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

ADVERSARIAL AUGMENTATION FOR ROBUST SPEECH SEPARATION

ADVERSARIÁLNÍ AUGMENTACE PRO ROBUSTNÍ SEPARACI ŘEČI

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. JÁN PAVLUS**
AUTOR PRÁCE

SUPERVISOR **Ing. KATEŘINA ŽMOLÍKOVÁ**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2021/2022

Master's Thesis Specification

Student: **Pavlus Ján, Bc.**
Programme: Information Technology and Artificial Intelligence
Specialization: Machine Learning
Title: **Adversarial Augmentation for Robust Speech Separation**
Category: Speech and Natural Language Processing
Assignment:

1. Get acquainted with the problem of speech separation using neural networks.
2. Get acquainted with generative adversarial networks.
3. Train baseline for speech separation on several datasets.
4. Implement and train learnable augmentation for robust speech separation.
5. Evaluate the method and compare with baseline results.
6. Discuss the results and suggest potential ways to improve them.

Recommended literature:

- Luo, Yi, 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 (2019): 1256-1266.
- Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems* 27 (2014).
- Qian, Yanmin, Hu Hu, and Tian Tan. "Data augmentation using generative adversarial networks for robust speech recognition." *Speech Communication* 114 (2019): 1-9.

Requirements for the semestral defence:

- 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, 2021
Submission deadline: May 18, 2022
Approval date: November 1, 2021

Abstract

Speech separation is the task of separating single signals from the given mixture of multiple speakers. Neural networks trained for speech separation usually work well on artificial data but they often fail on real-world examples. To improve their behavior on real-world mixtures it is possible to use training data augmentations such as noise addition. Nevertheless, the power of these augmentations is limited as they have to be manually designed.

In this work, the modified version of the generative adversarial networks (GAN) model could improve this process by generating augmentations depending on the separation performance on these data. Speech separation could be then made more robust with each generator and separator training step. This system was subjected to experimentation. During these experiments, the parameters have been tuned to find the best setting that will successfully train the GAN model without collapsing. This setting was found and the most robust model from the training was selected and evaluated. Results show that the separator model trained by the GAN model does not achieve any significant improvement from the original separator model pretrained on the WSJ0-2mix dataset during the testing on the WHAM dataset. Nevertheless, another evaluation shows that the separator model trained by the GAN model is significantly more robust than the original one towards the generated noises.

Abstrakt

Separace řečníků se zabývá separací signálů jednotlivých řečníků z dané směsi vícero řečníků. Neuronové sítě trénované pro separaci řečníků fungují většinou dobře na uměle smíchaných nahrávkách, ovšem při použití směsí z reálného světa často selhávají. Pro zlepšení tohoto chování, je možné použít augmentaci trénovacích dat, jako je například přidání šumu. Nicméně tyto augmentace jsou limitovány tím, že musí být ručně navrhnuty.

V této práci je použita modifikovaná verze modelu generativních adversarialních sítí (GAN), která může zlepšit tuto vlastnost tak, že generuje augmentace na základě míry zmatení separačního systému. Po každém kroku trénování generátoru a separátoru se systém separace řečníků stává více robustní. Takto navrhnutý model byl podroben experimentům. Během těchto experimentů byly různě nastavovány parametry GAN modelu, aby se nalezlo jejich nejlepší nastavení, které by vedlo ke správnému natrénování modelu, bez zkolabování do žádného módu. Během experimentů bylo takové nastavení nalezeno. Z takto natrénovaného modelu byl vybrán nejvíce robustní separátor a ten poté vyhodnocen. Výsledky hodnocení neukázaly zlepšení funkčnosti zrobustněného separačního systému vůči samému nezrobustněnému systému předtrénovanému na WSJ0-2mix datasetu, během testování na datasetu WHAM. Nicméně výsledky jiného hodnocení experimentů ukázaly, že separátor vybrán z trénování GAN modelu je značně zrobustněn oproti původnímu.

Keywords

speech separation, GAN, adversarial augmentations, robust neural network

Klíčová slova

separace řečníků, GAN, adversarialní augmentace, robustní neuronová síť

Reference

PAVLUS, Ján. *Adversarial Augmentation for Robust Speech Separation.* Brno, 2022. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Kateřina Žmolíková

Rozšířený abstrakt

Separace řečníků se zabývá separací signálů jednotlivých řečníků ze směsí obsahujících více hovořících řečníků. Takovéto systémy se mohou použít například pro zlepšení výsledků systémů pro rozpoznávání řeči, které často selhávají na nahrávkách obsahujících překrývající se promluvy vícero řečníků.

V dnešní době se pro systémy separace řečníků používají neuronové sítě. Pro jejich trénink jsou třeba směsice více řečníků, u kterých známe i signály jednotlivých řečníků. U směsi pořízené nahráním z reálného světa je těžké zjistit signály jednotlivých řečníků. Pokud by to bylo jednoduše možné, systémy separace řečníků by postrádali svůj smysl. Proto jsou používány nahrávky uměle smíchané ze signálů jednotlivých řečníků. Systémy trénované na těchto směsích fungují dobře na dalších uměle smíchaných směsích, ovšem často selhávají na nahrávkách z reálného světa. To je dáno tím, že umělé směsi neobsahují šumy, ozvěny a další vlastnosti reálného prostředí.

Pro zlepšení výsledků systémů separace řečníků na směsích z reálného světa je možné přidat do trénovacích dat různé vlastnosti reálných prostředí, tedy provést augmentaci dat. Sytémy, které jsou trénovány na augmentovaných datech jsou robustnější a dosahují lepších výsledků na datech z reálného světa. Existuje mnoho klasických praktik pro augmentování řečových signálů, jako jsou inverze signálu, přidání šumu, přídání ozvěn, změna tónu, změna tempa, atd. Klasické praktiky augmentace dat musí být manuálně navrhnuty a z tohoto hlediska jsou nevýhodné.

Tato práce se zabývá použitím modifikovaného modelu generativních adversariálních sítí (GAN) pro augmentaci dat. Výhoda GAN modelu je ve schopnosti generování nových augmentací, které zatím nemusely být manuálně navrhnuty. Model použitý v této práci se od původního liší diskriminátorem. V původním modelu je diskriminátorem neuronová síť, která udává, zda jsou generovaná data reálná nebo falešná. Diskriminátor použitého GAN modelu se skládá ze dvou částí: systému separace řečníků a funkce podobnosti. Generátor se při tréninku snaží vygenerovat takové augmentované směsi, které co nejvíce zmatou separátor a budou co nejvíce podobné původní směsi.

Experimenty s navrhnutým GAN modelem používají předtrénovaný systém separace řečeníků na původních datech a generátor předtrénovaný na úloze vlastní identity. Experimenty používají dataset WSJ0-2mix nebo jeho augmentovanou verzi WHAM. Během experimentů bylo nejdříve nalezeno správné nastavení všech parametrů. Nastavit parametry GAN modelu není lehké, model je velice náchylný na jakékoliv změny a rychle kolabuje do jednoho z módů špatného trénování, jako je například příliš silný generátor nebo separátor. Z experimentů bylo zjištěno, že parametry, které manuálně přepínájí trénink jsou prakticky nenastavitelné a místo nich je třeba použít parametrů dynamického přepínání. U tohoto druhu přepínání parametry určují cíl, který musí jednotlivé části během svého trénování dosáhnout, nežli je trénink přepnut. Nakonec bylo správné nastavení, při kterém GAN model nekolaboval do žádného z módů špatného trénování, nalezeno.

Z takto natrénovanáho GAN modelu je vybrán nejrobustnější separátor a ten poté evaluován. Evaluace prokázala, že zrobustněný separátor, který byl předtrénován na datasetu WSJ0-2mix neprokazuje zlepšení při evaluaci na testovací sadě datasetu WHAM. Na druhou stranu zrobustněný separátor je z výsledků evaluace viditelně robustnější vůči generovaným augmentovaným směsím než-li původní předtrénovaný separátor. Toto chování je dané tím, že generátor není schopný generovat tak sofistikované šumy jako ty, které jsou obsaženy v rámci datasetu WHAM. Separátor zrobustněný GAN modelem navrhnutým v této práci nezrobustní systém vůči všem šumům, ovšem může významně pomoci jako další systém pro augmentaci dat při zrobustňování systému separace řečníků.

Adversarial Augmentation for Robust Speech Separation

Declaration

I hereby declare that this Master'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 16, 2022

Acknowledgements

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

Contents

Chapter 1

Introduction

Speech separation is the task of separating the signals from the given mixed signal. One of the possible usage of these systems is pre-processing for speech recognition systems which often fail on more overlapped speech. In these cases, the speech separation system could improve the result of the recognition system by separating individual signals from the mixed speech. The speech separation systems are nowadays based on neural networks. To train such a neural network it is necessary to have a mixed signal with well-known original source signals. For real-world mixtures, single-speaker signals are usually unavailable, and thus it is necessary to use artificial mixtures. This leads to a problem with bad performance of speech separation systems on the real-world mixtures. This creates the need to make the speech separation systems more robust towards the real-world mixtures.

This is caused by the absence of the different noises, echoes and other features of real environments in the artificial mixtures. To improve the behavior of the speech separation systems in these situations it is possible to do data augmentation. Using the data augmentations could add these real-world features into the mixtures on which the system is trained on. There are several well-known methods how to do the data augmentation for the speech signal, such as the signal inversion, noise addition, etc. The disadvantage of these methods is that they do not cover all possible augmentations and each new augmentation needs to be manually designed.

Nevertheless, the generative adversarial networks [8] concept could be used also for the data augmentation as this concept was originally proposed for the data generation. Generative adversarial networks could be used to perform data augmentations for speech separation systems automatically. Their advantage is that they can generate augmentations depending on the response of the speech separation system.

In this work, a modified version of the generative adversarial networks is used. It consists of the generator network generating the augmentations, the separator network that should be trained to be more robust, and the similarity loss function that constraints the generator network. The separator network and the similarity loss function represent the discriminator role. For both networks, the ConvTasNet [18] architecture (with different parameters) has been used. The aim of this system is to generate some new augmentations on the given mixtures that have never been seen before and force the speech separation system to adapt to them.

In Chapter 2 the speech separation is described in detail together with the training of the neural networks. Chapter 3 describes the generative adversarial networks model, its training and problems occurring during the training. The usage of the generative adversarial networks for data augmentation is also explained there. Chapter 4 outlines how to make the

speech separation system more robust, i.e. the classic methods for data augmentation, and the modified version of the generative adversarial networks. Chapter 5 describes how the model is implemented in more detail. The last Chapter 6 shows the experiments proving the concept of the presented generative adversarial networks model to make the robust speech separation system.

Chapter 2

Speech separation using neural networks

The sound is represented by a vector of samples called signal. It is possible to hear multiple sounds in one moment. Such mixture can be modelled as:

$$y_t = \sum_{n=1}^{N} s_{t,n} \tag{2.1}$$

where y_t is the mixture to be separated, $s_{t,n}$ is the speech signal of a single speaker or noise, t is the time index, n is the source index, and N is the number of sources. The main task in speech separation is to reconstruct signals $s_{t,n}$ from the mixture y_t with no information about the signals $s_{t,n}$.

The speech separation task could be also explained as a Cocktail party problem. Imagine a cocktail party where a lot of people talk over each other. The listener present at the party is trying to focus on one specific speech. The human ear and brain are well adapted to solve this task, but for computer systems, it is very difficult. In the past, there were attempts to solve this task with classic methods such as principal component analysis [1] or independent component analysis [25]. These classic methods usually work well when the task is greatly simplified, but they fail when silent blocks, echoes, and delays are present.

2.1 Using neural networks

Nowadays neural networks are used for speech separation tasks. These methods usually work either with the signal in time-domain or its short time Fourier transform. The most common neural network architectures used for speech separation are convolutional neural networks in combination with recurrent blocks as for example long short term memory blocks (LSTM) [13], which work quite well.

Convolutional neural networks contain specific type of layers called convolutional layers. These layers are composed of convolutional filters with trainable parameters. Filters in convolutional layer are used to extract useful information from the given signal. For example the convolutional layer could be trained to perform pseudo short Fourier transform of the signal.

Recurrent neural networks contain loops that allow information to be stored within the network. Due to this property recurrent neural networks have the possibility to use context.

In other words, they are able to predict next step using the previous information. Recurrent neural networks are divided to two groups:

- short recurrent neural networks that contain only simple loops. They have possibility to store only little amount of information and use only a short context given by the stored information,
- long term recurrent neural networks that contain blocks that are able to store more information. This leads to the possibility of using longer context. They are two basic blocks commonly used: Gated Recurrent Units (GRU) and Long short time memory blocks (LSTM). The difference between them is that GRU contains two gates and LSTM contains four of them. Gates are used to determine which information the network should remember or forget.

Considering that the context information in speech processing is very important, using LSTM blocks in neural networks for speech separation could improve its performance.

2.2 ConvTasNet

In this work, the ConvTasNet [18] neural network architecture is used. This architecture is mostly used for speech separation. It consists from three parts as it is shown in Figure 2.1:

1. Encoder.
2. Separator.
3. Decoder.

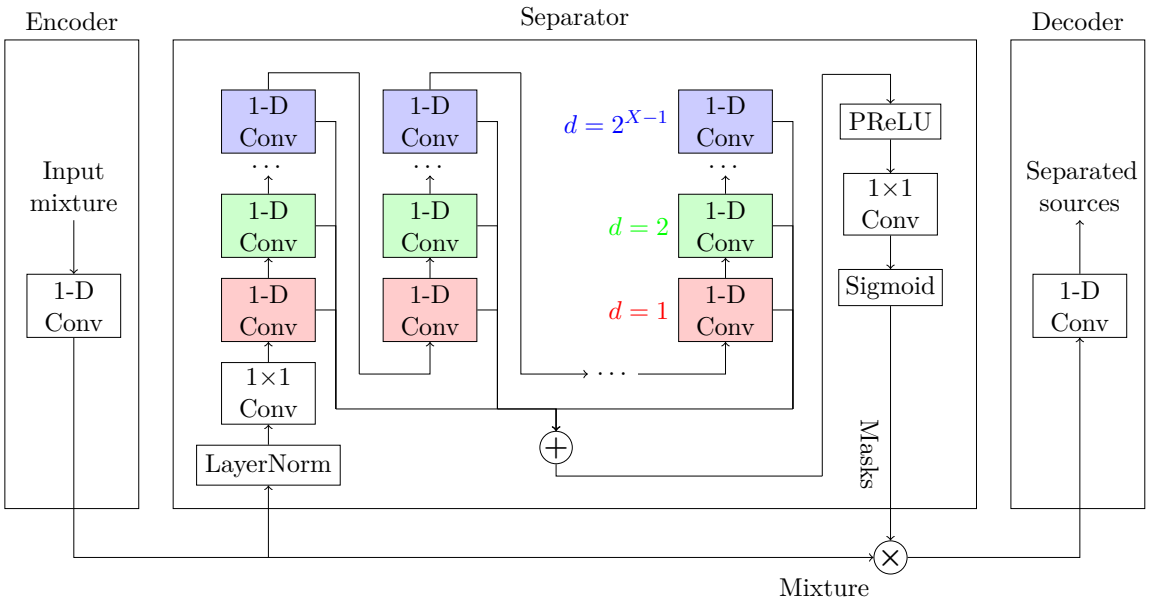


Figure 2.1: ConvTasNet neural network architecture.

Encoder is the first block of the ConvTasNet. It gets raw speech signal on input. This block consist of the convolutional block trained to produces representation that resembles mentioned STFT. We will denote this representation as pseudo-STFT.

The second part ensures the separation. It consists of a series of consecutive convolutional blocks that are placed in series. Each of them stands for a filter that is applied on longer and longer parts of the context from both sides. The number of these convolutional blocks in each series determines how much context will be taken into account. The separation part takes a pseudo-STFT as an input. Series of convolutional blocks then provides feature extraction and generation of the separation masks. Separation masks are two dimensional matrices of the same shape as the input pseudo STFT. Each mask contains values between the zero and one. The value represents the probability that the current frequency block belongs to the speaker separated by the current mask. Separation is then performed by element wise multiplication of each mask and the input mixture pseudo STFT given by the encoder. These multiplications produce N separated outputs one for each speaker.

Finally it is necessary to rebuild time-domain signal again from the pseudo STFT representation. This is provided by the last part of the ConvTasNet architecture, the decoder. This part consists of the convolutional block similarly to the encoder part. This block is trained to reverse the pseudo-STFT. Decoder takes separated pseudo STFTs one by one and generates separated time-domain signals from them.

2.3 Training of neural networks for speech separation

Training the neural network for speech separation is performed on mixtures where the original single speaker signals are known. This kind of training is called supervised learning. It is hard to obtain such data from the real-world mixtures. So it is necessary to use artificial ones. Training the system only on artificial mixtures often leads to bad performance in real-world usage. This issue will be addressed in more detail in Chapter 4.

The training consists of several steps that are shown in Figure 2.2. Firstly, the neural network estimates the N separated signals from the given mixture to the N outputs, where N is the fixed number of speakers. It is necessary to know how many speakers have to be separated before the training. Then the loss function is computed between the estimated signals and the targets. The computed loss value is used for the training.

The loss function used for the training is scale-invariant signal-to-noise-ratio (SI-SNR) function [4], which is defined as:

$$\vec{s}_{\text{target}} := \frac{\langle \hat{\vec{s}}, \vec{s} \rangle \vec{s}}{\|\vec{s}\|^2} \tag{2.2}$$

$$\vec{e}_{\text{noise}} := \hat{\vec{s}} - \vec{s}_{\text{target}} \tag{2.3}$$

$$\text{SI-SNR}(\vec{s}, \hat{\vec{s}}) := 10 \log_{10} \frac{\|\vec{s}_{\text{target}}\|^2}{\|\vec{e}_{\text{noise}}\|^2} \tag{2.4}$$

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. The $\|\vec{s}\|^2 = \langle \hat{\vec{s}}, \vec{s} \rangle$ denotes the signal power, where $\langle \hat{\vec{s}}, \vec{s} \rangle$ denotes the dot product between estimated and original source. The function is scale-invariant because the scale of the estimated signal does not influence the result. The neural network is trained to maximize this function.

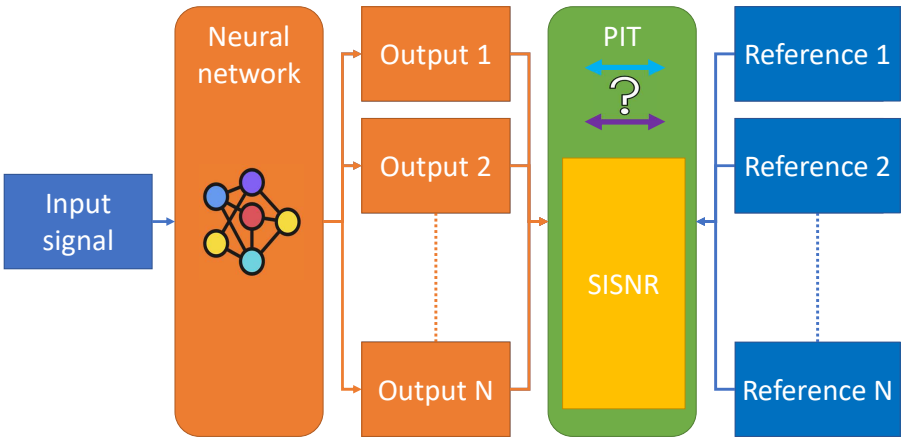


Figure 2.2: Basic training of the speech separator using neural networks.

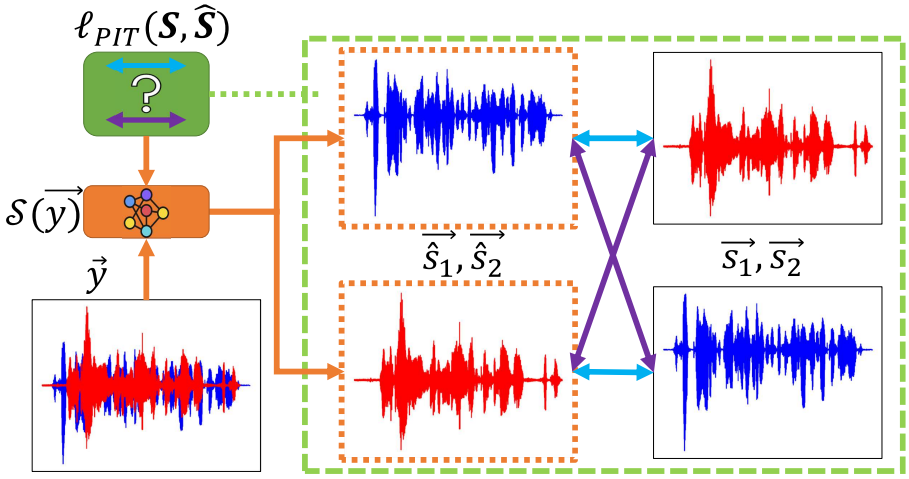


Figure 2.3: Example of PIT method used on the mixtures that consist of the speakers.

When evaluating the output of the speech separation neural network, several estimates need to be compared to several reference signals in all permutations. This is due to the fact that neural network does not know which estimated output signal belongs to which

reference signal. In the other words signal of the speaker A could be estimated arbitrary to the first or second output. This gives rise to a permutation problem.

The solution is the permutation invariant training (PIT) method [31] shown in Figure 2.3. This method computes the loss function between all permutations of original and estimated signals. The best-computed value corresponds to the right permutation of the estimated outputs and this value is also used for the training of the neural network. This method is defined as:

$$\hat{\vec{s}}_1, \hat{\vec{s}}_2, \cdots, \hat{\vec{s}}_N = \mathcal{S}(\vec{y}) \tag{2.5}$$

$$l_{PIT}(\mathbf{S}, \hat{\mathbf{S}}) = \min_i \sum_{j=1}^{N} -\text{SI-SNR}(\vec{s}_{\sigma_{i,j}}, \hat{\vec{s}}_j) \tag{2.6}$$

where $\mathcal{S}(\vec{y})$ is separator function, that estimates separated signal as outputs from the given mixture $\vec{y}$. These signals are represented by vectors $\hat{\vec{s}}_1, \hat{\vec{s}}_2, \cdots, \hat{\vec{s}}_N$. The $l_{PIT}(\mathbf{S}, \hat{\mathbf{S}})$ function takes two parameters: $\mathbf{S}$, which is matrix of vectors of target signals and matrix $\hat{\mathbf{S}}$, which consists of vectors of estimated separated signals. Variable N represents the number of single speakers present in the mixture. Permutation $\sigma_{i,j}$ is the index of j-th target signal in the i-th permutation of target vectors given by the matrix $\mathbf{S}$. All computed SI-SNR loss values are compared. The best (minimum) of them marks the best permutation of references. This value is used for the training of the speech separation neural network.

Chapter 3

Generative adversarial networks

A generative adversarial networks (GAN) [8] is a model which is used for data generation. The GAN model is able to generate new data which have never been seen before. Nevertheless the generated data are from the same domain as the dataset the GAN is trained on. There are a lot of ways how the GAN could be used. For example data reconstruction, where the GAN generates missing parts of the data [23]. It is also possible to use it for grayscale images colorizing [26], artificial face generation [5], face aging [3], text translation [10], etc. And finally, there is an option to use the GAN for data augmentation, which is what we used it for in this work.

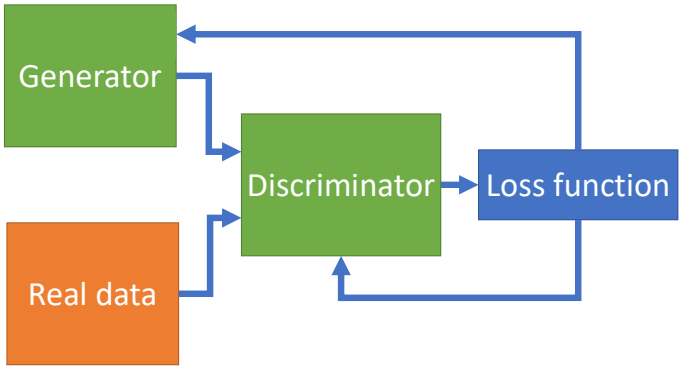


Figure 3.1: Architecture of the base concept of the generative adversarial networks model.

3.1 Training generative adversarial networks

The base GAN model architecture is shown in Figure 3.1, it consists of two parts: the generator and the discriminator. The first part is the generator which is the neural network that is trained to generates a new data, called fake data. The generator neural network architecture depends on the type of data that will be generated. In general, the generator is defined as follows:

$$\vec{g} = \mathcal{G}(\vec{z}) \tag{3.1}$$

where $\vec{z}$ is random vector generated from some random distribution, for example normal distribution. The $\mathcal{G}(\vec{z})$ is the generator function which takes the random vector $\vec{z}$ as an input and generates the fake data as an output. These generated data are represented by the vector $\vec{g}$.

The second part of the GAN model is the discriminator, which is also a neural network. This network is used to tell whether the given data is fake or real. Formally the discriminator is defined as follows:

$$p_{real}(\vec{x}, \mathcal{D}) = \mathcal{D}(\vec{x}) \tag{3.2}$$

where $\vec{x}$ is vector of real or fake data. The $\mathcal{D}(\vec{x})$ is the discriminator function that takes the $\vec{x}$ as an input and estimates the probability of the given data $\vec{x}$ is real, which is defined by $p_{real}(\vec{x}, \mathcal{D})$. Conversely, the probability that the data is fake is represented as:

$$p_{fake}(\vec{x}, \mathcal{D}) = 1 - p_{real}(\vec{x}, \mathcal{D}). \tag{3.3}$$

To train the GAN model the loss function is also needed. The discriminator output is used as the input for this loss function, which computes loss values for both networks. It is defined as follows:

$$\mathcal{L}(\mathcal{G}, \mathcal{D}) = E_{\vec{x}\sim p(\vec{x})}[\log(p_{real}(\vec{x}, \mathcal{D}))] + E_{\vec{z}\sim p(\vec{z})}[\log(p_{fake}(\mathcal{G}(\vec{z}), \mathcal{D}))] \tag{3.4}$$

where $\mathcal{G}$ is generator function, $\mathcal{D}$ is discriminator function. The $p(\vec{x})$ is the real data distribution. The generator generates fake data from random inputs $\vec{z}$ that are given by the distribution $p(\vec{z})$. Then $E_{\vec{x}\sim p(\vec{x})}$ is expected value over all given real data and in contrary $E_{\vec{z}\sim p(\vec{z})}$ is the expected value over all given fake data.

These two networks then play the min-max game. This game provides the GAN model training, which consists of two steps:

1. Discriminator neural network training.
2. Generator neural network training.

The first step can be seen in Figure 3.2. In this step, the generator neural network weights are locked. This is shown by the gray color of the generator box in the figure. The generated data $\vec{g} = \mathcal{G}(\vec{z})$ and the real data x are used as the input for discriminator neural network.

Discriminator is trained to maximize this loss function, this is defined as:

$$\mathcal{D}^* = \max_{\mathcal{L}(\mathcal{G},\mathcal{D})} \tag{3.5}$$

On the other hand in the second step presented by Figure 3.3, the discriminator neural network's weights are locked. In this step, the generator neural network is trained to deceive the discriminator by generating such similar fake data to real ones that the discriminator will not recognize them as fake. In other words, generator is trained to minimize loss function, which is defined as:

$$\mathcal{G}^* = \min_{\mathcal{L}(\mathcal{G},\mathcal{D})} \tag{3.6}$$

Unlike discriminator training, only generated data is used in generator training. These two steps are repeated and both networks are getting better and better in their tasks.

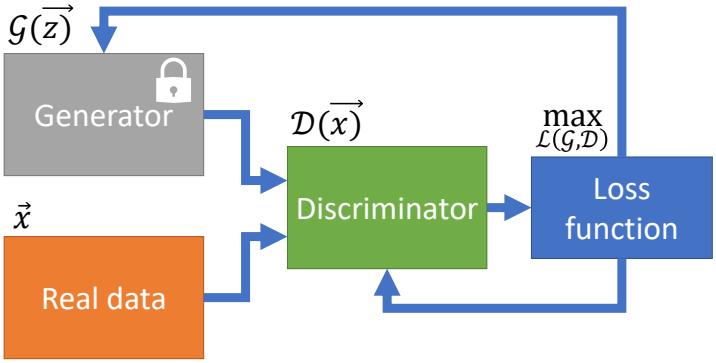


Figure 3.2: The first step of training the architecture of the generative adversarial networks. The generative neural network is locked. The discriminator network is trained on generated (fake) and real data.

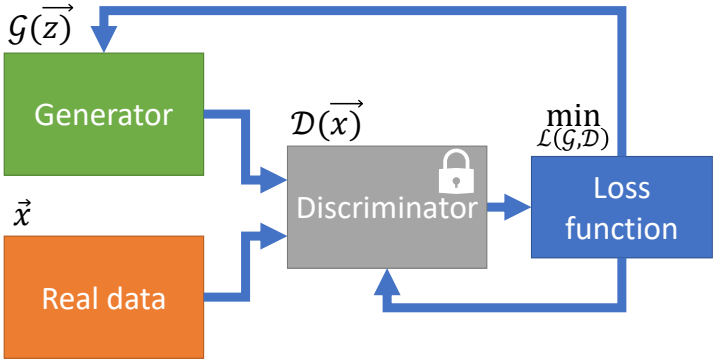


Figure 3.3: Architecture of generative adversarial networks training in the second step, where discriminator neural network is locked and generator is trained to deceive the discriminator.

3.2 Training problems of generative adversarial networks

There are several problems in training of the generative adversarial networks [14]:

1. Mode collapse.
2. Non convergence.
3. Diminished gradients.

3.2.1 Mode collapse

Number of data classes in the dataset can be represented as modes. The generator is then trained to generate the data from these different modes. In the mode collapse training

problem, the generator is unable to generate data from all of these modes. Instead it only generates data from a few of them.

For example, in case of the generator that is trained to generate single-digit numbers, the modes that data could be generated from are numbers from zero to nine. When the GAN model fall in the mode collapse problem during the training, then the model is only able to generate, for example, the numbers two and five.

The reason why this problem occurs is described in following text. The generator is trained to generate such data that will confuse the discriminator as much as possible. If the discriminator weights will be locked during training and only the generator will be trained, it will in some point converge to the state, where it will generate the high quality data that strongly confuses the discriminator. Nevertheless this data will be generated independently on the given random vector $\vec{z}$, which is defined in Section 3.1. The generator will collapse here to the single point which gives the highest quality data.

Now if the discriminator weights will be unlocked and it will start to train again, it will only get the data generated from the single mode. Therefore, the discriminator will detect as true or fake the generated data only in the single mode. The reason is that the generator makes the vector $\vec{z}$ irrelevant.

The described case is the extreme one, but it may occur when the generator gets too much space during the training and is much stronger then the discriminator.

3.2.2 Non convergence

The generator and separator parts of the GAN model are playing the min-max game during the training. In this game the first player is trying to maximize its actions and conversely the second player is trying to minimize them. The point where the one player will not change its action regardless of what the opponent may do is called Nash equilibrium. According to game theory, GAN should converge to this point. Nevertheless, it is very difficult to find a such parameter values that will make the GAN model converge during the training. This is caused by the fact that the adjustable parameters of the GAN model are very sensitive to any changes.

3.2.3 Diminished gradients

This problem occurs when the discriminator is much stronger then the generator. Then the discriminator get to the state where the gradients for the generator training are vanished. In other words, the generator is unable to confuse the discriminator and due to the high quality of the discriminator, it is unable to learn anything.

The loss function mentioned in the first article [9] that introduced the GAN model encountered the problem of vanishing gradients. This means that the gradients are too small that the neural network is unable to learn anything from them. There is also second loss function mentioned in the article, which could solve the vanishing gradiens problem. Nevertheless, it encounters problem of fluctuating gradients, which causes the GAN model instability. There were many attempts to create loss function which solves these two problems such as LSGAN [19], WGAN [2], WGAN-GP [11], BEGAN [6], etc. Nevertheless, none of them solve these problems as it is described in the paper „Are GANs Created Equal?“ [17].

3.3 Generative adversarial networks for data augmentation

Generative adversarial networks (GAN) could be also used for data augmentation. For example in the article „Low-Shot Learning from Imaginary Data“ [27], GAN is used as a hallucinator. The hallucinator works as follows: When some person looks at image, the person will also imagine another similar images. For example, if the image is a dog then the person imagine another dogs or the same dog in a lot of different positions. If the computer could do this imagination then it could learn better from fewer data. In this article, GAN model takes image as the real data input $\vec{x}$ and noise as the fake data input $\vec{z}$. Then it provides estimation (hallucination) of similar data, for example, the mentioned dog in different positions. Generated hallucinations are then classified with discriminator classification neural network. Outputs of classification are used to compute loss function and train the generator (hallucinator).

In the another article „GAN-based, Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification“ [7] GAN model is used to generate synthetic lesions on livers. In this article two GAN architectures are used DCGAN [24] and ACGAN [21]. The difference between these two architectures is that ACGAN trains generators to generate samples from given class instead of random ones. This is achieved by getting the second output from the discriminator part. The second output estimates the most probable class of the given data. This information is then used with value of the real fake loss function to train the generator. In medicine it is hard to obtain large datasets, this is due to the fact that data are sensitive and they contain private information about the patients. Another fact is that some data are expensive to obtain. Therefore, using a GAN in the way to extend the dataset is very useful here. Classifier trained on the original dataset has resulted in a sensitivity value of 78.6% and specificity value of 88.4%. With the classifier trained on the dataset extended by the presented GAN model, there is an improvement in both values to the sensitivity of 85.7% and specificity of 92.4%.

There are a many more examples of the GAN model usage. Nevertheless, the mentioned usages show that it is possible to use the GAN model for the data augmentation and these augmentations could significantly improve the results of the original models.

Chapter 4

Robust speech separation

As it is mentioned in Section 2.3 neural networks used for speech separation are trained on the mixtures with known single speaker signals. These mixtures are hard to obtain from the real world. If the single speaker signals could be easily obtained from the real-world mixture then the speech separation task will be solved and the speech separation system will be useless. Thus, mixtures used for the speech separation system training are artificially mixed from the recorded speaker signals. However, the speech separation systems trained on artificial mixtures achieve good results on these mixtures, they often fail on real-world mixtures. The reason of the bad results on real-world mixtures is due to the fact that real-world mixtures contain echoes and noises given by the environment where the mixture was recorded.

4.1 Data augmentation

To improve the speech separation system behaviour on the real-world mixtures it is possible to make the speech separation system more robust. The main idea of making a robust system is to add some other input mixtures, that extend the variance of the training and validation data. The separation system trained on this extended dataset could better manage the real-world mixtures. The mixtures that extend the dataset are created from the original mixtures by the method called data augmentation.

4.2 Classic practices

There are many classic practices to augment the data and extend the original dataset. It is possible to split these practices into two categories:

1. Practices that add noises and echoes to signal.
2. Practices that work with signal itself.

In this section, some practices from each category will be described.

4.2.1 Noise addition

Examples of noises that could be used for data augmentation are Gaussian noise, Color noise, White noise, etc. It is also possible to record noises from real-world places such as

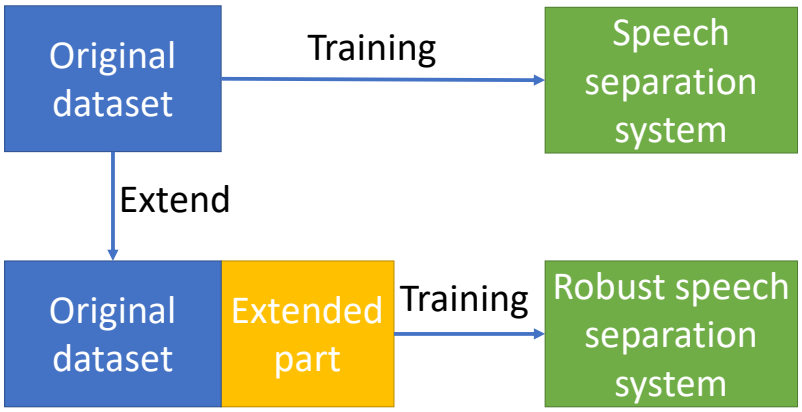


Figure 4.1: Base idea of robust speech separation

streets, rooms, airports, and many others. All of these noises are mixed with the original mixture using weighted sum, which is defined as:

$$y_t^{(\text{aug})} = w_s y_t + w_n n_t \tag{4.1}$$

where $y_t^{(\text{aug})}$ is augmented mixture, y_t is original mixture, n_t is selected noise for augmentation and w_s and w_n are mixtures and noise ratios, that determines mixing ratio of the original mixture and selected noise.

4.2.2 Echoes

Echoes, also called sound reverberation [30] are sound reflections caused by objects placed the real world spaces. These objects could be for example walls, hills, furniture, etc. For human ear reverberation with delay between $1.5s$ to $2.5s$ is still clearly understandable. The reverberation with these delays could be heard in concert halls and it is possible to simulate it by the hall reverberation algorithm [20]. Mixtures with simulated reverberation could be used to make speech separation systems more robust toward environments that give these echoes.

4.2.3 Frequency filters

All the following techniques described in this section are from the second category. The first of them is the technique that uses frequency filters. These filters are applied to the signals and cause changes in the signal's frequency spectrum. Four well-known types of filters:

a) Low pass filter, which allows passing only signal frequencies lower than some selected threshold.

b) High pass filter, which allows only these signals frequencies that are higher than the selected threshold.

c) Band pass filter, which combines two previous filters. This filter allows passing signal's frequencies that are inside some selected frequencies range, called bandwidth.

d) Band stop filter, which works opposite to band pass filter. This filter allows passing signal frequencies that are not inside some selected bandwidth.

Using frequency filters for data augmentation could make the speech separation systems more robust towards different microphones that are used for data recording. Not all microphones are sensitive to the same range of frequencies.

4.2.4 Gain

Gain is an augmentation practice that works with increasing or decreasing signal volume. Changes in gain of the signal are done by multiplying or dividing the signal amplitude by a random amplitude factor. Higher values of the signal amplitude mean a higher volume of the signal and conversely lower amplitudes mean lower volume. So it is obvious that moving with signal's gain should make the model more robust towards different volumes of recorded signals.

4.2.5 Pitch

Furthermore, it is also possible to shift the pitch of the signal. Shifting the pitch is in other words shifting of tone. If the pitch is raised, then signal sounds in higher tones, and conversely when the pitch is reduced, the signal sounds in lower (bass) tones. Considering that different people speak in different tone, the pitch shift augmentation could make speech separation system robust towards the people voices that are not presented in the current dataset.

4.2.6 Tempo

Another data augmentation can be provided by working with the signal's tempo. Tempo could be sped up or slowed down. When the tempo is simply sped up, then the signal's pitch is also getting higher. Conversely, with the tempo slowing down, the pitch is going lower. It is possible to use the simple tempo augmentation with this behavior but it is not always wanted. So to change the tempo but make the signal to sound in the same tones, it is necessary to use pitch shifting together with tempo changes.

4.2.7 Polarity inversion

The polarity inversion augmentation is provided by multiplying the signal by -1, so in other words to flip it upside down. This change is not distinguishable from the original signal by human ears but it can be a problem for the neural networks where this change may make a big difference in their behavior. So this augmentation can help to make neural networks more robust towards to that simple change.

4.2.8 Audio shift

The audio shift augmentation is used to shift the audio signal forwards or backwards on time axis. This technique could be provided with rollover or not. When the rollover is applied, then the signal is moved as the cyclic buffer. This could raise some problems in

neural networks, which use context information. If the rollover is not applied then the overflowed parts of the signal are lost and zeros are pad in front or back of the signal.

4.2.9 Channels shuffling

In multi-channel signal, it is possible to shuffle channels. For example, in a stereo signal, it is possible to switch the left and right channels. Channel shuffling could help machine learning models to combat different positional biases.

4.2.10 Vocal tract length

The speaker signal frequency values also depend on the length and shape of the speaker vocal tract. Each speaker has different vocal tract which produces different vocal tract features. It is possible to work with the vocal tract feature in two ways.

The first way is to get rid of this information by using the vocal tract length normalization method (VTLN) [15]. This method works by the warping the frequency axis in the filter bank analysis, which is warped by the normalization scale factor α. This factor is derivated from an estimate of the length of the speaker's vocal tract. Thus, the vocal of the each speaker with different vocal tract length (VTL) is scaled to the „standard“ vocal tract.

On the other hand, the second way is to use this information and to extend the dataset. The data added to the dataset should increase the variance of VTL. The method that could generate data with different VTL features is called vocal tract length perturbation (VTLP) [16]. This method use the same warping method as the VTLN method mentioned above, but it used various scale factors α. Each of that factors simulates different VTL and generates data which will be added to the dataset and increase the dataset VTL variance.

4.3 Using generative adversarial networks

In this work the Generative adversarial networks (GAN) are used for data augmentation on adversarial mixtures. The generator here takes the adversarial mixture as an input, generates augmentation and put the augmented mixture as an output. The generator is defined as:

$$\vec{y}^{\,(\mathrm{aug})} = \mathcal{G}(\vec{y}) \tag{4.2}$$

where $\mathcal{G}(\vec{y})$ is the generator function that takes the original mixture $\vec{y}$ as an input and generates augmented mixture $\vec{y}^{\,(\mathrm{aug})}$ as an output.

Instead of the common GAN architecture, where the discriminator consists of the second neural network and one loss function, in this architecture, the discriminator consists of two parts as can be seen in Figure 4.2. The first part uses the speech separation neural network and the SI-SNR loss function, which is computed between original targets and separated signals from the augmented mixture. It is defined as:

$$\hat{\mathbf{S}} = (\hat{\vec{s}}_1, \hat{\vec{s}}_2, \cdots, \hat{\vec{s}}_N) = \mathcal{S}(\vec{y}^{\,(\mathrm{aug})}) \tag{4.3}$$

$$p = -l_{\mathrm{PIT}}(\mathbf{S}, \hat{\mathbf{S}}) \tag{4.4}$$

where $\vec{y}^{\,(\text{aug})}$ is the generated noisy mixture which is defined by Equation 4.2. The matrix $\mathbf{S}$ consists of all separated signals from the augmented mixture by separator function $\mathcal{S}(\vec{y}^{\,(\text{aug})})$. And finally p is loss value computed by $l_{\text{PIT}}(\mathbf{S}, \hat{\mathbf{S}})$ defined by Equation 2.6.

The second part represents the similarity between the original $\vec{y}$ and the augmented $\vec{y}^{(\text{aug})}$ mixture, otherwise the generator could generate any signal which does not even corresponds to the reference source. This similarity is computed by the SI-SNR function defined by Equation 2.4. The second part of the discriminator is defined as:

$$s = -\text{SI-SNR}(\vec{y}^{\,(\text{aug})}, \vec{y}) \tag{4.5}$$

where $\text{SI-SNR}(\vec{y}^{\,(\text{aug})}, \vec{y})$ is defined by Equation 2.4. The $\vec{y}$ is the original mixture defined by Equation 2.1 and the $\vec{y}^{\,(\text{aug})}$ which is defined by Equation 4.2.

The generator goal is to confuse the separator system as much as possible with the generated augmented mixtures that are as similar as possible to the original mixtures. In other words, the generator tries to minimize the values p and s. The loss value that is used for generator training is computed as the weighted sum of these two values p and s. It is defined as:

$$l = w_{\text{sim}} s + w_{\text{sep}} p \tag{4.6}$$

where w_{sim} is weight of the similarity value s and w_{sep} is the weight of the separator loss value p.

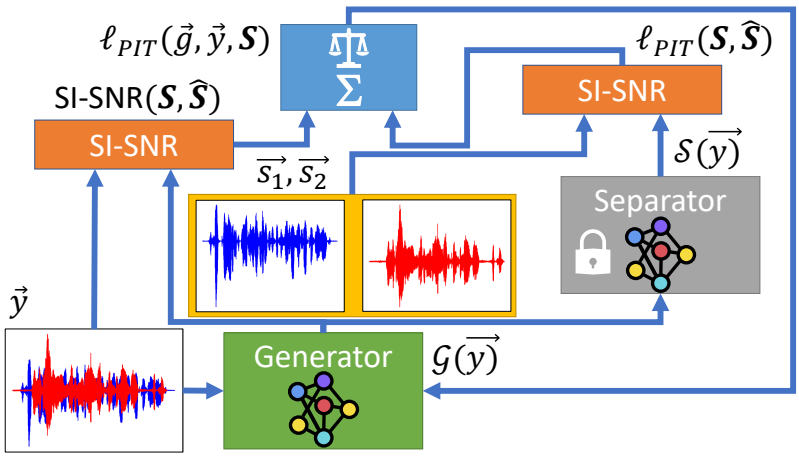


Figure 4.2: Architecture of generative adversarial networks training used in this work. This figure shows the step where the generator is trained.

As it was mentioned in Section 3.3, the generator and the discriminator play the min-max game. Thus, the separator should be also trained. The training of the separator is very similar to the classic speech separation training described in Section 2.3. The only change is that the generated augmented mixtures are also used during the training. As it is shown in Figure 4.3 they are added to the training by flipping the unfair coin. The percentage of the generated augmented mixtures is a hyperparameter that should be tuned.

In each step of the training, the augmented mixture is used with probability c, while the original one with probability $1 - c$. The probability c is set as a hyperparameter. The augmented or original mixture is given to the separator as an input with the same ground truth single speaker signals. This should make the speech separation system be more robust towards the augmented mixtures generated by the generator neural network.

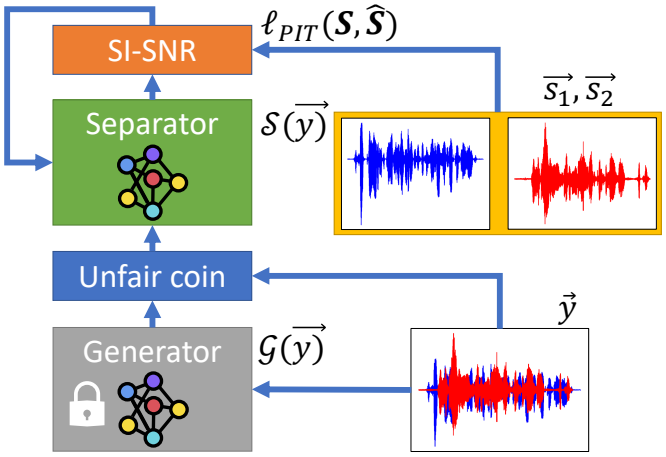


Figure 4.3: Architecture of generative adversarial networks training used in this work. This figure shows the step where the discriminator is trained.

4.3.1 Problems with the best model selection

Training of such a system raises a new problem. Which separator and which generator from which epoch should be chosen as the best one. If the chosen generator will be the generator with the best-computed loss value on validation data during training, it is possible that the separator network was weak in this training epoch and the generator network gets a better result that does not correspond to its trained quality. In other words, with the poor separator, the poor generator could get better results than the quality generator with the quality separator.

As mentioned, the problem with the best model choice also applies to the separator model. With the poor generator which generates practically the same augmented mixtures as the original ones the separator model will get nice results even when it is not the most robust one.

In the case of this work, the generator model is not needed. The aim is to make robust speech separator model, so only the best separator model should be selected. The solution is to use all generators from the training to generate the augmented mixtures. Then next step is to use different separators to separate signals from the augmented mixtures and choose the separator with the best overall score. The whole model selection pipeline is also shown in Figure 4.4. Generating of augmented mixtures is processed by generating the random number between the zero and the number of epochs. This number represents the used generator which will be used to generate the augmented mixture. This is defined as:

$$i \sim \mathcal{U}(0, E); i \in \mathbb{N} \tag{4.7}$$

$$\vec{y}_{ij}^{\,(\mathrm{aug})} = \mathcal{G}_i(\vec{y_j}) \tag{4.8}$$

where the i is the random integer generated from the uniform discrete distribution $\mathcal{U}$ of integers between the zero and E. The E is the number of epochs for which the GAN model were trained on, it is also the number of all saved generators. The $\vec{y_j}$ is the j-th original mixture from the validation set. The $\mathcal{G}_i$ is the generator function of the i-th generator. Finally, the $\vec{y}_{ij}^{\,(\mathrm{aug})}$ is the j-th generated augmented mixture be the i-th generator from the j-th original mixture.

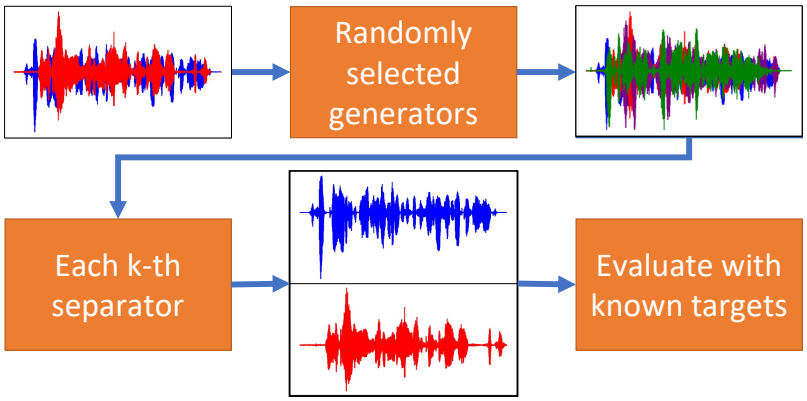


Figure 4.4: The best separator model selection pipeline.

All the generated augmented mixtures will be separated by the selected separator and the SI-SNR values will be computed, using the PIT, between the separated outputs and the known targets. This is defined as:

$$k_{\text{best}} = \arg\min_{k}\left(\frac{1}{J}\sum_{j}^{J}\left(l_{\text{PIT}}\left(\mathcal{S}_k\left(\vec{y}_{ij}^{\,(\text{aug})}\right), \mathbf{T}_j\right)\right)\right); k \in (20, 30, 40, \ldots, E) \tag{4.9}$$

where $\mathbf{T}_j$ is the matrix of target signals for the j-th mixture. The $\mathcal{S}_k$ is the k-th separator function, where k is from the $(20, 30, 40, \ldots, E)$ which is the vector of indexes of the validated separators. The l_{PIT} is the PIT loss function defined by Equation 2.6 and k_{best} is the index of the best separator.

Chapter 5

Implementation

The implementation of the presented generative adversarial networks (GAN) model is based on Jian Vu's PyTorch implementation of the ConvTasNet [29]. The PyTorch library [22] is Python based machine learning library, that contains neural network layers, optimizers, schedulers, and tensors that can work with the CUDA platform. The used ConvTasNet module contains the ConvTasNet neural network implementation.

5.1 Main process

The system consists of the Model, Dataloader, and Trainer class. Firstly the mixtures and the corresponding reference single speaker signals are loaded by the Dataloader. Then the model is initialized with randomly set weights. It is also possible to load the weights of the trained model and use them for further training. The Model and the Dataloader are put into the Trainer class which can train the model.

To train the model the *run* function of the Trainer class should be called as it is shown in Figure 5.1. This function loops over the set number of epochs and in each epoch calls two functions: *train* and *eval*. The *train* function loops over the batches of training data provided by the given Dataloader. For each batch the *compute_loss* function is called that provides the forward pass and loss computation. This function returns a computed loss value, which is used by the backpropagation algorithm to train the generator or the separator neural network. The *eval* function works similarly as the *train* function, but the backpropagation algorithm is not called.

5.2 Switching of training

Training of the GAN model includes the min-max game between the generator and separator part. This is done by switching the training between these two parts as it is shown in Figure 5.2. In this work, the generator and separator training is switched several times during the epoch after at least one batch. The number of batches after which the training is switched is different for the generator and separator parts and is defined by the hyper-parameter. There is also a possibility to control the training switch according to results achieved by each part. In this case, the generator or the separator is trained until it achieves the results that satisfy the defined goal.

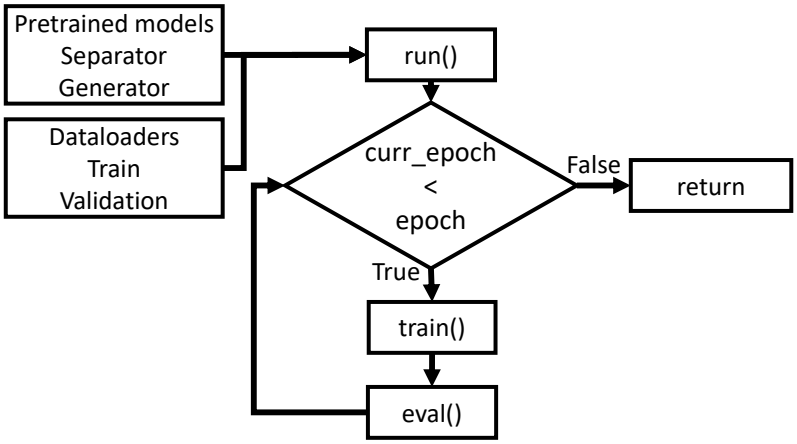


Figure 5.1: Diagram of the main training cycle that is done inside the function *run*.

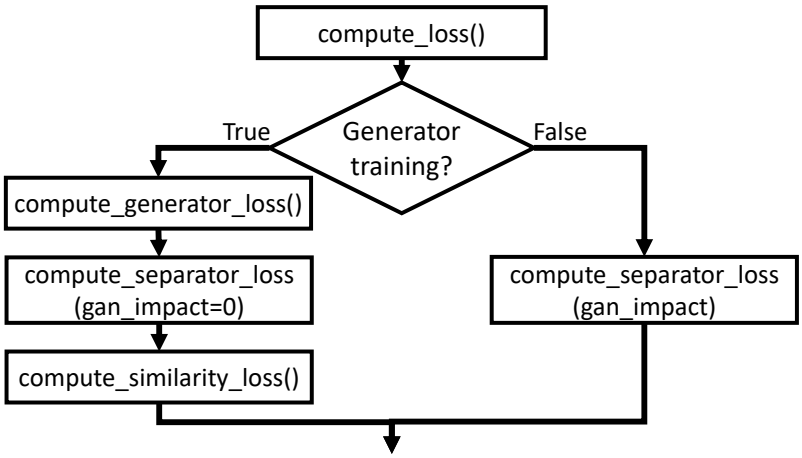


Figure 5.2: Diagram of the loss computation during the generator and separator training.

5.3 Loss computation

The loss computation part of the Trainer is called depending on which part is currently trained. As it is mentioned in Section 4.3 the generator loss function is composed of two parts, the separator and similarity loss functions. To compute these, the generator first generates an augmented mixture. Then, the separated signals are estimated by the separator from the augmented mixture. The *gan_impact* is here set to 0, because the mixture is already generated. Then the separator loss function takes these signals as an input and compares them with the known targets. The second part of the generator loss function is computed as the SI-SNR between the original and generated augmented mixture. These two computed values are then summed up with different weights and the backpropagation is called on the generator neural network.

On the other hand, in the case of the separator training, only the separator loss function should be computed. The only difference is that some amount of the generated augmented mixtures must be included in the training. This is implemented by generating the random number between zero and one. The generated number is then compared to the defined hyperparameter of the *gan_impact* of the augmented mixture. If the generated number is

smaller than the defined impact then the generated augmented mixture is used as an input of the separator network and vice versa. The SI-SNR value is then computed between the estimated outputs and the target signals and the backpropagation is called on the separator neural network.

5.4 Evaluation

The trained GAN model should be evaluated. The evaluation process is shown in Figure 5.3 Firstly the best separator should be chosen from all epochs. The *generate* script uses generators from random generator epochs to augment the validation part of the dataset. Then the *evaluate* script evaluates each n-th separator on these augmented data.

The second part of the GAN model evaluation is to evaluate the selected robust separator on the testing part of the dataset. The *separate* script estimates single speaker signals from the testing mixtures. The second script *compute_si_snr* compares the estimated signals with references. The SI-SNR values are computed during this comparison and then the average is computed from them. The computed mean is the evaluation result.

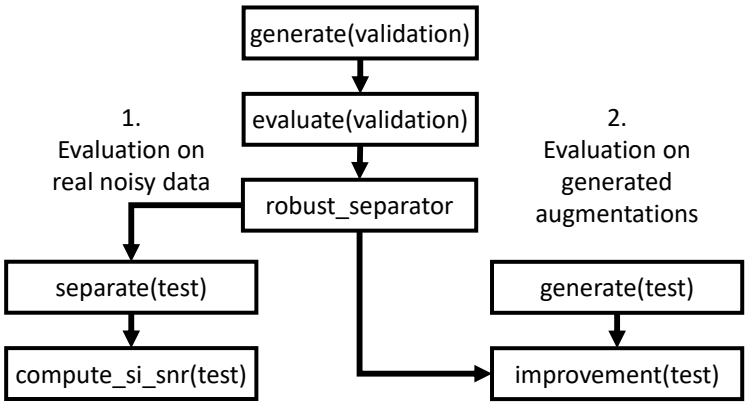


Figure 5.3: Diagram of the evaluation process for both evaluations. The left side shows the evaluation on real noisy data and the right side shows the evaluation on generated augmentations.

Another evaluation is set to compute the improvement between the original and robust separator on the augmented data. For this type of evaluation, the *generate* script is again used, but now the testing part of the dataset is augmented. Then the *separate* and *compute_si_snr* scripts are used to evaluate the original and robust separator. The computed average SI-SNR values are then used to compare the quality of the robust model.

Chapter 6

Experiments

Experiments with a generative adversarial networks (GAN) model used to make the speech separation more robust are performed with the setup described below. The first experiments try to find the applicable combinations of parameters, which do not lead to one of the GAN training problems. The second experiments evaluates the GAN model on the WSJ0-2mix and WHAM dataset and solve problems raised during experimenting. The last experiments train the GAN model on the WHAM dataset and evaluate it as well, to proof that the concept is easily applicable to the different dataset.

6.1 Dataset

Experiments are done on the Wall Street Journal dataset (WSJ0-2mix) [12]. 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 with various signal-to-noise ratios (SNR) between 0 dB and 5 dB. All data are downsampled to 8 kHz sample rate. For training, there are 20000 mixtures corresponding to 30 hours, for cross-validation, there are 5000 mixtures corresponding to 10 hours, and there are 3000 mixtures corresponding to 5 hours for testing. The testing part contains different speakers than the training and cross-validation parts.

The WSJ0 Hipster Ambient Mixtures (WHAM) [28] is the second dataset used in the experiments. This dataset pairs each two-speaker mixture in the WSJ0-2mix dataset with the unique noise background scene. Noises are recorded in San Francisco Bay in urban environments such as coffee shops, restaurants, bars, office buildings, parks, etc. They are recorded at a sampling rate of 48 kHz and then downsampled to the 8 kHz to match with the WSJ0-2mix dataset. There are 80 hours of noises recorded at 44 different locations and they are separated to the four bins: very quiet, quiet, normal and loud. Each part of the dataset (training, cross-validation and testing) contains noises from at least two unique locations in each bin. The sizes of training, cross-validation, and testing parts are the same as in the raw WSJ0-2mix dataset. Each background noise is mixed into the mixture by sampling SNR value between -6 dB and 3 dB and then by applying gain to the louder speaker of the mixture such that the SNR between the louder speaker and the noise is equal to the sampled SNR value.

6.2 Initial separator and generator networks setups

The neural network architecture used for both networks (separator and generator) is ConvTasNet [29]. Parameters of the ConvTasNet are shown in Table 6.1. The table also shows the parameter values for the generator and separator neural networks used in the experiments.

Table 6.1: Hyperparameters of the ConvTasNet network [18]

Symbol	Description	Generator	Separator
F	Number of filters in autoencoder	128	128
L	Length of the filters (in samples)	40	40
B	Number of channels in bottleneck	128	128
H	Number of channels in convolutional blocks	192	192
P	Kernel size in convolutional blocks	3	3
X	Number of convolutional blocks in each repeat	3	7
R	Number of repeats	1	3
O	Number of outputs	1	2

The separator network used for training has been pretrained on one of the above-mentioned datasets or their combination. So there are three separator networks used in experiments, each pretrained on one of them. The baseline pretrained scores are shown in Table 6.2. From the given results it is obvious that the WHAM dataset is much more difficult than the WSJ0-2mix. This is caused by the noises added to the mixtures. If the system is trained on the WSJ0-2mix and tested on the WHAM, then the results are quite poor.

Table 6.2: Baseline results of pretrained separator neural networks. Results are computed by SI-SNR loss function using PIT method. Datasets in rows are the training ones. Testing datasets are in columns.

	WSJ0-2mix SI-SNR [dB]	WHAM SI-SNR [dB]
WSJ0-2mix	12.46	-2.99
WHAM	9.04	6.09
WSJ0-2mix + WHAM	12.34	6.45

The generator network is much smaller than the separator one and it has been pretrained for the self-identity task. This is due to the fact that the untrained encoder and decoder parts of the ConvTasNet leads to the generating nonsense augmented mixtures. It is possible to train the generator network properly during the GAN model training but it is very difficult and often leads to a bad GAN model training. Since the aim of the GAN training is not to train the encoder and decoder parts but to generate augmented mixtures, pretraining the self-identity removes the encoder and decoder training problem from the GAN model training process.

6.3 Adjustable parameters

The first task of the experiments is to find the right combination of parameters that will train GAN properly. The adjustable parameters are:

- Separator loss weight w_{sep}, which sets the importance of the separator loss in generator training. The generator training loss function is defined by Equation 4.6. The separator loss value is computed during generator training on the generated augmented mixtures. The generator is trained to maximize the separator loss in order to confuse the separator as much as possible.
- Similarity loss weight w_{sim} is used to indicate the importance of the similarity between the generated augmented mixture and the original one. This loss function is also computed during training and its role is to constrict the generator so that it would not generate complete nonsense.

GAN model is switching between the separator and generator training during each epoch. Two parameters control this:

- Separator batch cap c_{sep}, which sets how many batches will be used in separator training turn.
- Generator batch cap c_{gen}, which sets how many batches will be used in generator training turn.

For example, when c_{sep} and c_{gen} are set to 10, the generator will be trained on the first ten batches. After this, the training is switched to the separator training, which uses other ten batches and then switches back. The number of batches for each model can significantly influence the training and these parameters are difficult to set properly.

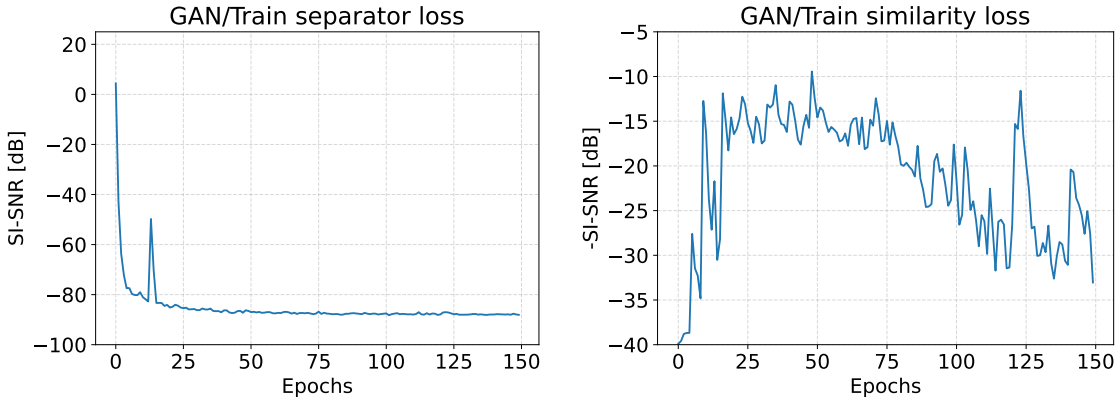
The last two adjustable parameters are the ratio of the augmented mixtures during the separator training r_{aug} and the similarity loss SI-SNR cap c_{sim}, which sets the value that is used to clip the similarity loss to a maximum value. This serves to prevent the similarity function to be too strong in comparison with the separator loss function.

6.4 Initial experiment

The initial experiment is set with following parameters:

- $w_{\text{sep}} = w_{\text{sim}} = 1.0$,
- $c_{\text{sep}} = c_{\text{gen}} = 10$,
- $r_{\text{aug}} = 0.5$ and
- $c_{\text{sim}} = 40$.

The purpose of the initial experiment was to inspect the basic behavior of the loss functions during the training. The results are shown in Figure 6.1. The separator loss function curve shows that the generator network managed to completely confuse the separator network. Although this is the task of the generator, in this case, the generator completely dominated the training to the point that the separator was unable to adapt to the augmented mixtures. The strength of the generator network is possibly caused by:



(a) Training curve of the separator loss function (b) Training curve of the similarity loss function

Figure 6.1: Training curves of the separator and similarity loss computed during the generator training move in each epoch. These curves shows collapse of the GAN model with initial parameters setting to the imbalance state, where the generator is too strong for the separator.

1. Too much emphasis on the separator network confusion, which could be adjusted by the parameters w_{sep} and w_{sim}. These adjustments will be examined in Section 6.5, or
2. Too much training space for the generator network, which could be adjusted by the parameters c_{gen} and c_{sep}. These adjustments will be examined in Section 6.6.

6.5 Generator loss weights

The base experiments lead to the imbalance between the generator and separator networks. There is a chance to solve this imbalance problem by finding the correct weights w_{sep} and w_{sim}.

Therefore, next experiments set different combinations of values of the weight parameters. Thus the separator weight w_{sep} value is reduced by tenths to 0.1 with similarity weight w_{sim} locked at 1.0. This could reduce the generator strength and help to a better system balance.

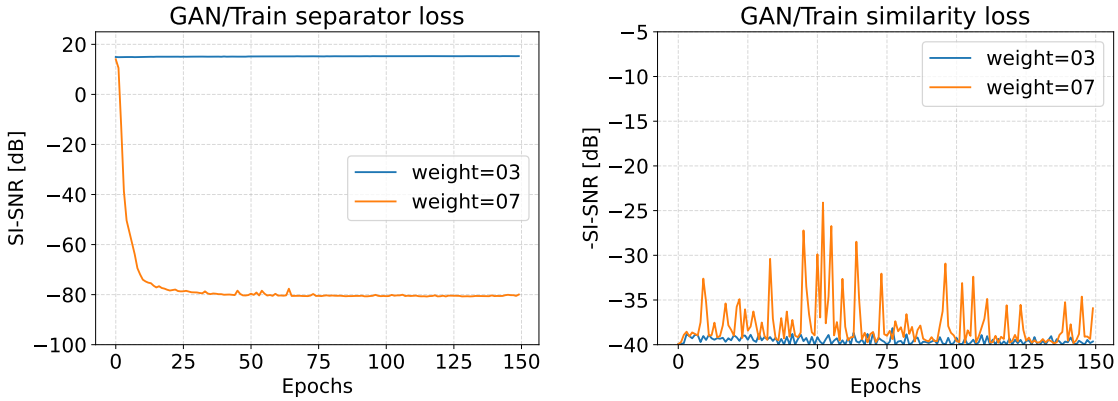
Experiments collapse to two modes, where the generator network:

1. Is too strong and overwhelms the separator network.
2. Generates very similar mixtures to the original ones and does not make any changes.

The first mode is achieved when the c_{sep} is above the value 0.5 as shown by the orange curves in Figure 6.2. Lower values collapse to the second mode, where the similarity loss function has a big influence on the generator as it is shown by the blue curves in Figure 6.2.

6.6 Separator and generator batch caps

Another idea is to solve the collapsing to the first mode by the right combination of the separator and generator batch cap parameter values. The main issue is that the generator



(a) Training curve of the separator loss function (b) Training curve of the similarity loss function

Figure 6.2: Training curves of the separator and similarity loss computed during the generator training move in each epoch. The orange curves show the collapse of the GAN model with $w_{sep} = 0.7$ to the imbalance state, where the generator is too strong for the separator. The blue curves show the collapse of the GAN model with $w_{sep} = 0.3$ to the imbalance state, where the generator does not generate any augmentations.

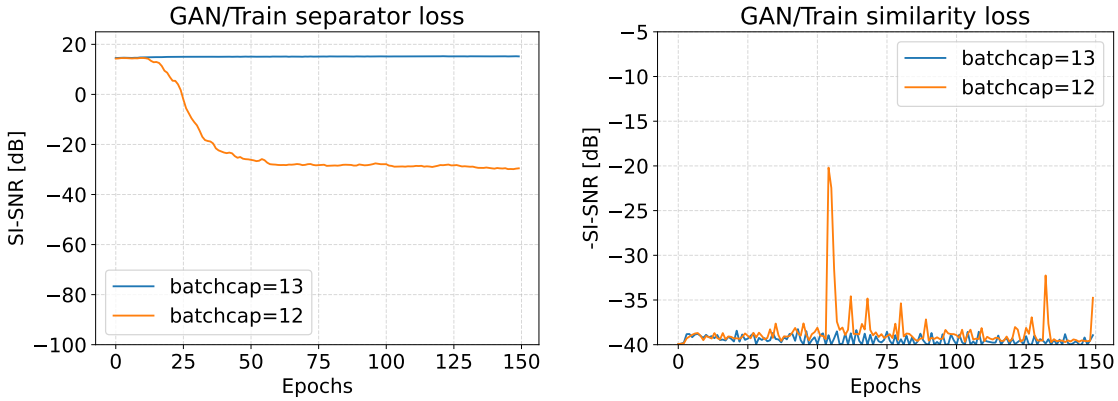
confuses the separator too much. Therefore, increasing the separator batch cap c_{sep} could help the separator to better adapt to the augmented mixtures. The fixed value $w_{\text{sep}} = 0.7$ was used in batch cap experiments. This value has been chosen because although the system with these settings collapses to the first mode, it reduces the impact of the separator loss function on the generator. The c_{sep} value is incremented by the unit. Nevertheless, the system does not stabilize again, it collapses to the second mode when the $c_{\text{sep}} >= 13$ as it is shown by the blue curves in Figure 6.3. With lower values of c_{sep}, the system stays in the first collapse mode as it is shown by the orange curves in Figure 6.3. Therefore, it means that the batch cap parameters are not distinguished finely enough.

6.7 Automatic separator and generator batch caps

After these experiments, it turned out that adjusting the batch caps c_{sep} and c_{sim} is not enough to stabilize the training. The constant batch cap values still lead to one of the two collapse modes described in Section 6.5, i.e. one of the models is too strong while the other does not learn anything. Here, we explore another way to balance the training, where the number of batches for each model is adjusted dynamically, based on the SI-SNR value of the separator. The generator is thus trained until the separator obtains SI-SNR values higher than parameter c_{snrgen} and the separator is trained until the separator does not achieve a -SI-SNR value lower than parameter c_{snrsep} on the augmented mixtures.

Experiments using this method are initially set with parameters:

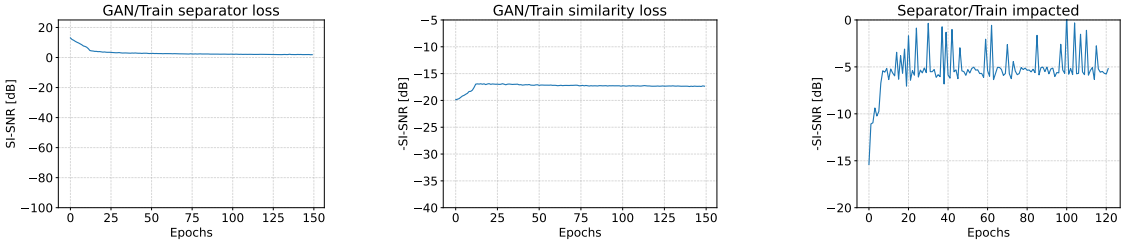
- $w_{\text{sep}} = 0.6, 0.7, 0.9$
- $w_{\text{sim}} = 1.0,$
- $c_{\text{snrgen}} = 0.0$
- $c_{\text{snrsep}} = -5.0,$



(a) Training curve of the separator loss function (b) Training curve of the similarity loss function

Figure 6.3: Training curves of the separator and similarity loss computed during the generator training move in each epoch. The blue curves show the collapse of the GAN model with $c_{sep} = 13$ to the imbalance state, where the generator does not generate any augmentations. The orange curves show the collapse of the GAN model with $c_{sep} = 12$ to the imbalance state, where the generator is too strong for the separator.

- $r_{\text{aug}} = 0.5$ and
- $c_{\text{sim}} = 20$.



(a) Training curve of the separator loss function computed during the generator training.

(b) The training curve of the similarity loss values computed during the generator training.

(c) Training curve of the separator loss function computed on data impacted by the generator during the separator training.

Figure 6.4: The training curve of the separator loss computed during the generator training moves each epoch and the training curve of the computed separator loss function during the separator training on the augmented mixtures. These curves show, that the curves converge to the set parameters c_{snrgen} and c_{snrsep}. The c_{snrsep} parameter value is inverted during the training. Nevertheless, the generated augmented mixtures are very similar to the original ones, thus the generated augmentations are not so strong.

The c_{sim} value follows the knowledge from the previous experiment, where the similarity values around the 40 dB overweight values of the separator loss function during the generator training. From training curves shown in Figure 6.4 it is evident, that systems trained by using this method do no longer collapse to the modes mentioned in Section 6.5. The training

curve in Subfigure 6.4a that stands for the level of the separator confusion converges to the SI-SNR value 0 dB which is set by the parameter c_{snrgen}. The Subfigure 6.4c shows that the training curve of the separator gained values on the augmented data, which converges to the value of -5 dB -SI-SNR set by the parameter c_{snrsep}.

6.8 First evaluation

Figure 6.5: Results from finding the best separator model from trained GAN with parameter $w_{\text{sep}} = 0.7$. The first chart shows the SI-SNR means of evaluated separators. The other charts show the results on generated augmented mixtures of the separator models from the each selected epoch

Since the GAN model does not collapse to the one of the above mention modes it is possible to evaluate it. Firstly, the best-trained separator model has to be found. This is provided by generating augmented mixtures by randomly chosen trained generators. Then

each tenth separator model is evaluated on all generated augmented mixtures. The model with the best result is then chosen for the evaluation using the test data. This is described in more detail in Section 4.3.1. The results of the separator selection are shown in Figure 6.5. The first chart shows the overall results of each evaluated separator model. The bar charts show how successful the separator is on the generated augmentations by each generator. Bars represent groups of generators, which are grouped by their epoch number. From these bars it is evident that the separators works well on the first ten generator but elsewhere gain a poor results. Nevertheless, the results achieved by the different separators on the data generated by the different generators are very similar. This means that the separator is not being more and more robust during the epochs.

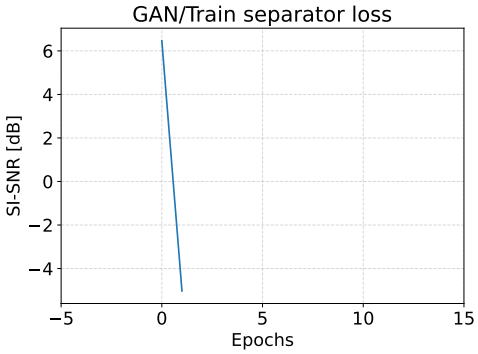
Table 6.3: Results from the first evaluation of the experiments with automated c_{gen} and c_{sep} parameter settings. In first column there are different w_{sep} parameter settings. Other columns represent results from the evaluation on the tested part of the WSJ0-2mix or WHAM dataset. The columns with original annotation contain evaluation results of the raw pretrained separator model on the WSJ0-2mix dataset. The columns with the augmented annotation contain the evaluation results of the best separator model chosen from the GAN training.

	WSJ0-2mix original SI-SNR [dB]	WSJ0-2mix robust SI-SNR [dB]	WHAM original SI-SNR [dB]	WHAM robust SI-SNR [dB]
$w_{\text{sep}} = 0.6$	12.46	12.18	-2.99	-2.89
$w_{\text{sep}} = 0.7$	12.46	11.95	-2.99	-2.78
$w_{\text{sep}} = 0.9$	12.46	11.36	-2.99	-3.05

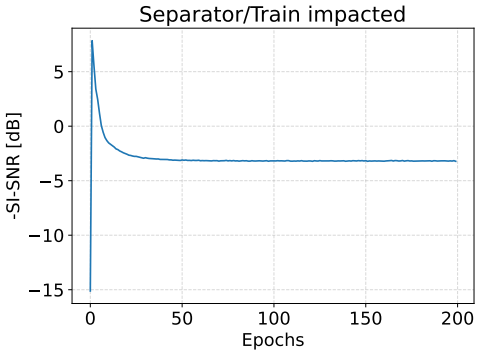
The separator with the best score is chosen as the best one. This separator is then evaluated on the WSJ0-2mix and WHAM datasets. The results are shown in Table 6.3. They show that the trained GAN model with the above-mentioned parameters setting does not train the separator to be more robust. This is tested by evaluating the best separator model on the testing part of the WHAM dataset. The SI-SNR result achieved by the separator model pretrained on the WSJ0-2mix dataset is similar to the model trained by the GAN, when evaluating both the WSJ0-2mix and WHAM datasets. There are three experiments using different separator weights, but none of them has achieved better results.

The bad evaluation results could be caused by the fact that although the GAN model does not collapse to the above mentioned two modes, the generated augmented mixtures are very similar to the original ones as it is shown in Figure 6.4b, where the curve represents the similarity loss values. Thus, it would be necessary to move with the w_{sim} parameter instead of the w_{sep}. The w_{sim} parameter constraints generator from generating very different mixtures, which also constraints the generator's possibility of generating difficult augmentations. Considering this other experiments are set with the fixed w_{sep} at value 1.0 and w_{sim} set to the values $0.1, 0.2, \ldots, 1.0$. These experiments could show that the decreasing w_{sim} parameter may release the generator to generate the more difficult augmented mixtures.

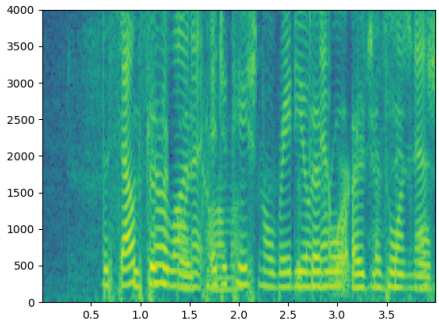
Experiments show that with high values of w_{sim} the generator has the major part of the training time and the separator could adapt to the generated augmentation in a few batches. With decreasing value of the w_{sim} the GAN model reaches the point, where hard augmentation is found in the first few epochs and the separator is then unable to adapt to this augmentation with the SI-SNR value set by the parameter c_{snrsep}. This is shown in



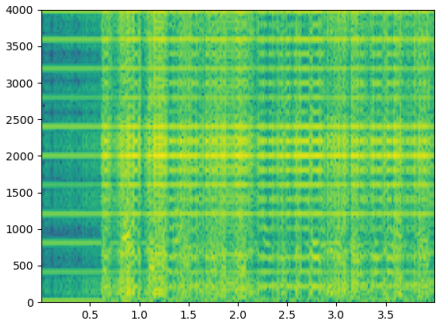
(a) Training curve of the separator loss function during the generator training.



(b) Training curve of the separator loss function on data impacted by the generator during the separator training.



(c) The spectrum of the original not augmented mixture.



(d) Generated augmented mixture spectrum from the original one on the left.

Figure 6.6: The training curve of the separator loss computed during the generator training moves each epoch and the training curve of the computed separator loss function during the separator training on the augmented mixtures. These curves show, that the generator generates, in first epoch, so strong augmentation that could not be adapted by the separator with the SI-SNR result which reaches the set c_{snrsep} value.

Figure 6.6, where the first curve is the separator loss during the generator training, which shows the creation of the strong augmentation in the first two epochs, and the second curve is the separator loss on the generated augmented mixtures during the separator training. The second curve shows that the separator is unable to achieve the value -10 set by the parameter c_{snrsep}. In Figure 6.6 the original and augmented mixture is also shown. From the spectrogram, it is obvious that the separator does not have a chance to adapt. The w_{sim} value that collapse to the strong generator mode is not stable, in some training it is a value 0.7 another training makes this collapse at a value 0.6.

6.9 Instability in the collapse to the strong generator mode

The instability in collapse to the strong generator mode during the training was inspected. The problem is that at the moment when the generator generates the strong augmentation training is switched to the separator training. The separator is then unable to achieve the

required SI-SNR value and does not switch back to the generator training. If the generator training could be switched on, the similarity loss could make the augmentation slightly easier and the separator could try again to adapt. There are a total of two ways to achieve this. The first one is to set the timeout parameter, which switches the training if the separator is unable to reach the required score in the set number of the maximum provided batches (timeout). Although this is an elegant solution it raises another parameter that should be tuned. Thus, in this work, the second solution is used. This solution starts each epoch with the generator training, which forces the generator to be trained even when the separator does not achieve the required score.

6.10 Problem with the proper switching of the generator and separator training

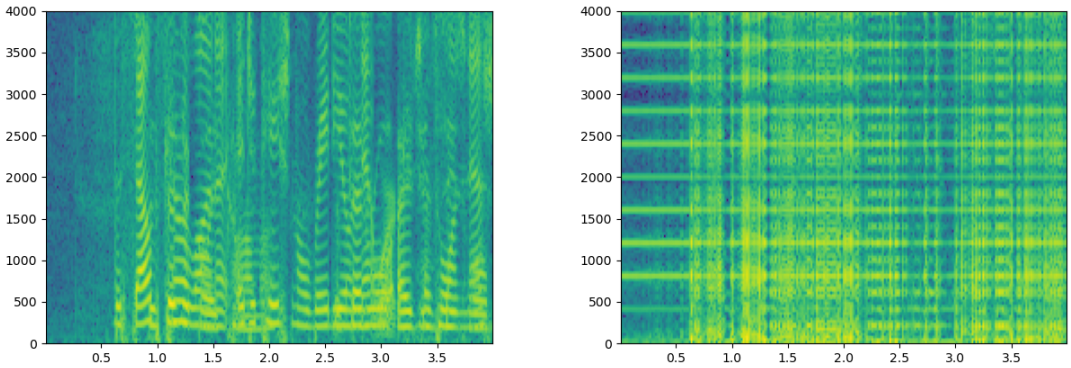
The second problem which arises during the experimentation is that the generator and separator training is not properly switched although they reach the required SI-SNR score. This score is computed as mean from the score of all batches used during each separator and generator training. The score values are reset after the training is switched and other score values computed during the current part of the training are taken into account for the mean computation. Nevertheless, when for example the values start from the 5 and then decrease to the -10, the computed mean is not equal to -10 until the values stabilize at that value. The higher starting value the longer time should the values be stable so that the computed mean will switch the training. This behavior leads to GAN model training that will never switch the training between the generator and separator parts.

The solution is to select only the last few batch results that will be used for the switching decision. The number of the batch results is defined by the parameter m_{window} called the window. Then the median should be computed from the selected values. The median value and the threshold set by the parameter $m_{\text{threshold}}$ creates space around the median value. Only values within this space would be used for the mean computation which reduces the influence of the outliers on the computed mean. The computed mean value is then used for the train switching decision and makes it more stable. This solution is called median filtering.

6.11 Forced generator training and the median filtering

The experiments that force the generator training on the start of each epoch and also use the median filtering during the training switch decision use the following parameters:

- $w_{\text{sep}} = 1.0$,
- $w_{\text{sim}} = 0.1, 0.2, \ldots, 1.0$
- $c_{\text{snrgen}} = -5.0$
- $c_{\text{snrsep}} = -10.0$
- $r_{\text{aug}} = 0.5$
- $c_{\text{sim}} = 20$
- $m_{\text{window}} = 10$



(a) Spectrum of the original mixture.

(b) Spectrum of the generated augmented mixture.

Figure 6.7: Spectrums of the original and generated augmented mixture. These spectrums shows how the generator can augment the mixture in way that it is impossible to adapt to it by the separator. it.

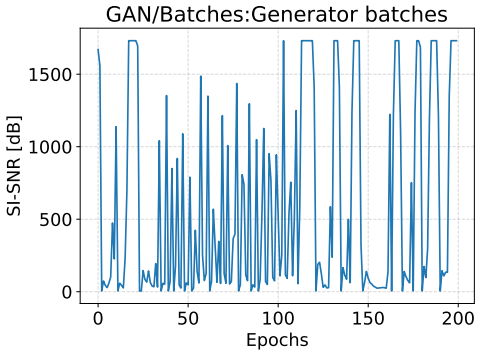
- $m_{\text{threshold}} = 5$

Experiments with the above settings and w_{sim} parameter value higher than 0.7 are unable to achieve the set c_{snrgen} score during the generator training. However, lower w_{sim} parameter values achieves the mentioned score, their similarity loss values are lower than 0 dB. This means that the generated augmented mixtures are very different from the original mixtures. Thus, the generated augmentations are very difficult and the separator is unable to learn anything from them. Figure 6.7 shows spectrum of the original and augmented mixtures with the low w_{sim} parameter value. It is evident that the generated augmented mixture in Subfigure 6.7a is different from the original one, which is in Subfigure 6.7b. To avoid these difficult augmentations the c_{snrgen} parameter value should be set to a higher value. The higher value of this parameter stops the generator training earlier than it could generates very difficult augmented mixtures.

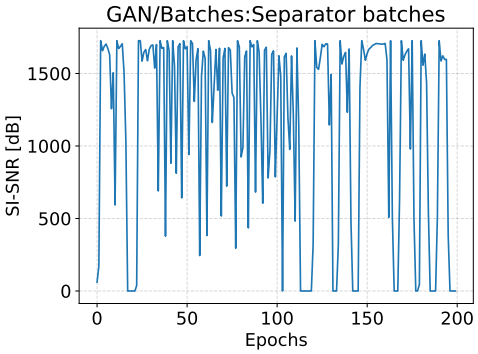
6.12 Higher generator SI-SNR target value

Other experiments are set with the $c_{\text{snrgen}} = 0dB$. Experiments are again set with the w_{sim} parameter values between the 0.1 and 1.0 like in the previous experiments. The separator adaptation works better in these experiments because generator does not confuse it too much.

The experiments with w_{sim} set to the value 0.7 will be described in detail. The experiments have been chosen to give the generator more freedom during the augmented mixtures generation. In Figure 6.8 it is shown how the number of batches for the training of each part is switching during the epochs. It is evident that the generator and separator are finally properly switching the training and do not collapse to any above-mentioned modes. An example of the generated augmented mixture is shown in Figure 6.9. From the spectrum, it is evident that the signal contains some augmentation but at the same time, the original signal is visible. The lines that the generated augmented mixture contains are caused by



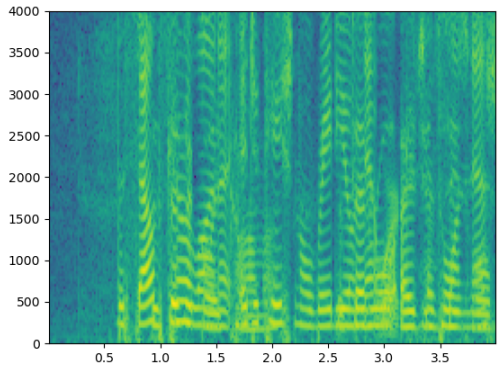
(a) Number of batches provided for generator training during the GAN model training.



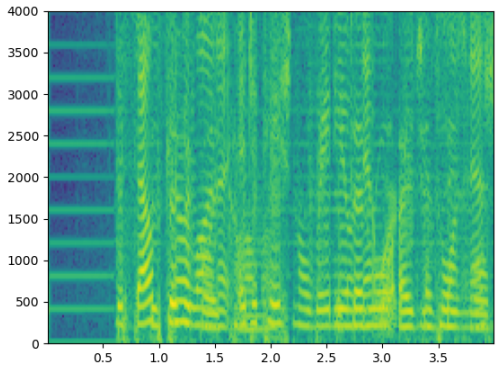
(b) Number of batches provided for separator training during the GAN model training.

Figure 6.8: These figures show how many batches are provided to each part of the GAN model during its training.

the ConvTasNet neural network architecture. They appeared during many experiments. The higher value of w_{sim} suppresses them but they are still there.



(a) Spectrum of the original mixture.

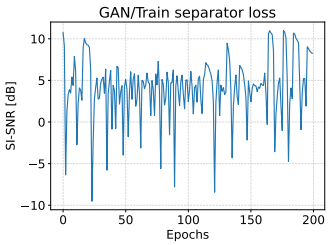


(b) Spectrum of the generated augmented mixture.

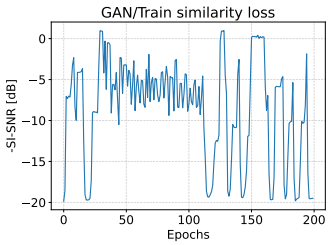
Figure 6.9: Spectrums of the original and generated augmented mixture. These spectrums shows how the generator can augment the mixture in way that the separator can adapt to it.

Fortunately, the separator network can adapt to them as is shown in Figure 6.10 it can be seen that the separator can adapt to the generated augmented mixtures. The left Subfigure 6.10a shows how the generator tries to confuse the separator and it is successful in many epochs, for example in the epoch 15. It is also evident that the generator generally confuses the separator to the 5 dB though the c_{snrgen} value is set to the 0 dB. This is caused by the generator switching the training more times during one epoch and the confused value is computed as the mean of all batches given for the generator training.

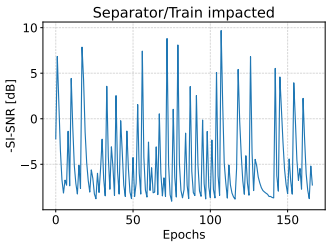
The Subfigure 6.10b shows the similarity loss curve and it is evident how in the middle of the training the similarity value fluctuates around the value of -5 dB, then in the end of



(a) Training curve of the separator loss function computed during the generator training.



(b) The training curve of the similarity loss values computed during the generator training.



(c) Training curve of the separator loss function computed on data impacted by the generator during the separator training.

Figure 6.10: The training curve of the separator loss computed during the generator training moves each epoch and the training curve of the computed separator loss function during the separator training on the augmented mixtures. These curves show, that the curves converge to the set parameters c_{snrgen} and c_{snrsep}. The c_{snrsep} parameter value is inverted during the training. Nevertheless, the generated augmented mixtures are very similar to the original ones, thus the generated augmentations are not so strong.

the training it switches between the very similar generated augmented mixtures and very different ones. From this similarity curve it is evident that although the similarity function helps the GAN model from the collapse to the strong generator mode, it also leads to the states where the generator forget about the learned augmentation. This behaviour will be discussed later.

The Subfigure 6.10c shows the separator loss value during the separator training on the generated augmented mixtures in the other words on the data impacted by the generator. It should be mentioned that the Subfigure 6.10c shows the -SI-SNR value instead of the Subfigure 6.10a. It is evident that the separator can adapt to the generated augmented mixtures and train but it could not achieve the value -10 dB which is set to the parameter c_{snrsep}. Nevertheless, the generator is forced to train at the start of each epoch as it is mentioned in Section 6.9. This rightly rescued the GAN model from the collapse to the strong generator mode.

The best separator selection is shown in Figure 6.11. In contrast with the best separator selection during the first evaluation which is shown in Figure 6.5, the performance of the separator on the different generated mixtures by the different generators is not that stable. Thus, the generator and separator have different strengths during the training. The separator from the epoch 180, and later, fails more than the previous ones. This could be caused by the fact that they are less robust but they can also try to adapt to the more difficult augmentation. This more difficult adaptation could be generated by the generator that also forgets some other augmentation from the past. If the augmentation is so strong the separator could also forget the adaptation to these old augmentations to be able to handle the difficult new one. This raises the generator forgetting problem.

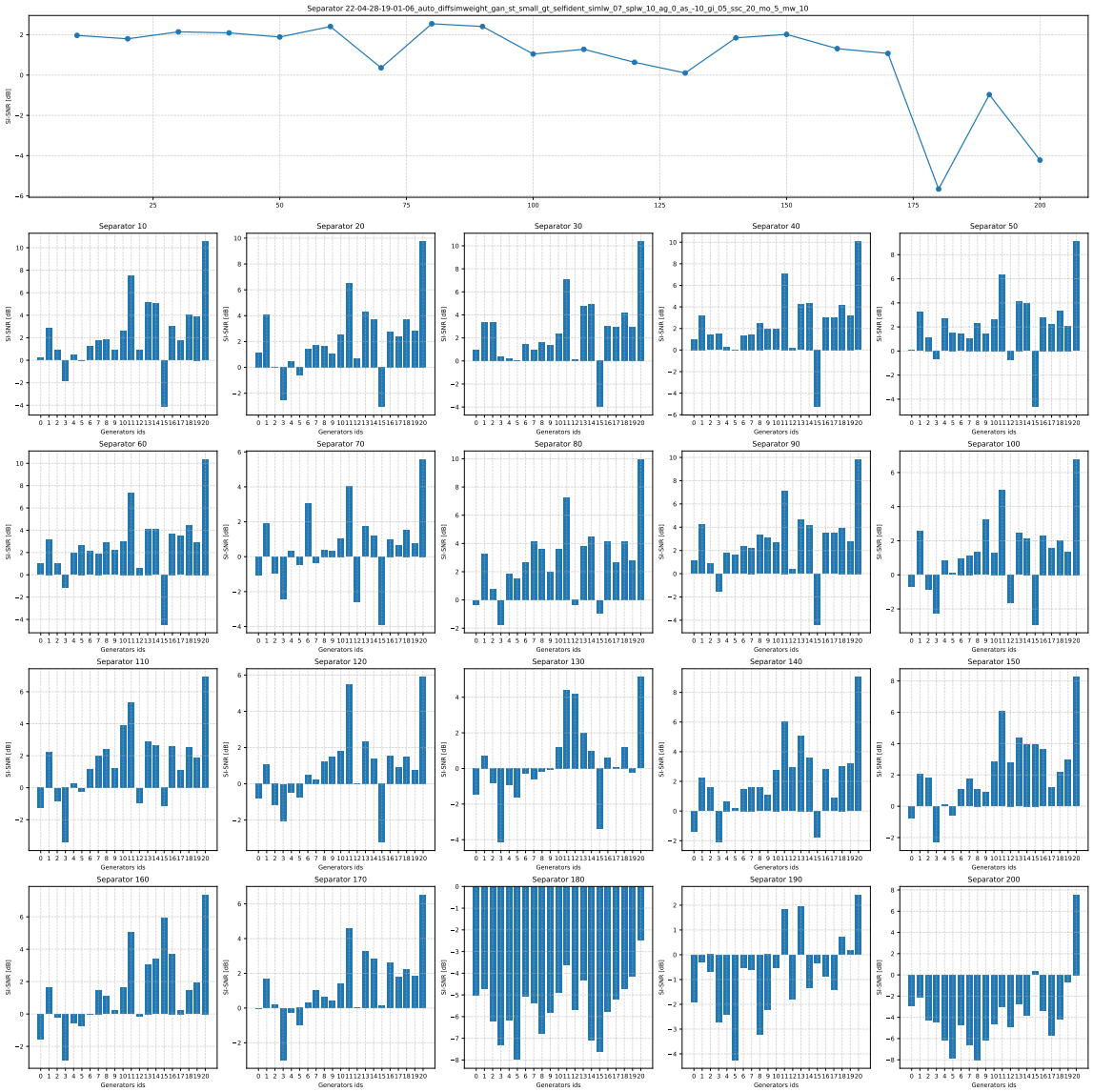


Figure 6.11: Results from finding the best separator model from trained GAN with parameter $w_{\text{sim}} = 0.7$ and used automated training switch with set parameters $c_{\text{snrgen}} = 0\,\text{dB}$ and $c_{\text{snrsep}} = -10\,\text{dB}$. The first chart shows the SI-SNR means of evaluated separators. The other charts show the results on generated augmented mixtures of the separator models from the each selected epoch

6.13 Generator forgetting problem

Using the similarity loss function together with the separator loss function during the generator training prevents the generator from generating total nonsense. With the combination of the forced generator training at the start of each epoch, the similarity also helps to the rescuing the GAN model from the collapse to the strong generator mode. Unfortunately, it also leads the generator to forget the learned augmentations during the training. The generator then learns a new augmentation, but training of the separator is in this GAN model designed to get generated augmented mixtures only from the last generator state.

This leads to the forgetting of the separator. If the newly learned augmentation by the generator is not difficult or it is similar to the previous ones, the separator model still works well and it is getting more robust. Nevertheless, if the new augmentation is different the separator decides that it is better to forget the old augmentation to adapt to this new one. This leads to the not robust separator towards the previous generator states. If the adaptation to these augmentations takes only a few epochs and then the end of the training follows or it is quickly forgotten, the separator will not be selected during the best separator selection. However, the selected separator will be more robust it will probably fail on the augmentations generated by the generators in after forget states.

To solve this problem the GAN model training should be redesigned. Older versions of the generator should be stored and used during the training to generate new augmentations. This should protect the separator from the forgetting adaptation to the previous augmentations and it should be redesigned in future work.

6.14 Evaluation problem

The best separator from each of experiments from Section 6.12 was chosen and then evaluated. The results are shown in Table 6.4. The results show that with the high value of the parameter w_{sim} the robust separator achieves a little bit lower results on the testing part of the WSJ0-2mix dataset in comparison with the pretrained separator model. However the numbers for the WSJ0-2mix dataset are still acceptable, results also show that on the testing part of the WHAM dataset any of the trained separators do not achieve better results than the pretrained separator model. The results are even worse with low value of the parameter w_{sim}.

Table 6.4: Results of the experiments with automated $c_{\text{gen}} = 0\,\text{dB}$ and $c_{\text{sep}} = -10\,\text{dB}$ parameter settings. In first column there are different w_{sim} parameter settings. Other columns represent results from the evaluation on the tested part of the WSJ0-2mix or WHAM dataset. The columns with original annotation contain evaluation results of the raw pretrained separator model on the WSJ0-2mix dataset. The columns with the augmented annotation contain the evaluation results of the best separator model chosen from the GAN training.

w_{sim}	WSJ0-2mix original SI-SNR [dB]	WSJ0-2mix robust SI-SNR [dB]	WHAM original SI-SNR [dB]	WHAM robust SI-SNR [dB]
0.1	12.46	3.01	-2.99	-5.93
0.2	12.46	7.10	-2.99	-5.22
0.3	12.46	9.99	-2.99	-6.46
0.4	12.46	8.91	-2.99	-4.55
0.5	12.46	10.17	-2.99	-3.21
0.6	12.46	10.29	-2.99	-2.20
0.7	12.46	9.67	-2.99	-2.94
0.8	12.46	11.23	-2.99	-2.946
0.9	12.46	10.93	-2.99	-2.35
1.0	12.46	10.09	-2.99	-3.44

Nevertheless, the WHAM dataset which is mentioned in Section 6.1 contains noises recorded in the real environments. The generator is unable to generate these noises, but

it is possible to generate new different noises. The separator that is made robust by this generator could fail on the WHAM dataset but still, be robust towards the generated noises. To evaluate this behavior each mixture in the testing set of the WSJ0-2mix dataset is augmented by the randomly chosen generator from the training. The raw pretrained model is then evaluated on these augmented mixtures and then also on the robust one. The robust separator should perform better on these augmented mixtures than the raw pretrained separator.

Table 6.5: Results of the experiments on WSJ0-2mix dataset with automated $c_{\text{gen}} = 0.0\,\text{dB}$ and $c_{\text{sep}} = -10.0\,\text{dB}$ parameter settings. In first column there are different w_{sim} parameter settings. The second column represents values of the not robust separator pretrained on WHAM dataset. The third column contains results of the robust separator. These results shows the comparison between the not robust and robust separator which should prove the concept of this GAN.

w_{sim}	Augmented WSJ0-2mix original SI-SNR [dB]	Augmented WSJ0-2mix robust SI-SNR [dB]
0.1	-7.68	-3.17
0.2	-6.16	-2.02
0.3	-3.73	-1.69
0.4	-2.69	-0.78
0.5	-1.34	-0.10
0.6	2.15	2.23
0.7	1.99	2.46
0.8	5.15	5.64
0.9	4.22	5.02
1.0	7.12	6.63

The results of this evaluation are shown in Table 6.5. There is a visible improvement between the original and robust separator for all set w_{sim} values except the value of 1.0. Lower values of this parameter give results with a bigger difference between the original and robust separator. These values are under the zero SI-SNR which is caused by the low impact of the similarity loss function during the training. Nevertheless, they show that the GAN model training makes the separator network robust toward the augmented mixtures generated by the generators during the training.

Figure 6.12 shows the improvement in results for the robust separator in experiment with parameter $w_{\text{sim}} = 0.7$. This experiment is described in detail in Section 6.12. In Figure, the improvement is most evident in the augmented mixtures generated by the generators from 0-10, 40-110, 120-130, and also 150-160 epochs. These improvements are shown in two bottom bar charts, where the left bar chart shows the results for the not robust separator and the right bar chart results for the robust separator.

6.15 Experiments on WHAM dataset

Other experiments are performed on the WHAM dataset with the separator pretrained on the WHAM dataset. These experiments should firstly show how much the parameters depend on the current dataset. They should also show the difference in behavior on the already augmented dataset. Experiments are performed with the following parameters:

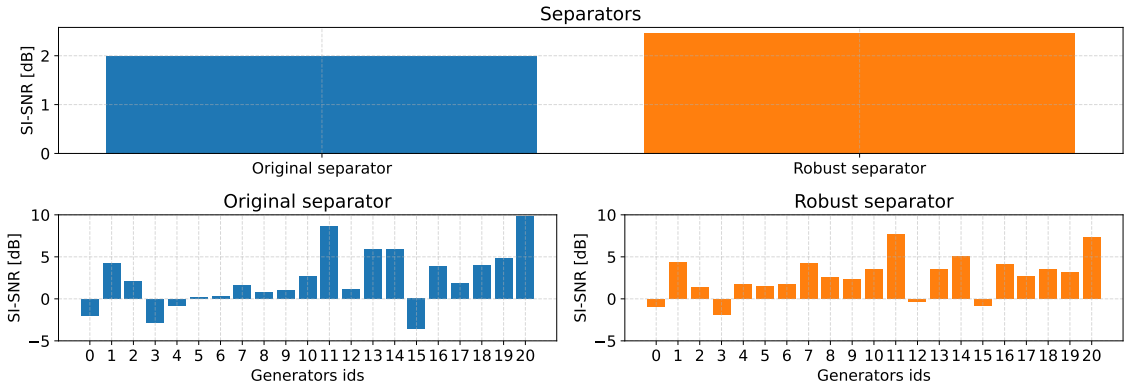


Figure 6.12: Improvement results for the experiment with parameter $w_{\text{sim}} = 0.7$. The upper bar chart shows the results for the not robust (blue) and robust separator (orange). The two bottom bar charts shows how the not robust and robust separators perform on the augmented mixtures generated by the generators from different epochs of the GAN model training. Each bar represents the group of the generators from ten neighboring epochs.

- $w_{\text{sep}} = 1.0$,
- $w_{\text{sim}} = 0.1, 0.2, \ldots, 1.0$
- $c_{\text{snrgen}} = 0.0$
- $c_{\text{snrsep}} = -5.0$
- $r_{\text{aug}} = 0.5$
- $c_{\text{sim}} = 20$
- $m_{\text{window}} = 10$
- $m_{\text{threshold}} = 5$

The parameter setting is similar to the experiments in Section 6.12. Only the parameter c_{snrsep} is changed to the value -5.0 dB (-SI-SNR). This is due to the base results of the separator on the WHAM dataset, where it is possible to achieve 6.09 dB. The experiments with the value -10.0 dB are also run, but they suffer from the problem that the separator is unable to achieve such a high value during the adaptation to the generated augmented mixtures.

The results of these experiments are shown in Table 6.6. These results show the same behavior as the experiments with the WSJ0-2mix dataset. The only difference is in the better results for the WHAM dataset, which is caused by the pretraining of the separator on these data. If the selected separator is more robust than the original separator could be seen in the evaluation which compares the results of these two separator states.

These results are shown in Table 6.7. The improvements are higher than the 4 dB. There are bigger improvements than during the experiments on WSJ0-2mix dataset, where the improvements are around 3 dB or lower.

The GAN model concept for robust separator networks is proved by the experiments to work. The separator chosen from the GAN model training is more robust towards

Table 6.6: Results of the experiments performed on the WHAM dataset with automated $c_{\text{gen}} = 0\,\text{dB}$ and $c_{\text{sep}} = -10\,\text{dB}$ parameter settings. In first column there are different w_{sim} parameter settings. Other columns represent results from the evaluation on the tested part of the WSJ0-2mix or WHAM dataset. The columns with original annotation contain evaluation results of the raw pretrained separator model on the WSJ0-2mix dataset. The columns with the augmented annotation contain the evaluation results of the best separator model chosen from the GAN training.

w_{sim}	WSJ0-2mix original SI-SNR [dB]	WSJ0-2mix robust SI-SNR [dB]	WHAM original SI-SNR [dB]	WHAM robust SI-SNR [dB]
0.1	9.04	8.192	6.09	5.76
0.2	9.04	8.08	6.09	5.44
0.3	9.04	7.77	6.09	5.29
0.4	9.04	7.90	6.09	4.68
0.5	9.04	8.11	6.09	5.42
0.6	9.04	7.96	6.09	5.22
0.7	9.04	8.23	6.09	5.22
0.8	9.04	8.29	6.09	5.42
0.9	9.04	7.69	6.09	4.95
1.0	9.04	5.55	6.09	3.42

Table 6.7: Results of the experiments on WHAM dataset with automated $c_{\text{gen}} = 0.0\,\text{dB}$ and $c_{\text{sep}} = -5.0\,\text{dB}$ parameter settings. In first column there are different w_{sim} parameter settings. The second column represents values of the not robust separator pretrained on WHAM dataset. The third column contains results of the robust separator. These results shows the comparison between the not robust and robust separator which should prove the concept of this GAN.

w_{sim}	Augmented WSJ0-2mix original SI-SNR [dB]	Augmented WSJ0-2mix robust SI-SNR [dB]
0.1	-9.30	-1.42
0.2	-7.06	-1.75
0.3	-6.43	0.78
0.4	-4.36	2.30
0.5	-3.96	3.46
0.6	-4.08	3.09
0.7	-3.44	3.91
0.8	-3.39	4.02
0.9	-3.33	4.01
1.0	-3.26	-2.82

the generated augmentations. This behaviour is evident on the both of tested datasets. Nevertheless, the robust separator from the experiments performed on the WSJ0-2mix dataset does not performs better on the WHAM dataset.

Chapter 7

Conclusion

In this work, the generative adversarial networks (GAN) model is used to make a robust speech separation system. The used model is slightly different from the original one. The discriminator is the neural network for speech separation pretrained on the base speech separation task and also the similarity loss which is represented by the SI-SNR computed between the original and generated augmented mixture.

The GAN model was tuned during the experiments. The model is very sensitive to changes in parameters, but the right setting was found. Experiments show that the concept works and makes the system more robust. The evaluation was set to compare the original pretrained and the robust system results achieved on the testing part of the dataset augmented by random generators from the training. For the model trained on the WSJ0-2mix dataset, there is an improvement around 3 dB SI-SNR and for the model trained on the WHAM dataset, there is an improvement around 4 dB and more. Nevertheless, there is no improvement in the WHAM dataset for the model trained on the WSJ0-2mix dataset. This is caused by the fact that the noises generated by the trained generators are not as sophisticated as the noises added by the WHAM dataset. Nevertheless, the trained speech separation system from the GAN model is still more robust than the original one. The experiments also show that the parameter setting found for the WSJ0-2mix dataset is reusable with only a few changes for the WHAM datasets.

During the experiment, two problems in the concept were found. The first is that the generator and separator are join information about the trained augmentation during the training. This is caused because the separator is trained on the original mixtures and generated augmented mixtures by the last state of the generator network. To solve this problem the generated augmented mixtures from the different previous states of the generator should be also used during the separator training. The second problem is that the generator is limited by the similarity loss function. The limitation is for example generating a flipped signal, changes in tempo, etc. It is not possible to make these changes only for mixtures, because the original single speaker signals will not match. Thus, the solution is to generate augmented original single speaker signals instead of the mixtures.

These changes lead to the rework of the proposed concept to a less or greater extent. The new experiments should be also provided to find the right setting and prove if these suggested solutions work. This could be done in future work. Another dataset could be also used to prove that the parameter settings are reusable for the different datasets with only a few changes.

Bibliography

[1] Abdi, H. and Williams, L. J. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*. Wiley Online Library. 2010, vol. 2, no. 4, p. 433–459.

[2] Adler, J. and Lunz, S. Banach wasserstein gan. *Advances in Neural Information Processing Systems*. 2018, vol. 31.

[3] Antipov, G., Baccouche, M. and Dugelay, J.-L. Face aging with conditional generative adversarial networks. In: IEEE. *2017 IEEE international conference on image processing (ICIP)*. 2017, p. 2089–2093.

[4] Bahmaninezhad, F., Wu, J., Gu, R., Zhang, S.-X., Xu, Y. et al. A comprehensive study of speech separation: spectrogram vs waveform separation. *ArXiv preprint arXiv:1905.07497*. 2019.

[5] Bao, J., Chen, D., Wen, F., Li, H. and Hua, G. CVAE-GAN: fine-grained image generation through asymmetric training. In: *Proceedings of the IEEE international conference on computer vision*. 2017, p. 2745–2754.

[6] Berthelot, D., Schumm, T. and Metz, L. Began: Boundary equilibrium generative adversarial networks. *ArXiv preprint arXiv:1703.10717*. 2017.

[7] Frid Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J. et al. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*. Elsevier. 2018, vol. 321, p. 321–331.

[8] Goodfellow, I., Pouget Abadie, J., Mirza, M., Xu, B., Warde Farley, D. et al. Generative adversarial nets. *Advances in neural information processing systems*. 2014, vol. 27.

[9] Goodfellow, I., Pouget Abadie, J., Mirza, M., Xu, B., Warde Farley, D. et al. Generative adversarial nets. *Advances in neural information processing systems*. 2014, vol. 27.

[10] Gu, J., Im, D. J. and Li, V. O. Neural machine translation with gumbel-greedy decoding. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018, vol. 32, no. 1.

[11] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems*. 2017, vol. 30.

[12] 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.

[13] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*. MIT Press. 1997, vol. 9, no. 8, p. 1735–1780.

[14] Hui, J. *Gan - why it is so hard to train generative adversarial networks!* Medium, Oct 2019. Available at: https://jonathan-hui.medium.com/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b.

[15] Johnson, K. Vocal tract length normalization. *UC Berkeley PhonLab Annual Report*. 2018, vol. 14, no. 1.

[16] Kim, C., Shin, M., Garg, A. and Gowda, D. Improved Vocal Tract Length Perturbation for a State-of-the-Art End-to-End Speech Recognition System. In: *Interspeech*. 2019, p. 739–743.

[17] Lucic, M., Kurach, K., Michalski, M., Gelly, S. and Bousquet, O. Are gans created equal? a large-scale study. *Advances in neural information processing systems*. 2018, vol. 31.

[18] 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.

[19] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z. et al. Least squares generative adversarial networks. In: *Proceedings of the IEEE international conference on computer vision*. 2017, p. 2794–2802.

[20] McPheron, B., Cintorino, K., Benoit, N., Hasan, A., Oliveira, K. et al. The Use of Digital Reverberation Projects to Teach Audio Signal Processing. In:. October 2015.

[21] Odena, A., Olah, C. and Shlens, J. Conditional image synthesis with auxiliary classifier gans. In: PMLR. *International conference on machine learning*. 2017, p. 2642–2651.

[22] 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., Alché 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.

[23] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T. and Efros, A. A. Context encoders: Feature learning by inpainting. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 2536–2544.

[24] Radford, A., Metz, L. and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *ArXiv preprint arXiv:1511.06434*. 2015.

[25] Stone, J. V. Independent component analysis: an introduction. *Trends in cognitive sciences*. Elsevier. 2002, vol. 6, no. 2, p. 59–64.

[26] Suárez, P. L., Sappa, A. D. and Vintimilla, B. X. Infrared image colorization based on a triplet dcgan architecture. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, p. 18–23.

[27] Wang, Y.-X., Girshick, R., Hebert, M. and Hariharan, B. Low-shot learning from imaginary data. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, p. 7278–7286.

[28] Wichern, G., Antognini, J., Flynn, M., Zhu, L. R., McQuinn, E. et al. WHAM!: Extending Speech Separation to Noisy Environments. In: *Proc. Interspeech*. September 2019.

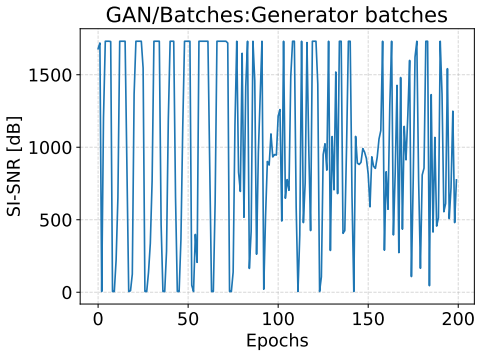
[29] Wu, J. *FUNCWJ/Conv-tasnet: A pytorch implementation of „Tasnet: Surpassing ideal time-frequency masking for speech separation“ (see recipes in APS Framework https://github.com/funcwj/aps)*. Dec 2018. Available at: https://github.com/funcwj/conv-tasnet.

[30] Yoshioka, T., Sehr, A., Delcroix, M., Kinoshita, K., Maas, R. et al. Making machines understand us in reverberant rooms: Robustness against reverberation for automatic speech recognition. *IEEE Signal Processing Magazine*. IEEE. 2012, vol. 29, no. 6, p. 114–126.

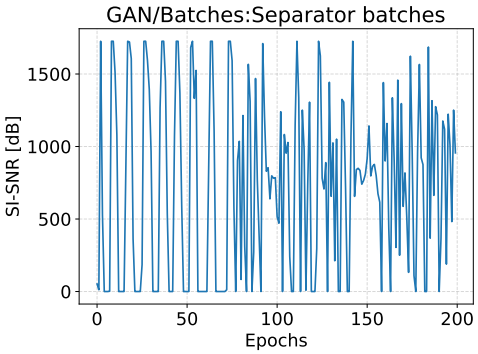
[31] Yu, D., Kolbæk, M., Tan, Z.-H. and Jensen, J. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In: IEEE. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, p. 241–245.

Appendix A

Experiments using WHAM dataset

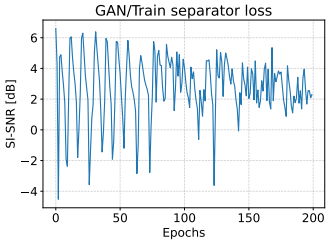


(a) Number of batches provided for generator training during the GAN model training.

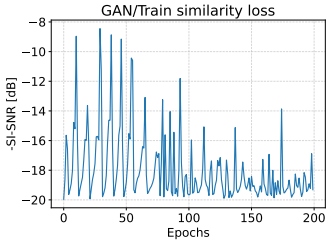


(b) Number of batches provided for separator training during the GAN model training.

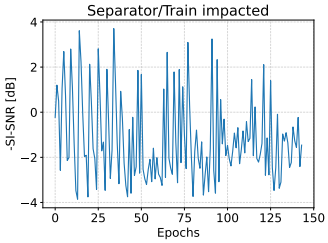
Figure A.1: These Figures show how many batches are provided to each part of the GAN model during its training on WHAM dataset with parameter $w_{\text{sim}} = 0.7$ and used automated training switch with set parameters $c_{\text{snrgen}} = 0\,\text{dB}$ and $c_{\text{snrsep}} = -5\,\text{dB}$.



(a) Training curve of the separator loss function computed during the generator training.

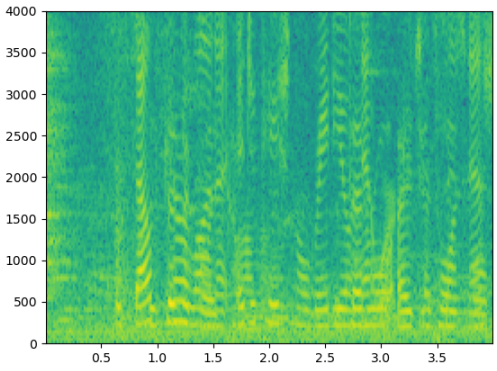


(b) The training curve of the similarity loss values computed during the generator training.

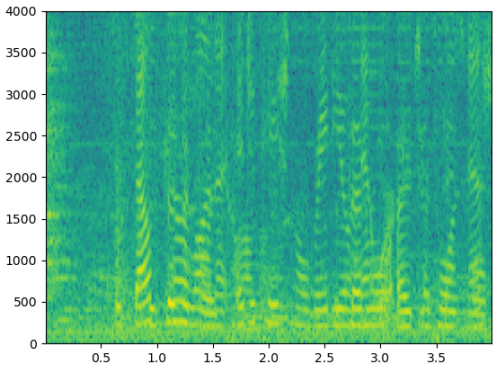


(c) Training curve of the separator loss function computed on data impacted by the generator during the separator training.

Figure A.2: These Figures show the process of the GAN model training on the WHAM dataset with parameter $w_{\text{sim}} = 0.7$ and used automated training switch with set parameters $c_{\text{snrgen}} = 0\,\text{dB}$ and $c_{\text{snrsep}} = -5\,\text{dB}$.. Firstly the training curve of the separator loss computed during the generator training moves each epoch then similarity loss during the generator training and the training curve of the computed separator loss function during the separator training on the augmented mixtures.



(a) Spectrum of the original mixture.



(b) Spectrum of the augmented mixture.

Figure A.3: These Figures show the example of the generated augmented mixture during the GAN model training on the WHAM dataset.

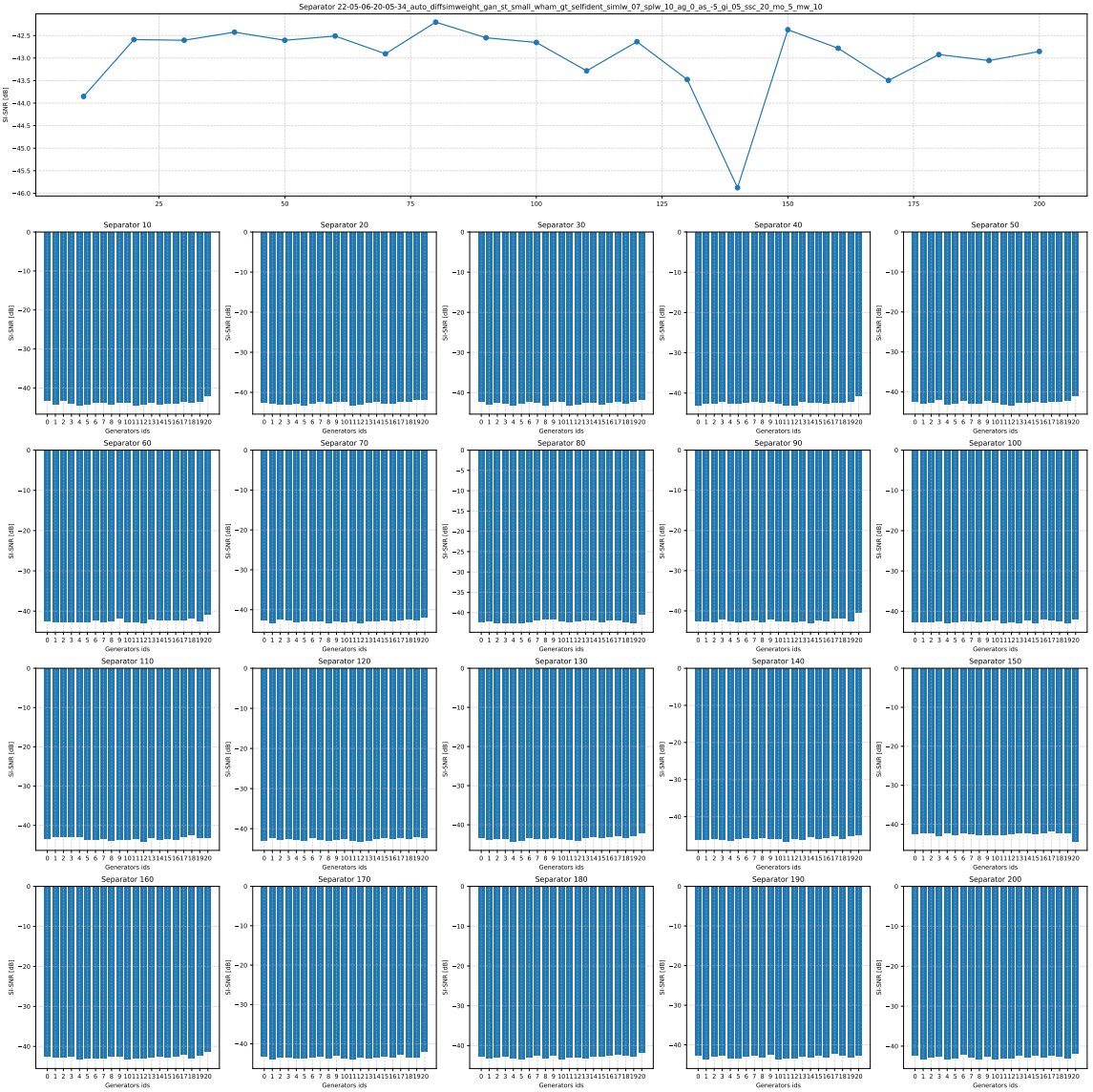


Figure A.4: Results from finding the best separator model from trained GAN on the WHAM dataset with parameter $w_{\text{sim}} = 0.7$ and used automated training switch with set parameters $c_{\text{snrgen}} = 0\,\text{dB}$ and $c_{\text{snrsep}} = -5\,\text{dB}$. The first chart shows the SI-SNR means of evaluated separators. The other charts show the results on generated augmented mixtures of the separator models from the each selected epoch.

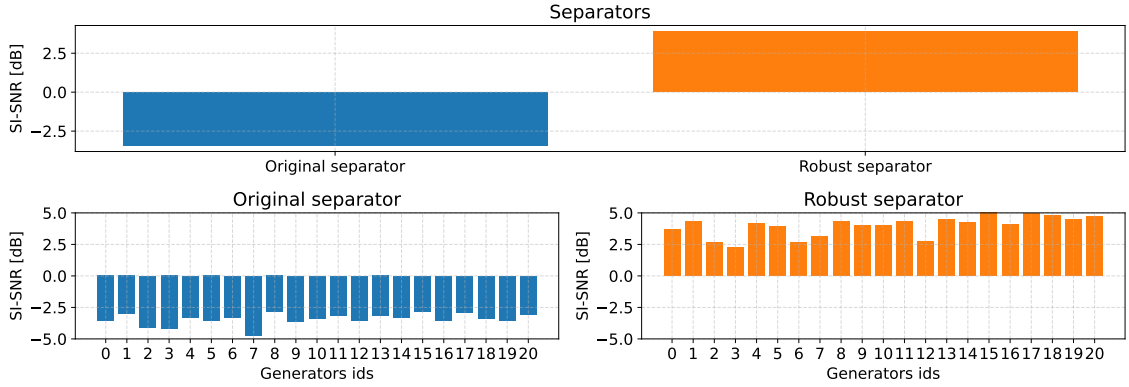


Figure A.5: Improvement between the robust separator and the original one tested on the testing part augmented by the random generators from the GAN model training on WHAM dataset with parameter $w_{\text{sim}} = 0.7$ and used automated training switch with set parameters $c_{\text{snrgen}} = 0\,\text{dB}$ and $c_{\text{snrsep}} = -5\,\text{dB}$. The top Figure shows the mean achieved by each separator. The bottom Figures shows how the each separator works on the augmentations from the different generators.