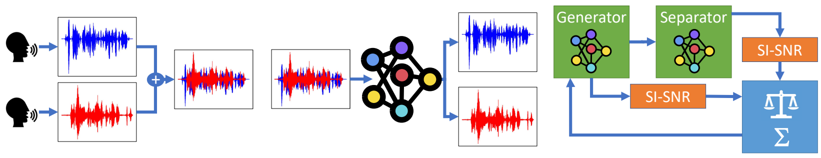
BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2022
http://excel.fit.vutbr.cz

Using generative adversarial networks to make robust speech separation systems

Bc. Ján Pavlus



Abstract
Speech separation is a task of separating single signals from the given mixture of multiple speakers. Speech separation systems are trained on artificial mixtures generated from single speaker's signals. These signals are then used as targets for the training. Neural networks trained this way 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 for example noise addition. Nevertheless, the power of these augmentations is limited as they have to be manually designed. Using generative adversarial networks (GAN) could improve this process by generating augmentations for data depending on the success of confusing the separation system using these data. Speech separation could be then made more and more robust with each generator and separator training step.
This paper describes experiments that are used to find the right parameters and their combination for the GAN model training. Although the experiments do not yet lead to a more robust speech separation, they provide an analysis of the pitfalls of training the GAN, which is the necessary first step towards a successful system. These experiments show that training the GAN model to the stable state is difficult by adjusting the exact number of batches, after which the separator and generator training is switched. On the other hand, adjusting the to-be-achieved scores of the generator or separator training move could work much better and train the GAN model properly. Other experiments have to be done to prove the correctness of these parameters and their settings.

Keywords: speech-separation — GAN — robust — adversarial augmentations

Supplementary Material: Downloadable Code

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

1. Introduction

Speech separation systems are useful as a 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. However, as speech separation systems today are based on neural networks, they need to be trained on the mixed signals for which single signals are well known. 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.

The more robust speech separation system could be achieved by using different data augmentations on the training data. Classic data augmentations are performed by well-known methods, that are more described in Section 3. The disadvantage of these methods is that they do not cover all possible augmentations and each new augmentation needs to be manually designed. 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 paper, a modified version of the generative adversarial network 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 [1] architecture (with different parameters) has been used.

2. Speech separation using neural networks

This section introduces the speech separation task and the way neural networks are used to tackle it.

2.1 Speech separation task

The speech separation task could be 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. More formally, there is a mixture defined as:

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

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 $s_{t,n}$.

In the past, there were attempts to solve this task with classical methods such as principal component analysis [2] or independent component analysis [3]. These classical methods usually work well when the task is greatly simplified, but they fail when silent blocks, echoes, and delays are present. Nowadays neural networks are used for speech separation tasks and they work well. The most used neural network architectures are convolutional neural networks and recurrent neural networks.

2.2 Training neural networks for speech separation

Neural network is used to estimate the single signals from the given mixture. During training, estimated signals are compared with the known original ones using the scale-invariant signal-to-noise-ratio (SI-SNR) function [4], which is defined as:

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

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

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

where $\hat{\vec{s}} \in \mathbb{R}^{1\times T}$ is the estimated source. $\vec{s} \in \mathbb{R}^{1\times T}$ is the original source signal used as the target. 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.

The neural network estimates the N separated signals from the given mixture to the N outputs. Nevertheless the neural network can output the signals of individual speakers in any order. This gives rise to a permutation problem.

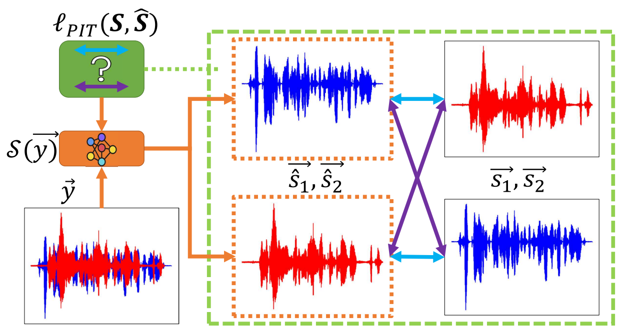


Figure 1. Example of permutation invariant training (PIT) with two estimated outputs by neural network in the orange dotted boxes and two ground truth targets on the right from them. PIT compares estimations and ground truth in all permutations and chooses the permutation with the best result.

The solution is the permutation invariant training (PIT) method [5] shown in Figure 1. 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:

$$\hat{\vec{s}}_1, \hat{\vec{s}}_2, \cdots, \hat{\vec{s}}_N = \mathscr{S}(\vec{y}) \tag{5}$$

$$l_{PIT}(\mathbf{S}, \hat{\mathbf{S}}) = \min_i \sum_{j=1}^{N} -\text{SI-SNR}(\vec{s}_{\sigma_{i,j}}, \hat{\vec{s}}_j) \tag{6}$$

where $\mathscr{S}(\vec{y})$ 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 $\mathbf{S}$.

3. Robust speech separation

To train the speech separation system on real-world mixtures is really difficult as it is hard to obtain single speech signals from them. Therefore, the artificial mixtures are used instead with known single speech signals.

Using solely artificial mixtures during training usually leads to worse system results on real-world mixtures. Real-world mixtures contain echoes, noises, and reverberations obtained from real-world spaces like concert halls, public places, congress rooms, airports, etc. The neural network does not experience all the variety during training, so it often leads to improper separation of the speech signals.

To prevent the neural network from confusion there are methods to make the speech separation system more robust. The common way is to use data augmentations for the training data. There are many classical data augmentations [6] such as polarity inversion augmentation, frequency filters, decreasing or increasing the volume, adding noises, adding reverberation etc.

3.1 Generative adversarial networks

The above mentioned data augmentation methods are well known and they help the speech separation system to work better on real-world data. These methods, however, can not cover all the features of real-world mixtures, so there is a potential to use generative adversarial networks (GAN).

The principle of a GAN models is to use two neural networks, where the first one is used as the generator and the second one is used as the discriminator. These two networks are then trained by playing a min-max game against each other. This principle is slightly modified in this paper. There are also two neural networks: the first one is the generator network, which takes an artificial mixture signal as input and provides an augmented signal, while the second network is the speech separation neural network which should be made robust. The basic idea is to make speech separation more robust epoch by epoch by generating better and better-generated data augmentations. These generated augmentations are constrained by maximizing the similarity between the original mixture and the augmented mixture.

The training of GAN model consists of two steps:

1. Generator training, which is shown in Figure 2. The weights of the separator network are locked. Mixtures are given to the generator network, which generates the augmented mixtures on the output. Augmented mixtures are then submitted to a separator network, which provides separated signals. The l_{PIT} loss function which is defined in Equitation 6 is then computed between the separated signals and the target ones while the generator is trained to maximize it. In parallel, the SI-SNR is computed between the augmented mixtures and the original ones, which gives a similarity value the generator tries to maximize. This is necessary to prevent the generator from completely destroying the information in the mixture. The loss value used for training of the generator network is computed as a weighted sum of the similarity value and the separator loss value, which is defined as:

$$\vec{g} = \mathscr{G}(\vec{y}) \tag{7}$$

$$x_{\text{sim}} = -\text{SI-SNR}(\vec{y}, \vec{g}) \tag{8}$$

$$x_{\text{sep}} = l_{pit}(\mathbf{S}, \mathscr{S}(\vec{g})) \tag{9}$$

$$l_{\text{gen}}(\vec{g}, \vec{y}, \mathbf{S}) = -w_{\text{sep}} * x_{\text{sep}} + w_{\text{sim}} * x_{\text{sim}} \tag{10}$$

where $\vec{g}$ is the generated augmented mixture by the generator function $\mathscr{G}(\vec{y})$, which takes mixture $\vec{y}$ as an input. The x_{sim} is the similarity value computed by the SI-SNR function between the generated augmented mixture $\vec{g}$ and the original mixture $\vec{y}$. The x_{sep} is the value computed by the PIT loss function between the target signals in the matrix $\mathbf{S}$ and the separated signals by the separator function $\mathscr{S}(\vec{g})$, that takes the generated augmented mixture $\vec{g}$ as an input. Then the generator loss function $\ell_{PIT}(\vec{g}, \vec{y}, \mathbf{S})$, which takes the generated augmented mixture $\vec{g}$, the original mixture $\vec{y}$ and the matrix of the target signals $\mathbf{S}$ as an input is computed by the weighted sum of the similarity x_{sim} and the x_{sep} values. The weights are set by the parameters w_{sep}, which sets the importance of the separator loss, and w_{sim}, which sets the importance of the similarity value in the generator loss function.

2. Separator training, which is shown in Figure 3. The weights of the generator network are locked in this step. The separator neural network is trained in the classical speech separation way, but it receives a certain amount of augmented mixtures, for example, 50%. The ratio of the augmented mixtures will be denoted as r_{aug}. In this step, the separator network is trained to handle the augmented mixtures to be more robust.

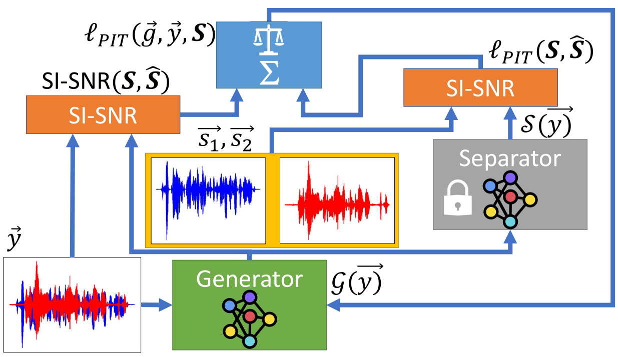


Figure 2. Architecture of generative adversarial network training used in this paper. This figure shows the step where the generator is trained.

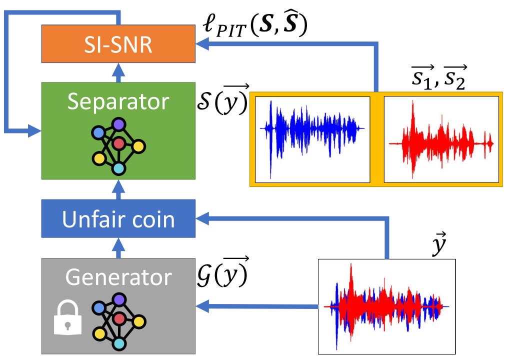


Figure 3. Architecture of generative adversarial network training used in this paper. This figure shows the step where the separator is trained.

3.2 Problems with generative adversarial networks

There are several problems with training GAN models [7]. The first of them is non-convergence, which means that the GAN weights oscillate and never converge to the one best state. The second problem is mode collapse, where GAN is collapsing to the few generating modes. For example, the model which is trained to generate numbers only generates numbers two and five. Another problem is called diminished gradient. In this problem, the discriminator gets too successful so the generator is unable to learn anything. In this paper, we experienced a problem with the imbalance between the separator and generator networks. The imbalance problem and the problem of finding the right parameters are described in Section 4 in a more detailed manner.

Another problem that appears when training the separator together with the generator is the right choice of the best model. In the classical neural networks training methods, the best model is chosen depending on the best cross-validation loss value during the training. This is not applicable in this paper. It is necessary to find the right separator model, which works well on both the original and the augmented data. The generated augmentations are changing during the training due to the changes in the generator. Therefore, it is not possible to compare the validation loss values through separator models from different epochs. This causes a problem with the best separator network selection.

The solution is to save generator and separator models from each epoch during training. They can be used to generate augmented mixtures by randomly choosing a generator from a uniform distribution for each mixture of the cross-validation set. Thus augmented mixtures now represent all possible augmentations learned during training. Now it is necessary to choose the right separator, which will be the most robust one. This task could be achieved by evaluating each separator on mentioned augmented mixtures. The separator with the best evaluation result should be the most robust one.

4. Experiments

Experiments with a generative adversarial network 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.

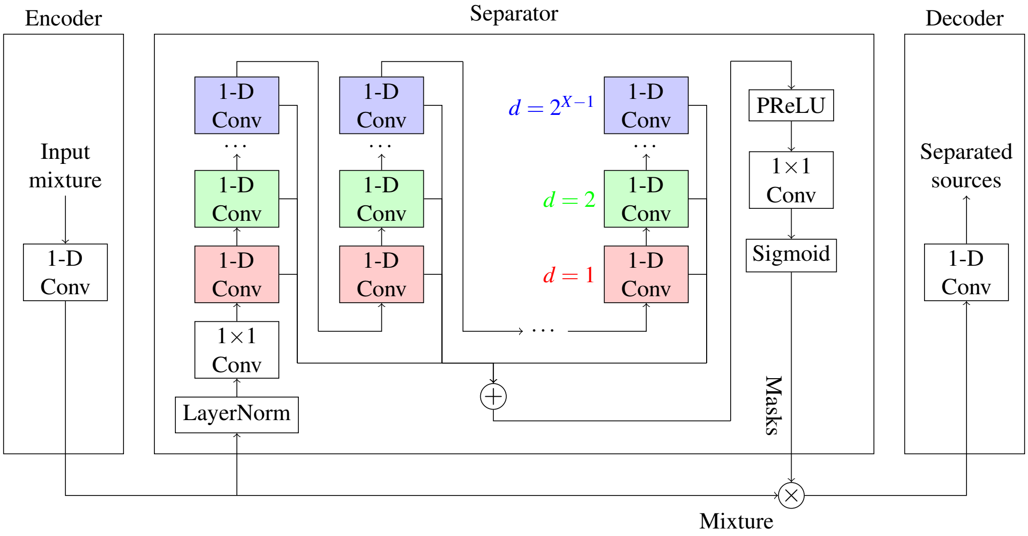


Figure 4. ConvTasNet neural network architecture. Image adapted from [1]

4.1 Dataset

Experiments are provided on the Wall Street Journal dataset (WSJ) [8]. 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. 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 second dataset used in the experiment is WSJ0 Hipster Ambient Mixtures (WHAM) [9]. This dataset pairs each two-speaker mixture in the WSJ dataset with the unique noise background scene. The sizes of training, cross-validation, and testing parts are the same as in the raw WSJ dataset.

4.2 ConvTasNet

The neural network architecture used for both networks (separator and generator) is ConvTasNet [10]. This architecture consists of three parts as it is shown in Figure 4.

The first part is the encoder which consists of one convolutional layer. This layer takes the mixture signal as an input and provides pseudo short-time Fourier transform (STFT) transformation of the given signal. An encoded signal is then given to the second part of the architecture.

This part is called separator and provides separation as the name reveals. The separation part consists of the blocks of the convolutional layers that are applied to the larger and larger context. The output of this part is a series of masks for each speaker. These masks are applied to the encoded signal and given to the last part, decoder, as an input.

The decoder part consists of one convolutional layer like the encoder part. The task of this part is to reassemble the signal from the encoded pseudo STFT format. Parameters of the ConvTasNet are shown in Table 1. The table also shows the parameter values for the generator and separator neural networks used in the experiments.

4.3 Initial separator and generator networks setups

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 2. From the given results it is obvious that the WHAM dataset is much more difficult than the WSJ. This is caused by the noises added to the mixtures. If the system is trained on the WSJ and tested on the WHAM, then the results are quite poor.

The generator network is much smaller than the separator one and it has been pretrained for the self-identity task. Since the aim of the GAN training is not to train the encoder and decoder parts of the ConvTasNet, pretraining the self-identity task removes the encoder and decoder training problem.

4.4 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:

Table 1. Hyperparameters of the ConvTasNet network [1]

Symbol	Description	Generator	Separator
F	Number of filters in autoencoder	128	128
L	Length of the filters (in samples)	40	40
B and the residual paths' 1×1-*conv* blocks	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

Table 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.

	WSJ	WHAM
WSJ	12.46	-2.99
WHAM	9.04	6.09
WSJ + WHAM	12.34	6.45

- 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 10. 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 results.

4.5 Initial experiment

The initial experiment is set with following parameters:

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

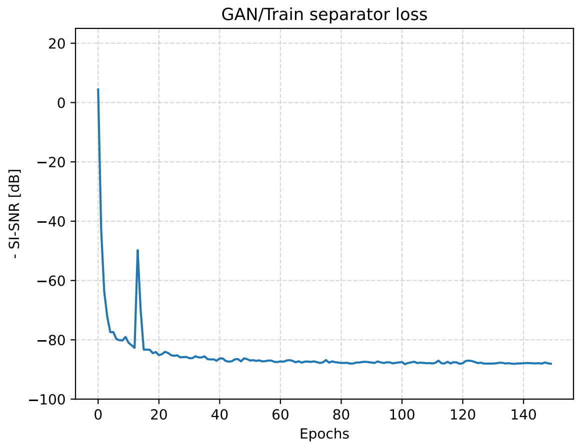
The purpose of the initial experiment was to inspect the basic behavior of the loss functions during the training. The results are shown in Figure 5. The separator loss function curve shows that the generator network managed to completely confuse the separator network. Although this is the task of the 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:

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 4.6, 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 4.7.

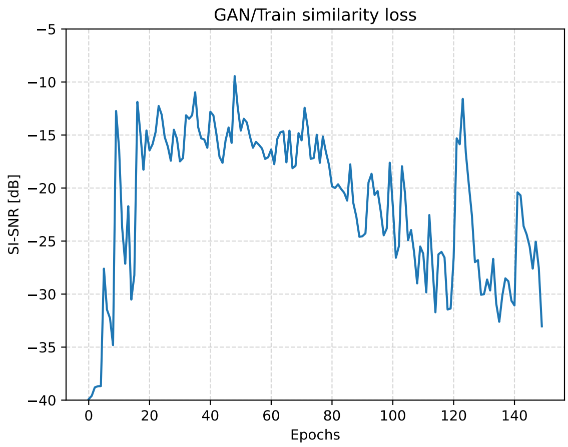
4.6 Generator loss weights

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

Therefore, other experiments are set with 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

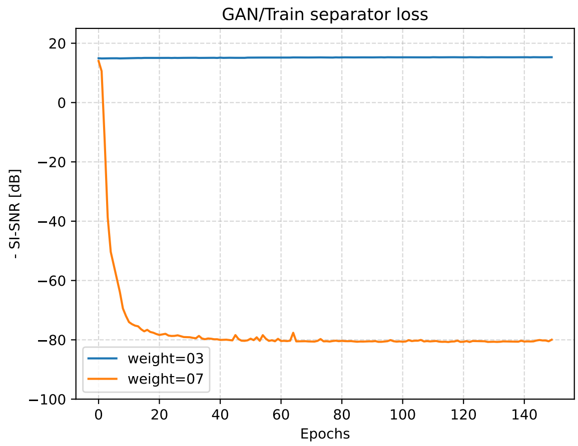


(a) Training curve of the separator loss function

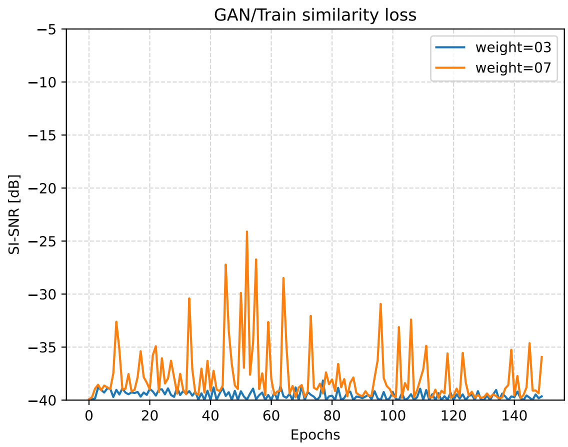


(b) Training curve of the similarity loss function

Figure 5. Training curves of the separator and similarity loss computed during the generator training move 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.



(a) Training curve of the separator loss function



(b) Training curve of the similarity loss function

Figure 6. Training curves of the separator and similarity loss computed during the generator training move each epoch. These 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.

could reduce the generator strength and help to a better system balance.

Experiments collapse to two modes, where the generator network:

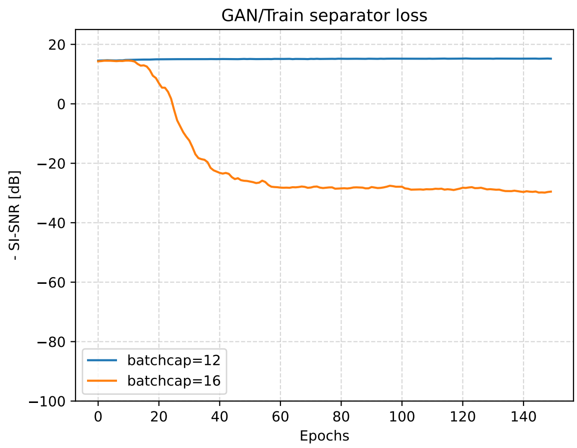
1. Is too strong and overwhelms the separator network, or
2. Generates very similar mixtures to the original ones and does not make any changes.

The first mode is achieved when the c_{sep} is above the value 0.5 as shown by the orange curves in Figure 6. 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.

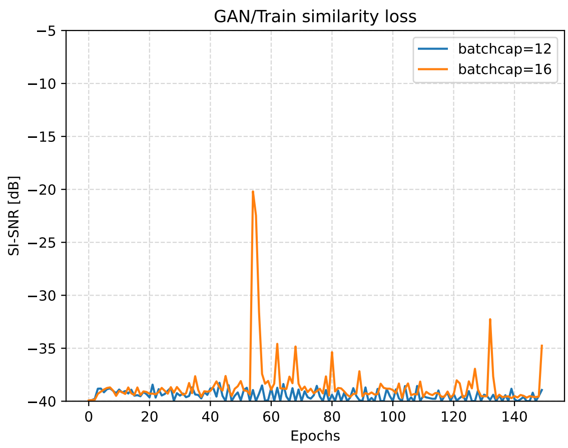
4.7 Separator and generator batch caps

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 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 7. With lower values of c_{sep}, the system stands in the first collapse mode as it is shown by the orange curves in Figure 7. Therefore, it means that the batch cap parameters are not distinguished finely enough.



(a) Training curve of the separator loss function



(b) Training curve of the similarity loss function

Figure 7. Training curves of the separator and similarity loss computed during the generator training move each epoch. The blue curves show the collapse of the GAN model with $c_{sep} = 12$ to the imbalance state, where the generator does not generate any augmentations. The orange curves show the collapse of the GAN model with $c_{sep} = 13$ to the imbalance state, where the generator is too strong for the separator.

4.8 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 4.6, 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 gains loss values higher than parameter c_{snrgen} and the separator is trained until the separator does not achieve a loss value higher than parameter c_{snrsep} on the augmented mixtures.

Experiments using this method are initially set with parameters:

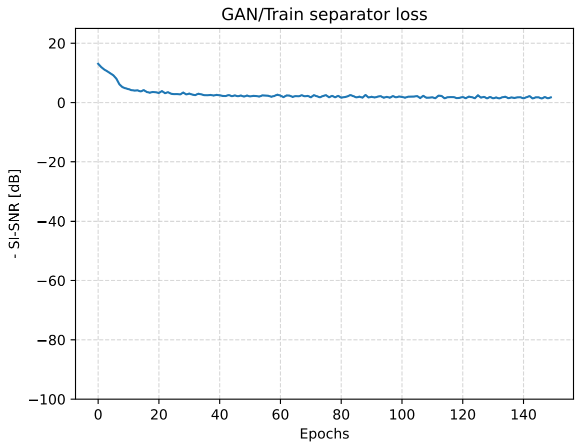
- $w_{\mathrm{sep}} = 0.6, 0.7, 0.9$
- $w_{\mathrm{sim}} = 1.0,$
- $c_{\mathrm{snrgen}} = 0$
- $c_{\mathrm{snrsep}} = 5.0,$
- $r_{\mathrm{aug}} = 0.5$ and
- $c_{\mathrm{sim}} = 20.$

The c_{sim} value follows the knowledge from the previous experiment, where the similarity values around the $40dB$ overweight values of the separator loss function during the generator training. These experiments use the pretrained separator model on the WSJ dataset. From training curves shown in Figure 8 it is evident, that systems trained by using this method do no longer collapse to the modes mentioned in Section 4.6. The training curve that stands for the level of the separator confusion converges to the value set by the parameter c_{snrgen}. The second chart shows that the training curve of the separator gained values on the augmented data, which converges to the value set by the parameter c_{snrsep}.

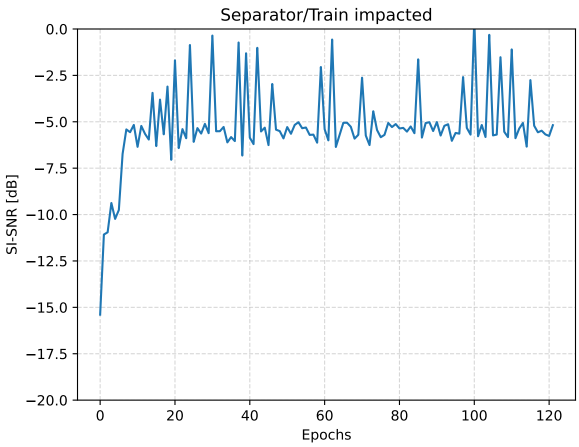
4.9 Final results

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. The results of the separator selection are shown in Figure 9. The first chart shows the overall results of each evaluated separator model. The bar charts show how the separator was successful on the generated augmentations by each generator. Bars represent groups of generators, which are grouped by their epoch number.

The separator with the best score is chosen as the best one. This separator is then evaluated on the WSJ and WHAM datasets. The results are shown in Table 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 WSJ dataset is similar to the



(a) Training curve of the separator loss function



(b) Training curve of the similarity loss function

Figure 8. 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.

model trained by the GAN, when evaluating on both the WSJ and WHAM datasets. There are three experiments using different separator weights, but none of them has achieved better results. Further experimenting is to be done to examine if there is any parameters settings combination that leads to better final results.

5. Conclusions

This paper investigates the usage of generative networks to automatically augment training data for speech separation systems. This could replace manually designed data augmentation methods and make the speech separation system more robust.

Table 3. Final results 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 WSJ or WHAM dataset. The columns with original annotation contain evaluation results of the raw pretrained separator model on the WSJ dataset. The columns with the augmented annotation contain the evaluation results of the best separator model chosen from the GAN training.

	WSJ original	WSJ robust	WHAM original	WHAM robust
$w_{\text{sep}} = 0.6$	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 main obstacle to training such a system is finding the correct parameters. These parameters should be chosen experimentally. The presented experiments show, that the system collapses to the two modes. Other experiments are set to solve this collapsing show that adjusting exactly the amount of the training space, which is given for both separator and generator networks during the training epoch is ineffective. Therefore, it is better to use the automated adjusting method of the amount of the training space. The automated method sets the goals of the generator and separator networks that should be achieved by each of them during their training turn.

The model using this automated method no longer collapses to one of the instability modes. However, as the final results reveal it does not train the separator network to be more robust properly. Further experiments have to be done to find the right parameters combination, that will train the separator network to be more robust. Adjusting the weight of the similarity function or different values of the generator and separator goals parameters could improve these results.

If such a parameters combination will be discovered, it will fit only for the current model with current settings and the current dataset. If there will be any change in of those three things, it is necessary to find the right parameters combination again. Therefore, the GAN parameters are very sensitive, it is very difficult to find such a combination. Nevertheless it is possible to use some algorithms to find the right parameters such as evolutionary algorithms [11] or bayesian hyperparameter optimization [12]. If these methods will work how they should then the using a GAN to make speech separation system robust could be used widely.

Figure 9. 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

References

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

[2] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[3] James V Stone. Independent component analysis: an introduction. *Trends in cognitive sciences*, 6(2):59–64, 2002.

[4] Fahimeh Bahmaninezhad, Jian Wu, Rongzhi Gu, Shi-Xiong Zhang, Yong Xu, Meng Yu, and Dong Yu. A comprehensive study of speech separation: spectrogram vs waveform separation. *arXiv preprint arXiv:1905.07497*, 2019.

[5] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 241–245. IEEE,

2017.

[6] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Sixteenth annual conference of the international speech communication association*, 2015.

[7] Divya Saxena and Jiannong Cao. Generative adversarial networks (gans) challenges, solutions, and future directions. *ACM Computing Surveys (CSUR)*, 54(3):1–42, 2021.

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

[9] Gordon Wichern, Joe Antognini, Michael Flynn, Licheng Richard Zhu, Emmett McQuinn, Dwight Crow, Ethan Manilow, and Jonathan Le Roux. Wham!: Extending speech separation to noisy environments. In *Proc. Interspeech*, September 2019.

[10] Jian Wu. A PyTorch implementation of the Conv-tasnet: Surpassing Ideal Time-Frequency Masking for Speech Separation, November 2019.

[11] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the workshop on machine learning in high-performance computing environments*, pages 1–5, 2015.

[12] Franck Dernoncourt and Ji Young Lee. Optimizing neural network hyperparameters with gaussian processes for dialog act classification. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 406–413. IEEE, 2016.