ATTENTIONPIT: SOFT PERMUTATION INVARIANT TRAINING FOR AUDIO SOURCE SEPARATION WITH ATTENTION MECHANISM

Hirokazu Kameoka, Shogo Seki, Li Li, and Chihiro Watanabe

NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation

ABSTRACT

Permutation invariant training (PIT) has recently attracted attention as a framework to achieve end-to-end time-domain audio source separation. Its goal is to train a separation network that takes a mixture signal as input and produces the J underlying source signals. Since the order of the output signals is arbitrary, the idea of PIT is to first find the best output-target assignment and then update the network parameters based on the error given by that assignment at each iteration. However, there are two known problems with PIT: One is that it has a time complexity of $\mathcal{O}(J!)$, which makes it infeasible as J increases, and the other is that it is prone to getting stuck in bad local optimal solutions due to the hard output-target assignment process. To overcome these problems simultaneously, in this paper, we propose AttentionPIT, which uses an attention mechanism to find soft output-target assignments for separation network training, and can be run in polynomial time in J, as with the recently proposed fast PIT variants such as SinkPIT and HungarianPIT. The training loss of AttentionPIT is fully differentiable, allowing us to simultaneously perform processes corresponding to soft output-target assignment and network parameter update through backpropagation. Experiments on the LibriMix corpus revealed that while AttentionPIT works reasonably well on its own, it works even better when combined with SinkPIT and HungarianPIT so that AttentionPIT is run only in the early stages of training.

Index Terms— End-to-end audio source separation, permutation invariant training (PIT), attention

1. INTRODUCTION

The task of separating and extracting the signal of each sound source from a monaural mixture of multiple sources is called monaural source separation. Since the emergence of powerful methods based on deep neural networks (DNNs), the level of sound source separation performance has been dramatically improved. The methods proposed so far can be broadly classified into two approaches: spectral-based and time-domain-based ones. The spectral-domain approach aims to predict binary, soft, or complex-valued time-frequency masks that extract the components corresponding to the source signals from an input mixture signal [1–3]. The time-domain approach, on the other hand, aims to perform separation in an end-to-end fashion by designing and training a neural network model that takes a mixture signal as input and outputs a set of J signals (hereafter referred to as a separation network). One problem in training a separation network is that the order of the output signals depends on the architecture and parameters of the network and it is usually difficult to predict in advance which source each output signal corresponds to. Therefore, simply using the error between the output signals and the target signals arranged in a predetermined order as the training loss will not provide satisfactory performance to the network. To tackle this problem, a framework called permutation invariant training (PIT) [4] has been proposed. The idea is to first find the best output-target assignment for each training sample, and then update the network parameters based on the error given by that assignment at each iteration. This technique, or its utterance-level extension called uPIT [5], has been shown to be effective for the above problem and is currently being employed to train many types of separation networks, including the time-domain audio separation network (TasNet) [6], convolutional TasNet (ConvTasNet) [7], dual-path recurrent neural network (DPRNN) [8], Wavesplit [9], as well as for the successive downsampling and resampling of multi-resolution features (SuDoRM-RF) [10].

However, since the time complexity of the exhaustive search for the best output-target assignment is $\mathcal{O}(J!)$, PIT becomes more challenging as the number of sources increases (say, to more than ten). For example, when $J = 20$, $J! = 2.4 \times 10^{18}$, which makes training infeasible. Some methods have recently been proposed to overcome this obstacle. One of them is called SinkPIT [11], which exploits the fact that permutation matrices are a special case of doubly stochastic matrices, and that the Sinkhorn-Knopp algorithm can converge an arbitrary positive matrix to a doubly stochastic matrix. The authors showed that the Frobenius inner product between the pairwise error matrix (namely, a matrix with each element being the error between a different pair of the output and target signals) and the doubly stochastic matrix converged from a certain matrix expressed using the pairwise error matrix is equivalent to the PIT loss. Using this loss leads to a time complexity of $\mathcal{O}(J^2)$. Another example is HungarianPIT [12], which uses the Hungarian algorithm [13–15], a

This work was supported by JST CREST Grant JPMJCR19A3, Japan.

method that can solve the assignment problem in polynomial time, to find the best output-target assigment instead of an exhaustive search. The time complexity of HungarianPIT is $\mathcal{O}(J^3)$, which is still at a level sufficient to handle realistic scenarios (e.g., up to twenty sources).

Another problem that PIT poses is convergence to a bad local optimal solution. Since PIT alternates between output-target assignment and network parameter update, it is inherently prone to getting stuck in bad local solutions, compared to the optimization in ordinary regression model training. This is in some sense analogous to the k-means algorithm, a well-known clustering method that alternates between cluster assignment and cluster centroid update. Especially in the early stages of training, when the separation network tends to still be immature and produces only seemingly noisy signals, it would be better to leave the chance for every output signal to correspond to any source, rather than assigning each output signal to a different source exclusively. For the same reason that the expectation-maximization algorithm, or soft k-means algorithm, is less prone to local solutions than the k-means algorithm, we believe that soft assignment is the key to avoiding bad local solutions in PIT. SinkPIT and another PIT variant called ProbPIT[1] [16] are examples of methods designed to find soft output-target assignments in the PIT framework, and the method proposed in this paper takes a similar approach in that regard. In particular, motivated by the fact that the attention mechanism [17, 18] has been proven to be very good at finding soft assignments between the items in a sequence or a pair of sequences, in this paper, we propose AttentionPIT, which uses an attention mechanism for finding soft output-target assignment in separation network training. As will be explained later, the complexity of AttentionPIT is $\mathcal{O}(J^2)$ or $\mathcal{O}(J^3)$ depending on the choice of regularization loss. We will also show that while AttentionPIT works reasonably well on its own, it works even better when combined with HungarianPIT or SinkPIT so that AttentionPIT is run only in the early stages of training.

2. METHOD

2.1. Notation and Problem Formulation

By using $s_j(n)$ to denote the signal of source j, the mixture $y(n)$ of J source signals, $s_1(n), \ldots, s_J(n)$, is given by

$$y(n) = \sum_j s_j(n) \ (n = 1, \ldots, N), \tag{1}$$

where n and N denote the time index and the number of sample points, respectively. The goal of monaural source separation is to restore $\mathbf{s}_j = [s_j(1), \ldots, s_j(N)] \in \mathbb{R}^{1\times N}$ $(j = 1, \ldots, J)$ solely from $\mathbf{y} = [y(1), \ldots, y(N)] \in \mathbb{R}^{1\times N}$.

[1]The complexity of ProbPIT is the same as PIT, namely, $\mathcal{O}(J!)$.

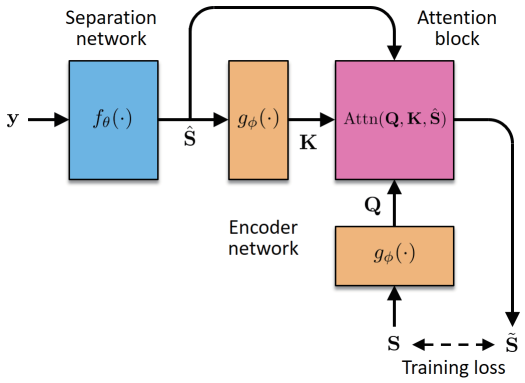

Fig. 1. Illustration of AttentionPIT

The DNN-based time-domain approach tackles this problem by designing a separation network $f_\theta(\cdot)$

$$\hat{\mathbf{S}} = f_\theta(\mathbf{y}) \tag{2}$$

with some architecture parametrized by θ, where $\hat{\mathbf{S}} = [\hat{\mathbf{s}}_1; \ldots; \hat{\mathbf{s}}_J] \in \mathbb{R}^{J\times N}$, and by training θ so that $\hat{\mathbf{S}}$ matches the target $\mathbf{S} = [\mathbf{s}_1; \ldots; \mathbf{s}_J] \in \mathbb{R}^{J\times N}$. Note that here we have used ; to denote vertical concatenation of vectors or matrices. However, which source each output signal $\hat{\mathbf{s}}_j$ corresponds to depends on the architecture and parameters of f_θ, and cannot be easily predicted prior to training. The idea of PIT [4] is to find the best output-target assignment at each iteration before updating θ. We can write this in the form of a training loss as

$$L(\theta) = \mathbb{E}_{\mathbf{S},\mathbf{y}}\left[\min_{\mathbf{P}\in\mathcal{P}} D(\mathbf{P}f_\theta(\mathbf{y}), \mathbf{S})\right], \tag{3}$$

where $\mathbb{E}_{\mathbf{S},\mathbf{y}}[\cdot]$ means the sample mean over all training examples, and $\mathbf{P} \in \mathcal{P}$ denotes a permutation matrix that corresponds to a particular output-target assignment. D represents a measure of the errors between the output and target signals. One widely used measure is the mean of the negative scale-invariant signal-to-distortion ratios (SI-SDRs) [19].

HungarianPIT [12] provides a way to search for the best $\mathbf{P}$ in a $\mathcal{O}(J^3)$ running time under fixed θ. Hence, local optimality can be efficiently achieved by alternately updating $\mathbf{P}$ and θ. However, for the reason mentioned earlier, it may not necessarily be the best strategy for finding the jointly optimal solution of $\mathbf{P}$ and θ.

2.2. AttentionPIT

Here, we describe AttentionPIT, which provides yet another way to find soft output-target assignments during model training, different from SinkPIT [11] and ProbPIT [16]. In AttentionPIT, an additional encoder network g_ϕ is introduced and trained along with the separation network. The encoder is configured to take $\hat{\mathbf{S}}$ or $\mathbf{S}$ as input and to output

$$\mathbf{K} = g_\phi(\hat{\mathbf{S}}), \tag{4}$$

$$\mathbf{Q} = g_\phi(\mathbf{S}). \tag{5}$$

Each row of $\mathbf{K}, \mathbf{Q} \in \mathbb{R}^{J \times N'}$ is expected to be a sequence consisting of the features of the corresponding input signal, where N' denotes the length of the output sequences. Using $\mathbf{K}$ and $\mathbf{Q}$, we can compute an "attention" matrix $\mathbf{A}$ with each element $a_{jj'}$ representing the similarity between the output signal j and the target signal j'. If we use the expression of the scaled dot-product attention [18], $\mathbf{A}$ is given by

$$\mathbf{A} = \mathrm{softmax}(\mathbf{K}\mathbf{Q}^\mathsf{T}/\sqrt{N'}), \tag{6}$$

where $\mathrm{softmax}$ denotes a softmax function applied to each column in a matrix so that the elements in each column sum to 1. Here, it is important to note that the way our attention matrix is computed differs from the regular attention mechanism employed in sequence-to-sequence models in that $\mathbf{A}$ represents the correspondence between channels rather than between time points: In the context of sequence-to-sequence models, the scaled dot-product attention is given as $\mathbf{A} = \mathrm{softmax}(\mathbf{K}^\mathsf{T}\mathbf{Q}/\sqrt{J})$, when $\mathbf{S}$ and $\hat{\mathbf{S}}$ are interpreted as J-dimensional vector sequences. In our $\mathbf{A}$, element $a_{jj'}$ represents how likely the target signal j' is to correspond to the ouput signal j, which takes a value close to 1 if the output j and the target j' are similar, and 0 otherwise, and

$$\tilde{\mathbf{S}} = \mathbf{A}^\mathsf{T}\hat{\mathbf{S}} \tag{7}$$

becomes a matrix such that each row is a convex combination of the output signals (the rows of $\hat{\mathbf{S}}$) that approximates one of the target signals. Our goal is to train θ and ϕ so that $\tilde{\mathbf{S}}$ fits $\mathbf{S}$. If we put (6) and (7) together and denote them as

$$\tilde{\mathbf{S}} = \mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \hat{\mathbf{S}}), \tag{8}$$

the training objective to be minimized becomes

$$\begin{aligned} L(\theta, \phi) &= \mathbb{E}_{\mathbf{S},\mathbf{y}}\big[D(\tilde{\mathbf{S}}, \mathbf{S})\big] \\ &= \mathbb{E}_{\mathbf{S},\mathbf{y}}\big[D(\mathrm{Attn}(g_\phi(\mathbf{S}), g_\phi(f_\theta(\mathbf{y})), f_\theta(\mathbf{y})), \mathbf{S})\big]. \end{aligned} \tag{9}$$

The time complexity for computing this loss is $\mathcal{O}(J^2)$.

There should be a one-to-one correspondence between the output and target signals. This corresponds to the case where $\mathbf{A}$ satisfies the conditions for a permutation matrix. Since using the above loss alone does not necessarily converge $\mathbf{A}$ to a permutation matrix, we need a regularization term that encourages $\mathbf{A}$ to become one. To this end, we introduce

$$\mathcal{R}(\theta, \phi) = \lambda \mathbb{E}_{\mathbf{S},\mathbf{y}}\big[\tfrac{1}{J^2}\|\mathbf{A}\mathbf{A}^\mathsf{T} - \mathbf{I}\|_1\big], \tag{10}$$

as the regularization loss, where $\|\cdot\|_1$ denotes the sum of the absolute values of all the elements in a matrix and λ denotes the regularization weight. In the following experiment, we gradually increased λ from 0 to 50 according to $\lambda = \min(1.05^e - 1, 50)$ during training, where e denotes the number of epochs. The time complexity for computing this loss is $\mathcal{O}(J^3)$. Note that as an alternative, we also tried a sparsity-inducing loss [20]

$$\mathcal{R}(\theta, \phi) = \lambda \mathbb{E}_{\mathbf{S},\mathbf{y}}\Big[\tfrac{1}{J}\textstyle\sum_j \frac{(\sum_{j'}|a_{jj'}|)/\sqrt{\sum_{j'}|a_{jj'}|^2} - 1}{\sqrt{J} - 1}\Big], \tag{11}$$

which takes a value of zero if each row of $\mathbf{A}$ contains only one non-zero element and requires a time complexity of $\mathcal{O}(J^2)$ to compute, but there was no significant difference in the actual training time up to $J = 20$ compared to (10).

The training loss of AttentionPIT is fully differentiable, allowing us to simultaneously perform processes corresponding to soft output-target assignment and network parameter update through backpropagation. Here, it is interesting to compare how SinkPIT and AttentionPIT differ in the way they find soft output-target assignments during training. The difference between the two is the point in the loss computation process at which the weighted mean is taken: SinkPIT uses as the loss the weighted mean of the SI-SDRs of all possible output-target pairs, whereas AttentionPIT uses as the loss the SI-SDR between each target signal and a weighted mean (convex combination) of the output signals. Of course, this difference alone is not conclusive enough to argue which is superior, so they need to be compared experimentally.

While the time complexities of SinkPIT, HungarianPIT, and AttentionPIT in terms of J are as decribed above, the actual training time is influenced not only by J but also by other factors such as the hyperparameter settings and input signal lengths, unless we assume a considerably large J. Therefore, the superiority of each method should be discussed after the source separation accuracy and actual training time have been experimentally compared.

2.3. Architecture

The encoder network g_ϕ was designed to consist of four 1D strided convolution layers with the kernel size of 8, stride size of 2, and J output channels, each followed by an instance normalization layer and a sigmoid linear unit (SiLU) function except for the last layer. Hence, the length of the output sequence becomes $1/16$th of the input signal (i.e., $N' = N/16$). For the separation network f_θ, we chose to use the ConvTasNet architecture [7].

3. EXPERIMENT

3.1. Datasets

For the experiment, we used the speech signals in the LibriSpeech corpus [21] and the script[2] made available by the authors of [12, 22] to create training, validation, and evaluation datasets for the clean mixtures (without noise) of 5, 10, 15, and 20 speakers. All the signals were sampled at 8 KHz, and all the mixtures were created in the min mode [23]. These datasets are referred to as Libri$J$Mix.

[2]https://github.com/ShakedDovrat/LibriMix

3.2. Baselines and Experimental Setup

SinkPIT, HungarianPIT, and AttentionPIT (hereafter abbreviated as Sink, Hun, and Attn, respectively) were compared in terms of computation time and separation accuracy. Note that if the computation time is unlimited, Hun is equilvalent to PIT in terms of separation accuracy. In addition to these, versions in which Attn was run up to 20 epochs and Sink or Hun was run thereafter (abbreviated as Attn→Sink and Attn→Hun, respectively) were also considered for comparison. For the implementations of Sink and Hun, the scripts[3] provided in the Asteroid platform [23] were used as they are. Attn was implemented on the same platform to make the condition as consistent as possible. All the methods were run using the Adam optimizer [24] with the batch size of 8 and learning rate of 0.001. For all the methods, the learning rate was reduced by a factor of 0.5 if the loss was not improved for five consecutive epochs. The β scheduler for SinkPIT was configured in the same manner as in [11]. The negative mean SI-SDR was used as D. The mean SI-SDR under the best output-target assignment (found using the Hungarian algorithm [13–15]) was used for the validation and evaluation metrics. For each method, the model that obtained the best validation metric within 200 epochs was used for evaluation. The average number of iterations per second was used as the computational efficiency metric. All the methods were run on a single Tesla V100 SXM2 GPU with a 32.0 GB memory and an Intel(R) Xeon(R) Platinum 8160 24-core CPU @ 2.10GHz.

In a previous work [12], Hun and Sink were only compared under different separation networks. In this experiment, the separation network was fixed to ConvTasNet in order to make a pure comparison of the training methods.

3.3. Results

Table 1 shows the average number of iterations run per second for each method. The results show that Sink and Attn were equally efficient and Hun was slightly more efficient than the other two when $J \leq 15$, but none of them had any practical problems in terms of training efficiency, at least up to $J = 20$. Note that Attn was slightly faster when $J = 20$ than when $J = 15$ because the length of each mixture signal was shorter on average due to the min mode, and probably because the computational efficiency of Attn depends slightly more on the length of each input signal than the other two: In the min mode, each mixture signal stops with the shortest source signal [22]. Table 2 shows the SI-SDR improvement obtained with each method in each condition. Interestingly, the method that showed the best performance differed depending on J: Hun tended to show relatively higher performance with smaller J, while Sink tended to show relatively higher performance with larger J. On the other hand, Attn showed the best performance when $J = 10$, but as with Hun, the performance tended to become lower as J became larger.

[3] https://github.com/asteroid-team/asteroid

Table 1. Average number of iterations per second.

Dataset	Sink	Hun	Attn
Libri5Mix	2.91	3.06	2.95
Libri10Mix	2.77	2.92	2.77
Libri15Mix	2.62	2.77	2.62
Libri20Mix	2.45	2.37	2.66

Table 2. SI-SDR improvement (dB).

Dataset	Sink	Hun	Attn	Att→Sin	Att→Hun
Libri5Mix	8.16	8.24	8.22	8.30	**8.43**
Libri10Mix	4.97	4.91	5.20	**5.29**	5.17
Libri15Mix	4.29	4.06	4.07	**4.36**	4.22
Libri20Mix	3.91	3.66	3.51	**3.92**	3.89

The relatively lower performance of Hun compared to Sink as J increases may be due to the fact that Hun is more prone to bad local optima, as mentioned earlier. In Attn, there were many cases where the attention matrices did not converge to permutation matrices as J became larger, probably due to an insufficient regularization effect. This is thought to be the cause of the relative performance degradation as J becomes larger. Of particular note is that Attn→Sink and Attn→Hun performed better than Sink and Hun, respectively, which implies the effect of Attn in avoiding bad local optima. It may be possible to further improve the performance of Attn alone by improving the λ scheduler or the loss definition for regularization, but given the above results, it seems reasonable to use Attn in combination with Sink or Hun. The above results were obtained for the case where the separation network was ConvTasNet, but a similar trend was observed for the case of DPRNN [8].

4. CONCLUSION

In this paper, we proposed AttentionPIT, a new PIT variant for end-to-end audio source separation network training that can be run in polynomial time with respect to the number of sources. The attention mechanism in AttentionPIT allows for a seamless process of updating the network parameters while finding soft output-target assignments through backpropagation. While the original PIT is inherently prone to getting stuck in bad local optimal network parameters due to its hard output-target assignment process, AttentionPIT is expected to be effective in avoiding them. In experiments on the LibriMix corpus, we found that while AttentionPIT works reasonably well on its own, it works even better when combined with SinkPIT or HungarianPIT so that AttentionPIT is run only in the early stages of training. In the current experiment, we provided only the results of AttentionPIT applied to ConvTasNet. Although we preliminarily tested its behavior when applied to DPRNN as well, we plan to more thoroughly verify its effectiveness on models other than ConvTasNet in the future.

5. REFERENCES

[1] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. ICASSP*, pp. 31–35, 2016.

[2] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *Proc. ICASSP*, pp. 246–250, 2017.

[3] Z.-Q. Wang, J. Le Roux, D. L. Wang, and J. R. Hershey, "End-to-end speech separation with unfolded iterative phase reconstruction," in *Proc. Interspeech*, 2018.

[4] D. Yu, M. Kolbæk, Z.-H. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *Proc. ICASSP*, pp. 241–245, 2017.

[5] M. Kolbæk, D. Yu, Z.-H. Tan, and J. Jensen, "Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks," *IEEE/ACM Trans. ASLP*, vol. 25, no. 10, pp. 1901–1913, 2017.

[6] Y. Luo and N. Mesgarani, "TasNet: Time-domain audio separation network for real-time, single-channel speech separation," in *Proc. ICASSP*, pp. 696–700, 2018.

[7] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time-frequency magnitude masking for speech separation," *IEEE/ACM Trans. ASLP*, vol. 27, no. 8, pp. 1256–1266, 2019.

[8] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-Path RNN: Efficient long sequence modeling for time-domain single-channel speech separation," in *Proc. ICASSP*, pp. 46–50, 2020.

[9] N. Zeghidour and D. Grangier, "Wavesplit: End-to-end speech separation by speaker clustering," *arXiv:2002.08933 [eess.AS]*, 2020.

[10] E. Tzinis, Z. Wang, and P. Smaragdis, "Sudo rm-rf: Efficient networks for universal audio source separation," in *Proc. MLSP*, pp. 1–6, 2020.

[11] H. Tachibana, "Towards listening to 10 people simultaneously: An efficient permutation invariant training of audio source separation using sinkhorn's algorithm," in *Proc. ICASSP*, pp. 491–495, 2021.

[12] S. Dovrat, E. Nachmani, and L. Wolf, "Many-speakers single channel speech separation with optimal permutation training," *arXiv:2104.08955 [cs.SD]*, 2021.

[13] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[14] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[15] N. Tomizawa, "On some techniques useful for solution of transportation network problems," *Networks*, vol. 1, no. 2, pp. 173–194, 1971.

[16] M. Yousefi, S. Khorram, and J. H. Hansen, "Probabilistic permutation invariant training for speech separation," in *Proc. Interspeech*, pp. 4604–4608, 2019.

[17] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. EMNLP*, 2015.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Adv. NeurIPS*, 2017.

[19] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR –half-baked or well done?," in *Proc. ICASSP*, pp. 626–630, 2019.

[20] P. O. Hoyer, "Non-negative matrix factorization with sparse constraints," *J. MLR*, vol. 5, pp. 1457–1469, 2004.

[21] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015.

[22] J. Cosentino, M. Pariente, S. Cornell, A. Deleforge, and E. Vincent, "LibriMix: An open-source dataset for generalizable speech separation," *arXiv:2005.11262 [eess.AS]*, 2020.

[23] M. Pariente, S. Cornell, J. Cosentino, S. Sivasankaran, E. Tzinis, J. Heitkaemper, M. Olvera, F.-R. Stoter, M. Hu, J. M. Martin-Donas, D. Ditter, A. Frank, A. Deleforge, and E. Vincent, "Asteroid: The PyTorch-based audio source separation toolkit for researchers," in *Proc. Interspeech*, 2020.

[24] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.