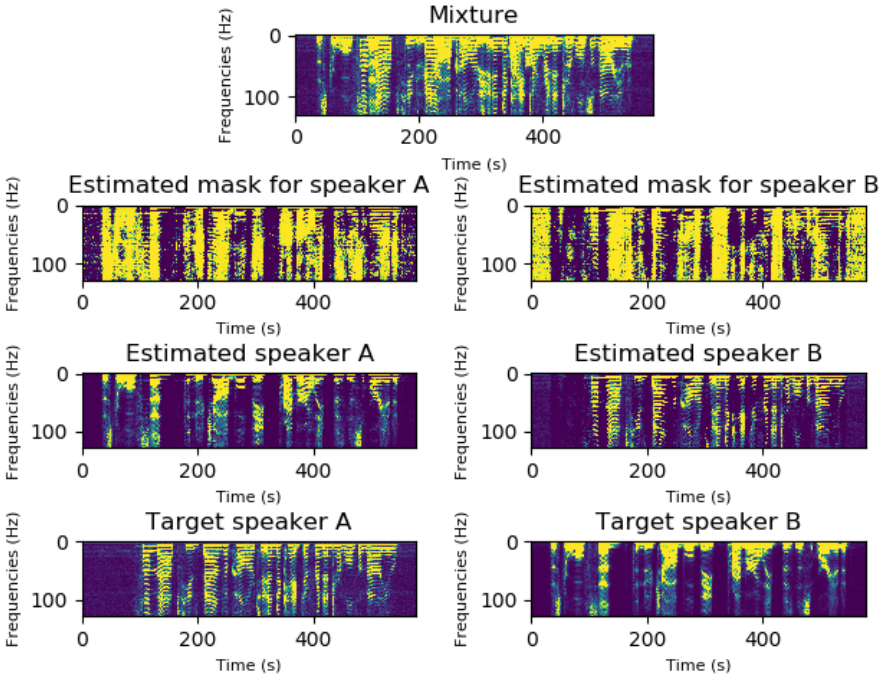

Figure 6.5: Example of target masks estimated by spatial cues algorithm.

6.4.2 Reference experiments

Reference experiments are provided on multichannel data with known target speakers for each artificial mixed mixture. Results of reference experiments are used for comparison with spatial cues experiments. There are spectrograms obtained by testing the PIT trained NN showed on Figure 6.6 and Figure 6.7. On these figures there is a tested mixture spectrogram showed on the first line. On the second line there are target masks estimated by the NN. The third line contains speaker signals obtained by applying estimated masks on the tested mixture. For reference, there are known target signals showed on the last line of the figure.

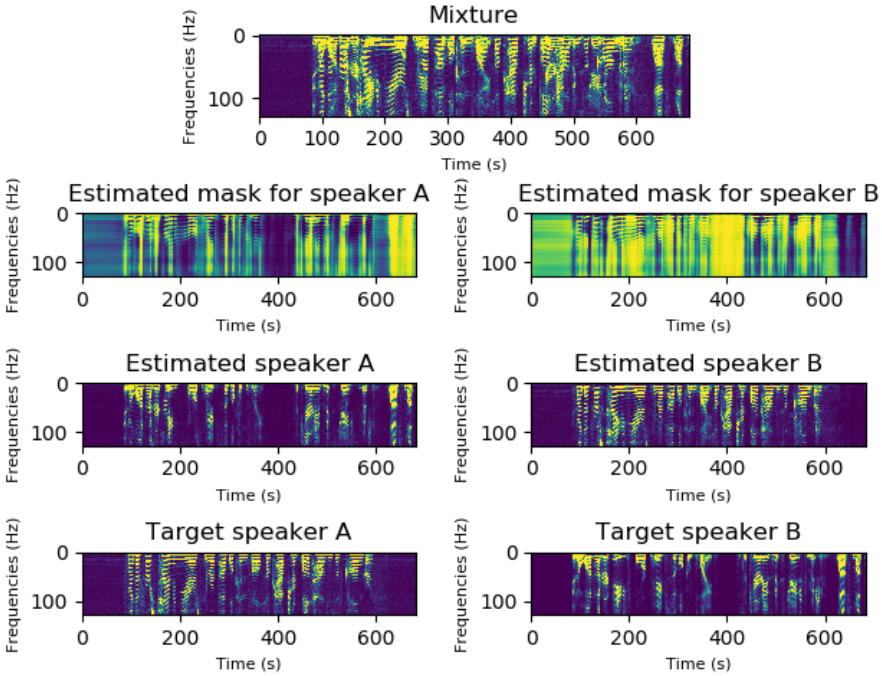

Figure 6.6: Example of estimated masks and target signals spectrograms from the PIT testing.

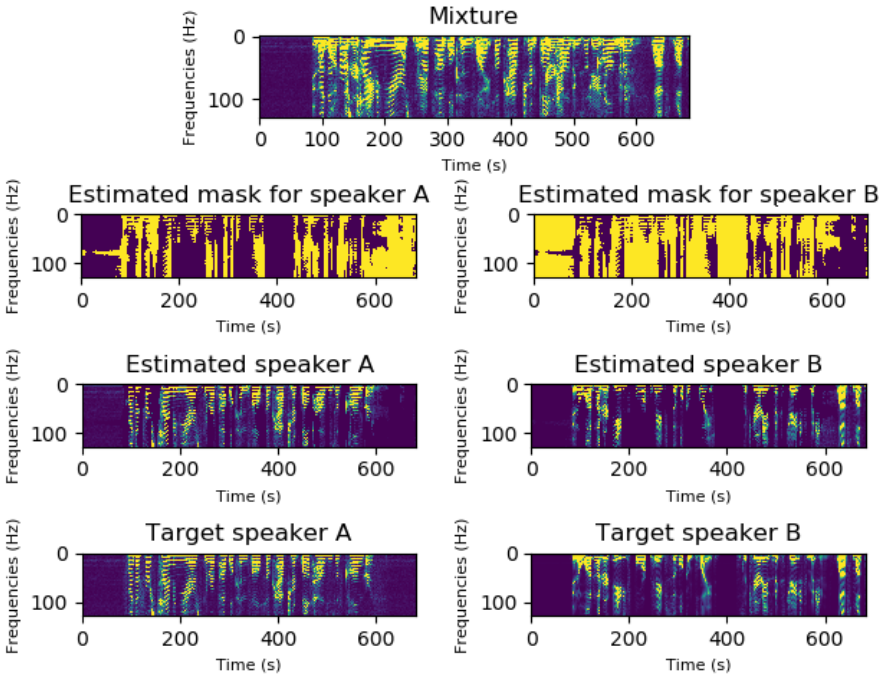

Figure 6.7: Example of estimated masks and target signals spectrograms from the DC testing.

The shape of the masks estimated by both methods is very similar, but it is possible to see differences in low energy time-frequency bins. The DC method may miss some bins by using zero or one masks (zero for the other speaker, one for the same speaker), but this behavior is balanced by false determined bins by the PIT method.

The results of testing showed on table Table 6.1, are average of SDR values of all 3000 tested masks, which are presented on Figure 6.8 for PIT based training and on Figure 6.9 for the training using DC method. On each figure are two separated histograms for the speaker A and for the speaker B. These results were obtained on epoch with the lowest cross-validation loss value.

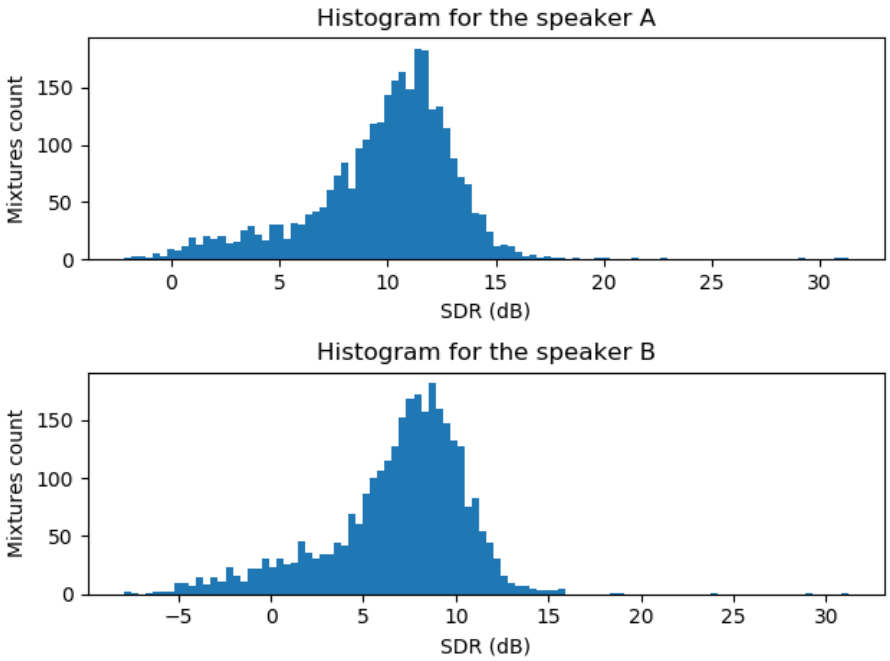

Figure 6.8: Histogram of PIT trained NN test results.

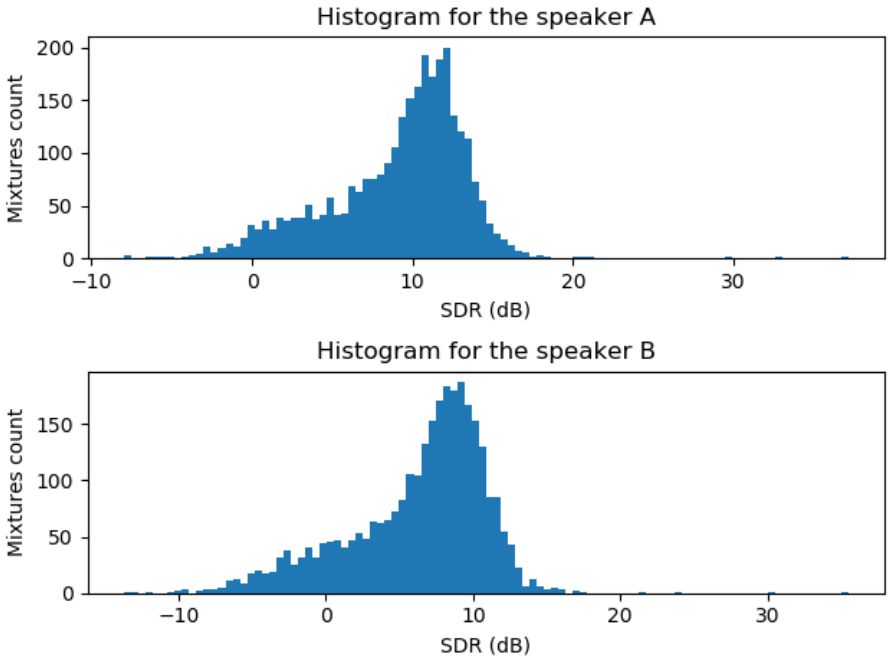

Figure 6.9: Histogram of DC trained NN test results.

Training type	SDR [dB]
PIT	8.18
DC	7.68

Table 6.1: Table of results of reference experiments.

The results of experiments are very similar, but they are lower than expected, so more training with different values of parameters was tested. Adjusted parameters were batch size, which was tried with a value of 20, 40, 100, also different learning where used with values 0.001, 0.0001. But none of these adjustments affect SDR results of the experiment. Also learning rate scheduler was used, which tries to lower the learning rate after every 100 epochs, but it was also not helpful. Finally batch of size 40, the learning rate of 0.001 was used without scheduling, which gives the best results.

6.4.3 Spatial cues experiments

Spatial cues experiments were made for the testing of how successful was the training with using target masks for speakers obtained by spatial cues blind separation algorithm described in section 2.2. In these experiments, the same system parameters as in reference experiments subsection 6.4.2 were used. Results of experiments were computed on epoch with the lowest cross-validation loss value pointed by the blue dot on Figure 6.10 for the training with the PIT method and Figure 6.11 for the training with the DC method. PIT cross-validation loss function continues to decline in 400 epochs, where the training procedure was stopped. On the contrary the cross-validation shape of DC training is stopped declining in epoch 108 and start to increase back slowly. This means that the NN starts overfitting, so the training process was terminated earlier.

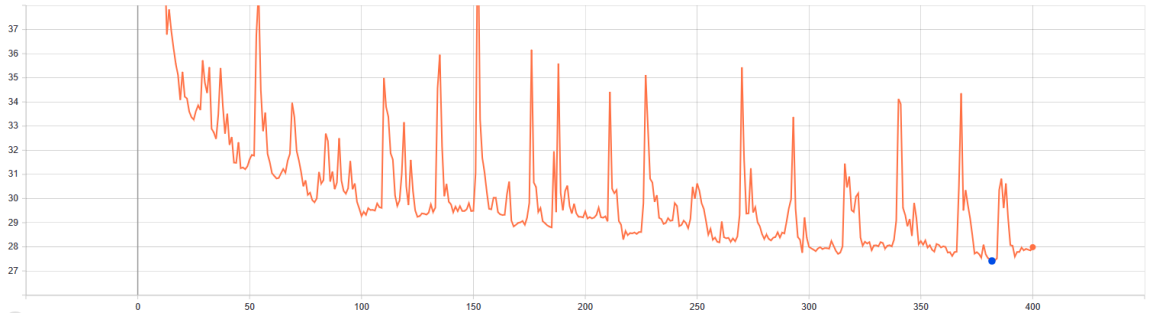

Figure 6.10: Graph of cross validation loss function results in epochs for multichannel data with using PIT trained on spatial cues obtained target masks.

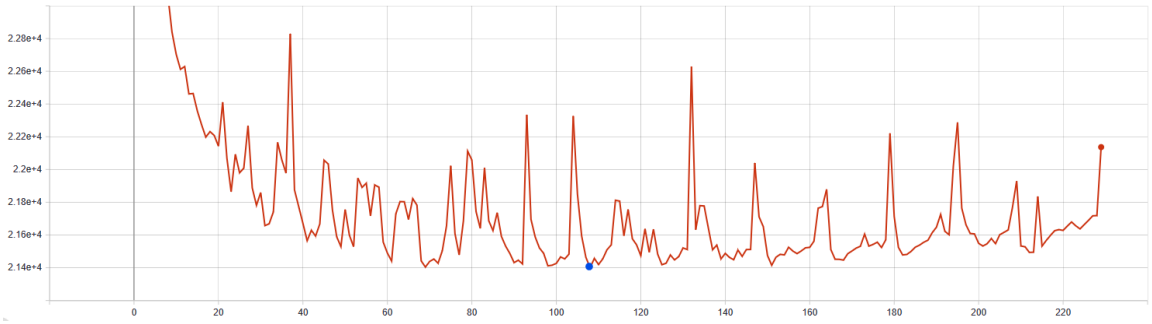

Figure 6.11: Graph of cross validation loss function results in epochs for multichannel data with using DC trained on spatial cues obtained target masks.

The Figure 6.12 and the Figure 6.13 shows the examples of estimated masks by the trained NN with spatial cues estimated target masks. The first figure shows estimated masks and targets for speakers from the NN trained with the PIT method. The second one shows the estimation of the trained NN with the DC method. In both figures there is a mixture spectrogram showed on the first line, then there are estimated target masks for speaker A and speaker B. Third and the fourth line contains estimated signals of speaker A and speaker B and original speakers signals for reference. Target masks estimated by the DC training method do not have soft values as masks estimated by the PIT training method. It is because embedding vectors in the DC are clustered by the KMeans. The process of obtaining masks from embedding vectors is described in section 4.2. But essentially the same pattern is visible on masks estimated by both methods.

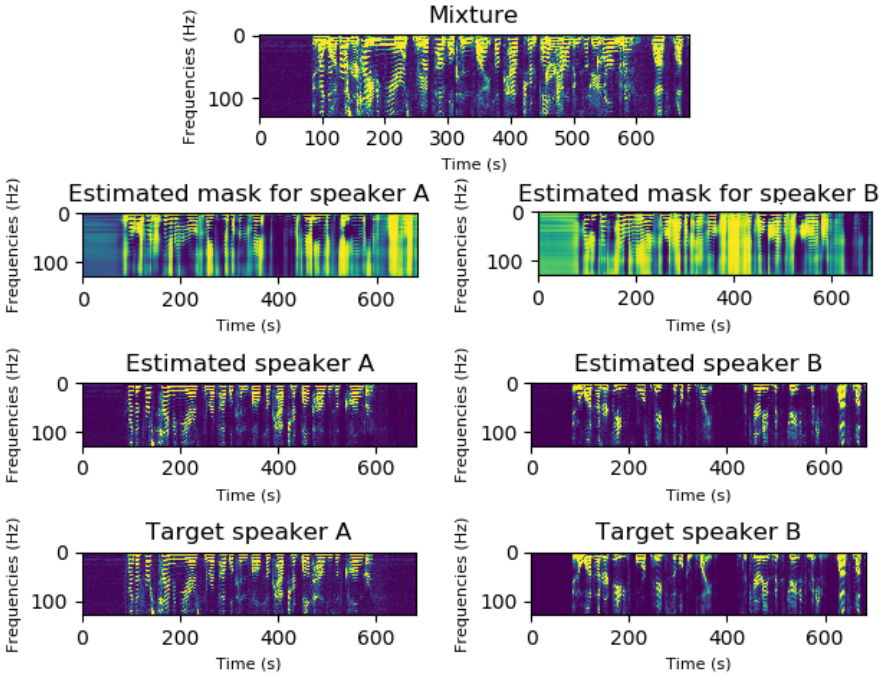

Figure 6.12: Example of estimated masks and target signals spectrograms from the SC PIT testing.

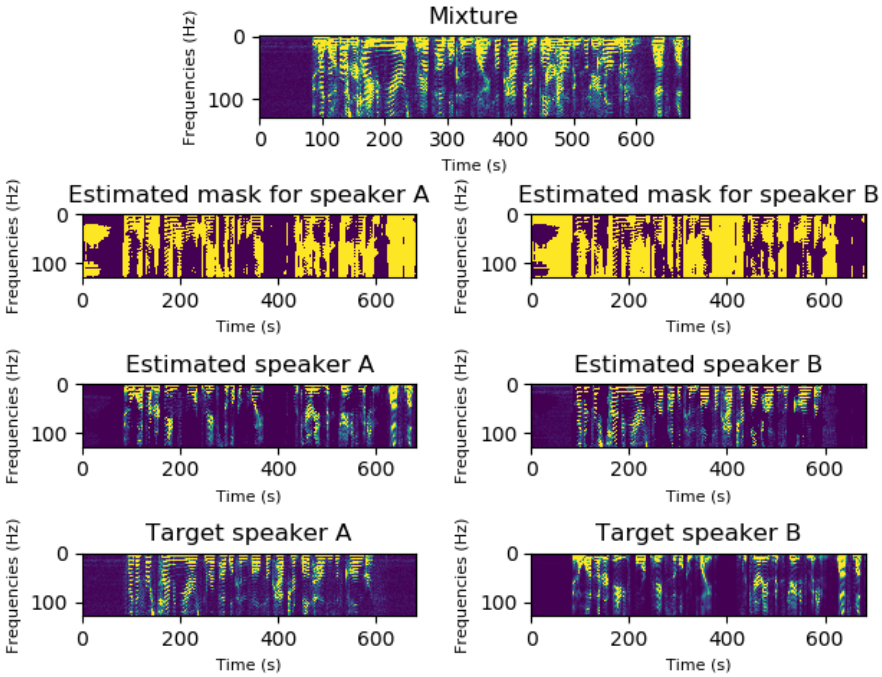

Figure 6.13: Example of estimated masks and target signals spectrograms from the SC DC testing.

The results of testing showed on table Table 6.1, are also average of SDR values of all 3000 tested masks. These results are also showed on histograms. On Figure 6.14 are histograms of results for the training with PIT and spatial cues estimated target mask. With the DC trained NN that also uses spatial cues target masks have results histograms showed on Figure 6.15.

The result values of these experiments, presented in Table 6.2 are slightly worse in comparison with the original article *Bootstrapping single-channel source separation via unsupervised spatial clustering on stereo mixtures* results. In the article there is only the DC method used for training of the NN, and it gives the SDR result of $9.2dB$. In this thesis it is only $8.08dB$ SDR obtained from the training that uses the DC method, but also the PIT method was used in this thesis for training, and it gives $9.31dB$ as an SDR result. This value is similar to value from the original article.

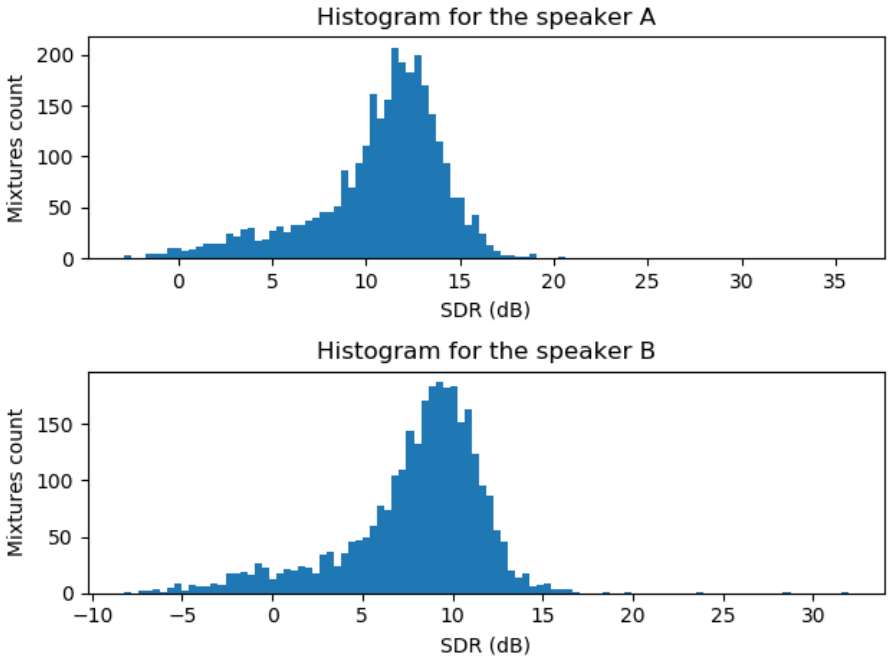

Figure 6.14: Histogram of PIT trained NN using spatial cues estimated masks test results.

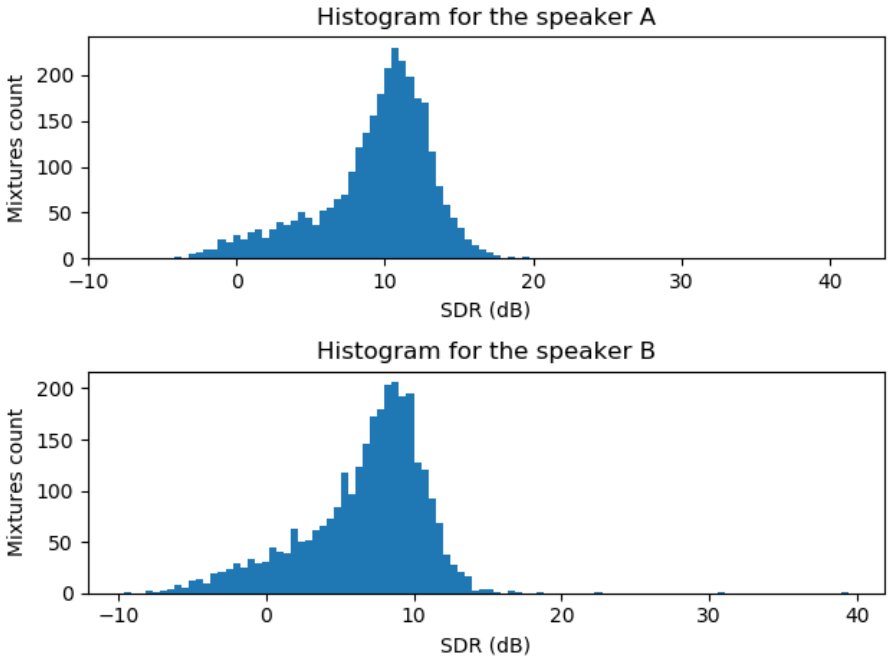

Figure 6.15: Histogram of DC trained NN using spatial cues estimated masks test results.

Training type	SDR [dB]
PIT	8.18
DC	7.68
SC PIT	9.31
SC DC	8.01

Table 6.2: Table of results of spatial cues (SC) and reference experiments.

From Table 6.2 it is also visible that results for training with known target signals give slightly worse results, than training that uses spatial cues estimated masks to obtain target signals. This could be caused by a system of cutting loaded mixtures and targets. With this zeros For non spatial cues using systems, mixtures are cut to 4000ms segments in data loader section 5.2. Some mixtures are padded with zeros and then STFT is counted on those fragments. Also loss function counts with these zero paddings, which could possibly causes low results of these experiments. On the contrary for spatial cues training, STFT is counted on the whole loaded mixture and then is cut or pad to the specified time shape size. This is caused by the need of complete mixtures for spatial cues estimating algorithm because it fails on a cut or padded mixtures. So an improvement in results could be achieved by better cutting and preparing input mixtures.

Chapter 7

Conclusion

The goal of the thesis was to implement the deep clustering training method and spatial cues algorithm for estimating target masks, mentioned in the article *Bootstrapping single-channel source separation via unsupervised spatial clustering on stereo mixtures* [8]. The deep clustering training method is used for training neural networks used for speech separation. The spatial cues estimation algorithm is used for estimating of target mask for speakers signals from mixtures where the original speaker's signals are not known e.g. the real-world mixtures.

The spatial cues estimation algorithm was implemented and tested with bad quality results. However another algorithm was used, described in section 2.2, that works much better. The estimated masks were used for the training of neural networks in experiments.

Experiments were provided by training the NN on the multi-channel wall street journal dataset, described in section 6.1. NN was trained using two different methods of learning, the PIT method and the DC method, both described in chapter 4.

Results of executed experiments were $8.01dB$ SDR for the NN trained with the DC method using spatial cues estimated target masks, which is a little bit worse than in the original article. But for the trained NN with the PIT method using spatial cues estimated target masks given result was $9.31dB$ SDR which is better than the article's result. The comparison of reference and spatial cues experiments showed that the results of spatial cues experiments were better than reference results. That was maybe caused by the different cutting of dataset files in the system. The reasons for the result of this comparison were more described in chapter 6.

As a way to improve the trained NN results, there is also *confidence measure* mentioned in the original article. This method measure how successful was the spatial cues algorithm in estimating target masks from the mixture. Based on this confidence measurement, poor quality estimated target masks can be separated from the training and the cross-validating dataset and do not be used in the training procedure. This could help the NN to be trained only on good quality estimated target masks, which could improve the quality of this trained NN.

However spatial cues method can be used for real-world mixtures, all mixtures used for the training and the testing in the thesis were artificial ones. It may be interesting to try to obtain real-world mixtures and train the NN on them. Also it will be very interesting to test trained systems from the thesis on real-world mixtures, to see how good they work for them.

Bibliography

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A. et al. Tensorflow: A system for large-scale machine learning. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, p. 265–283.

[2] Amidi, A. and Amidi, S. *Recurrent Neural Networks cheatsheet* [online]. Stanford, january 2019 [cit. 2019-12-20]. Available at: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

[3] Drude, L., Hasenklever, D. and Haeb Umbach, R. Unsupervised training of a deep clustering model for multichannel blind source separation. In: IEEE. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, p. 695–699.

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

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

[6] HKDH, B. Neural networks in materials science. *ISIJ international*. The Iron and Steel Institute of Japan. 1999, vol. 39, no. 10, p. 966–979.

[7] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J. et al. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. A field guide to dynamical recurrent neural networks. IEEE Press, 2001.

[8] Hochreiter, S. and Schmidhuber, J. LSTM can solve hard long time lag problems. In: *Advances in neural information processing systems*. 1997, p. 473–479.

[9] Ito, N., Araki, S. and Nakatani, T. Complex angular central Gaussian mixture model for directional statistics in mask-based microphone array signal processing. In: IEEE. *2016 24th European Signal Processing Conference (EUSIPCO)*. 2016, p. 1153–1157.

[10] Kriesel, D. *A Brief Introduction to Neural Networks*. 2007. Available at: availableathttp://www.dkriesel.com.

[11] Luo, Y. and Mesgarani, N. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing*. IEEE. 2019, vol. 27, no. 8, p. 1256–1266.

[12] Ni, Z. and Mandel, M. I. Onssen: an open-source speech separation and enhancement library. *ArXiv preprint arXiv:1911.00982*. 2019.

[13] Olah, C. Understanding lstm networks, 2015. *URL http://colah. github. io/posts/2015-08-Understanding-LSTMs*. 2015.

[14] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach, H., Larochelle, H., Beygelzimer, A., Álché Buc, F. d, Fox, E. et al., ed. *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, p. 8024–8035. Available at: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[15] Raffel, C., Mcfee, B., Humphrey, E. J., Salamon, J., Nieto, O. et al. mir_eval: a transparent implementation of common MIR metrics. In: *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. 2014.

[16] Sawada, H., Araki, S. and Makino, S. Measuring dependence of bin-wise separated signals for permutation alignment in frequency-domain BSS. In: IEEE. *2007 IEEE International Symposium on Circuits and Systems*. 2007, p. 3247–3250.

[17] Seetharaman, P., Wichern, G., Le Roux, J. and Pardo, B. Bootstrapping Single-channel Source Separation via Unsupervised Spatial Clustering on Stereo Mixtures. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, p. 356–360.

[18] Seetharaman, P., Wichern, G., Le Roux, J. and Pardo, B. Bootstrapping single-channel source separation via unsupervised spatial clustering on stereo mixtures. In: IEEE. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, p. 356–360.

[19] Venkatachalam, M. *Recurrent Neural Networks* [online]. Towards Data Science, march 2019 [cit. 2019-12-20]. Available at: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce.

[20] Yildirim, Ö. A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification. *Computers in biology and medicine*. Elsevier. 2018, vol. 96, p. 189–202.

[21] Yu, D., Kolbæk, M., Tan, Z.-H. and Jensen, J. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In: IEEE. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, p. 241–245.