The Costs of Reproducibility in Music Separation Research: a Replication of Band-Split RNN

Paul Magron, Romain Serizel, Constance Douwes

***Abstract*—Band-split RNN is commonly reported as a strong baseline for music source separation due to its high performance. However, it is hard to reproduce since its full code is not available. In this paper, we propose a replication of this system, and we conduct extensive experiments to study several design choices and training strategies. Although unsuccessful in reproducing the original paper's results, we propose additional variants that ultimately reach a slightly superior performance. Our contributions are three-fold. First, this study yields several insights on the model design and training pipeline, which sheds light on potential improvements. Second, it reveals and discusses reproducibility issues, both methodological and in terms of time and energy costs. Third, our code and pre-trained models are released publicly to foster reproducible research.**

***Index Terms*—Class, IEEEtran, LaTeX, paper, style, template, typesetting.**

I. Introduction

Music source separation [1] aims to extract the instrumental tracks that add up to form an observed music song. This finds application in e.g., automatic remixing for users with cochlear implants [2], [3], karaoke/backing track generation [4], [5], or music information retrieval [6], [7].

Despite their reported remarkable performance, most recent methods suffer from several drawbacks from a *reproducible research* perspective [8]. For instance, winners of the latest music demixing challenges (MDX) [9], [10] are bags of models, which can be tedious to train as they depend on multiple base architectures. Other examples are best performing models obtained via fine-tuning on a private dataset, which makes replication impossible [11], or architectures that require such a large computing capacity that retraining is prohibitively long when only a fraction of that capacity is available [12].

As a result, band-split recurrent neural network (BSRNN) [13] appears as an appropriate candidate for a replication study. Firstly, it outperforms (more) recent methods, as well as earlier open frameworks such as Open-Unmix (UMX) [14], thus it is a rather strong baseline. Secondly, even though the model is fine-tuned on a private dataset, its base performance considered here is obtained using the openly available MUSDB18-HQ dataset [15]: this allows for fair comparison with competing techniques. Thirdly, it is a single architecture, which is easier to train and fine-tune than ensemble models. Lastly, the reported training time and hardware are rather reasonable compared to more recent variants such as [12].

P. Magron and R. Serizel are with Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France. Constance Douwes is with Laboratoire d'Informatique et Systèmes, UMR 7020, CNRS, Aix-Marseille University, Marseille, France.

Unfortunately, while the authors have shared some code for the model definition,[1] to the best of our knowledge there is no available implementation of the *full pipeline* that allows to reproduce the results reported in the paper. This includes the code for data preprocessing, detailed training scripts with optimizer parameters, and evaluation functions. Yet, these are of paramount importance for reproducing any deep learning-based system's performance.

In this paper, we propose to replicate BSRNN as closely as possible to the original paper. In particular, we describe the training and evaluation protocol in details, in order to outline difficulties encountered while replicating the work. We conduct extensive experiments to study various parameters, design choices, and training strategies. Since we were unsuccessful in reaching the performance reported in the original paper, we implement several variants that could potentially bridge this gap. This ultimately yields an *optimized* BSRNN model whose performance is slightly superior to the paper's results. An additional important contribution of our work is to discuss issues encountered while conducting such a replication, in terms of both methodology and time/energy costs. We hope that this study will contribute to raise awareness on the importance of reproducible research in our community. To foster it, we release our code and pre-trained models publicly.[2]

The rest of this paper is structured as follows. Section III describes the BSRNN model, as well as some architecture variants we investigated. Section IV presents the experimental protocol that we set up in order to reproduce the results. We then detailed the results in Section V, and subsequently discuss the costs of reproducibility in Section VI. Finally, Section VII draws some concluding remarks.

II. Problem setting

In this section, we provide an overview of recent state-of-the-art music separation models and we outline several reproducibility issues. Since BSRNN appears as a good candidate, we motivate to reproduce it.

dire que BSRNN sert de base à des développements futurs: SIMO, RoFormer, DTTNet, extension to cinematic separation [16] + possibilité d'extension à des datasets plus gros et plus récents, comme Moisesdb

A. Overview of recent models' performance

Results in terms of chunk signal-to-distortion ratio (cSDR, see Section IV-C for details about the metric) are presented in

[1] gitlab.aicrowd.com/Tomasyu/sdx-2023-music-demixing-track-starter-kit
[2] github.com/magronp/bsrnn

TABLE I
SEPARATION PERFORMANCE (cSDR IN dB) FOR VARIOUS MODELS ON THE MUSDB18-HQ TEST SET.

	Vocals	Bass	Drums	Other	Average
CWS-PResUNet [17]	8.9	5.9	6.4	5.8	6.8
KUIELab-MDX-Net [18]	9.0	7.8	7.2	5.9	7.5
Hybrid Demucs [19]	8.1	8.8	8.2	5.6	7.7
HT Demucs [11]	7.9	8.5	7.9	5.2	7.5
BSRNN [13]	10.0	7.2	9.0	6.7	8.2
SIMO-BSRNN [20]	9.7	7.8	10.1	6.6	8.5
TFC-TDF UNet v3 [21]	9.6	8.5	8.4	6.9	8.3
BS-RoFormer [12]	10.7	11.3	9.5	7.7	9.8
DTTNet [22]	10.1	7.5	7.4	6.9	8.1
Training with extra data					
Hybrid Demucs [19]	8.8	9.1	9.3	6.2	8.3
HT Demucs [11]	9.4	10.5	10.8	6.4	9.2
BSRNN [13]	10.5	8.2	10.2	7.1	9.0
BS-RoFormer [12]	12.7	13.3	12.9	9.0	12.0

Table I. First, here we detail a few problems that arise when trying to compare these models' performance.

We do not report older models' performance such as UMX or Spleeter, as they are much lower. Besides, even though the ResUNet model [23] is commonly reported, its performance is known on the non-HQ version of MUSDB18, which is a different dataset. It is occasionally reported in the same table as other models tested on MUSDB18-HQ, which is confusing since no comparison can be made as numerical results correspond to different dataset. Sometimes results are presented as being on the HQ set, as in [22], which is simply wrong. We therefore outline the importance of being extra careful when copying pasting results from other papers.

Another factor that prevent from actually comparing models is the lack of information on the inference on whole songs (see Section IV-E). The technique is used is often not disclosed, while it might have a large impact on the test results. For instance, using various parameters yields average cSDR difference of about 0.2-0.3 dB [18], [20], which is of the order of the performance difference between competing models with a similar structure.

B. *Which model is a good candidate?*

KUIELab-MDX-Net and the reported Hybrid Demucs are ensemble or bags of models, which means they combine the outputs of different model, whether different architectures, or several instances of the same model with different hyper-parameters. These are particularly tedious to optimize, and besides they don't even yield the best performance. To quote the authors of Demucs: "While suitable for a competition, such a complex model does get in the way of reproducing easily the performance achieved." [19]

DTTNet, SIMO-BSRNN, and BS-RoFormer are models that build upon BSRNN. DTTNet is a lightweight version that yields slightly lower performance, SIMO-BSRNN exploits several variants (unfortunately without discussing in details the effect of each of them specifically), and BS-RoFormer notably replaces the RNNs with transformers. As such, BSRNN still appear as relevant because it yields close-to-SOTA results, it outperforms many others, and does not requires prohibite computing ressources.

Indeed, RoFormer is prohibitively long to train.

The lower part of the Table displays results obtained with models that are trained and/or fine-tuned with extra (private) data. We report these to have an idea of how important the extra data (and corresponding data processing/preparation) is, since the delta can be up to 2.2 dB (note however that using more data allows to increase the model size, which contributes to the performance gap), which is far basically as much as the largest difference between models in the upper part of the Table (trained using the same amount of data). Finally, one cannot even use these results to assess one model's best/maximum potential, since various models are trained with different private extra datasets. As a result, they only show the overall performance of a combination of dataset, data processing, training pipeline, evaluation inference, and model architecture and optimization. This does not allow for more accurate comparison of specific components.

ajouter mention sur le fait qu'on teste aussi des trucs liés à SIMO-BSRNN

C. *Issues with BSRNN*

The authors have shared some code for the model definition,[3] to the best of our knowledge there is no available implementation of the *full pipeline* that allows to reproduce the results reported in the paper. This includes the code for data preprocessing, detailed training scripts with optimizer parameters, and evaluation functions. Yet, these are of paramount importance for reproducing any deep learning-based system's performance, as outline above regarding the importance of data and inference procedure.

Note that unofficial implementations exist, but they either report substantially lower performance to the original's,[4] are tailored and adapted to a different music separation scenario,[5] or they are adapted to alternative tasks such as speech enhancement, hence using a different pipeline that is not applicable to music separation/datasets.[6] Nevertheless, these were useful for designing our own implementation.

Model definition is not correct (does not allow to compute the original loss).

Unofficial implementation yield much lower results. They report a 6.7 SDR on vocals, which is much lower than the worst competing model in Table I.

III. MODEL AND VARIANTS

In this section, we briefly present the original BSRNN model. In an attempt to bridge the performance gap between our implementation's and the paper's results, we then propose a few architecture variants. These are illustrated in Figure 1

[3]gitlab.aicrowd.com/Tomasyu/sdx-2023-music-demixing-track-starter-kit
[4]github.com/amanteur/BandSplitRNN-PyTorch
[5]github.com/crlandsc/Music-Demixing-with-Band-Split-RNN
[6]github.com/sungwon23/BSRNN

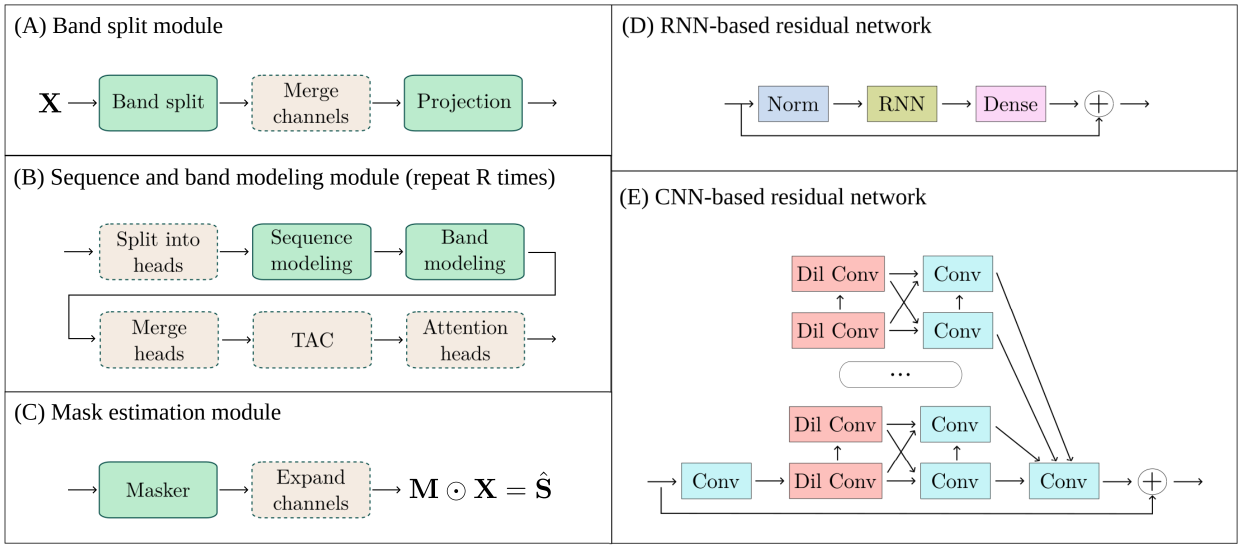

Fig. 1. Overview of BSRNN and variants, with the band split (A), sequence and band modeling (B), and mask estimation (C) modules. Blocs with dashed line contours denote variants from the original architecture. The sequence and band modeling modules can be either based on RNNs as in the original model (D), or using dilated convolutions (E).

A. *Original BSRNN architecture*

BSRNN takes as input the complex-valued short-time Fourier transform (STFT) of the mixture $\mathbf{X} \in \mathbb{C}^{F \times T}$, where F and T denote the number of frequency bands and time frames, respectively, and predict the STFT of a target source $\mathbf{S} \in \mathbb{C}^{F \times T}$. In practice, complex-valued STFTs are represented as stacks of their real and imaginary parts, since such a real-valued tensor can easily be processed with a neural network. BSRNN consists of the three following modules.

First, the *band split* module decomposes the input STFT into K subband spectrograms with variable bandwidth. These are subsequently projected into a deep latent space with dimension N, and the obtained deep subband features are then stacked back into a fullband tensor with dimensions $N \times K \times T$. The band split scheme is designed and optimized for each source specifically, which yields instrument-specific models of different size. Alternatively, one can use a finer bandsplit scheme that is shared across instruments, as proposed in a subsequent BSRNN variant paper [20]. Note however that in this paper, the shared scheme was consider because a shared encoder was used; we can instead use this shared scheme but with a source-specific encoder. On le fait pas ici. Ajouter ref à d'autres splits, comme SIMO, ou [16]

Then, the *sequence and band modeling* module applies two residual networks (one after the other), whose basic structure consists of a group normalization, a bidirectional long short-term memory (LSTM), and a dense layer. The sequence and band RNN act across the T time frames and K bands, respectively. This dual-path approach [24] optimally exploits the structure of audio/music signals by capturing dependencies across both time and frequencies. Such RNNs are stacked R times to form a deeper network.

Finally, the *mask estimation* module splits the output of the previous module into subbands, and each subband feature is fed to a multi-layer perceptron (MLP) that predicts a subband mask. The MLPs use a hidden size $\mu \times N$, where $\mu = 4$. Subband masks are then assembled into a fullband mask $\mathbf{M} \in \mathbb{C}^{F \times T}$ that is multiplied with the input STFT to yield the source estimate: $\hat{\mathbf{S}} = \mathbf{M} \odot \mathbf{X}$, which is finally reverted to time-domain via inverse STFT (iSTFT).

B. *Variants*

Here we present our proposed architecture variants, focusing on keys concepts for a clarity purpose. We refer the interested reader to our code for full implementation details.

1) Stereo modeling: By design, BSRNN is a single-channel model that processes the two channels of the (stereo) input mixture independently, as if they were two single-channel streams coming from different music songs. Not accounting for the cross-channel information might limit the system's performance, and processing the left and right channels independently might increase the computational burden unnecessarily.

To alleviate this issue, we propose a stereo variant of BSRNN that is based on simple modifications of the band split and mask estimation modules. We consider a stereo input mixture as a 2-channel STFT $\mathbf{X} \in \mathbb{C}^{2 \times F \times T}$, which is first split into subbands. Then, the left and right channels of each subband spectrogram are concatenated and jointly projected into a deep subband feature with dimension N. The sequence and band modeling module is identical, and the masker's MLP has twice as many outputs, corresponding to the two channels of the estimated source. In what follows we refer to this strategy as the "naive" stereo model. It corresponds to the "merge channels" and "expand channels" blocs in Figure 1.

Alternatively, we propose a stereo-aware variant that is based on leveraging a transform-average-concatenate (TAC) module, which was originally tailored for speech separation [25], and recently proposed for music separation in a BSRNN variant [20]. In a nutshell, the TAC module first *transforms* a multichannel input representation via a dense layer; then applies *averaging* over channels and a second dense layer; and finally *concatenates* this second layer's output with the first layer's transformed representation, before passing it to a third dense layer. This module yields a representation that shares information across channels, which makes it suitable for stereo-aware modeling. The TAC module is applied after sequence and band modeling [20], and the dense layers' activation function is either a hyperbolic tangent (TanH), as in the BSRNN variant [20], or a parametric rectified linear unit (PReLU), as in the original TAC paper [25].

2) Alternative layers: We investigate alternative layers to LSTMs in the sequence and band modeling module. Preliminary experiments showed that using gated recurrent units yield similar results to LSTMs (as outlined in other studies [16]), and replacing each RNN with a simple one-layer convolutional neural network (CNN) induce a large performance drop, although these are much faster to train because of reduced memory constraints. However, this approach is quite naive, as such CNNs cannot properly model sequential data with long-term dependencies. Therefore, we propose to use a stack of dilated CNNs, since they have shown promising for source separation, and are competitive with RNNs due to their increased receptive field [19], [26]. We implement the architecture from Wang [26], which yields a fully-convolutional model denoted band-split CNN (BSCNN). It is illustrated in Figure 1-(E).

3) Self-attention: The self-attention mechanism has shown promising in many tasks involving sequential data, including music and speech separation [27]. In particular, it was used in TFGridNet [28], a model that shares similarities with BSRNN, as it projects frequencies bands into deep embeddings, and learns dependencies across bands and time frames via a dual-path-like architecture. In TFGridNet, the self-attention mechanism was observed to yield a noticeable performance boost, with a negligible impact in terms of model size and computational burden. Therefore, we incorporate N_a attention heads with an encoding dimension E_a within each sequence and band modeling module, using the implementation from the ESPnet toolbox [29].

4) Multi-head mechanism: Chen et al. [22] proposed to replace the sequence and band modeling modules with so-called "improved" modules. These boil down to splitting the input feature of dimensions $N \times K \times T$ into H heads, thus with dimensions $H \times N/H \times K \times T$. The RNNs then operate in parallel across heads, and use a hidden dimension that is divided by a factor H. This corresponds to the "split into heads" and "merge heads" blocs in Figure 1-(B). This effectively reduces inference time as well as the number of (redundant) parameters. We implement and test such a multi-head mechanism.

IV. Experimental protocol

This section describes our experiment protocol. Since it is largely similar to that of the original BSRNN paper, we mostly focus on aspects for which variants are considered, or details that need special care and clarification.

A. Data

We use the openly available MUSDB18-HQ dataset [15] for all experiments. It consists of 150 stereo songs sampled at 44100 Hz, with pairs of mixtures and corresponding four isolated sources: `vocals`, `bass`, `drums`, and `other`. The songs are split into 86, 14, and 50 tracks for training, validation, and testing, respectively. The STFT is computed using a 2048 sample-long Hanning window with a hop size of 512.

To generate training data, we implement the same procedure as in the BSRNN paper [13]. Training tracks are first processed by a source activity detector (SAD) that removes the silent regions. Then, training data samples are generated on-the-fly as follows (we generate 20000 samples per epoch):

(i) Randomly mixing sources from different songs. Preliminary experiments did not reveal any difference between shuffling tracks only once when training starts, or re-shuffling tracks at each epoch. We use the former as it is slightly faster.
(ii) Extracting random 3 s-long chunks for each source.
(iii) Randomly adjusting each chunk's energy in a $[-10, 10]$ dB range.
(iv) Dropping each chunk with probability 0.1 to simulate silence sources.

We also consider an alternative training data generation process, without SAD preprocessing nor random chunk dropping, that uses the same augmentations as in UMX [14], i.e., randomly swapping the channels with a probability of 0.5, and rescaling the chunks' energy using a linear gain between 0.25 and 1.25.

B. Model configuration

In order to accelerate prototyping, most experiments use a small model, i.e., $N = 64$ and $R = 8$, while the original model (herein denoted *large*) uses $N = 128$ and $R = 12$. All other hyper-parameters are chosen as per the original paper [13], unless specified explicitly.

C. Metrics

We assess source separation quality in terms of signal-to-distortion ratio (SDR) expressed in dB [30]. We consider the following two variants of the SDR:

- The *utterance* SDR (uSDR) is used in MDX challenges [9], [10]. It is equal to a basic signal-to-noise ratio. We report the mean across songs.
- The *chunk* SDR (cSDR) is used in the signal separation evaluation campaign [31]. We report the median across the median over 1 s-long chunks in all songs, computed using the museval toolbox [32].

D. Training

1) Loss: BSRNN is trained using a combination loss $\mathcal{L} = \mathcal{L}_{\text{time}} + \mathcal{L}_{\text{TF}}$ that consists of a time-domain term:

$$\mathcal{L}_{\text{time}} = |\text{iSTFT}(\mathbf{S}) - \text{iSTFT}(\hat{\mathbf{S}})|_1, \tag{1}$$

and a time-frequency (TF) domain term:

$$\mathcal{L}_{\text{TF}} = |\mathbf{S}_r - \hat{\mathbf{S}}_r|_1 + |\mathbf{S}_i - \hat{\mathbf{S}}_i|_1, \tag{2}$$

where $|.|_1$ denotes the ℓ_1 norm, and the subscripts r and i respectively denote the real and imaginary parts.

2) Optimizer and hardware: In the original paper, the model is trained using the Adam algorithm with an initial learning rate $\lambda = 10^{-3}$, a batch size of 2, and 8 GPUs in parallel, yielding a *global* batch size $B = 16$. Unfortunately, we do not have access to enough (large) GPUs to obtain the same global batch size. As a result, we resort to adjusting the learning rate in order to preserve the same *effective* learning rate λ/B, which ensures similar gradient descent steps [33]. The learning rate is decayed by 0.98 every two epochs, and gradient clipping by a maximum gradient norm of 5 is applied.

Small models are trained using 4 Nvidia RTX 2080 Ti (11 GB) GPUs, except for the `drums` model with self-attention, which is trained using 4 Nvidia Tesla T4 (15 GB) GPUs because of memory constraints. Large models are trained using 2 Nvidia Tesla L40S (45 GB) GPUs.

3) Monitoring: Training is conducted with a maximum of 100 epochs in [13], which we did not observe to be enough for convergence in most cases, thus, we set this number at 200. Besides, the authors apply early stopping "when the best validation is not found in 10 consecutive epochs" [13, IV-A], which does not clearly state which quantity (loss or SDR) is monitored. By default we chose to monitor the uSDR since computing the cSDR is time-consuming, and we keep a patience of 10 epochs.

E. Inference of whole songs for evaluation

At the evaluation stage (i.e., validation or test), songs cannot be processed entirely at once by the model because of memory constraints. Consequently, they are divided into small (overlapping) segments that are fed to the separation model, and then the estimated chunks are assembled using some overlap-add (OLA) strategy to recover whole source estimates. Here, we split the songs into segments of 10 s with 10% overlap, and the estimated chunks are assembled using a linear fader to smooth their edges [19].

Note that in the BSRNN paper, the evaluation segments are 3 s-long with a 0.5 s hop size, and the exact OLA/reconstruction procedure (i.e., linear fading, windowing, trimming/concatenation) is not specified. We implement it using a rectangular window whose scale factor is set to ensure perfect reconstruction, and we will discuss its impact on test results in Section V-D. However, the exact parameters of the inference procedure was observed to have no impact on validation in preliminary experiments, as the best selected model is the same regardless of mild absolute differences in terms of validation uSDR.

F. Monitoring energy consumption

Reporting the energy consumption is crucial for evaluating the potential climate impacts of machine learning research, as such a deep learning-based project is likely to to have a substantial electricity and carbon footprint [34], [35]. As a result, we monitor the energy consumption of training the various models considered in this work. We track the energy (measured in kWh) during training via the codecarbon toolbox,[7] using a power usage effectiveness factor of 1.5, according to our computing platform's recommendation.[8] Note that even though models are trained using different GPU models (see Section IV-D), which has an impact on energy consumption [36], we use this setup as it allows to optimally exploit our available hardware.

We also estimate the total consumed energy for the project, including all other models variants (not reported in this paper), preliminary and additional experiments, prototyping, and inference. We followed the methodology proposed in the green algorithm online calculator [37], which approximates energy consumption based on hardware specifications (we consider a 3 W power per 8 GB of memory). Note that this method tends to overestimate energy compared to codecarbon, but it is closer to actual power meter readings [38].

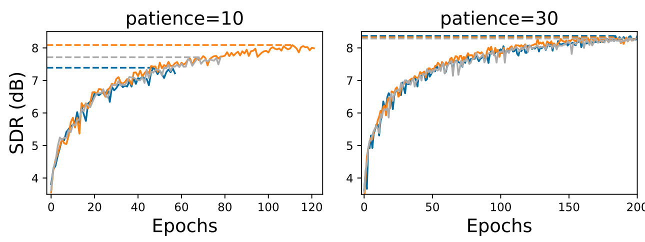

Fig. 2. Validation uSDR over epochs for the base model on the `vocals` track, with a patience of 10 (left) or 30 (right). Each color corresponds to a different run, and the dashed lines correspond to each run's best uSDR.

V. Results

This section describes our experimental results. Note that we do not report all conducted experiments, as this paper would be prohibitively long. We rather encourage the interested reader to experiment with our code for conducting further investigation, and we detail additional results in a specific document.[9]

A. Preliminary experiment

Let us first consider a *base model*, which corresponds to the small-size version of BSRNN (see Section IV-B). We train it using three different random seeds, and display the validation uSDR over epochs in Figure 2 (left) for the `vocals` track (similar results are obtained for the other sources). We observe that all runs exhibit the same trend, but they yield some variance in terms of best uSDR. This variance is explained by instabilities in the uSDR that stem from the different initial random seed [39], which ultimately causes training to stop at different epochs.

To alleviate this issue, we increase the patience parameter at 30, and we display the results in Figure 2 (right). We observe that the mean best uSDR is increased, but more importantly, its variance is largely reduced. Increasing patience is therefore effective to continue training and reduce the impact of the random seed onto the best uSDR.

[7]github.com/mlco2/codecarbon
[8]intranet.grid5000.fr/stats/indicators
[9]github.com/magronp/bsrnn/docs/analysis.md

Based on these findings, one could either reduce variance by tuning the random seed as any other hyperparameter [40], or average multiple runs / increase patience as done above. Nevertheless, to keep our experiments cost-effective and consistent with the original paper, we report a single run's results with a patience of 10, unless specified explicitly (an exception is the base model, for which we report the mean value over the three initial runs, since we have conducted these already).

B. *Validation results*

We report the best model's validation uSDR for all variants in Table II. Each line describes the difference with the base model (reported at line 1), which serves as a comparison reference for each variant.

1) Training parameters:

a) Accumulating gradients: Instead of adjusting the learning rate as described in Section IV-D2, we can accumulate gradients over steps before performing descent, which artificially increases the global batch size. Results from line 2 show that both strategies perform similarly. In what follows we adjust the learning rate as we observed it to be more stable when training larger models.

b) Monitoring criterion: While by default early stopping is performed via monitoring the uSDR (see Section IV-D3), we can instead monitor the validation loss. Both approaches yield similar results (*cf.* line 3), but we chose the former as it allows training to continue for more epochs, which is beneficial in ensuring convergence.

c) Loss: We conduct training by minimizing each term of the loss individually (see Section IV-D1). Interestingly, all loss domains (*cf.* lines 1, 4, and 5) yield similar results, which contrasts with previous studies that outlined the importance of time-domain training [41]. One explanation is that the TF domain loss (2) treats both the real *and* imaginary parts separately. This likely enforces *phase consistency* [42], [43] implicitly, which makes it equivalent to a time-domain loss.

2) Original architecture parameters:

a) STFT parameters: We consider an STFT with a 4096 sample-long window and a hop size of 1024 samples, as this setup is common among music separation models [14], [19]. As shown at line 6, this yields poor result, which we attribute to a reduced time resolution that is not compensated by the increased frequency resolution. Indeed, the latter has little impact since the band split scheme and projection in a latent space of fixed dimension occur early in the network. This notably explains why the drop is more pronounced for the `drums` than for the `bass` track, since percussive events are localized in time and thus require a refined time resolution to be properly modeled. Consistently, a larger window of 6144 points [22] yields worse results.

b) Masker size: We propose to reduce the MLP masker size by setting $\mu = 2$ (vs. $\mu = 4$ by default, see Section III-A). While this negatively affects performance for the `drums` and `other` tracks, it substantially improves the `bass` results, thus yielding a similar average performance (*cf.* line 7). Further decreasing μ at 1 degrades performance more importantly.

c) Large model: Using the original model's parameters ($N = 128$ and $R = 12$) yields a large uSDR improvement of 1.1 dB on average compared to the base model (line 8). Nonetheless, such a model was not fully converged (except for the `drums` track), thus we continue training after increasing the patience, which further improves performance by 0.3 dB (line 9). Nevertheless, this model's test set performance is still lower than the original results [13] (see Section V-D), which motivates the next experiments.

3) Architecture variants:

a) Stereo models: Our naive approach to stereo modeling results in a performance decrease, except for the `bass` track, as attested by the results from line 10. One way to bridge this gap consists in increasing the masker size ($\mu = 8$) to compensate for the larger number of outputs, as two channels must be recovered. While it partly mitigates the performance drop for `drums` and `other`, it yields worse results for the `bass` track (*cf.* line 11), and the model becomes prohibitively large when increasing N and R.

Besides, leveraging a TAC module as described in the SIMO-BSRNN variant [20] also exhibits a performance drop (see line 12). However, replacing the TanH activation with a PReLU, as when TAC was originally proposed [25], outperforms the base model, with a more significant improvement for the `bass` and `drums` tracks (line 13).

b) BSCNN: We first consider a small variant of our proposed BSCNN ($N = 32$, $R = 4$, no band modeling), as this allows to conduct faster preliminary experiments to tune the corresponding hyperparameters (number of dilated layers, kernel sizes, etc.). We then report at line 14 the performance of such an optimized BSCNN model with larger size ($N = 64$, $R = 8$). We observe that, while it is faster to train and slightly lighter, BSCNN is outperformed by the RNN-based network. A refined and source-specific tuning of the convolution parameters (e.g., considering different parameters for band and sequence modeling) could improve performance, which we leave to future investigation.

c) Self-attention: As attested by lines 15 and 16, incorporating self-attention heads is beneficial, except for the `other` track. In particular, such a small-size `drums` model with attention performs similarly to a large model with no attention (*cf.* line 8). This approach is therefore an interesting alternative to larger and much more computationally demanding models such as those that completely replace RNNs with transformers [12].

d) Multi-head module: Results from line 17 show that the multi-head module performs worse for most sources, except for the `vocals` track. Setting $R = 4$ [22] or further increasing H degrades performance more importantly, though the model gets much lighter.

4) Data generation: Instead of randomly dropping each *chunk* to simulate silent sources when generating data [13, IV-A-2], we instead only drop the *target* source, which improves performance for the `bass` track (*cf.* line 18). Besides, not performing SAD and applying the UMX-like augmentations further improves performance on average, as observed at line 19. This suggests that there is room for improvement for our SAD implementation, since this preprocessing is alleged

TABLE II
COMPARISON OF VARIOUS MODELS' PERFORMANCE: BEST MODEL'S VALIDATION uSDR (IN dB), TOTAL NUMBER OF PARAMETERS (IN MILLIONS, "-" DENOTES THE SAME VALUE AS IN THE PRECEDING LINE), AND ESTIMATED ENERGY FOR TRAINING ALL SOURCES (IN kWh).

		Validation uSDR (dB)					# Parameters (M)	Energy (kWh)
		Vocals	Bass	Drums	Other	Average		
1	Base model: $N = 64$, $R = 8$	7.7	6.1	9.7	4.8	7.1	32.3	127
	Training parameters							
2	Accumulating gradients	8.0	5.8	9.6	4.9	7.1	-	129
3	Monitoring with the loss	7.5	6.4	9.3	4.8	7.1	-	120
4	Loss domain: time	7.9	6.1	9.4	4.9	7.1	-	116
5	Loss domain: TF	7.9	6.4	9.6	4.9	7.2	-	131
	Base parameters							
6	STFT: window=4096, hop=1024	7.3	5.9	8.7	4.4	6.6	37.1	58
7	Masker factor $\mu = 2$	7.9	6.8	9.4	4.4	7.1	20.6	110
8	Large model: $N = 128$, $R = 12$	9.2	7.3	10.3	5.8	8.2	146.7	230
9	Large model and patience=30	9.5	7.8	10.3	6.3	8.4	-	354
	Stereo models							
10	Naive	7.7	6.6	8.4	4.0	6.7	37.1	78
11	Naive, with $\mu = 8$	7.9	6.1	8.7	4.3	6.7	81.1	87
12	TAC with TanH activation	7.6	6.0	9.6	4.3	6.8	34.7	117
13	TAC with PReLU activation	7.9	6.5	10.0	4.7	7.3	34.7	128
	Architecture variants							
14	BSCNN	7.3	5.9	9.0	4.2	6.6	29.7	113
15	Attention: $N_a = 1$, $E_a = 8$	7.7	7.4	10.4	4.8	7.6	33.0	151
16	Attention: $N_a = 2$, $E_a = 16$	8.2	7.7	10.4	4.9	7.8	33.2	157
17	Multi-head module, $H = 2$	7.6	5.5	9.1	4.0	6.6	22.0	91
	Data generation							
18	Silent target (instead of all sources)	7.9	6.6	9.5	4.4	7.1	32.3	110
19	No SAD; UMX-like augmentations	8.2	6.9	9.5	5.3	7.5	-	135
	Optimized models							
20	Attention, patience = 30, no SAD	10.1	9.1	10.9	6.7	9.2	149.9	426
21	+ TAC	10.2	10.2	11.3	6.9	9.6	164.1	508

to greatly benefit the separation. Note however that it is not clear from [13] to what extent, since its impact is not evaluated in a specific experiment.

We use this alternative data generation process to train a large model ($N = 128$ and $R = 12$) with attention ($N_a = 2$ and $E_a = 16$), with an increased patience of 30. According to the results displayed at line 20, this *optimized* model outperforms our previous large model (line 9), at the cost of a moderate increase in number of parameters.

C. Energy consumption

We report the energy consumed for each model's training in the last column of Table II. Note that line 1 corresponds to the mean over the three runs (see Section V-A). Most small-size model variants exhibit a similar energy cost, except for those with lighter architecture / memory requirements, or generally faster convergence (lines 6, 10, and 17), which all reduce energy consumption. Conversely, the larger energy costs of models with attention (lines 15 and 16) is due to requiring more epochs for reaching convergence. This highlights the importance of considering both performance *and* energy consumption when performing model selection.

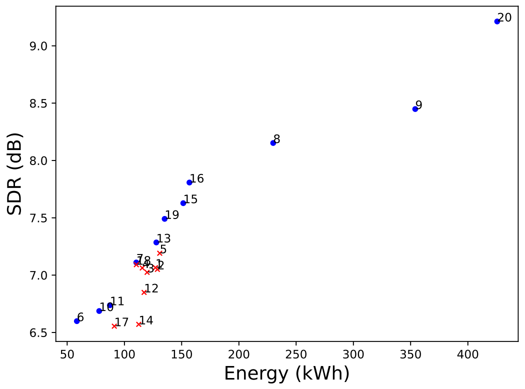

Fig. 3. Performance vs. energy for model variants (each number corresponds to the line number in Table II). Blue dots highlight models that are optimal in a Pareto sense.

We further illustrate this by displaying performance vs. energy in Figure 3, where we highlight points that are optimal in a Pareto sense [44], that is, such that there is no other point yielding both a higher SDR and a lower energy. commenter

Overall, training all the models discussed in this table amounts to xx kWh, which represents about xx times the energy cost of the single best model (line 20). While this ratio might seem reasonable at first glance, we also estimate the total consumed energy for the project, as described in Section IV-F. This amounts to xx kWh, which is more than xx times the energy consumption of training the best model, or xx times that of the base model. This is equivalent to the yearly electricity consumption of about xx persons in France (where computations have been performed)[10], or about xx km of electric car.[11]

In all fairness, part of this energy cost is due to our own implementation errors, which resulted in, e.g., interrupted or redundant training runs. However, we believe that most music/audio researchers are not machine learning or coding experts, therefore these pitfalls are likely common. Be that as it may, these are typical ratios associated with project development: for instance the authors from a previous study [35] report that the electricity cost of their whole project is about 2000 times that of training a single model, and they report running almost 5000 jobs in total, including many that crashed.

Overall, such a project has a substantial energy cost, which should be systematically reported, analyzed, and taken into account when performing model selection. Besides, this project's footprint would have been substantially reduced upon availability of the code, as many trial and error experiments could have been avoided.

D. Test results

The separation results on the test set are reported in Table III. First, we compare our linear fader-based inference procedure to an OLA-based procedure, similar to that of BSRNN (see Section IV-E). Varying hop sizes using this OLA procedure yields a 0.1 dB difference on the `vocals` track, which is consistent with the original results [13, Table II]. The OLA and fader techniques yield similar results on average, but using OLA with a 0.5 s hop size largely increases inference time by a factor of 6. This confirms the advantage of our linear fader for faster inference. Similar conclusions can be drawn using the optimized model instead.

We also report the original test results [13] in Table III, and we observe that our implementation falls behind these by 0.5 dB on average. This motivates further refining our implementation to better match these. Nevertheless, our optimized model is able to bridge this performance gap, as it largely improves our implementation of the model, and it even outperforms the original paper's results by 0.2-0.3 dB. This optimized model is therefore an interesting alternative baseline to BSRNN, as it is openly available and yields similar to slightly better results.

VI. Discussion

This replication study has shown insightful in many regards. It notably extended the original paper's analysis by investigating a variety of parameters such as architectural (e.g., the masker's MLP size), training-related (e.g., the contribution of each term in the loss), or data-related (e.g., the SAD preprocessing). Explicitly reporting such results, even though sometimes *negative*, is beneficial to other researchers as it spare them a costly (re)investigation. Besides, these detailed results pave the way for potential improvements in terms of model reduction. For instance, reducing the masker size or leveraging our stereo approach yielded a similarly performing but lighter `bass` model. Alternatively, using a small `drums` model with attention is both competitive with a larger variant and significantly lighter. Our experiments also illustrate that one cannot reuse *as-is* building blocks from other papers. For instance, the dilated convolution [26], or the multi-head modules [22], were shown effective in these papers, but they were used in conjunction with other architectural blocks and integrated in a different pipeline. The same applies to the TAC module, which was incorporated to BSRNN specifically [20], but in conjunction with other stuffs (a common encoder and BS scheme for all sources, a mixing constraint, and a context-aware masker). Here we observed that it does not yield any substantial improvement.

While our analysis mostly focused on performance results in terms of numerical values of SDRs, the fundamental reproducibility issues lie beyond this sole aspect, which echoes prior work on reproducibility in music information retrieval [8], [45]. A first problem is that the discrepancy between results is currently unexplained, and can come from whether one or a combination of factors including, but not limited to, all variants considered in this study. Since no official implementation for the whole pipeline is available, there is no guarantee that we did not make any mistake or that there is not a substantial mismatch between implementations. A second problem is that the replication process itself is prohibitively time-consuming, which hampers research development. Indeed, the whole implementation of the project (including prototyping and debugging), as well as all extensive training to tune hyper-parameters, could be avoided if the code (along with training routine, hyper-parameters and eventually weights) was released. This in turn would promote faster research developments that build upon this project, rather than re-implementing it.

Finally, the specific deep learning nature of this project brings additional reproducibility problems. Indeed, while deep learning research has a considerable environmental footprint and should move towards its systematic monitoring and reduction [34], [35], non-reproducible research exacerbates this issue, as discussed in Section V-C. Besides, deep models' performance strongly depend on the hardware, as it affects the training protocol. On the one hand, given the impact of the (global) batch size on training speed and convergence, and thus on the final performance, accommodating the available hardware requires to consider and carefully fine-tune adaptation strategies (e.g., scaling the learning rate or accumulating gradients), which represents a substantial extra experimental burden. On the other hand, reproducibility might be plain impossible in practice for public institutions with limited computing capacity. As outlined in Section II, training the four source models of the BS-RoFormer [12] would take about 1.5 years using our largest cluster. In a nutshell, these various costs

[10]www.data.gouv.fr/reuses/consommation-par-habitant-et-par-ville-delectricite-en-france/

[11]ev-database.org/cheatsheet/energy-consumption-electric-car

TABLE III
SOURCE SEPARATION PERFORMANCE ON THE MUSDB18-HQ TEST SET (USDR AND CSDR, IN DB).

	Vocals		Bass		Drums		Other		Average	
	uSDR	cSDR	uSDR	cSDR	uSDR	cSDR	uSDR	cSDR	uSDR	cSDR
Paper's results [13]	10.0	10.0	6.8	7.2	8.9	9.0	6.0	6.7	7.9	8.2
Our implementation										
Fader	9.2	9.1	6.5	7.7	8.6	8.1	5.4	5.7	7.4	7.7
OLA, hop = 1.5 s	9.1	-	6.5	-	8.3	-	5.4	-	7.3	-
OLA, hop = 0.5 s	9.2	-	6.6	-	8.4	-	5.5	-	7.4	-
Optimized model	9.7	9.9	7.4	8.9	9.6	9.2	5.8	6.1	8.1	8.5
+ TAC	9.8	9.8	8.1	9.8	10.0	10.3	5.8	6.3	8.4	9.1

and issues advocate for a more reproducible research.

This echoes similar works from other machine learning fields [46], [47], and could also be extended to similar tasks such as speech enhancement and separation, which are prone to similar pitfalls.

VII. CONCLUSION

In this paper, we have replicated and extensively analyzed the BSRNN model for music source separation. We implemented and released a fully functioning model, whose performance (whether optimized or not) makes it an interesting alternative baseline to that of the original BSRNN, as it is openly available, thus reproducible. We also discussed the various costs associated with this project, notably in terms of energy consumption.

Beyond its focus on performance, the core contribution of this work is to recall that reproducibility is a fundamental aspect of the scientific endeavour. We encourage our colleagues to adopt open research practices, notably via releasing their code with proper documentation [8]. When this is not possible due to copyright matters or company policy, we respectfully advocate that papers resulting from such research should be considered mostly for their methodological or theoretical merits, and numerical performance results should be reported for comparison with extra care - or not at all. Such practices will foster a more transparent, reliable, and cost-effective research.

VIII. ACKNOWLEDGEMENTS

All computation were carried out using the Grid5000 (https://www.grid5000.fr) testbed, supported by a French scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations. We thank Jianwei Yu (author of the BSRNN paper) for answering some of our implementation-related questions. We also thank Christopher Landschoot for fruitful discussion related to his own unofficial BSRNN implementation.

REFERENCES

[1] E. Cano, D. FitzGerald, A. Liutkus, M. D. Plumbley, and F. Stöter, "Musical source separation: An introduction," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 31–40, Jan. 2019.

[2] J. Pons, J. Janer, T. Rode, and W. Nogueira, "Remixing music using source separation algorithms to improve the musical experience of cochlear implant users," *The Journal of the Acoustical Society of America*, vol. 140, no. 6, p. 4338–4349, Dec. 2016.

[3] S. Tahmasebi, T. Gajęcki, and W. Nogueira, "Design and evaluation of a real-time audio source separation algorithm to remix music for cochlear implant users," *Frontiers in Neuroscience*, vol. 14, no. 434, May 2020.

[4] H. Tachibana, Y. Mizuno, N. Ono, and S. Sagayama, "A real-time audio-to-audio karaoke generation system for monaural recordings based on singing voice suppression and key conversion techniques," *Journal of Information Processing*, vol. 24, no. 3, pp. 470–482, May 2016.

[5] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, D. FitzGerald, and B. Pardo, "An overview of lead and accompaniment separation in music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1307–1335, August 2018.

[6] N. Ono, K. Miyamoto, H. Kameoka, J. Le Roux, Y. Uchiyama, E. Tsunoo, T. Nishimoto, and S. Sagayama, *Harmonic and Percussive Sound Separation and Its Application to MIR-Related Tasks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 213–236.

[7] L. Lin, Q. Kong, J. Jiang, and G. Xia, "A unified model for zero-shot music source separation, transcription and synthesis," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, November 2021, pp. 381–388.

[8] B. McFee, J. W. Kim, M. Cartwright, J. Salamon, R. M. Bittner, and J. P. Bello, "Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 128–137, Jan. 2019.

[9] Y. Mitsufuji, G. Fabbro, S. Uhlich, F.-R. Stöter, A. Défossez, M. Kim, W. Choi, C.-Y. Yu, and K.-W. Cheuk, "Music demixing challenge 2021," *Frontiers in Signal Processing*, vol. 1, 2022.

[10] G. Fabbro, S. Uhlich, C.-H. Lai, W. Choi, M. Martínez-Ramírez, W. Liao, I. Gadelha, G. Ramos, E. Hsu, H. Rodrigues, F.-R. Stöter, A. Défossez, Y. Luo, J. Yu, D. Chakraborty, S. Mohanty, R. Solovyev, A. Stempkovskiy, T. Habruseva, N. Goswami, T. Harada, M. Kim, J. H. Lee, Y. Dong, X. Zhang, J. Liu, and Y. Mitsufuji, "The sound demixing challenge 2023 – music demixing track," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, Apr. 2024.

[11] S. Rouard, F. Massa, and A. Défossez, "Hybrid transformers for music source separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2023.

[12] W.-T. Lu, J.-C. Wang, Q. Kong, and Y.-N. Hung, "Music source separation with band-split rope transformer," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2024.

[13] Y. Luo and J. Yu, "Music source separation with band-split RNN," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 1893–1901, May 2023.

[14] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix - a reference implementation for music source separation," *Journal of Open Source Software*, 2019.

[15] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "MUSDB18-HQ - an uncompressed version of MUSDB18," Dec. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3338373

[16] K. N. Watcharasupat, C.-W. Wu, Y. Ding, I. Orife, A. J. Hipple, P. A. Williams, S. Kramer, A. Lerch, and W. Wolcott, "A generalized bandsplit neural network for cinematic audio source separation," *IEEE Open Journal of Signal Processing*, vol. 5, pp. 73–81, 2024.

[17] H. Liu, Q. Kong, and J. Liu, "CWS-PResUNet: Music source separation with channel-wise subband phase-aware ResUNet," in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, 2021.

[18] M. Kim, W. Choi, J. Chung, D. Lee, and S. Jung, "KUIELab-MDX-Net: A two-stream neural network for music demixing," in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, 2021.

[19] A. Défossez, "Hybrid spectrogram and waveform source separation," in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, 2021.

[20] Y. Luo and R. Gu, "Improving music source separation with simo stereo band-split rnn," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2024.

[21] M. Kim, J. H. Lee, and S. Jung, "Sound demixing challenge 2023 music demixing track technical report: TFC-TDF-UNET V3," 2023, arXiv preprint arXiv:2306.09382.

[22] J. Chen, S. Vekkot, and P. Shukla, "Music source separation based on a lightweight deep learning framework (DTTNET: DUAL-PATH TFC-TDF UNET)," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2024.

[23] Q. Kong, Y. Cao, H. Liu, K. Choi, and Y. Wang, "Decoupling magnitude and phase estimation with deep ResUNet for music source separation," in *Proc. of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*. ISMIR, Nov. 2021, pp. 342–349.

[24] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path RNN: Efficient long sequence modeling for time-domain single-channel speech separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[25] Y. Luo, Z. Chen, N. Mesgarani, and T. Yoshioka, "End-to-end microphone permutation and number invariant multi-channel speech separation," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6394–6398.

[26] J. Wang, "An Efficient Speech Separation Network Based on Recurrent Fusion Dilated Convolution and Channel Attention," in *Proc. INTERSPEECH 2023*, 2023, pp. 3699–3703.

[27] A. Pandey and D. Wang, "Dense CNN with self-attention for time-domain speech enhancement," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, p. 1270–1279, March 2021.

[28] Z.-Q. Wang, S. Cornell, S. Choi, Y. Lee, B.-Y. Kim, and S. Watanabe, "TF-GRIDNET: Making time-frequency domain models great again for monaural speaker separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2023.

[29] B. Yan, J. Shi, Y. Tang, H. Inaguma, Y. Peng, S. Dalmia, P. Polák, P. Fernandes, D. Berrebbi, T. Hayashi, X. Zhang, Z. Ni, M. Hira, S. Maiti, J. Pino, and S. Watanabe, "ESPnet-ST-v2: Multipurpose spoken language translation toolkit," in *Proc. Annual Meeting of the Association for Computational Linguistics*, 2023, pp. 400–411.

[30] E. Vincent, R. Gribonval, and C. Févotte, "Performance Measurement in Blind Audio Source Separation," *IEEE Transactions on Speech and Audio Processing*, vol. 14, no. 4, pp. 1462–1469, July 2006.

[31] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *Proc. International Conference on Latent Variable Analysis and Signal Separation*, 2018.

[32] F.-R. Stöter, A. Liutkus, D. Samuel, L. Miner, F. Voituret, pyup.io bot, S. Bot, and tobeperson, "sigsep/sigsep-mus-eval: museval 0.4.0," Feb. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4486535

[33] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," *ArXiv*, vol. abs/1706.02677, 2017.

[34] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *Journal of Machine Learning Research (JMLR)*, vol. 21, no. 1, Jan. 2020.

[35] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 09, pp. 13 693–13 696, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/7123

[36] R. Serizel, S. Cornell, and N. Turpault, "Performance above all ? energy consumption vs. performance for machine listening, a study on dcase task 4 baseline," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2023.

[37] L. Lannelongue, J. Grealey, and M. Inouye, "Green algorithms: quantifying the carbon footprint of computation," *Advanced science*, vol. 8, no. 12, p. 2100707, 2021.

[38] M. Jay, V. Ostapenco, L. Lefèvre, D. Trystram, A.-C. Orgerie, and B. Fichel, "An experimental comparison of software-based power meters: focus on CPU and GPU," in *CCGrid 2023-23rd IEEE/ACM international symposium on cluster, cloud and internet computing*. IEEE, 2023, pp. 1–13.

[39] X. Bouthillier, P. Delaunay, M. Bronzi, A. Trofimov, B. Nichyporuk, J. Szeto, N. Mohammadi Sepahvand, E. Raff, K. Madan, V. Voleti, S. Ebrahimi Kahou, V. Michalski, T. Arbel, C. Pal, G. Varoquaux, and P. Vincent, "Accounting for variance in machine learning benchmarks," in *Proc. Machine Learning and Systems*, vol. 3, 2021.

[40] S. Bethard, "We need to talk about random seeds," 2022, arXiv preprint arXiv:2210.13393.

[41] J. Heitkaemper, D. Jakobeit, C. Boeddeker, L. Drude, and R. Haeb-Umbach, "Demystifying TasNet: A dissecting approach," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[42] J. L. Roux, N. Ono, and S. Sagayama, "Explicit consistency constraints for STFT spectrograms and their application to phase reconstruction," in *ITRW on Statistical and Perceptual Audio Processing (SAPA 2008)*, 2008, pp. 23–28.

[43] S. Wisdom, J. Hershey, K. Wilson, J. Thorpe, M. Chinen, B. Patton, and R. A. Saurous, "Differentiable consistency constraints for improved deep speech enhancement," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019.

[44] C. Douwes, G. Bindi, A. Caillon, P. Esling, and J.-P. Briot, "Is quality enough$f$ integrating energy consumption in a large-scale evaluation of neural audio synthesis models," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2023.

[45] J. Six, F. Bressan, and M. Leman, "A case for reproducibility in MIR: Replication of 'a highly robust audio fingerprinting system'," *Transactions of the International Society for Music Information Retrieval*, Sep 2018.

[46] M. Ferrari Dacrema, P. Cremonesi, and D. Jannach, "Are we really making much progress? a worrying analysis of recent neural recommendation approaches," in *Proceedings of the 13th ACM Conference on Recommender Systems*, ser. RecSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 101–109. [Online]. Available: https://doi.org/10.1145/3298689.3347058

[47] A. Belz, S. Agarwal, A. Shimorina, and E. Reiter, "A systematic review of reproducibility research in natural language processing," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, P. Merlo, J. Tiedemann, and R. Tsarfaty, Eds. Online: Association for Computational Linguistics, Apr. 2021, pp. 381–393. [Online]. Available: https://aclanthology.org/2021.eacl-main.29/