Report: MixCycle, Unsupervised Speech Separation

Adhémar de Senneville

March 26, 2024

Abstract

This report examines MixCycle's novel approach to unsupervised speech separation. I chose this paper despite its difficulty because I was very interested in how we could separate sources in an unsupervised way. That even seemed impossible to me. Despite your advice, I tried to re-implement the whole paper and it didn't work, however it was a great learning experience.

1 Summary of MixCycle

1.1 Introduction to source separation

Audio source separation involves disentangling individual audio sources from a mixed signal. Consider a scenario where we have N sources, $s_1, \ldots, s_N$, and their mixture x. The mixture can be represented as:

$$x = \sum_{i=1}^{N} s_i \tag{1}$$

The objective of audio source separation is to estimate the individual sources $\hat{s}_i$ from the mixture x. In this report, we will consider only two sources for simplicity and clarity without loss of generality, however, two sources separations already has many applications like denoising and voice extraction. Denoising removes unwanted noise from audio signals, enhancing clarity in noisy environments. Voice extraction isolates vocal elements from mixed sounds, useful in music production, karaoke, and improving voice recognition systems.

In deep learning, the separation model f_θ provided with a mixture x has to predict M sources, $\hat{s} = f_\theta(x)$. The focus is on optimizing θ to minimize a loss function L, subject to θ

1.2 Evaluation Metrics

The first thing to settle for this problem is a metric to evaluate the quality of a source separation. Two primary metrics used to evaluate the quality of audio source separation are the Signal-to-Noise Ratio (SNR) and Scale-Invariant Signal-to-Noise Ratio (SI-SNR). These metrics are differentiable, making them suitable for optimization in deep learning frameworks.

1.2.1 Signal-to-Noise Ratio (SNR)

SNR measures the level of the desired signal relative to the background noise. It is defined as:

$$\mathrm{SNR}(s, \hat{s}) = 10 \log_{10} \left(\frac{\|s\|^2}{\|\hat{s} - s\|^2 + \tau \|s\|^2} \right) \tag{2}$$

where s is the original signal, $\hat{s}$ is the separated signal and τ is a threshold used for numerical stability and to stabilise the gradient in a batch. It is set to $\tau = 10^{-\frac{\text{SNR}_{\max}}{10}}$. A higher SNR indicates better separation quality, therefore, the objective is to minimize the negative SNR.

1.2.2 Scale-Invariant Signal-to-Noise Ratio (SI-SNR)

SI-SNR extends SNR by being invariant to the scale of the target and estimated signals. It is calculated as:

$$\text{SI-SNR}(s, \hat{s}) = 10 \log_{10} \left(\frac{\|\alpha s\|^2}{\|\alpha s - \hat{s}\|^2 + \tau \|\alpha s\|^2} \right) \tag{3}$$

With, $\alpha = \frac{\langle \hat{s}, s \rangle}{\|s\|^2} = argmin_r \|ry - \hat{y}\|$, SI-SNR is particularly useful in scenarios where the amplitude of the signal may vary, ensuring a consistent measure of separation quality irrespective of signal scale.

For the rest of the report, we will consider SNR as the loss to evaluate the quality of the separation between two audio signal.

1.3 Assumptions

In audio source separation, the main assumption is that sources are independent. The independence assumption means the statistical distribution of one source does not change based on the values of the other sources. This independence assumption simplifies the separation process because it allows the use of statistical techniques.

1.4 Supervised case: PIT

In PIT [4], [7], the separation loss is expressed as:

$$L_{\text{PIT}}(s, \hat{s}) = \min_P \sum_{m=1}^{2} -\text{SNR}(s_m, [P\hat{s}]_m), \tag{4}$$

Here, P is a 2×2 permutation matrix. In this context, there is no predefined order of the source signals (sources are independent). The loss is computed using the permutation that yields the best match between ground-truth reference sources s and estimated sources $\hat{s}$. There are only two 2×2 permutation matrix which makes the calculation of the loss function negligible compared to the calculation time of the neural network. The loss function minimization problem is therefore trivial to solve and does not pose a problem during training.

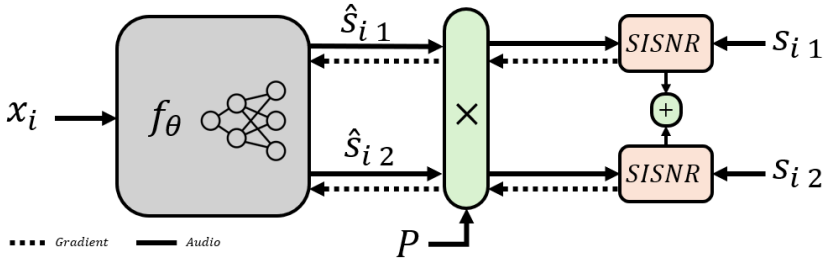


Figure 1: Permutation Invariant Training (PIT)

1.5 Supervised case: PIT-DM

Permutation Invariant Training with Dynamic Mixing (PIT-DM) is a data augmentation technique that improves PIT by mixing sources differently each time during training. It is possible thanks to the ground truth source separation available.

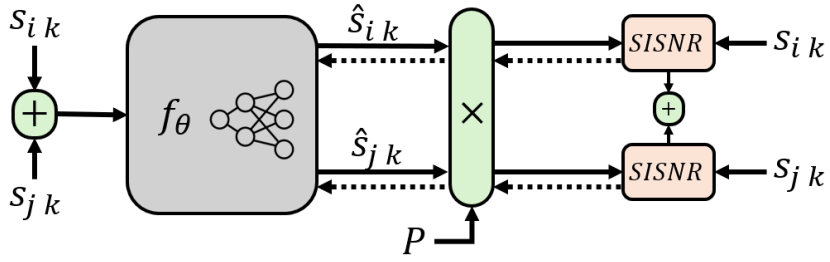


Figure 2: Permutation Invariant Training with Dynamic Mixing (PIT-DM)

1.6 Unsupervised case : MixIT

Permutation Invariant Training relies on ground truth source signals s. The paper MixIT [6] addresses this by considering two mixtures, x_i and x_j, each comprising up to 2 sources. Drawn randomly from an unsupervised dataset, these mixtures are combined to form a Mixture of Mixtures (MoM): $\bar{x} = x_i + x_j$. The separation model f_θ processes $\bar{x}$, outputting $2 * 2$ sources to ensure adequacy for any $\bar{x}$.

The unsupervised MixIT loss for estimated sources $\hat{s}$ and mixtures x_i, x_j is defined as:

$$L_{\text{MixIT}}(x_i, x_j, \hat{s}) = \min_A \sum_{i=1}^{2} -\text{SNR}(x_i, [A\hat{s}]_i)$$

Here, A is a 2×4 binary matrices with column sums to 1, allocating each source $\hat{s}_m$ to either x_i or x_j. MixIT minimizes the total loss between the mixtures x and the remixed separated sources $\hat{x} = A\hat{s}$, analogously to the process in PIT.

The main issue with MixIT is at inference.Because when you want to extract the two sources of a signal, you end up with a neural network that extract 4 sources. This increases the inference time without being useful.

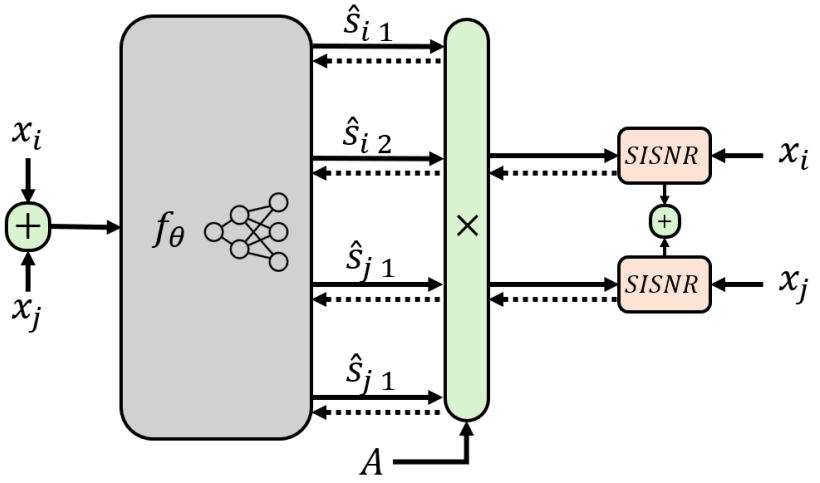


Figure 3: Mixture Invariant Training (MixIT)

1.7 MixPIT

MixPIT addresses the over-separation issue inherent in MixIT by aligning the number of model outputs with actual source quantities. In MixPIT, the model, $f_\theta(s_{i1}+s_{i2}+s_{j1}+s_{j2}) = f_\theta(x_i + x_j)$, is designed to process a mixture of four distinct sources but generates only two outputs. This model outputs three potential pair combinations: $\{\hat{s}_{i1} + \hat{s}_{i2}, \hat{s}_{j1} + \hat{s}_{j2}\}$, $\{\hat{s}_{i1} + \hat{s}_{j1}, \hat{s}_{i2} + \hat{s}_{j2}\}$, and $\{\hat{s}_{i2} + \hat{s}_{j1}, \hat{s}_{i1} + \hat{s}_{j2}\}$.

Due the statistical independence of sources s_i, s_j, s_k, and s_l, the likelihood of the three output pairs is equal. This equality arises because the model cannot differentiate between possible source combinations in the input mixtures.

The training of f_θ uses the PIT loss function defined as

$$L_{MixPIT}(x_i, x_j, \hat{S}) = L_{PIT}(x_i, x_j, \hat{S}),$$

where $\hat{S}$ represents one of the three output pairs.

In this framework, the PIT loss function L_{PIT} selects the permutation minimizing the loss. Consequently, the first (x_i, x_j) output pair indicates a perfect source match, whereas the second and third are partial matches. The design of the loss function ensures that at least two sources always match, enabling the model to effectively learn to separate two sources, even with noise from mismatched sources.

For MixPIT testing, the trained network is provided with a single mixture $f_\theta(x_i)$, to extract the individual source estimates $\hat{s}_{i1}$ and $\hat{s}_{i2}$. By maintaining a balance between the number of model outputs and the actual number of sources, MixPIT effectively mitigates the over-separation issue. Even if the problem of over-separation is solved, the model is not trained on the pure separation of sources, which means that performances may not be very good during inference. MixCycle counterbalance this problem by introducing a 2nd training phase of the neural network.

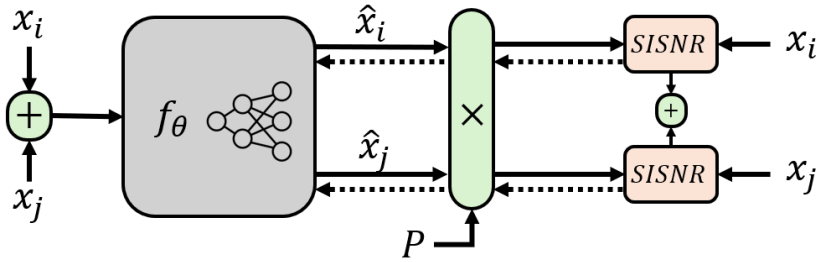


Figure 4: Mixture Permutation Invariant Training (MixPIT)

1.8 MixCycle

MixCycle is inspired by the same principle as PIT-DM. It uses predicted ground truth of the network as a training example for the next step. These techniques are often used in semi-supervised scenarios where we use predicted results of the neural network and add them to the dataset [5].

Inspired by continuous learning, the network (with freeze parameters) generate new pseudo ground truth data:

$$\{\tilde{s}_{i1}, \tilde{s}_{i2}\} = f_\theta(x_i), \quad \{\tilde{s}_{j1}, \tilde{s}_{j2}\} = f_\theta(x_j)$$

Now we can do a classic supervised training step like in PIT-DM. For example if we pick the pair $j1 - i2$:

$$\hat{s} = f_\theta(\tilde{s}_{j1} + \tilde{s}_{i2}), \quad s = \{\tilde{s}_{j1}, \tilde{s}_{i2}\}$$

$$L_{\text{MixCycle}}(s, \hat{s}) = \min_{P} \sum_{m=1}^{2} -\text{SNR}(s_m, [P\hat{s}]_m), \tag{5}$$

They use a trick to avoid divergence during the training. f_θ is constructed such that the sum of sources is equal to the original mixture. This is done through time frequency masking where the mask adds up to 1. However, the mask acts only on the amplitude of the short time Fourier transform (STFT) which puts constraint on the estimated sources because the function cannot change the phase from the original mixture. However, as seen in class the phase contains few information on the structure of the data for us humans.

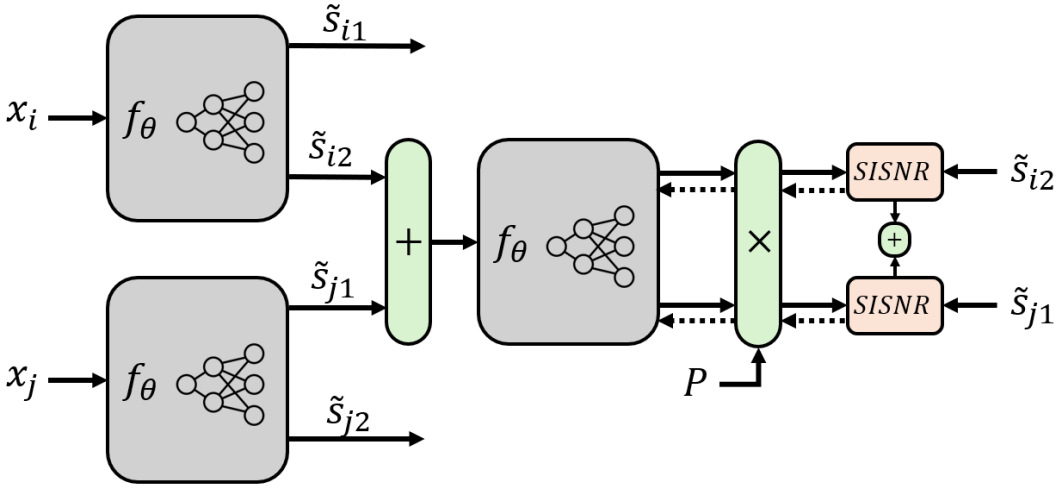


Figure 5: Cyclic Mixture Permutation Invariant Training (MixCycle)

1.9 Data

The paper uses 2 datasets: the standard LibriMix and the real-life mixtures REAL-M.

- LibriMix contain 212 hours of speech mixtures with gound truth. This is a classical dataset for audio source separation. - REAL-M contains only 74 minutes of unlabeled speech mixtures. This makes this new dataset particularly challenging. Because no ground truth is available on this dataset, an informal listening test was conducted to evaluate audio source separation on the REAL-M dataset. Ten participants, rated 10 validation mixtures for separation quality. They scored overall quality from 1 to 5. The resulting mean opinion scores (MOS) aligned with self-evaluation results, made available evaluation of the separation method's effectiveness.

1.10 Model

As said before, to have a stable training of MixCycle, the source estimation function must have the following property: $\hat{s}_{i1} + \hat{s}_{i2} = x_i$. To obtain very good results, the model used performs source separation in the time-frequency domain. In the time-frequency domain it predicts masks to be applied to the original signal to deduce the 2 sources. However the mask is only applied to the magnitude of the STFT. The phase remains the same as the mixture phase. The phase being considered negligible in the perception of harmonic sound like the voice, result are not altered a lot by this assumption. However, we can compute the best possible estimation of the neural network with this constrain on the phase. This is named ideal ratio mask (IRM) and it sets the lower bound on the loss during training.

The model used in the paper is Conv-TasNet [3]. The first operation on audio data before it enters the neural network is the STFT. The parameters are the followings: a window size of 512, a hop size of 128, and a Hann window. Knowing that sounds is sampled at 8000Hz and the fundamental frequency of a voice is about 160Hz, this allows to have a good temporal resolution. But mainly, this allows to have a frequency resolution degraded enough to only see the Formants of the voice and not harmonic peaks due to the harmonic nature of the voice. (In the case of voice separation)

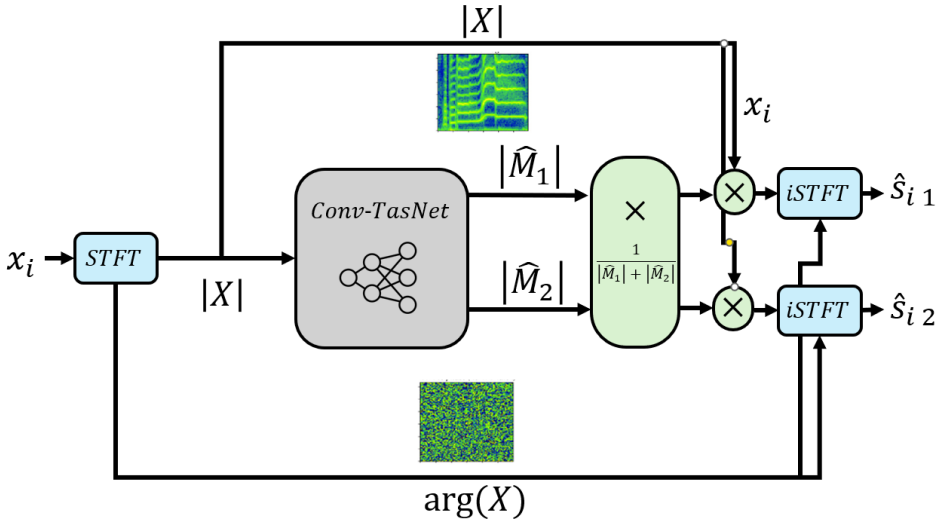


Figure 6: Time-Frequency Model

1.11 Results

1.11.1 LibriMix

In the study on LibriMix, the paper evaluate supervised baseline techniques like, IRM, PIT, PIT-DM, and unsupervised like MixIT. All methods were trained using Conv-TasNet with consistent parameters. The study's benchmarks were established using supervised methods—PIT, PIT-DM, and IRM setting the theoretical upper limit of performance that unsupervised techniques might not reach. Performance was assessed using the SI-SNRi metric, which quantifies the absolute quality improvement in SI-SNR.

Results show that the score of MixPIT is close to MixIT, despite a more challenging training objective and requiring less computation.

MixCycle, surpassed MixPIT as expected, and is the top-performing method among the unsupervised approaches. MixCycle nearly matched the performance of the supervised PIT-DM, illustrating the effectiveness of its unsupervised dynamic mixing process that leads to a model being trained on a dataset that closely simulates the supervised environment accessed by PIT-DM.

1.11.2 REAL-M

MixCycle technique, when combined with a teacher model and a remixing strategy, produces artificial mixtures that are close to original audio mixtures. Using self-evaluation, by generating and evaluating on unique artificial mixtures, it proved to be an effective alternative to traditional ground-truth evaluation, especially in scenarios where ground-truth data is not available.

Training	SI-SNR	Training Time
IRM	13.9	–
PIT	11.2	11.0h
PIT-DM	11.8	11.9h
MixIT	7.8	27.6h
MixPIT	7.1	4.2h
CycleMix	11.4	17.9h

Table 1: Performances of trainings (from the paper)

To validate the self-evaluation results, an informal listening test was conducted with 10 participants. The test confirmed the effectiveness of the MixCycle training, as the mean opinion scores (MOS) from the participants are consistent with the self-evaluation outcomes.

2 Experiments and critics (Voice Denoising)

2.1 Voice Denoising and Hypothesis relaxation

For our experiments, we will focus on speech denoising, which is a special case of source separation. It implies certain hypotheses compare to the general case. Firstly, we consider that there are only two sources: the voice v and the noise n such that:

$$x = \sum_{i=1}^{N} s_i = v + n \tag{6}$$

The noise is not considered to be white or to have any color because it has a structure, it can be the background noise of a train station or of a restaurant. The voice has we know has also a structure. This means that the classical hypothesise of source independence no longueur hold. If the first source is a voice, then the second can only be a noise. However, we can change the equations to our special case, and show that all those training methods still works.

2.1.1 PIT Loss

The PIT loss can be simplified, because the model can learn what is voice and noise:

$$L_{\text{PIT Simplified}}(s, \hat{s}) = \sum_{m=1}^{2} -\text{SNR}(s_m, [P\hat{s}]_m) \tag{7}$$

Here, P is the 2×2 identity matrix. There is no more optimisation problem to solve. This optimisation problem had a low computational cost compare to deep learning calculation done before and after the loss, however this means less computation overall during training. Notice that it is still possible to use classical PIT Loss during training, it can give different, or better result, especially when the two sources are close to each other.

2.1.2 MixIT Loss

We can expect the model to predict : $f_\theta(x_i + x_j) = [\hat{v}_i, \hat{v}_j, \hat{n}_i, \hat{n}_j]$ MixIT Loss can also be simplified:

$$L_{\text{MixIT}}(x_1, x_2, \hat{s}) = \min_{A} \sum_{i=1}^{2} -\text{SNR}(x_i, [A\hat{s}]_i)$$

Here, A is a 2×4 binary matrices which is a concatenation of two 2×2 permutation matrices, allocating each voice $\hat{v}_m$ to either x_i or x_j and each noise $\hat{n}_m$ to either x_i or x_j.

2.1.3 PITMix Loss

In a more interesting way, there is no simplification to be made with PITMix loss. Nonetheless the possible reconstruction result change. The model has no interest subject to the loss to reconstruct noise with noise and voice with voice which narrow down the possibilities.

Model output pair	Perm.	Loss function $L_{\text{MixPIT}}(\cdot)$
$S_1 = \{s_{i1} + s_{i2}, s_{j1} + s_{j2}\}$	1	$L(\hat{s}_{i1} + \hat{s}_{i2}, s_{i1} + s_{i2}) + L(\hat{s}_{j1} + \hat{s}_{j2}, s_{j1} + s_{j2})$
	2	~~$L(\hat{s}_{i1} + \hat{s}_{i2}, s_{i1} + s_{i2}) + L(\hat{s}_{j1} + \hat{s}_{j2}, s_{j1} + s_{j2})$~~
$S_2 = \{s_{i1} + s_{j1}, s_{i2} + s_{j2}\}$	1	$L(\hat{s}_{i1} + \hat{s}_{j1}, s_{i1} + s_{i2}) + L(\hat{s}_{i2} + \hat{s}_{j2}, s_{j1} + s_{j2})$
	2	$L(\hat{s}_{i1} + \hat{s}_{j1}, s_{i1} + s_{i2}) + L(\hat{s}_{i2} + \hat{s}_{j2}, s_{j1} + s_{j2})$
$S_3 = \{s_{i2} + s_{j1}, s_{i1} + s_{j2}\}$	1	$L(\hat{s}_{i2} + \hat{s}_{j1}, s_{i1} + s_{i2}) + L(\hat{s}_{i1} + \hat{s}_{j2}, s_{j1} + s_{j2})$
	2	$L(\hat{s}_{i2} + \hat{s}_{j1}, s_{i1} + s_{i2}) + L(\hat{s}_{i1} + \hat{s}_{j2}, s_{j1} + s_{j2})$

Practically, this will reduce the lower bound of the loss that will predict more often the good separation. And reduce noise in the training.

Model output pair	Perm.	Loss function $L_{\text{MixPIT}}(\cdot)$
$S_1 = \{s_{i1} + s_{i2}, s_{j1} + s_{j2}\}$	1	$L(\hat{s}_{i1} + \hat{s}_{i2}, s_{i1} + s_{i2}) + L(\hat{s}_{j1} + \hat{s}_{j2}, s_{j1} + s_{j2})$
	2	~~$L(\hat{s}_{i1} + \hat{s}_{i2}, s_{j1} + s_{j2}) + L(\hat{s}_{j1} + \hat{s}_{j2}, s_{i1} + s_{i2})$~~
$S_2 = \{s_{i1} + s_{j2}, s_{j1} + s_{i2}\}$	1	$L(\hat{s}_{i1} + \hat{s}_{j2}, s_{i1} + s_{i2}) + L(\hat{s}_{j1} + \hat{s}_{i2}, s_{j1} + s_{j2})$
	2	$L(\hat{s}_{i1} + \hat{s}_{j2}, s_{i1} + s_{i2}) + L(\hat{s}_{j1} + \hat{s}_{i2}, s_{j1} + s_{j2})$

Using a more appropriate notation :

Model output pair	Perm.	Loss function $L_{\text{MixPIT}}(\cdot)$
$S_1 = \{v_i + n_i, v_j + n_j\}$	1	$L(v_i + n_i, v_i + n_i) + L(v_j + n_j, v_j + n_j)$
	2	~~$L(v_i + n_i, v_j + n_j) + L(v_j + n_j, v_i + n_i)$~~
$S_2 = \{v_i + n_j, v_j + n_i\}$	1	$L(v_i + n_j, v_i + n_i) + L(v_j + n_i, v_j + n_j)$
	2	$L(v_i + n_j, v_i + n_i) + L(v_j + n_i, v_j + n_j)$

2.2 Data

The dataset [1] LibriMix is made for Generalizable Speech Separation. I chose Libri2Mix_min16k a reduced dataset of LibriMix with 1 speaker and background noise. The original Libri2Mix dataset is 430GB of data wile Libri2Mix_min16k is only 12GB. The sample rate is 16 kHz.

2.3 Models

Two distinct models were employed in the experimental phase to evaluate the source separation efficacy. The first model, ConvTasNet [3], aligns with the original paper's approach, predominantly utilizing a frequency-based methodology. ConvTasNet, renowned for its proficiency in frequency domain analysis, served as a benchmark for comparison. The second model employed, Hdemuc [2], stands out due to its hybrid architecture. This innovative design amalgamates raw audio and frequency domain techniques. Hdemuc is capable to get the strengths of both approaches.

2.4 Implementation

The following list encompasses all key components implemented within the project:

1. **Dataset Management**
    - Loading and preprocessing of audio data.

- STFT, Padding, Random-Filtering

2. **Losses**
   - SI-MSE (Scale-Invariant Mean Squared Error)
   - SNR (Signal-to-Noise Ratio)
   - SI-SNR (Scale-Invariant Signal-to-Noise Ratio)
3. **Conv-TasNet Model**
   - Encoder and decoder structure, incorporating convolutional layers. Implementation of skip connections.
4. **Training Pipeline**
   - PIT (Permutation Invariant Training)
   - MixIT (Mixture Invariant Training)
   - MixPIT (Mixture Permutation Invariant Training)
   - MixCycle (Cyclic Mixture Permutation Invariant Training)

Parameter	Value
GPU Used	1x P100
Model	ConvTasNet
Optimizer	Adam
Learning Rate	0.03
Early Stopping	3 epochs

Table 2: Training Configuration

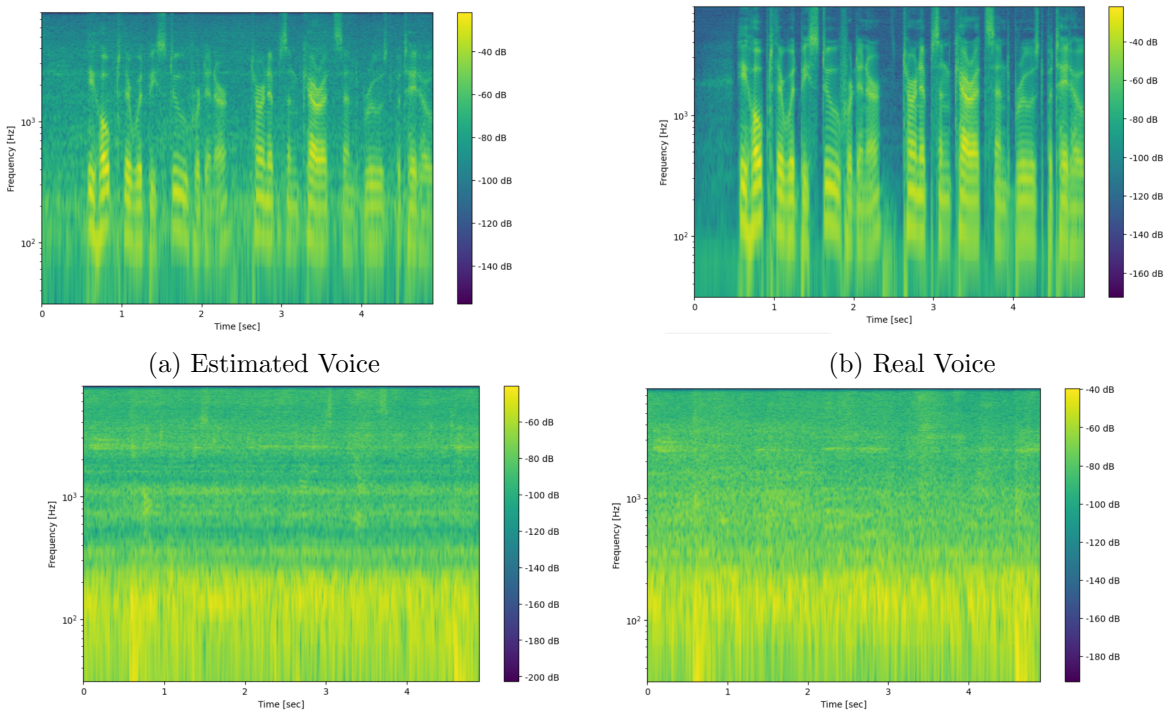
2.5 Results

2.5.1 Supervised : Without Deep-learning

One of the simplest ways to approach this problem with a classic method (without deep-learning) is to use a Wiener filter. As seen in class, by using a prior on the spectral density of noise and voice we can find the optimal filter for denoising.

$$H(f) = \frac{S_{ss}(f)}{S_{xx}(f)}$$

In this supervised experiment, we use the average spectral density of the dataset as a prior on the spectral density of voice and noise. The results are of average quality, nevertheless satisfactory.



(a) Estimated Voice

(b) Real Voice

(c) Estimated Noise

(d) Real Noise

Figure 7: Wiener Filter for voice denoising (STFT)

The main fault of this method is that it only uses one filter to do the denoising. However, a neural network can perform much more complex, non-linear operations on the time-frequency representation of the audio to extract the two signals.

2.5.2 Supervised : PIT

The results of training the model on PIT are satisfactory, however I encountered several problems. The audio time being very high, it was impossible to parallelize the training since the maximum batch size was 1 to avoid crashing the GPU. This significantly slowed down the training which for PIT lasted 8 hours.

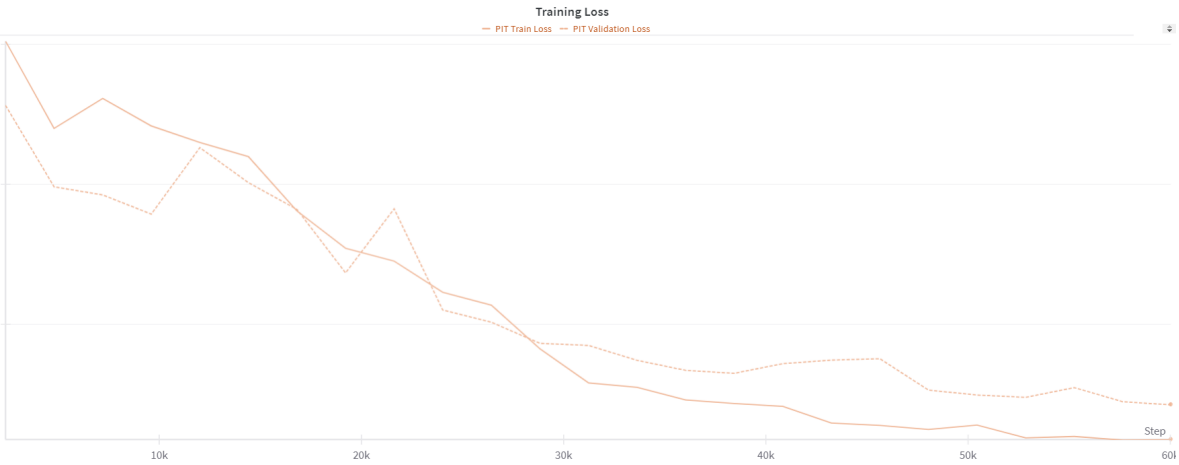
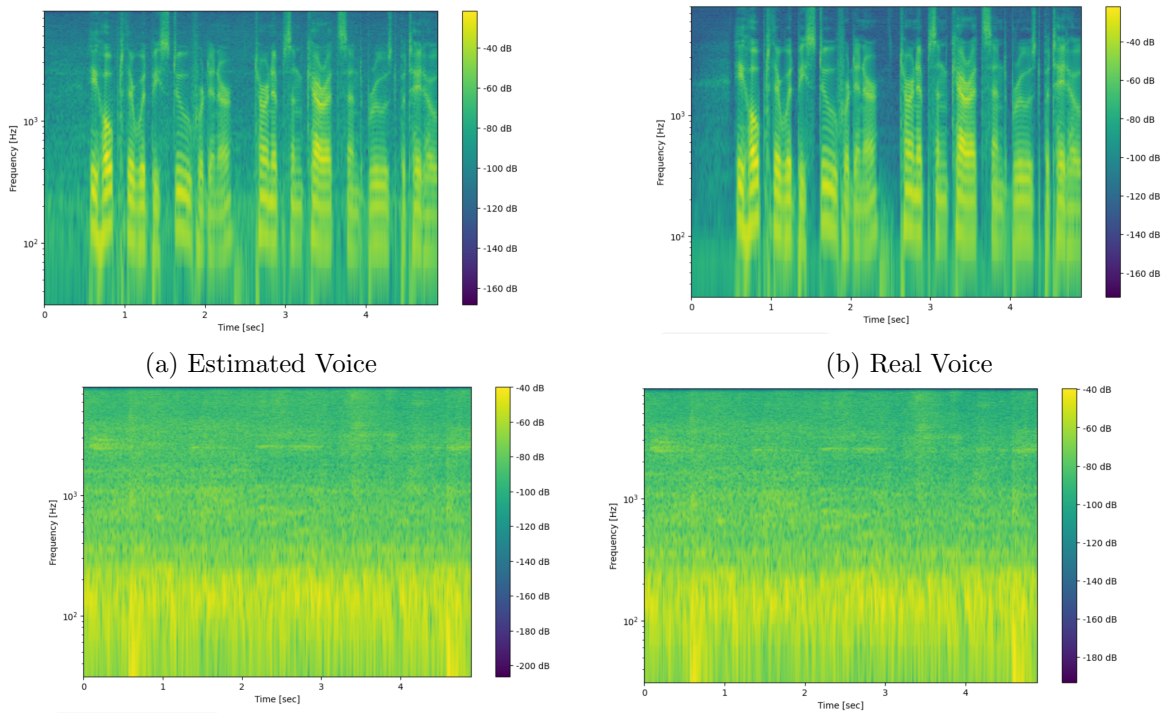


Figure 8: PIT Training



(a) Estimated Voice

(b) Real Voice

(c) Estimated Noise

(d) Real Noise

Figure 9: PIT for voice denoising (STFT)

2.5.3 Unsupervised

On all unsupervised techniques the results are quite disappointing. Due to the very noisy nature of the training of unsupervised methods, the descent of the cost function even at the start of training is very low even after several hours of training. I was therefore not able to obtain satisfactory results in unsupervised mode for source separation. The audio results were poor.

Figure 10: MixIT, MixPIT, MixCycle Training Losses

As you can see on the image the loss is very noisy, after about five hours of training we don't even know if the loss is decreasing. Despite the fact that each training method is

relatively simple to implement, the number of hyper parameters needed for training is far too high to have the right parameters that give the best results.

2.6 Proposed Experiment on MixPIT

On important critic of MixPIT (and all unsupervised training procedures) is it strong hypothesis that assumes that sources s_i, s_j, s_k, s_l are statistically independent from each others. Two significant real-world considerations can challenge its validity. Firstly, the acoustic environment in which the audio is recorded inherently imprints a shared acoustic signature upon the sources. This comprises similar resonance and impulse responses, like two voices recorded in the same room. Such environmental factors create acoustic characteristics that deviate from the assumption of statistical independence. Secondly, the heterogeneity in recording equipment quality and frequency response further complicates this assumption. Variations in microphone quality and their distinct frequency response profiles can lead to a form of frequency-dependent correlation between sources. This correlation manifests as a statistical dependency, driven by the equipment used in the recording process. In practical scenarios, the assumption of statistical independence may be overly simplistic and not fully representative of the complex inter-dependencies present in real-world audio recordings.

To investigate this identified limitations, I wanted to try an experiment wherein the audio quality in the dataset was altered to mimic the frequency response of a varying-quality microphone. This was achieved by applying a band-pass filter with variable bandwidth to simulate the effects of inferior recording equipment. The aim was to observe the impact of such alterations on the training process and the final quality of the model.

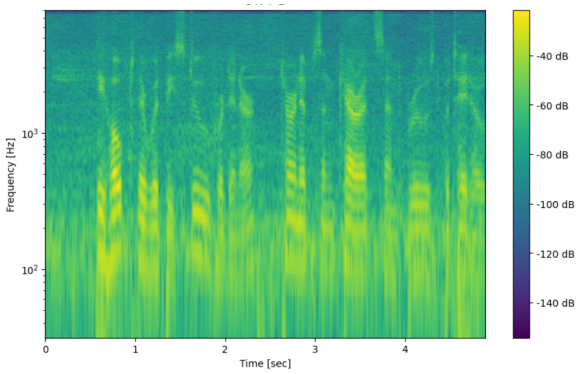


Figure 11: Mixture

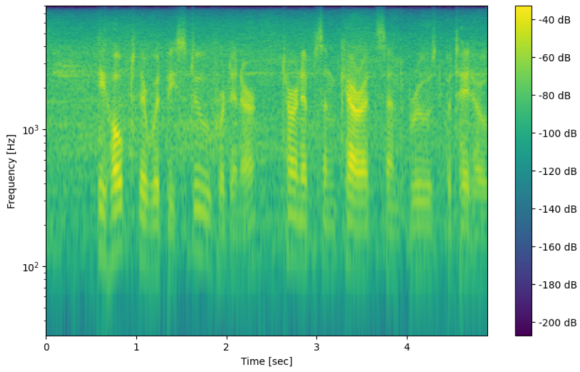


Figure 12: Filtered Mixture

In assessing the model's performance, a key metric was the minimal loss function value of the dataset in a mixed-cycle assuming or not statistical dependence of sources. The formula used for this calculation provides a theoretical benchmark for comparison:

$$MinL_{Dependence} = -2 \times \mathrm{SNR}_{\max}$$

$$\begin{aligned} MinL_{Independence} = & \frac{1}{N} \sum_{x_i, x_j \in Data} \frac{1}{3} (-2 \times \mathrm{SNR}_{\max} && (8) \\ & + max(L(\hat{s}_{i1} + \hat{s}_{i2}, s_{i1} + s_{i2}) + L(\hat{s}_{j1} + \hat{s}_{j2}, s_{j1} + s_{j2}), && (9) \\ & L(\hat{s}_{i1} + \hat{s}_{i2}, s_{j1} + s_{j2}) + L(\hat{s}_{j1} + \hat{s}_{j2}, s_{i1} + s_{i2})) && (10) \\ & + max(L(\hat{s}_{i2} + \hat{s}_{j1}, s_{i1} + s_{i2}) + L(\hat{s}_{i1} + \hat{s}_{j2}, s_{j1} + s_{j2}), && (11) \\ & L(\hat{s}_{i2} + \hat{s}_{j1}, s_{i1} + s_{i2}) + L(\hat{s}_{i1} + \hat{s}_{j2}, s_{j1} + s_{j2}))) && (12) \end{aligned}$$

The objective would have been to see if the cost function falls below the dependence threshold during unsupervised training. This would have made it possible to say that the network has learned the independence between the data and uses it to reduce the cost function without separating the sources. This is only of interesting if we know the sources in advance. This threshold is therefore only useful for experiments and does not allow choices to be made during unsupervised training.

Unfortunately, the lack of results on unsupervised learning does not allow me to conduct this experiment here is a graph of what I would have expected if the experiment had been conclusive.

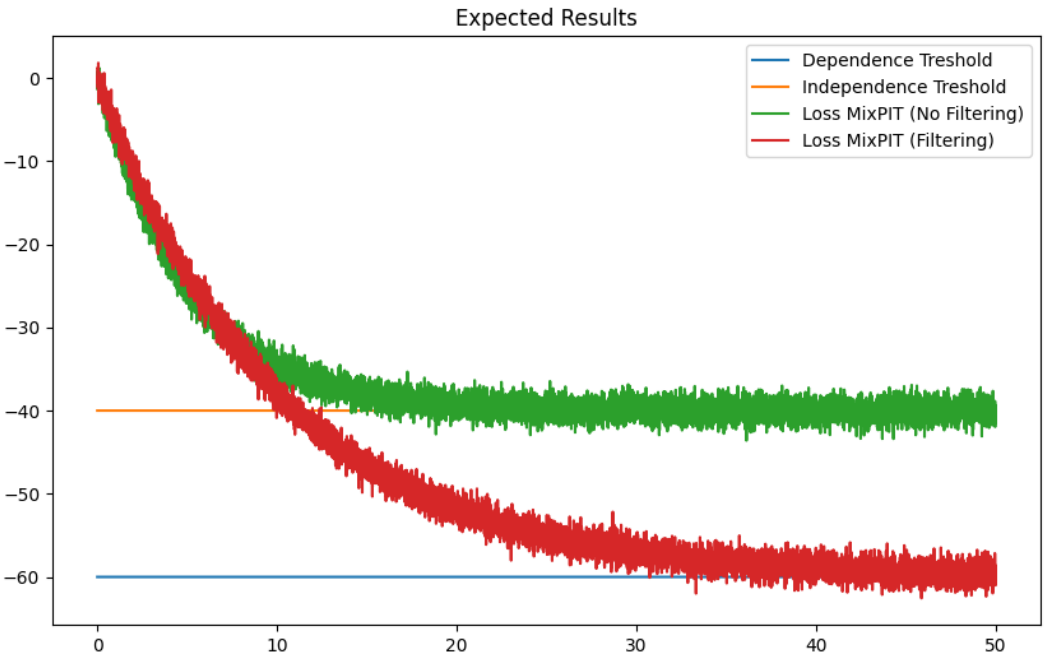


Figure 13: Expected Results

References

[1] Joris Cosentino, Manuel Pariente, Samuele Cornell, Antoine Deleforge, and Emmanuel Vincent. Librimix: An open-source dataset for generalizable speech separation, 2020.

[2] Alexandre Défossez. Hybrid spectrogram and waveform source separation, 2022.

[3] 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, August 2019.

[4] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[5] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence, 2020.

[6] Scott Wisdom, Efthymios Tzinis, Hakan Erdogan, Ron J. Weiss, Kevin Wilson, and John R. Hershey. Unsupervised sound separation using mixture invariant training, 2020.

[7] 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, 2017.