Master thesis on Sound and Music Computing

Universitat Pompeu Fabra

On Loss Functions for Music Source Separation

Enric Gusó Muñoz

Supervisor: Jordi Pons i Puig, PhD.

August 2020

Contents

Acknowledgement

I would like to express my sincere gratitude to:

- Jordi Pons, for his patience and very valuable advice.
- My mother Magda, for insisting on enrolling me in an Engineering degree some years ago.
- My partner Gemma, for paying half of the GPU electricity bill without complaining.

Abstract

Despite that L1 and L2 loss functions do not represent any perceptually-related information besides waveform-matching, these achieve remarkable results when used to train music source separation models. Our work contributes in extending the existing literature on loss functions for training deep learning audio models — to keep understanding of the pros and cons of several loss functions (including: L1, L2 and perceptually motivated losses) in a standardized evaluation framework.

In this work we focus on defining an evaluation framework for a fair comparison among losses — because we found difficult to extract conclusions out of the existing body of literature. Generally, loss improvements are presented along with additional model modifications (e.g. different data augmentation, or different model topology), making it difficult to assess the loss contribution to the results. This study focus on standardizing the evaluation process via employing the same dataset, the same data augmentation strategy and the same model topology — while varying its loss. The alternative losses we consider are based on cross-entropy, scale invariant SDR, multi-resolution STFT, and phase sensitive losses among others.

Keywords: Music Source Separation; Deep Learning for Audio; Loss Functions

Chapter 1

Introduction

The process of sound mixing consists in bringing together different audio signals (or sources) to create a mix (or mixture). Source separation is the process of undoing this, un-mixing it, with the goal to recover the original signals without little or any additional information. Humans do this process seamlessly. Our brain has the ability to focus its auditory attention on a specific component (e.g. a particular conversation in a noisy environment such as a in a *cocktail party* [1]) and filters out the rest of stimuli. Modeling this phenomenon has inspired researchers for decades, but there is no consensus for a solution to this problem.

While source separation is also of relevance for the speech processing field, e.g., to increase the intelligibility of the speakers, throughout our work we focus on music source separation — where our goal is to separate the sources (or *stems*: the outputs of a mix bus or the sub-mix of all the tracks corresponding to a particular instrument) from the master or mixture as depicted in Figure 1. Applications range from impacting musical genres that rely on remixing (i.e. to combine stems from different mixtures or with newly created sources), to automatically generating karaoke tracks, or to enabling spatial audio up-mixes of large music catalogs.

Traditionally, the popular approach to music source separation was to use matrix decomposition algorithms like Non-negative Matrix factorization (NMF) [2] or Independent Component Analysis (ICA) [3] on magnitude spectograms, but nowadays

deep learning dominates the field as documented in the Signal Separation Evaluation Campaign (SiSEC) [4], which establishes a common framework for model comparison [1]. In this framework, that we follow as a basis for the development of our work, the task consists in recovering four stereo stems ('drums', 'bass', 'vocals' and the residual 'other') from stereo mixes. Such task is formalized with the MUSDB18 dataset and the evaluation metrics as defined by the *bss_eval* toolkit [5].

Deep learning methods that are now widely used normally rely on supervised learning. Thus meaning that for training such models one needs to define a loss function, which maps a set of parameter values onto a scalar value that indicates how well the model performs by setting a distance between the model's *output* and the ground truth or *target*. Most successful methods presented on the SiSEC are based on the spectral magnitude estimation through supervised learning using L1 or L2 loss functions. So does Open-Unmix [6], the open-sourced model that will serve us as baseline.

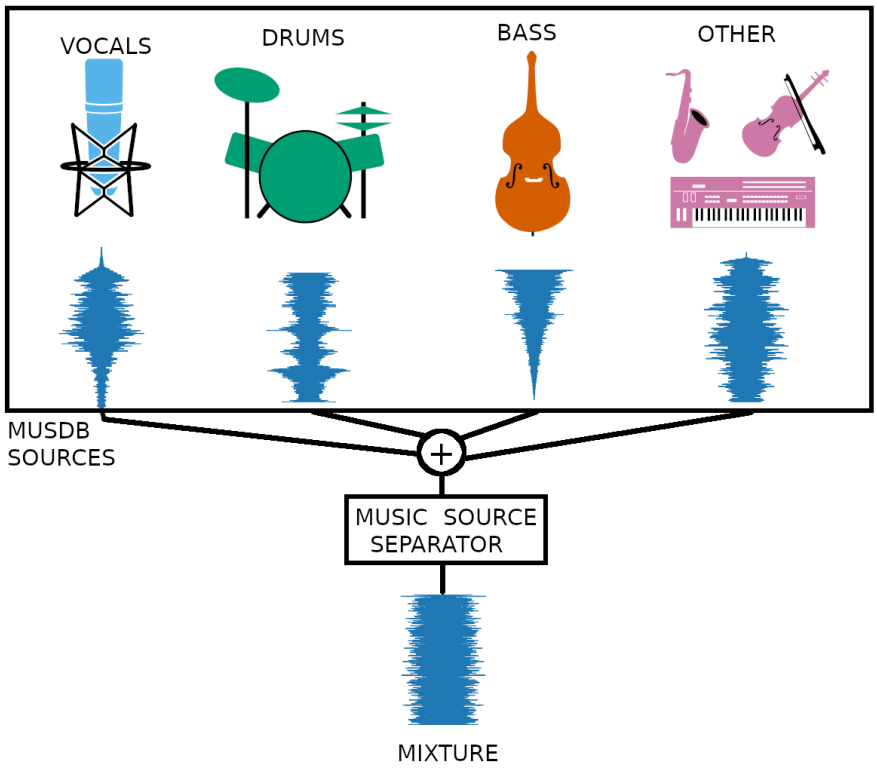

Figure 1: The SiSEC campaign employs the MUSDB dataset framework.

1.1 Motivation: the L2 waveform-loss problem

In practice, the L2 loss is used in its averaged version (mean squared error, MSE) in order to make it agnostic to signal length. Being Y the target and $\hat{Y}$ the model's output or estimate, and being N the signal length, they can be expressed as:

$$MSE = \frac{1}{N}\sum_{n=0}^{N}(Y_n - \hat{Y}_n)^2 \tag{1.1}$$

However, using the MSE as a loss for training deep neural networks presents some issues. Take, e.g., the recording of a guitar (a) from Figure 2 as a target signal, and signals (b), (c), and (d) as hypothetical outputs from a separator model:

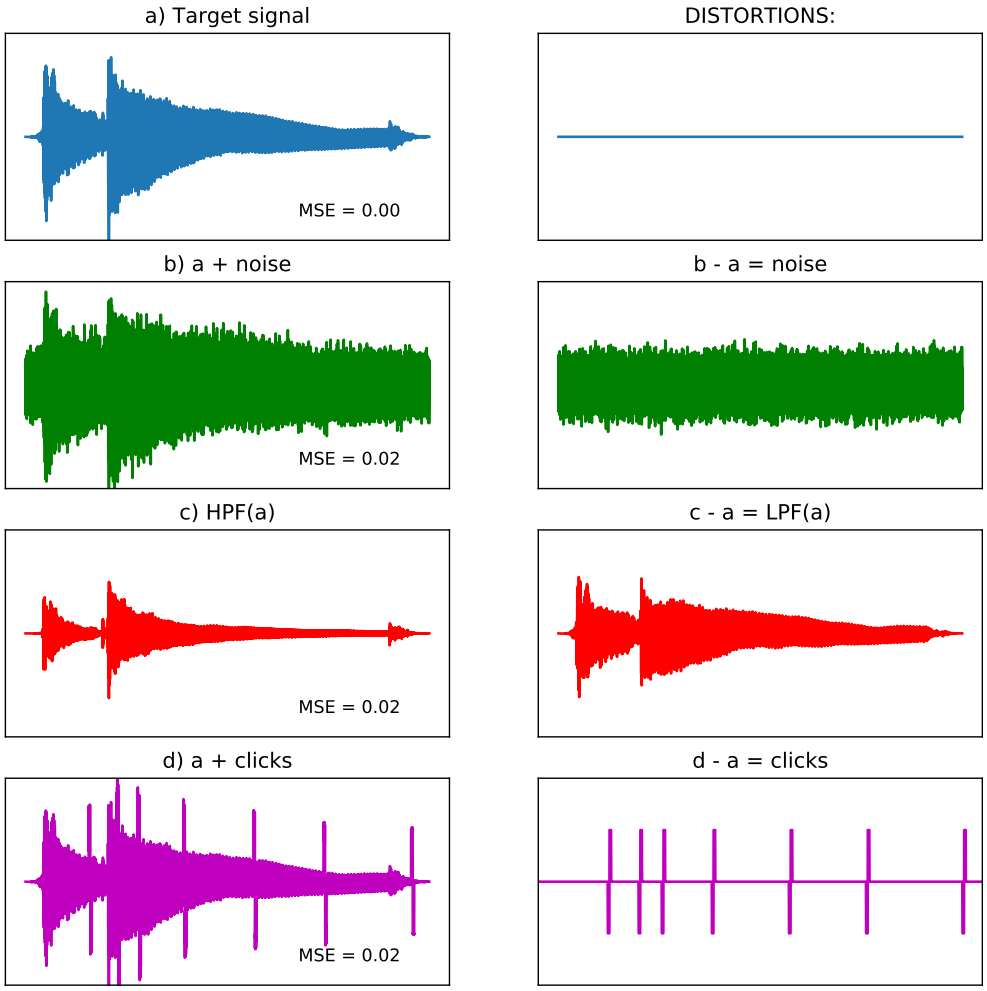

Figure 2: Perceptually different distortions with the same MSE. HPF stands for high-pass filter, and LPF stands for low-pass filter.

While all three outputs are at the same MSE distance from target (a) (i.e. they obtain the same MSE), it should be obvious to the experienced eye that all three will be perceptually very different. It is not trivial to assert which output is more preferable: while signals (b) and (d) would probably be less pleasant to the human ear compared to (c), it should be possible to de-noise and de-click them in a post-processing step, while the low frequencies filtered by a high-pass filter ($HPF(a)$) are irrecoverable.

Moreover, this MSE's lack of perceptual insight can be seen from another perspective. Take now a phase inversion and an all-pass filtered version of a target (a) that we depict in Figure 3:

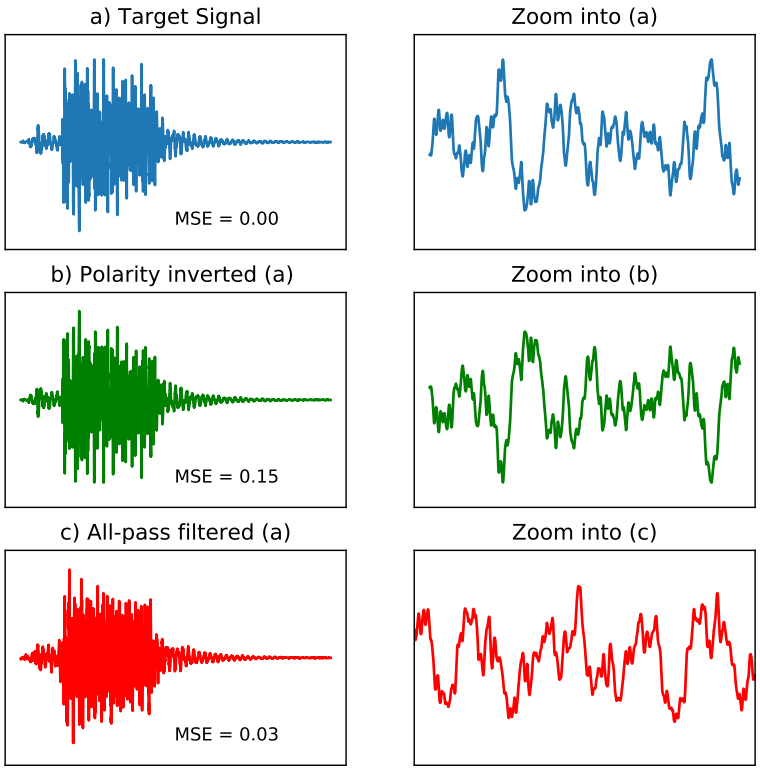

Figure 3: Inaudible phase distortions yield high MSEs.

As shown in the zoomed versions on the right, phase inversion (b) yields the same waveform but with flipped polarity, while (c) presents a different shape. Thus meaning that from the MSE perspective they will have different values. However, as these signals sound the same, the loss should at least have similar values among both outputs, and being perceptually identical to the target, the value should additionally

be very small. In reality, both outputs have different and high loss values, denoting how flawed the MSE is from a perceptual perspective. On top of that, note that the amplitude-distorted signal (see Figure 2 – c, with 0.02 MSE) achieves a lower loss than the perceptually identical polarity inverted (see Figure 2 – b, with 0.15 MSE).

Besides, we want to remark that such analysis could be easily extrapolated from the L1 waveform-loss (mean absolute error, MAE) case to the spectral domain L1 and L2 losses. Throughout the rest of the document, specially in Section 2, we will bring additional insights and discussion on how to address some of the aforementioned problems.

In the light of the above, regression losses like L1 and L2 present the problem that they are not necessarily correlated with human perception and therefore they could provide optimization paths that are not perceptually meaningful, constituting one major obstacle for the development of deep learning models for audio processing.

1.2 Research goals

Concluding this introduction, it is important to point out that most successful deep learning methods are based on optimizing regression losses with the intrinsic issues we just highlighted above. For this reason, and given that we might have identified an opportunity to improve the performance and generalization of music source separation models, we are interested in studying the different losses that have been proposed in the literature. With this goal in mind, and considering how sensible deep learning models are to their loss functions, our intention is:

- To review and summarize the different losses that have been proposed for training deep learning models for processing audio.
- To investigate the performance of those for the music source separation task.

Furthermore, we are not aware of any empirical study that consistently evaluates the losses landscape for music source separation. Most research articles focus on

proposing a new loss or an architecture, and only benchmark against a handful of models that might even consider different implementations and novel data augmentation setups. Throughout our work, we will focus on a reference open-source implementation, OpenUnmix [6], and simply elaborate on top of it to consistently benchmark an extensive set of losses. In short, our aim is:

- To adapt Open-Unmix model to consistently benchmark audio processing losses.
- To run an empirical study of the most promising losses we identify in the literature.

1.3 Structure of the Report

Once presented the general scope of this work, the rest of this thesis is structured into three blocks:

- In Chapter 2 we list the the different audio loss function we have identified in the literature that could be used for training music source separation models, while we describe and formalize them.
- In Chapter 3 we report our methodology: the baseline model and our adaptation (i.e. a modified *Open-Unmix* trained over the *MUSDB* dataset).
- Finally, in Chapters 4 and 5 we present and discuss the results, and draw some conclusions out of them.

Chapter 2

Losses for audio processing

Researchers have proposed several alternatives to L2 that can be found in recent literature. Before going into the details for each particular loss, we provide a glance to the state of the art in the following taxonomy:

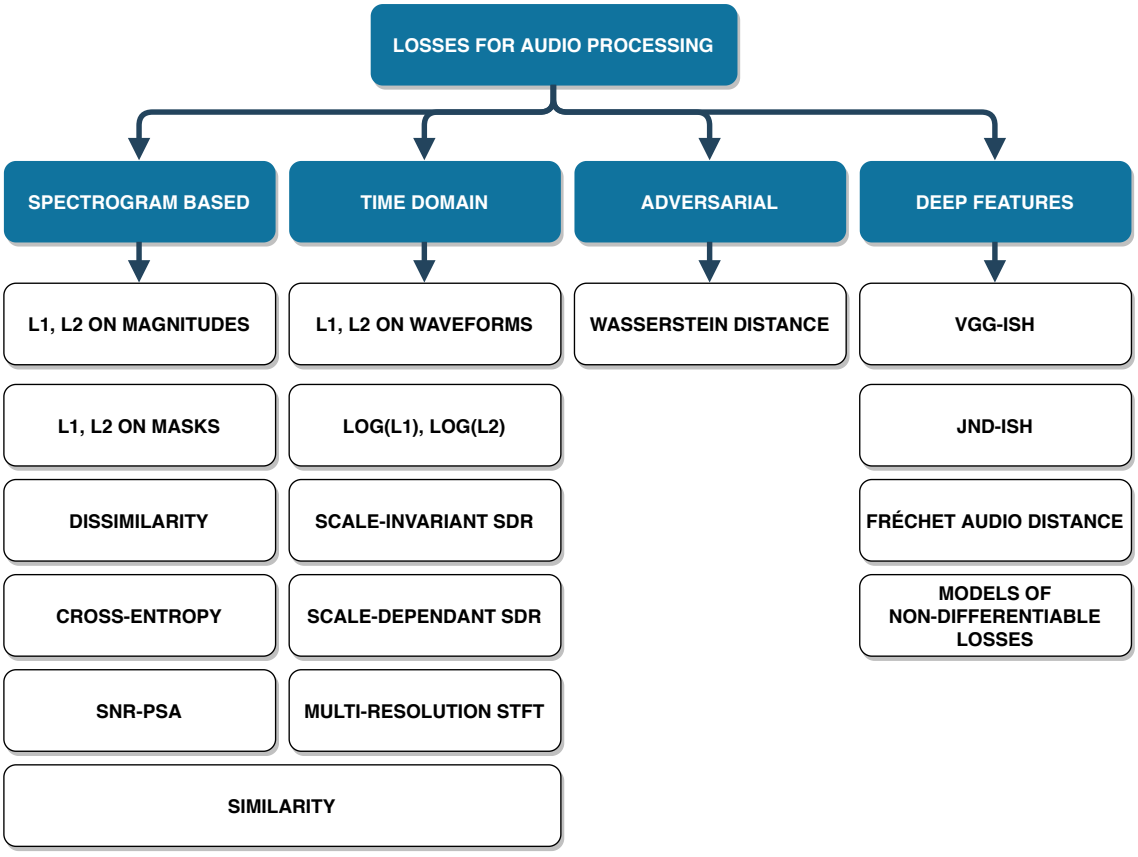

Figure 4: Taxonomy for the losses found in our literature review.

We can cluster the losses into four categories. First and foremost, traditional spectrogram losses may be computed with an spectral source separator such as Open-Unmix [6]. Next, performing the ISTFT inside the model additionally allows us to implement losses that are normally used in time domain models. Besides, we can adapt the whole training loop from supervised learning into an semi-supervised approach with adversarial losses. Lastly, deep features might be incorporated into the supervised loop through the use of pre-trained models' embeddings. This taxonomy categorization also depends on the point at which each loss is computed in the model's pipeline:

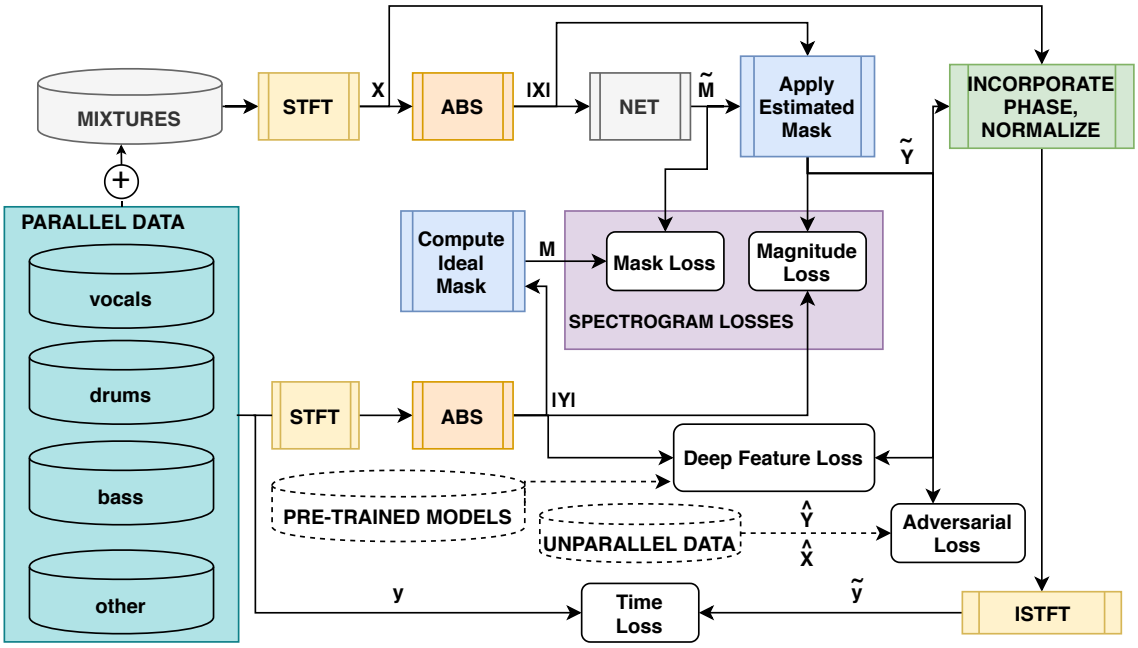

Figure 5: Measurement point for each loss category of our taxonomy in a spectral separator.

2.1 Spectrogram-based losses

Given the mixture x_t of N samples with $STFT(x) = X_{n,\omega}$, and ground truth source y with its corresponding Y for k sources, the first category from the taxonomy comprehends all the losses that follow the traditional approach: to estimate $|Y_{n,k,\omega}|$ with a neural network obtaining $\tilde{Y}_{n,k,\omega}$ and thus ignoring phase. During inference, the reconstructed source uses the mixture phase. Although the most straightforward approach is to directly apply it like $y_{t,k} = ISTFT(\tilde{Y}_{n,k,\omega}(cos(\angle X_{n,\omega}) + \mathbf{j}sin(\angle X_{n,\omega}))$,

one can additionally ensure that all the estimates add up to the mixture in what is known as *softmasking* [7]: normalizing the particular estimate with the energy of all estimates obtaining $\tilde{y}_k = ISTFT(X_{n,\omega}\tilde{Y}_{n,k,\omega}/\sum_0^k \tilde{Y}_{n,k,\omega})$ and incorporating the mixture phase at the same time. Regardless how the final mask (and therefore the mixture phase) is applied, separator models may also yield magnitude masks $\tilde{M}$ instead of magnitude estimates $\tilde{Y}$ as has been shown in Figure 5.

2.1.1 L1 / L2 on magnitude spectrograms

This is the most straightforward approach: to measure L1 or L2 norms between input-target magnitude spectrograms. Both functions have already been described for the general case in Equation 1.1. Extending them to multiple sources leads to:

$$\begin{aligned} L1_{freq} &= \frac{1}{NK\Omega}\sum_{n,k,\omega}|Y_{n,k,\omega} - \tilde{Y}_{n,k,\omega}| \\ L2_{freq} &= \frac{1}{NK\Omega}\sum_{n,k,\omega}|Y_{n,k,\omega} - \tilde{Y}_{n,k,\omega}|^2 \end{aligned} \tag{2.1}$$

2.1.2 L1 / L2 on spectrogram-masks

Returning to Figure 5, the output of the NET (which stands for our separator) may be a real mask $\tilde{M}$. Therefore, we can measure the error at this point in the pipeline just by re-computing the ideal masks from our target magnitudes. We ensure that they add to unity in the following way:

$$M_{n,k,\omega} = \frac{|Y_{n,k,\omega}|}{\sum_k |Y_{n,k,\omega}|} \tag{2.2}$$

Once with our target masks, we can bypass the mask application from Figure 5 and measure the L1 and L2 norms between M and $\tilde{M}$:

$$\begin{aligned} MAE_{mask} &= \frac{1}{NK\Omega}\sum_{n,k,\omega}|M_{n,k,\omega} - \tilde{M}_{n,k,\omega}| \\ MSE_{mask} &= \frac{1}{NK\Omega}\sum_{n,k,\omega}|M_{n,k,\omega} - \tilde{M}_{n,k,\omega}|^2 \end{aligned} \tag{2.3}$$

Optimizing the mask directly is proposed in works like [8].

2.1.3 L2 + Dissimilarity loss

While the above encourages the model to produce outputs similar to the targets, BSS is prone to suffer from interference between predictions (i.e. remnant components of the mixture remaining present in the separated estimates). To mitigate this, in [9] Huang et al. add a dissimilarity term that additionally encourages each prediction to be different from the rest of estimates, by adding an L2 norm for each additional source. In other words, in a particular source separation we would subtract the distance between the source estimate and the other estimates to the distance between the source estimate and the target. Extended to K sources as in [10] and with experimentally found γ coefficient, dissimilarity L2 can be expressed as:

$$L_{dissim} = \frac{1}{NK\Omega} \sum_{n,k,\omega} (L2 - dissimilarity)$$

$$= \frac{1}{NK\Omega} \sum_{n,k,\omega} ((\tilde{Y}_{n,k,\omega} - Y_{n,k,\omega})^2 - \gamma \sum_{\tilde{k}} |\tilde{Y}_{n,k,\omega} - Y_{n,\tilde{k}\neq k,\omega}|^2) \tag{2.4}$$

2.1.4 Cross-entropy loss

Besides, in [11] Lin et al. propose to use Binary Cross-Entropy (BCE) between output and target Ideal Binary Masks for singing voice separation, optimizing the average probability error for each T-F bin. Their work is based on pixel-wise classification for computer vision [12], and obtains slightly worse performance than SiSEC's UHL2 (Open-Unmix's predecessor). As they do not provide an ablation study on the loss, its impact is still unclear. They additionally study the use of cross-entropy as a regression loss by applying it to the ideal ratio masks described above, but report worse results than with IBMs. Given the ideal binary mask $B_{n,k,\omega}$ and the

network prediction $\tilde{B}_{n,k,\omega}$, our loss is the Binary Cross Entropy:

$$L_{BCE} = \frac{1}{NK\Omega} \sum_{n,k,\omega} (B_{n,k,\omega}(-log(\tilde{B}_{n,k,\omega}))+ (1 - B_{n,k,\omega})(-log(1 - \tilde{B}_{n,k,\omega}))) \tag{2.5}$$

As here we are dealing with multiple-sources separation, we will use Categorical Cross-Entropy instead of Binary Cross-Entropy, better suited for multi-class classifications. With $B_{n,\omega}$ as the binary mask of all sources:

$$L_{CE} = \frac{1}{N\Omega} - \sum_{n,\omega} B_{n,\omega} log(\tilde{B}_{n,\omega}) \tag{2.6}$$

For the sake of completeness, Cross-Entropy is also used in [13], where the outputs of BLSTM separators are fed to audio classifiers whose CE is used for the optimization of the separators. In this manner, sources are not restricted to the SiSEC's paradigm because targets become weak labels (clip and frame-level annotations of mixtures) instead of spectrograms for each source.

2.1.5 Phase-Aware Signal-to-Noise Ratio (SNR-PSA) loss

Moreover, in [14], Erdogan et al. propose to incorporate phase information into the loss for a speech enhancement task. The idea is that the magnitude estimates compensate for the errors that will later be introduced due to using the mixture phase by attenuating the amplitude of the most sensible bins (i.e. dimming regions where phase distortion of the separations would be less desirable than silence). This is attempted by optimizing a Phase-Sensitive Spectrum Approximation (PSA) of the magnitude with the following expression:

$$Y_{n,k,\omega}^{PSA} = |Y_{n,k,\omega}| cos(\angle X_{n,\omega} - \angle Y_{n,k,\omega}) \tag{2.7}$$

$$L_{PSA} = \frac{1}{NK\Omega} \sum_{n,k,\omega} |\tilde{Y}_{n,k,\omega} - Y_{n,k,\omega}^{PSA}|^2 \tag{2.8}$$

Given that the separation error can be seen as noise, Erdogan et al. propose to combine PSA with the common Signal to Noise Ratio in [15], where Phase-Aware Signal-to-Noise Ratio (SNR-PSA) is proposed. As they already compare their approach to directly predicting complex Ideal Ratio Masks (cIRM) [8] here we will not further consider them. SNR-PSA loss is saturated with A for avoiding the model to focus on easy utterances, setting it at 20dB for a speech enhancement task. This loss also uses power-law compression in order to mimic human perception as in [16]. With 0.5 power-law compression, SNR-PSA can be expressed as:

$$SNR_k'^{PSA} = -10log(\frac{\sum_{n,\omega} Y_{n,k,\omega}^{PSA}}{\sum_{n,\omega} (\sqrt{\tilde{Y}_{n,k,\omega}} - \sqrt{Y_{n,k,\omega}^{PSA}})^2})$$

$$SNR^{PSA} = \frac{1}{K} \sum_k (A\, tanh(SNR'^{PSA}/A)) \tag{2.9}$$

2.2 Time-domain losses

Another way to circumvent the lack of phase estimation than cIRM/PSA is to directly estimate the waveforms. In fact, in [17] Défossez et al. propose *Demucs*, which consists in a residual-connected convolutional encoder-decoder with BLSTMs at its bottleneck which, despite using L1 loss, obtains better objective and subjective scores than Open-unmix [6], being the first time-domain model to outperform spectrogram-based approaches and therefore becoming the new SOTA in music source separation. However, these losses may also benefit spectral models in the same way than PSA does, i.e. by trying to compensate in the magnitude estimates for the distortion introduced when using the mixture phase.

2.2.1 L1 / L2 on waveforms

To define this loss used in works like [17], we simply swap the spectral components from Equation 2.1 for their respective temporal counterparts as well as the STFT frames axis n for the sample axis t:

$$L1_{time} = \frac{1}{TK}\sum_{t,k}|y_{t,k} - \tilde{y}_{t,k}|$$
$$L2_{time} = \frac{1}{TK}\sum_{t,k}|y_{t,k} - \tilde{y}_{t,k}|^2 \tag{2.10}$$

2.2.2 Scale-Invariant Signal to Distortion Ratio (SI-SDR)

Although Source to Distortion Rartio (SDR) [5] is the most common evaluation metric for music source separation, in [18] Roux et al. point out that it has some potential problems, as the fact that *"obliterating some frequencies setting them to 0 could absurdly still result in near infinite SDR"*. In addition, making a differentiable adaptation from it is far from being trivial, making it unsuitable as a loos. Their proposal is a simplified yet more robust (despite allowing scaling) version of SDR calculated in its expanded form as:

$$\text{SI-SDR} = 10log_{10}(\frac{|\frac{\tilde{y}_{t,k}^T y_{t,k}}{|y_{t,k}|^2}y_{t,k}|^2}{|\frac{\tilde{y}_{t,k}^T y_{t,k}}{|y_{t,k}|^2}y_{t,k} - \tilde{y}_{t,k}|^2}) \tag{2.11}$$

As Scale-Invariant-SDR is more straightforward and less computationally expensive, in the present work we focus on SI-SDR as loss and leave SDR as an evaluation metric in order to make results comparable to SiSEC's. So do *Conv-Tasnet* [19] and Kim & El-Khamy in [20], where normalized L2 or Mean-Squared Error (MSE) is considered again to be sub-optimal for SDR maximization in a speech enhancement task, proposing to directly use SI-SDR on top of speech-tailored perceptual metrics like Perceptual Evaluation of Speech Quality (PESQ) and Short-time Objective Intelligibility (STOI) in the latter. Likewise, Kolbæk et al. [21] suggest SI-SDR

as the objectively best loss in a study similar to ours but for monoaural time-domain speech enhancement. Despite not appearing in the literature, the loss can be adapted to spectral domain just by concatenating the frequency components of the magnitudes $Y_{n,k,\omega}$ and $\tilde{Y}_{n,k,\omega}$ into $Y_{n\omega,k}$ and $\tilde{Y}_{n\omega,k}$ and substituting them for $y_{t,k}$ and $\tilde{y}_{t,k}$ in Equation 2.11.

Returning to the original work, Roux et al. also suggest a scale-dependant alternative using Scale-Dependant-SDR, which is sensitive to both up-scaling and down-scaling, combining a the traditional Signal to Noise Ratio with a Downscale-Dependent Signal to Distortion Ration (DsDSDR):

$$\begin{aligned} SNR &= 10log_{10}(\frac{|y_{t,k}|^2}{|y_{t,k} - \tilde{y}_{t,k}|^2}) \\ L_{down} &= SNR + 10log_{10}(\frac{\tilde{y}_{t,k}^T y_{t,k}}{|y_{t,k}|^2})^2 \\ SDSDR &= min(SNR, L_{down}) \end{aligned} \tag{2.12}$$

2.2.3 LOG-L1 and LOG-L2 on waveforms

However, in [22] Heitkaemper et al. describe how SI-SDR results in a more straightforward logarithmic version of the Mean Squared Error when all terms not dependent on learnable parameters are removed:

$$\text{SI-SDR} = \text{LOG-L2} = 10\frac{1}{K}\sum_k log_{10}\sum_t |y_{t,k} - \tilde{y}_{t,k}|^2 \tag{2.13}$$

The main idea behind compressing the loss with a logarithmic function is that distortions with low amplitude may be as perceptually unpleasant as high amplitude ones. With compression, the model is encouraged to focus on low amplitude distortions as well. Using L1 instead of L2 yields:

$$\text{LOG-L1} = 10\frac{1}{K}\sum_k log_{10}\sum_t |y_{t,k} - \hat{y}_{t,k}| \tag{2.14}$$

Again, as in the SI-SDR case, adapting it to spectral domain is very straightforward:

$$LOGL1_{mag} = \frac{10}{K}\sum_{k} log_{10} \sum_{n,\omega} |Y_{n,\omega} - \tilde{Y}_{n,\omega}|$$
$$LOGL2_{mag} = \frac{10}{K}\sum_{k} log_{10} \sum_{n,\omega} |Y_{n,\omega} - \tilde{Y}_{n,\omega}|^2 \tag{2.15}$$

2.2.4 Multi-resolution STFT

The last strictly time domain loss in the taxonomy is the Multi-resolution STFT loss L_{MRS} presented in [23], where Yamamoto et al. propose to optimize speech synthesis with the combination of several calculations of two losses: spectral convergence L_{sc} and log STFT magnitude L_{mag}, which is the spectral version of LOG-L1 loss described above. Firstly, for the single resolution we calculate L_{SRS}:

$$L_{SRS} = L_{sc} + L_{mag}$$
$$L_{sc} = \frac{\sqrt{\sum_{t,k} |STFT(y_{t,k}) - STFT(\tilde{y}_{t,k})|^2}}{\sqrt{\sum_{t,k} STFT(Y_{t,k})^2}} \tag{2.16}$$
$$L_{mag} = \frac{1}{TK}\sum_{t,k}(|log_{10}STFT(y_{t,k}) - log_{10}STFT(\tilde{y}_{t,k})|) \tag{2.17}$$

The point of introducing the STFT operation into the loss is that, given its the intrinsic time-frequency resolution trade-off, we can average several measures with different STFT parameters. Let O be the number of different parameters (i.e. FFT size, window size and or overlap), the final Multi-Resolution STFT loss is the average of L_{SRS} at every resolution:

$$L_{MRS} = \frac{1}{O}\sum_{o} L_{SRS}^{(o)} \tag{2.18}$$

In their speech synthesis task, L_{MRS} produces Mean Opinion Score gains between +0.33 and +2.7 points depending on the model they apply it to.

2.2.5 Similarity loss

Similarity consists in the opposite from dissimilarity and can be used in systems that follow an auto-encoder structure and when mini-batch training is used: as the batch contains several utterances of the same source, one can encourage the estimates to be similar to other targets from that batch. For example, on top of encouraging *vocals* to be different to the other instruments, we can additionally encourage them to be similar to the rest of *vocals* from the batch. The idea is proposed by Samuel & Ganeshan in [24], and is put into practice by encouraging embeddings of the same instruments to be close while discouraging similarity of embeddings from different instruments. Given batch dimension b, and the separated latent representation $h_{b,k}$, the loss can be expressed as:

$$L_{sim,dissim} = \frac{1}{BNK\Omega} \sum_{n,k,\omega,b} (L2 - dissimilarity + similarity)$$

$$= \frac{1}{BNK\Omega} \sum_{n,k,\omega,b} ((\tilde{Y}_{n,k,\omega,b} - Y_{n,k,\omega,b})^2 - \gamma \sum_{\hat{k}} \frac{abs(h_{b,k}) \cdot abs(h_{b,\hat{k}\neq k})}{|h_{b,k}||h_{b,\hat{k}\neq k}|} \tag{2.19}$$

$$+\epsilon \sum_{\hat{b}} \frac{h_{b,k} \cdot h_{\hat{b}\neq b,k}}{|h_{b,k}||h_{\hat{b}\neq b,k}|}) \tag{2.20}$$

As the loss per se does not care about the input features, we can put it in both spectral and temporal categories of our taxonomy. In fact, Samuel & Ganeshan combine temporal and STFT magnitude features that are encoded into a latent space by their model.

2.3 Adversarial loss

Our third loss category contains a single particular case: the adversarial loss. The middle ground between using paired multi-track data as in SiSEC and using mixtures alone with their respective labels as in [13] is presented in [25], were an adversarial semi-supervised framework is proposed for singing voice separation. Stoller et al. suggest that a source separator may play the generator role of a Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) [26] as depicted in

Figure 6. In this manner, nonparallel data (mixtures without their corresponding stems and vice versa) can be included in the supervised loop, alleviating the scarcity of multi-track data. As shown in [26], WGAN-GP mitigates some training instability issues from vanilla WGAN, which at the time tackled regular GANs' Jensen-Shanon Divergence problems.

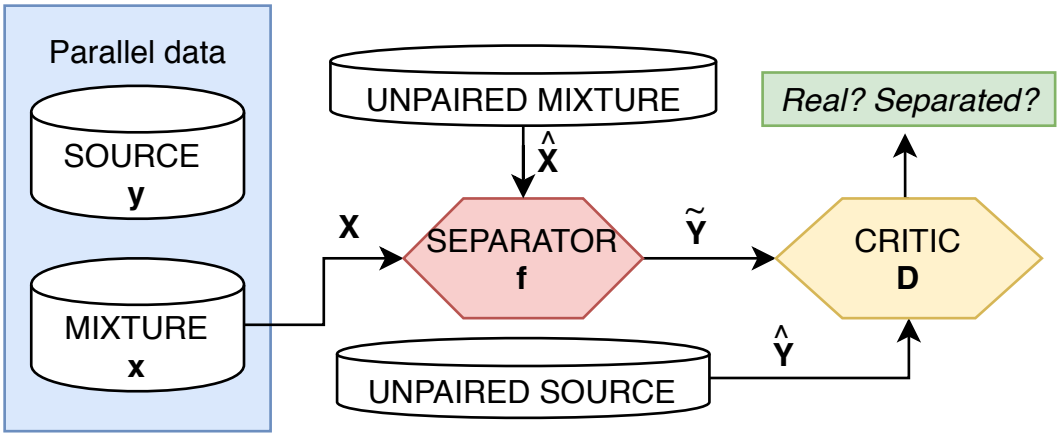

Figure 6: Adversarial Source Separation diagram.

In order to estimate the Wasserstein divergence, the critic is updated five times per separator step so two optimizers are needed. With α, β and γ as hyperparameters, with ϵ randomly sampled from $U[0,1]$ and $\hat{p} \leftarrow \epsilon \hat{Y}_{t,f} + (1-\epsilon) f(\hat{X}_{n,\omega})$, the semi-supervised loss for a single source can be expressed as the combination of the following elements:

- the L2 norm between parallel output and target
- the Wasserstein Distance between fake and real distributions
- a gradient penalty for ensuring the Lipschitz constraint
- an additive penalty which encourages all nonparallel separations to add up to their mixture

Each of these items correspond to a line of the overall semi-supervised loss equation:

$$\begin{aligned} L_{adv} = \frac{1}{N\Omega} \sum_{n,\omega,k} (|f(X_{n,\omega,k}) - Y_{n,\omega,k}|^2 \\ +\alpha(D(f(\hat{X}_{n,\omega,k})) - D(\hat{Y}_{n,\omega,k})) \\ +\beta(|\bigtriangledown_{\hat{p}} D(\hat{p})|_2 - 1, 0)^2 \\ +\gamma|\sum_{k=1}^{K} f_k(\hat{X}_{n,\omega,k}) - \hat{X}_{t,\omega,k}|_2) \end{aligned} \tag{2.21}$$

Stoller et al. perform an ablation study on the semi-supervised part, reporting $1,02dB$ of SDR increment in singing voice separation on the iKala dataset [27], but no SiSEC-like results are provided as their model has not been extended to the multi-source case. Besides, the impact of the adversarial loss without using additional nonparallel data (i.e. using the model topology as a kind of data augmentation) remains unknown.

2.4 Deep feature losses

The last loss category from the taxonomy comprehends the losses that use pre-trained external models, which had already shown to be valuable when modeling human perception in the Computer Vision field [28], and which have recently been explored for audio. As depicted below, each loss uses a particular pre-trained model ϕ and treats its outputs of the j-th layer in a particular way as well.

2.4.1 VGG-ish loss

On the one hand, the distance between a 16-layer Visual-Geometry-Group (VGG) [29] network's embeddings pretrained on ImageNet dataset is used in [30] by Sahai et al. as a loss function for music separation. In short, L_{VGG} loss is the weighted combination of $L_{feat}^{relu2\_2}$, $L_{sty}^{relu1\_2}$, $L_{sty}^{relu2\_2}$, $L_{sty}^{relu3\_3}$ and $L_{sty}^{relu4\_3}$. Firstly, as described in [31], *feature reconstruction* loss is extracted from the j-th layer of the VGG ϕ

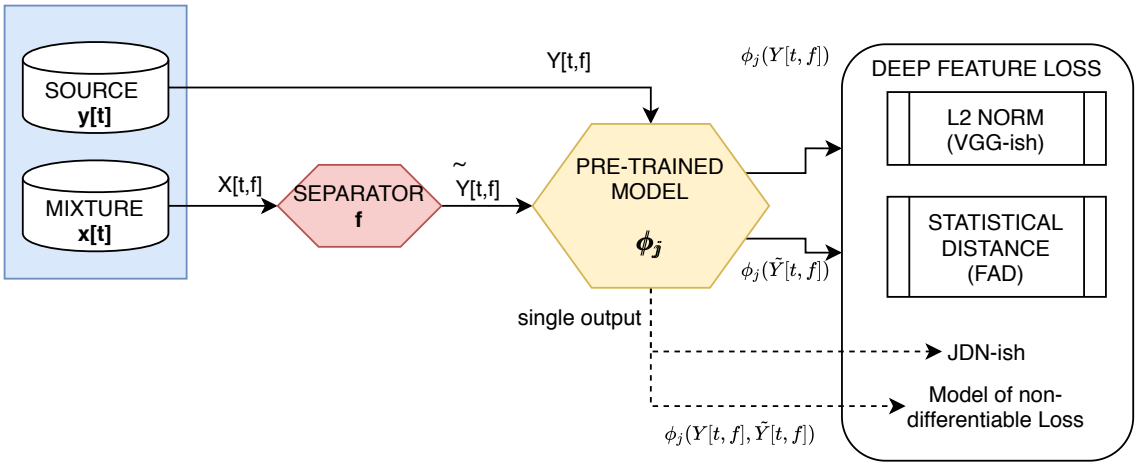

Figure 7: Deep Feature Losses pipeline.

through the euclidean distance between activations. If j is a convolutional layer, the output will be a feature map of shape $C_j \times H_j \times W_j$:

$$L_{feat}^{\phi,j}(\tilde{Y}_{n,\omega,k}, Y_{n,\omega,k}) = \frac{1}{C_j H_j W_j} |\phi_j(\tilde{Y}_{n,\omega,k}) - \phi_j(Y_{n,\omega,k})|^2 \tag{2.22}$$

Next, once that $\phi_j(Y_{t,\omega,k})$ is reshaped from the form $C_j \times H_j \times Wj$ to $\psi_Y = C_j \times H_j Wj$, *style reconstruction* loss is computed through Frobenius norm as:

$$L_{sty}^{\phi,j}(\tilde{Y}_{t,\omega,k}, Y_{t,\omega,k}) = |\frac{\psi_{\tilde{Y}}\psi_{\tilde{Y}}^T}{C_{j,\tilde{Y}} H_{j,\tilde{Y}} W_{j,\tilde{Y}}} - \frac{\psi_Y \psi_Y^T}{C_{j,Y} H_{j,Y} W_{j,Y}}|_F^2 \tag{2.23}$$

Finally, the weights for each of the five losses should be determined empirically. It is important to point out that this work reports improvements on audio processing performance despite using a model pre-trained on image classification.

2.4.2 JND-ish loss

On the other hand, in [32] Manocha et al. approach the L2 lack of perceptual insight by directly training a network on real human judgments for speech enhancement. After collecting a dataset of human Just Noticeable Differences for linear, reverb and compression perturbations (i.e. h labels on whether y_t and $\tilde{y}_t$ sound the same), distance model L_{JND} is optimized by a small classifier G through Binary Cross

Entropy loss introduced in Equation 2.5:

$$\mathcal{L}(G, D) = BCE(G(L_{JND}(y_t, \tilde{y}_t)), h) \tag{2.24}$$

While no details on G are given, L_{JND} consists of fourteen 3x1 layers trained on 22k pairs of human judgments for 1000 epochs. The 95.3% of subjective evaluations show L_{JND} to be better than a VGG-ish baseline.

2.4.3 Fréchet Audio Distance

Furthermore, in [33] Kilgour et al. propose Fréchet Audio Distance, a reference-free evaluation metric that computes the distance between statics of VGG embeddings when our model's output is inferred and the embeddings when a large database of clean music is classified, showing better correlation with MOS than SDR. In this case, the layer prior to the final one is taken as the embedding and inference is done with the MagnaTagATune [34] dataset z. Given tr as the trace of a matrix, FAD is calculated with the embeddings' statistics as follows:

$$FAD(\phi_z, \phi_{\tilde{y}}) = |\mu_z - \mu_{\tilde{y}}|^2 + tr(\Sigma_z + \Sigma_{\tilde{y}} - 2\sqrt{\Sigma_z \Sigma_{\tilde{y}}}) \tag{2.25}$$

2.4.4 Models of non-differentiable losses

Last, but not least, in [35] Elbaz et al. propose to model non-differentiable audio metrics with a neural network, obtaining a differentiable perceptual metric. They start by training Wavenet [36] on time-domain chunks with the respective Perceptual Evaluation Speech Quality labels that have been computed offline. Once pre-trained, Wavenet achieves 81% correlation with PESQ [37], being able to serve as an approximation of a perceptual metric suitable for optimization. Given the pre-trained model of a non-differentiable loss W:

$$L_{nondiff} = \frac{1}{K} \sum_{k} W(y_{t,k}, \tilde{y}_{t,k}) \tag{2.26}$$

Chapter 3

Methods and materials

Once described all the losses we have found in the literature, in this chapter we provide an overview of Open-Unmix (our baseline spectral model), the datasets we use and the experimental framework.

3.1 Open-Unmix

As explained in previous chapters, using Open-Unmix [6] as baseline implies that the present benchmark is done under optimal circumstances, with the best publicly available source separator. Open-Unmix can be considered the best performing system from SiSEC [4], tied with non-public systems like TAK1. In the present section, we will describe the Open-Unmix (UMX) topology and its main components.

Firstly, because audio is indeed a sequence, UMX is a Recurrent Neural Network, which, from [38], *"is suited especially well for machine perception tasks, where the raw underlying features are not individually interpretable"*. The main idea behind them is simple: adding a loop in the network should allow knowledge to persist. If we imagine how this loop unfolds over time in different states, we obtain Figure 8, where a network cell A receives an input x_t and produces an output h_t that can be decomposed in t time steps, with each instance passing the information to the next one. Early RNN designs suffer from difficulty of learning long-range dependencies

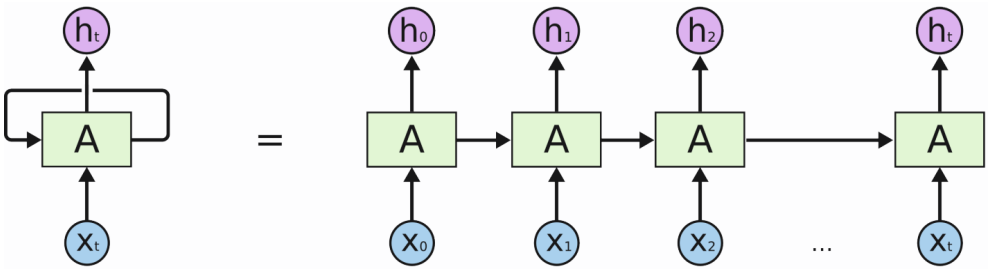

Figure 8: RNN scheme, courtesy of Cristopher Olah.

[39]. In order to address this issue, in 1997 Hochreiter et al. proposed the Long Short-Term Memory [40], specifically designed to learn long-term dependencies by using a cell state, a stable memory channel that gets modified through time by structures (additional neural network layers) called gates. This is depicted in Figure 9, where it is shown that the LSTM cell uses the previous output h_{t-1}, the current input x_t and the information in the cell state (the upper path) to compute the current output h_t, while it uses the sub-layers (the yellow squares in the diagram that use sigmoid and hyperbolic tangent activations) to regulate which old information from the cell state we forget (the forget gate) and which new information we store in it (the input gate). The current output is a *tanh*-filtered version of the cell state combined with some information from the previous output and the current input by the output gate. If we combine two LSTMs, one modeling the sequence from beginning to end and another one doing it from end to beginning, we obtain the Bidirectional Long Short-Term Memory (BLSTM) unit, the core component of UMX, with better temporal modeling capabilities than uni-directional LSTMs at the expense of the ability of running in real time.

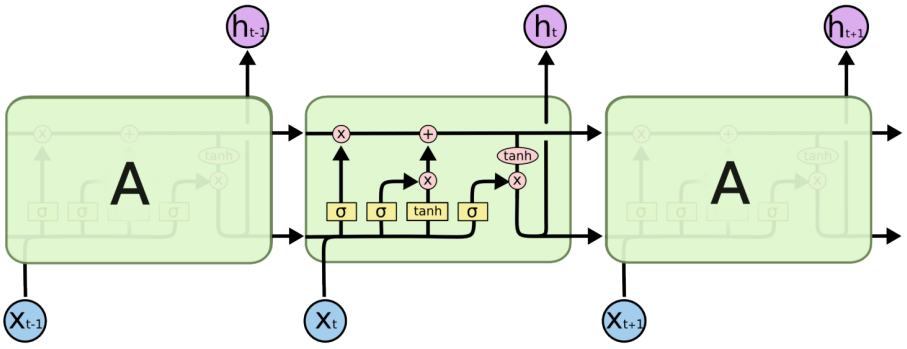

Figure 9: LSTM scheme, courtesy of Cristopher Olah.

The rest of layers present in UMX comprise the Fully Connected Layer (the core of the Multi-Layer Perceptron), the Rectified Linear Unit (ReLU) [41] and traditional Hyperbolic Tangent (tanh) activations to obtain non-linearity capabilities, the input scaler that standardizes inputs to match their scale to the randomly initialized small weights, and Batch Normalization [42], which applies the same concept from the input scaler to the inner layers of the model. The combination of these elements constitutes the following UMX's pipeline:

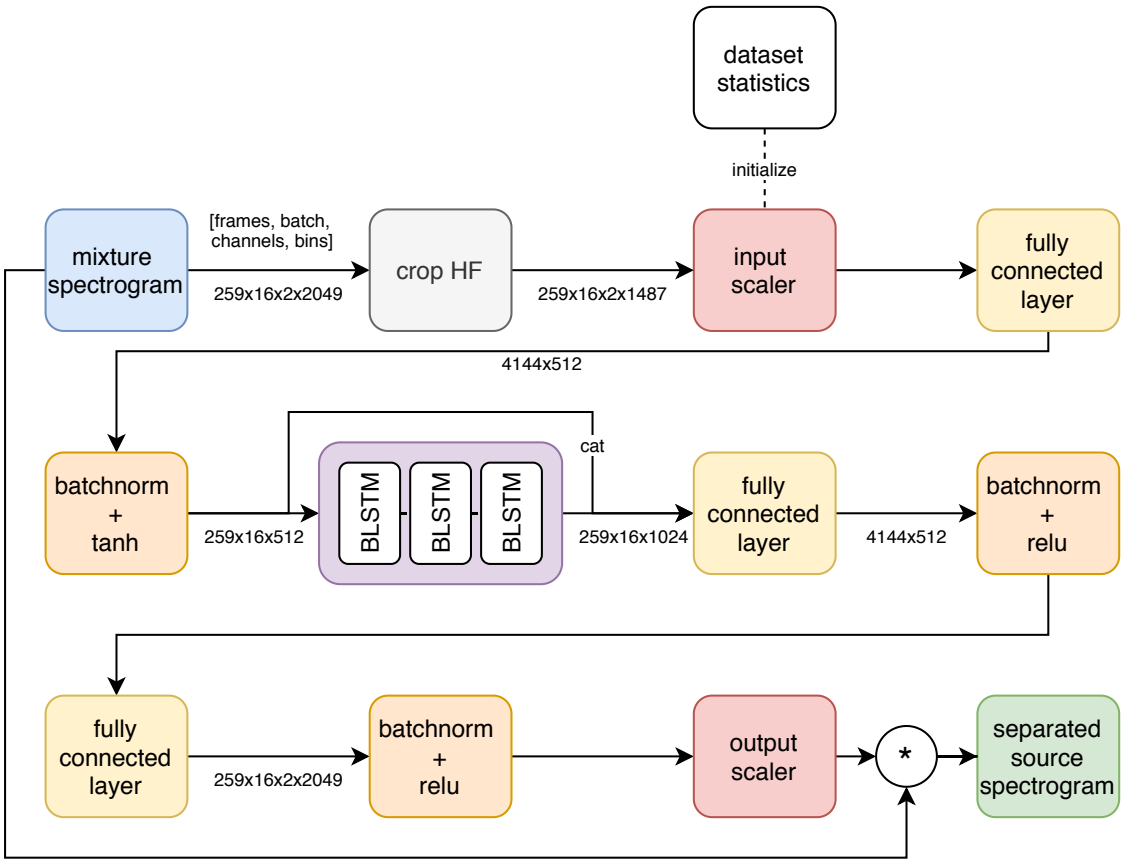

Figure 10: UMX topology.

In the first place, as we have shown above in Figure 5, UMX performs the Short-Time Fourier Transform operation to the mixture and target sources' waveforms, which follow $[16, 2, 6 \cdot 44100]$ shape corresponding to 16 utterances per batch, 2 channels as we are dealing with stereo signals, and randomly selected 6-seconds-long chunks of PCM audio sampled at 44100kHz. STFT is applied with an FFT size of 4096 and a hop size of 1024 samples, yielding 259 temporal frames and 2049 frequency

bins. By the complex norm and angle functions, magnitudes $Y_{n,k,\omega}$ and $X_{n,k,\omega}$ are extracted, as well as $\angle X_{n,k,\omega}$ which will be used when doing inference.

Following, UMX internally filters out the upper 562 frequency bins, as they contain no information due to the dataset being encoded with Advanced Audio Coding (AAC) compression. Prior to the start of the training process, bin-wise mean and standard deviations of the whole training set are computed and used to initialize the input scaler block (which subtracts mean and divides by standard deviation), while the output scaler is initialized to unity with *ones*. These scaler arrays have gradients, in order to let the model fine-tune the initial statistics.

Thirdly, the cropped mixture magnitude is encoded with a Fully-connected layer with shape $[4144, 512]$ corresponding to the number of bins times the channels, and to an experimentally set hidden size h_s of 512, followed by a batch normalization layer with trainable parameters and hyperbolic tangent activation. The batch normalization layer's output is reshaped into the frames, batch, and hidden size dimensions, and concatenated to the output of the model's core, comprised of a three-layer Bidirectional Long Short-Term Memory block of $h_s/2$. The BLSTM's output is concatenated with the input in a residual connection or identity shortcut introduced in [43] with the premise that when failing to extract a feature, the newtork should at least be able to learn the identity avoiding degrading the performance when using deep networks. This concatenation yields 1024 features, half corresponding to the BLSTMs output and half to the BLSTM's input. Again, the network uses the combination of fully-connected layer, batch normalization and activation, but this time twice and using ReLU instead of hyperbolic tangent, obtaining the estimated mask in standardized form.

Lastly, the output scaler will learn to de-standardize the masks in order that, once applied to the mixture spectrogram, their range is comparable to the target spectrograms. Given that masks are defined as shown in Equation 2.3, applying them is as easy as multiplying to the mixture spectrogram.

3.1.1 Baseline's implementation details

On top of all the above, UMX additionally uses several learning strategies. On the one hand, at training time, UMX uses two simple data augmentation transformers in order to fight over-fitting: swapping the sources' left and right channels and applying a random gain to each track before adding them into the mixture creating a custom mix. Additionally, it randomly selects the chunks from each track, making each epoch different. Gradient descent is optimized using Adam with 10^{-5} weight decay [44]. Initial learning rate is set to 10^{-3} and decreased with 0.3 factor when the validation loss does not decrease for 80 epochs, and early stopping is applied after 140 epochs without improvement. The trained model will be the one corresponding to the epoch with lower validation loss.

On the other hand, at testing time, a Multichannel Wiener Filter (MWF) is applied in order exploit the spatial information. Firstly, softmasking is optionally re-applied so all estimates add up to the mixture as previously shown in Equation 2.3, but this time filtering from the complex STFT. This may be skipped in case our model already provides good initial estimates by directly using the mixture phase with the estimated magnitudes. Next, Expectation Maximization algorithm is used to refine the estimates. The main idea behind it is that each source follows a Gaussian distribution in the multichannel space.

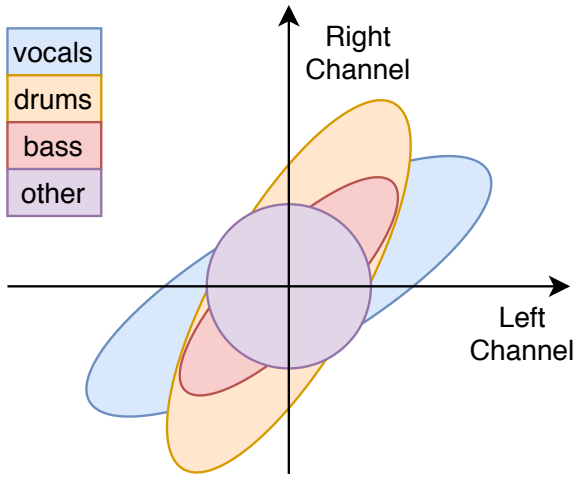

Figure 11: The bin-wise multichannel gaussian model.

The distribution parameters for each source are the magnitude or power spectral density and the *"stereo signature"* which corresponds to the covariance matrices. As both can be computed from UMX's output, refining may be obtained through the Expectation Maximization (EM) algorithm [45], which is a soft-clustering algorithm that will separate each source's power spectral density and covariance matrices from the mixture spatial distribution. In other words, given that our estimate belongs to the mixture and that UMX provides an initial source estimate, we can re-extract them from the mixture in a more spatially plausible way. Once EM provides new mean and covariance matrices, separations are re-computed again with the Wiener Filter, this time using the refined magnitudes.

3.2 Materials: MUSDB18 and MUSEVAL

As we have previously introduced, the main dataset we have used is MUSDB18 [46], which comprises DSD100 and the MedleyDB datasets compressed into AAC at 256kbps and encapsulated in STEMS mp4 format. From the 150 stems, the dataset splits into 100 tracks for training and 50 for validation. The authors also provide MUSDB18-HQ, a full-bandwith version directly available in WAV format, but here we will use the standard version in order to be comparable to participants from SiSEC 2018 [4]. Because online decompression constitutes a bottleneck during training, we have used the provided *musdbconvert* script to store the decompressed files in disk. UMX's default data handlers (and therefore ours) are using the *musdb* parser.

The present work builds upon the Pytorch version of UMX, written in Python and using Pytorch, NumPy and Scikit-Learn packages. On top of this, our work additionally uses Torchaudio's invertible and GPU-capable STFT. The code corresponding to the spectral and temporal losses is available here.

Despite, as introduced in Section 2.2.2, Source to Distortion Ratio (SDR) [5] presents some issues, it is still suited for assessing the performance of source separator models, as it decomposes the SNR into three categories: artifacts, noise and interferences.

This matches the kind of compromises found in source separators, quantifying the balance of how much instrument isolation we obtain and at which price (i.e. how much artifacts are introduced). In its first step, SDR projects the noise into the following energies:

$$\tilde{y} = y + e_{interference} + e_{noise} + e_{artifacts} \tag{3.1}$$

Those are used to build Source to Interferences Ratio (SIR) and Source to Artifacts Ratio, which combined with Sources to Noise Ratio produce the SDR:

$$\begin{aligned} SIR &= 10\ log_{10}\frac{|y|^2}{|e_{interf}|^2} \\ SNR &= 10\ log_{10}\frac{|y + e_{interf}|^2}{|e_{noise}|^2} \\ SAR &= 10\ log_{10}\frac{|y + e_{interf} + e_{noise}|^2}{|e_{artif}|^2} \\ SDR &= 10\ log_{10}\frac{|y|^2}{|e_{interf} + e_{noise} + e_{artif}|^2} \end{aligned} \tag{3.2}$$

Again, so as we are comparable with SiSEC systems, here we use the SDR version of BSSEval v3. From now on, SDR results will be median of frames and median of tracks for each instrument, and average among instruments' scores when providing a single value. Hence, in the present work, the evaluation relies on SDR scores and a subjective critical listen.

3.3 Our adaptation for joint estimation

Some of the losses described in Chapter 2 require some modifications to the original model. Specifically, mask losses from Section 2.1.2 need to compare the masks that are internally applied in the original UMX. To this purpose, in our adaptation the model outputs the mask before applying it, which is done in the training and testing loops.

The first test we have done while preparing this benchmark is to assess the impact of the MWF and an attempt to replace it for a *softmax* final layer (which makes

elements to lie in the [0,1] range and to sum to 1) in order to obtain a truly end-to-end model. As shown in Table 2, softmasking during training (i.e. softmax) only helps when no MWF is used, but still performs worse than original UMX version.

Table 1: Ablation on the Multichannel Wiener Filter (SDR).

	vocals	**drums**	**bass**	**other**
UMX + MWF	6,320	5,730	5,230	4,020
UMX - MWF	5,876	4,952	4,238	2,904
UMX + softmax - MWF	5,601	5,662	4,713	4,051
UMX + softmax + MWF	5,568	5,492	4,692	3,729

Secondly, because Dissimilarity from Section 2.1.3 requires all targets to be fed to the GPU, we have explored the possibility of jointly estimating all sources using a single model ($jUMX$) as we depict in Figure 12. To make the number of parameters comparable to using four separate models as in the original UMX, we have doubled the hidden size of the model, and split the final fully connected layer into the four outputs.

In addition, because the *other* class is the residual from *vocals*, *drums* and *bass*, we have tried to jointly training just these three classes and to build *other* from the mixture's residual, which also results in poorer performance:

Table 2: Joint UMX and residual training (SDR).

$model_{h_s}$	**vocals**	**drums**	**bass**	**other**
UMX_{512}	6,320	5,730	5,230	4,020
$jUMX_{512}$	6,257	5,671	5,137	4,399
$residual_{1024}$	5,466	5,196	4,284	2,653
$jUMX_{1024}$	**6,399**	**5,861**	**5,278**	**4,573**

Finally, the initialization of the input scaler needs to be adapted for the SNR-PSA loss from Section 2.1.5. Due to the loss operating in a power-law compressed domain, uncompressed statistics generate an exploding gradient problem during the

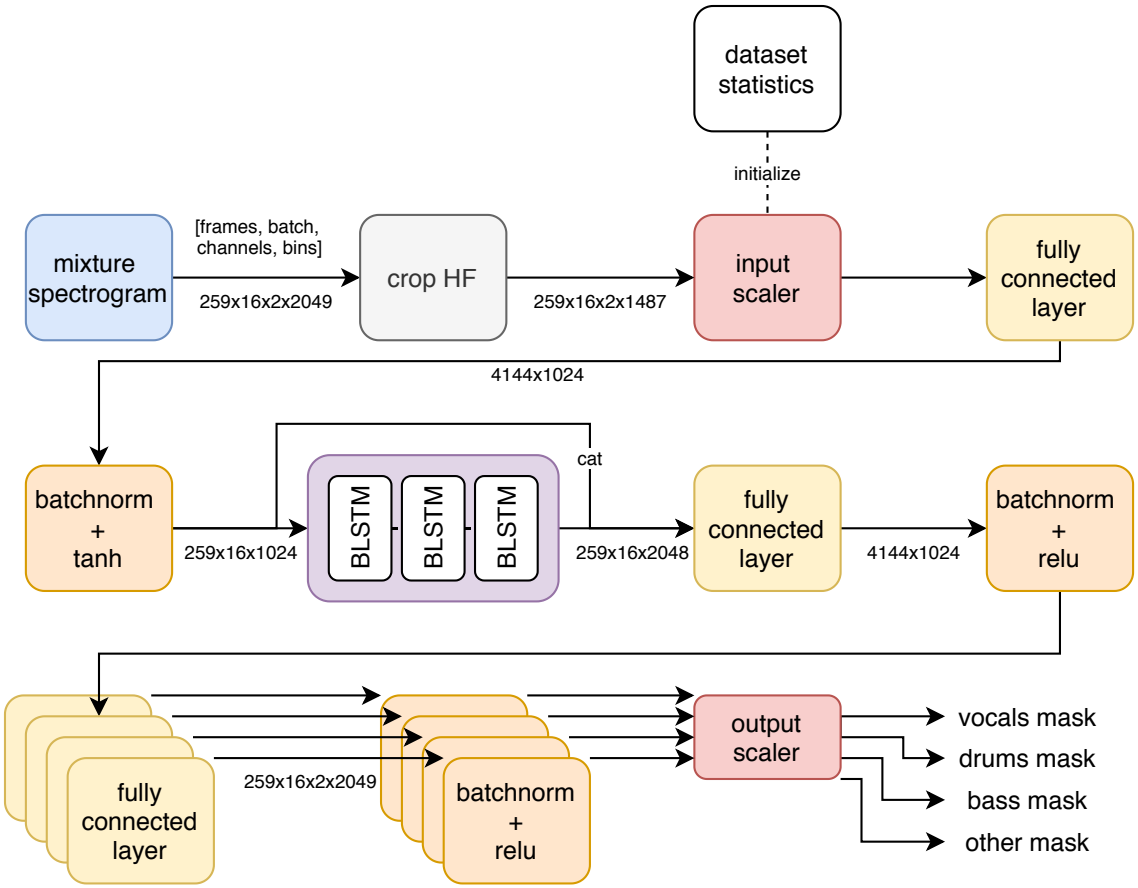

Figure 12: Jointly training UMX (jUMX) pipeline.

first iterations. In order to avoid this, we re-compute the statistics but with the power-law applied to the spectrograms.

In short, our adaptation jointly estimates the four sources using twice the hidden size and keeping the MWF, obtaining objective scores comparable to original UMX.

3.3.1 Hyperparameter selection

As in [21], each loss presents a varying sensitivity to the choice of learning rate. We have tested learning the following learning rates: 10^3, $5 \cdot 10^3$, 10^4, $5 \cdot 10^4$ and 10^5. In most cases, the the default learning rate of 10^{-3} performs the best, leaving the tuning to the learning rate scheduler. In some exceptions, we have observed unstable training during the first epochs, requiring a smaller initial learning rate:

Table 3: Learning rates.

$default$	$L2_{mask}$	$L1_{time}$	$L2_{time}$
10^{-3}	10^{-4}	10^{-5}	10^{-5}

Regarding the rest of hyperparameters, we have used $\gamma = \{0.05, 0.1, 0.15\}$ for the Dissimilarity loss, with 0.05 performing better, and added 2048 and 1024 STFTs with 512 and 256 hop sizes respectively in the Multi-Resolution STFT case.

Chapter 4

Results and discussion

Here we provide results for the first two categories from our taxonomy (i.e. spectral and temporal losses) and leave the *Adversarial* and *Deep Feature losses* for future work.

In the following we detail (see instrument-wise scores as reported in Table 4) and discuss the obtained results. An overall depiction of the average SDR scores (the higher, the better) we obtained for the studied losses is depicted in Figure 13.

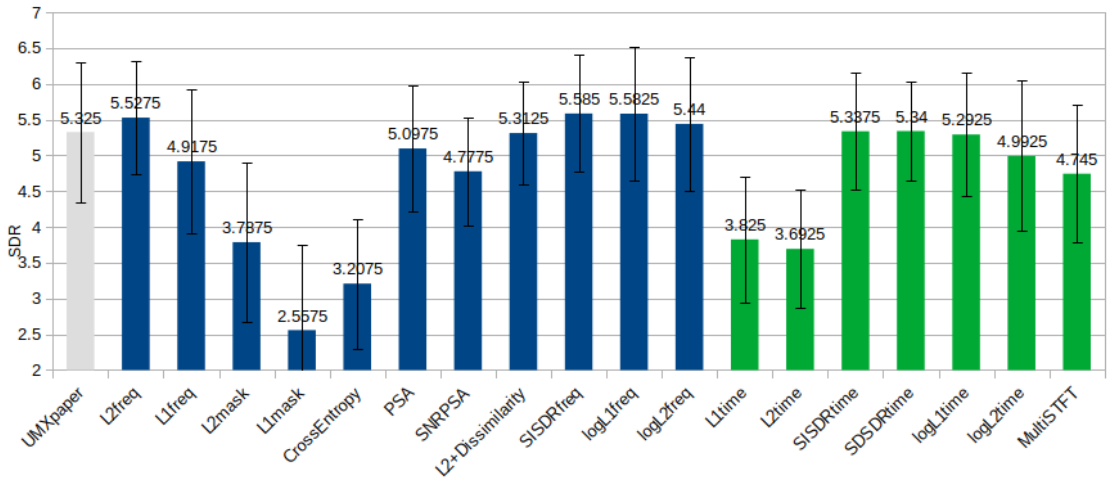

Figure 13: Results, average SDR among instruments. Spectral losses are depicted in blue, and temporal losses are in green. The higher the SDR, the better.

Table 4: SDR Results.

	vocals	drums	bass	other	average
UMX_{paper}	6.32	5.73	5.23	4.02	5.33
$L2_{freq}$	6.4	5.86	5.28	**4.57**	5.53
$L1_{freq}$	5.95	5.58	4.24	3.90	4.92
$L2_{mask}$	5.19	4.14	3.08	2.74	3.79
$L1_{mask}$	3.72	3.29	1.07	2.15	2.55
$L1_{time}$	4.79	4.33	3.26	2.92	3.82
$L2_{time}$	4.55	4.25	3.06	2.91	3.69
$SISDR_{time}$	6.24	5.72	5.04	4.35	5.34
$SISDR_{freq}$	6.29	6.09	**5.49**	4.47	**5.58**
$SDSDR_{time}$	5.90	5.75	5.34	4.37	5.34
$CrossEntropy$	4.32	3.53	2.26	2.72	3.21
$log(L1_{time})$	5.99	5.91	5.15	4.12	5.29
$log(L2_{time})$	5.95	5.73	4.44	3.79	4.98
$log(L1_{freq})$	**6.42**	**6.25**	5.25	4.41	**5.58**
$log(L2_{freq})$	6.51	5.82	5.10	4.33	5.44
PSA	5.87	5.79	4.65	4.08	5.10
$SNR(PSA)$	5.53	5.25	4.45	3.88	4.78
$L2_{freq} + Dissimilarity_{freq}$	6.04	5.66	5.17	4.38	5.31
$MultiSTFT$	5.82	5.11	4.48	3.57	4.75

4.1 Spectral losses

First, some of the worst performances are obtained when optimizing the *(i)* masks (i.e. in $L2_{mask}$, $L1_{mask}$) and with *(ii)* cross-entropy based losses. We argue that this could be caused because the problem is ill-defined when the mixture presents *silences*. In those cases, any set of masks is valid, hindering the optimization process.

Second, dimming amplitudes to mitigate phase distortions (Phase-Sensitive Amplitude estimation, also known as PSA) does not seem to work for music source

separation — while it has shown to be effective for speech source separation. Our hypothesis is that in music tracks there is much more time-frequency overlap than in the speech separation task. In addition, when comparing SNR-PSA and PSA we note that SNR-PSA achieves worse results — thus suggesting that the SNR-PSA modifications applied to PSA (power-law compression, signal-to-noise ratio and hyperbolic tangent clipping) don't generalize well for music source separation [15].

Third, adding a dissimilarity loss term (encouraging each prediction to be different from the rest of estimates) to strong baselines like UMX, does not seem to improve performance.

Four, when re-purposing the SISDR loss for the magnitude spectral domain we obtain slightly better results than $L2_{freq}$. We can find a similar case in the literature in [47], where Févotte et al. describe the benefits from scale invariance when working with spectrograms with the Itakura-Saito divergence in NMF: as the audio spectra typically exhibit exponential power decrease along frequency as well as comprehend low-power note attacks, scale invariance encourages the model to focus on these components as much as in the high-power tonal parts.

4.2 Temporal losses

Using temporal losses in a spectral model such as jUMX does not seem to improve the results obtained with spectral losses. Still, $log(L1_{time})$ and $SISDR_{time}$ perform reasonably well despite achieving worse results than their spectral counterparts. If we take a look at the training curves in Figure 14, we observe that the temporal loss curve presents more noise than the spectral one, suggesting that it is more challenging to optimize directly in the waveform-domain for jUMX — remember that jUMX separates sources via employing time-frequency masks.

We also observe that the worst results are obtained by $L1_{time}$ and $L2_{time}$ losses, followed by the Multi-Resolution STFT (MultiSTFT) loss. Again, we want to emphasize that these results are obtained via training a spectrogram-based model (jUMX) with a waveform loss. For this reason, in future work we would like to run a similar

study using a waveform-based model such as Demucs [17] — to asses whether the results are similar or not.

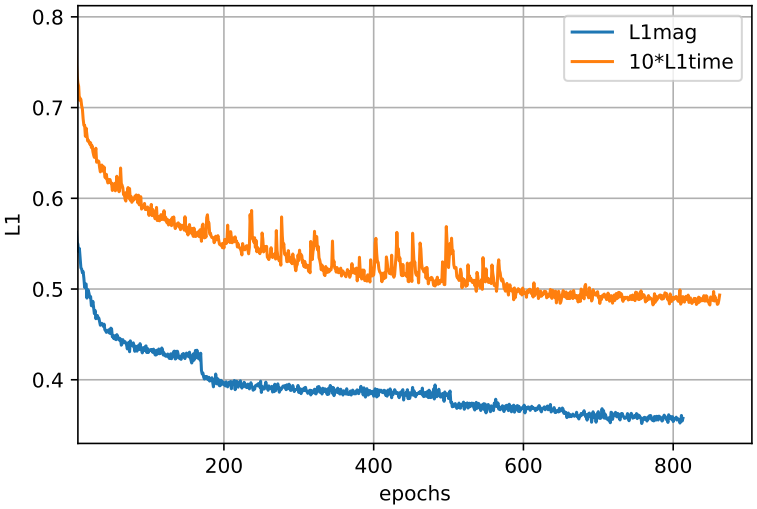

Figure 14: L1 training loss on temporal and frequency domains. We multiply $L1_{time}$ x10 for visualization purposes, since our goal is to visualize the shape of the learning curves (not its absolute values). Note that L1 losses are not comparable since they respectively operate on the magnitude and waveform domains.

Finally, we want to note all temporal losses that incorporate a logarithmic compression ($SISDR_{time}$, $SDSDR_{time}$, $log(L1_{time})$ and $log(L2_{time})$) obtain much better performance than their uncompressed counterparts ($L1_{time}$, $L2_{time}$ and $MultiSTFT$) — denoting the importance of this compression factor.

4.2.1 Logaritmic losses and its relation to SISDR

Heitkaemper et al. showed that $log(L2_{time})$ is closely related to $SISDR_{time}$:

$$SISDR = 10 \cdot log(L2_{time} / \sum_{t} |y_{t,k}|^2) \tag{4.1}$$

In particular, they suggested that because the denominator from the equation above does not depend on trainable parameters, both losses should perform the same. Our results seem to contradict this, since $SISDR_{time}$ achieves a better score (≈0.4 dB SDR more). Interestingly, though, the $log(L1_{time})$ loss achieves results that are

comparable to $SISDR_{time}$, suggesting that the energy of the target $\sum_t |y_{t,k}|^2$ or e_{target} plays a role in the optimization.

Note that in $log(L2)$ and $log(L1)$ compressing the loss through the logarithm encourages the model to focus in low-energy components in a similar fashion than with scale invariance. When we re-write both losses as $log(L2_{time}) = log_{10}(\sum_t |y_t - \tilde{y}_t|^2)$ and $log(L1_{time}) = log_{10}(\sum_t |y_t - \tilde{y}_t|)$ respectively, it is shown that in $log(L1)$ there is more compression with respect to $log(L2)$ due to the lack of exponent. Despite $log(L2_{freq})$ does not improve upon the baseline, the rest of losses perform better once compression is applied.

4.3 Scale invariance and Wiener Filter

Regarding scale invariance, it should be noted that we are avoiding volume-related issues thanks to the Multichannel Wiener Filter (MWF) post-processing step *OpenUnmix* relies on.

If we evaluate SISDR (scale-invariant) losses an compare with scale-dependent losses like $L2_{freq}$ or $SDSDR_{time}$, we observe that using scale-invariant losses without rebuilding scale through the MWF produces a dramatic drop in performance — due to bad volume estimates caused by the scale-invariance property of the loss:

Table 5: Ablation on the MWF for scale-dependent and invariant losses. Average SDR among all instruments.

	with MWF	without MWF
$L2_{freq}$	5.51	4.77
$SISDR_{freq}$	5.58	**2.74**
$SDSDR_{time}$	5.34	5.04
$SISDR_{time}$	5.11	**0.31**

4.4 Perceptual evaluation: critical listening of best candidates

If we perform a critical listening test with the three best candidates, i.e. by using $L2_{freq}$, $SISDR_{freq}$ and $log(L1_{freq})$ losses on top of the original UMX model to an out-of-sample test track. The first thing that seems relevant is that the waveforms are different even to the naked eye. As depicted in Figure 15, all candidates present interference issues, denoting that music source separation is a challenging research topic. Notably, $SISDRfreq$'s interferences have a large amplitude comparable to the amplitude of the separated source. Also note that the case of $log(L1_{freq})$ interferences are minimal.

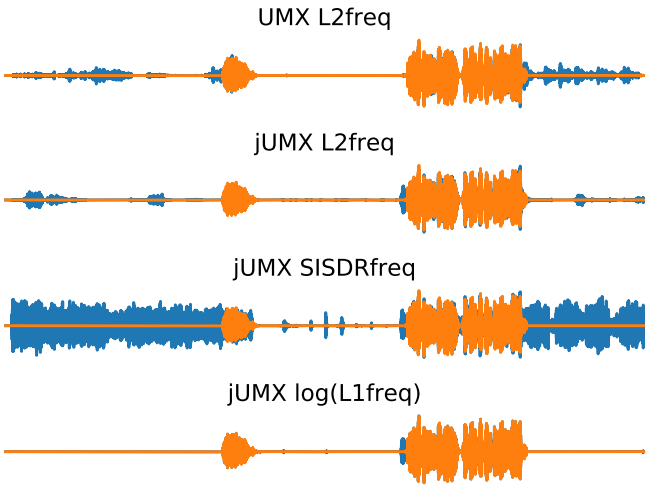

Figure 15: Resulting vocals' waveforms for an out-of-sample test track. Orange depicts the manually edited vocals on top of the separated waveform, in blue.

When listening across a set of ten evaluation tracks, we observe that the interference is usually coming from the hi-hat and cymbals high-frequencies present in the *drums* stem. However, this better separation of cymbals comes at the price of a duller low-frequency response, being the original UMX the one with better low-end response. To our understanding, $log(L1_{freq})$'s best separation seems to be the most desirable, but this should be confirmed with a Mean Opinion Score perceptual test.

Chapter 5

Conclusions

The two main contributions of this work are the following ones:

- We provide an extensive review of the different losses that have been proposed for training deep learning models for processing audio.
- We investigate the performance of those for the music source separation task.

Throughout our study, we find that despite L1 and L2 losses are limited to waveform and magnitude matching, these still outperform most of the proposed alternatives from the literature when isolated from additional techniques and modifications. Most of the proposals fail to improve results when using a standardized evaluation framework such as ours, suggesting that these might be suited for the original task and architecture it was proposed for but it does not generalize to music source separation and OpenUnmix (UMX).

While the lack of phase insight in spectral losses should yield to worse results than temporal losses, we observed the opposite: temporal losses don't work as well as spectral ones when using a magnitude mask-based model like OpenUnmix (UMX).

The original L2 in the magnitude spectrogram domain is still a strong baseline, only comparable to $log(L1_{freq})$. While $SISDR_{freq}$ generates some of the best objective

scores, in our informal listening test as well as in Figure 15 we find that the scale invariant SDR (SISDR) loss can be problematic — hence, L2 and $log(L1_{freq})$ losses are preferable. For these reasons, we argue that scale invariant losses should only be used together with scale reconstruction mechanisms like Wiener Filter.

Finally, we also observe that introducing a $log(\cdot)$ compression to the loss normally improves performance both in objective and subjective terms.

We investigated the impact of using one loss or another. However, advanced loss types (e.g., adversarial and deep feature losses) were not benchmarked. In line with that, it would also be interesting to benchmark the combination of the most promising losses we identified above. And, finally, as an alternative future research line, we would like to run a similar study but considering a waveform-based model — instead of OpenUnmix (UMX) that is based on filtering spetrograms.

List of Figures

List of Tables

Bibliography

[1] Bronkhorst, A. W. The cocktail party phenomenon: A review of research on speech intelligibility in multiple-talker conditions. *Acta Acustica united with Acustica* **86**, 117–128 (2000).

[2] Lee, D. D. & Seung, H. S. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, 556–562 (2001).

[3] Hyvärinen, A. & Oja, E. Independent component analysis: algorithms and applications. *Neural networks* **13**, 411–430 (2000).

[4] Ward, D. *et al.* Sisec 2018: State of the art in musical audio source separation-subjective selection of the best algorithm (2018).

[5] Vincent, E., Gribonval, R. & Févotte, C. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing* **14**, 1462–1469 (2006).

[6] Stöter, F.-R., Uhlich, S., Liutkus, A. & Mitsufuji, Y. Open-unmix-a reference implementation for music source separation (2019).

[7] Liutkus, A. & Badeau, R. Generalized wiener filtering with fractional power spectrograms. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 266–270 (IEEE, 2015).

[8] Williamson, D. S., Wang, Y. & Wang, D. Complex ratio masking for monaural speech separation. *IEEE/ACM transactions on audio, speech, and language processing* **24**, 483–492 (2015).

[9] Huang, P.-S., Kim, M., Hasegawa-Johnson, M. & Smaragdis, P. Deep learning for monaural speech separation. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1562–1566 (IEEE, 2014).

[10] Chandna, P., Miron, M., Janer, J. & Gómez, E. Monoaural audio source separation using deep convolutional neural networks. In *International conference on latent variable analysis and signal separation*, 258–266 (Springer, 2017).

[11] Lin, K. W. E., Balamurali, B., Koh, E., Lui, S. & Herremans, D. Singing voice separation using a deep convolutional neural network trained by ideal binary mask and cross entropy. *Neural Computing and Applications* 1–14 (2018).

[12] Long, J., Shelhamer, E. & Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440 (2015).

[13] Pishdadian, F., Wichern, G. & Roux, J. L. Finding strength in weakness: Learning to separate sounds with weak supervision (2019). `1911.02182`.

[14] Erdogan, H., Hershey, J. R., Watanabe, S. & Le Roux, J. Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 708–712 (IEEE, 2015).

[15] Erdogan, H. & Yoshioka, T. Investigations on data augmentation and loss functions for deep learning based speech-background separation. In *Interspeech*, 3499–3503 (2018).

[16] Kadioglu, B. *et al.* An empirical study of conv-tasnet. *arXiv preprint arXiv:2002.08688* (2020).

[17] Défossez, A., Usunier, N., Bottou, L. & Bach, F. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254* (2019).

[18] Roux, J. L., Wisdom, S., Erdogan, H. & Hershey, J. R. Sdr – half-baked or well done? *ICASSP 2019 - 2019 IEEE International Conference on Acoustics,*

Speech and Signal Processing (ICASSP) (2019). URL `http://dx.doi.org/10.1109/ICASSP.2019.8683855`.

[19] Luo, Y. & Mesgarani, N. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **27**, 1256–1266 (2019).

[20] Kim, J., El-Khamy, M. & Lee, J. End-to-end multi-task denoising for the joint optimization of perceptual speech metrics. *arXiv preprint arXiv:1910.10707* (2019).

[21] Kolbæk, M., Tan, Z.-H., Jensen, S. H. & Jensen, J. On loss functions for supervised monaural time-domain speech enhancement. *arXiv preprint arXiv:1909.01019* (2019).

[22] Heitkaemper, J., Jakobeit, D., Boeddeker, C., Drude, L. & Haeb-Umbach, R. Demystifying tasnet: A dissecting approach. *arXiv preprint arXiv:1911.08895* (2019).

[23] Yamamoto, R., Song, E. & Kim, J.-M. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6199–6203 (IEEE, 2020).

[24] Samuel, D., Ganeshan, A. & Naradowsky, J. Meta-learning extractors for music source separation. *arXiv preprint arXiv:2002.07016* (2020).

[25] Stoller, D., Ewert, S. & Dixon, S. Adversarial semi-supervised audio source separation applied to singing voice extraction. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2391–2395 (IEEE, 2018).

[26] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. Improved training of wasserstein gans (2017). `1704.00028`.

[27] Chan, T.-S. *et al.* Vocal activity informed singing voice separation with the ikala dataset. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 718–722 (IEEE, 2015).

[28] Zhang, R., Isola, P., Efros, A. A., Shechtman, E. & Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 586–595 (2018).

[29] Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[30] Sahai, A., Weber, R. & McWilliams, B. Spectrogram feature losses for music source separation. In *2019 27th European Signal Processing Conference (EUSIPCO)*, 1–5 (IEEE, 2019).

[31] Johnson, J., Alahi, A. & Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, 694–711 (Springer, 2016).

[32] Manocha, P. *et al.* A differentiable perceptual audio metric learned from just noticeable differences. *arXiv preprint arXiv:2001.04460* (2020).

[33] Kilgour, K., Zuluaga, M., Roblek, D. & Sharifi, M. Fréchet audio distance: A metric for evaluating music enhancement algorithms. *arXiv preprint arXiv:1812.08466* (2018).

[34] Law, E., West, K., Mandel, M. I., Bay, M. & Downie, J. S. Evaluation of algorithms using games: The case of music tagging. In *ISMIR*, 387–392 (2009).

[35] Elbaz, D. & Zibulevsky, M. Perceptual audio loss function for deep learning. *arXiv preprint arXiv:1708.05987* (2017).

[36] Oord, A. v. d. *et al.* Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[37] Rix, A. W., Beerends, J. G., Hollier, M. P. & Hekstra, A. P. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment

of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, vol. 2, 749–752 (IEEE, 2001).

[38] Lipton, Z. C., Berkowitz, J. & Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* (2015).

[39] Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* **5**, 157–166 (1994).

[40] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735–1780 (1997).

[41] Nair, V. & Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814 (2010).

[42] Ioffe, S. & Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[43] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).

[44] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[45] Dempster, A. P., Laird, N. M. & Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* **39**, 1–22 (1977).

[46] Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I. & Bittner, R. The MUSDB18 corpus for music separation (2017). URL `https://doi.org/10.5281/zenodo.1117372`.

[47] Févotte, C., Bertin, N. & Durrieu, J.-L. Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural computation* **21**, 793–830 (2009).