SINGING-VOICE SEPARATION USING DEEP RECURRENT NEURAL NETWORKS

Po-Sen Huang[†], **Minje Kim**[‡], **Mark Hasegawa-Johnson**[†], **Paris Smaragdis**[†‡§]
[†]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, USA
[‡]Department of Computer Science, University of Illinois at Urbana-Champaign, USA
[§]Adobe Research, USA
{huang146, minje, jhasegaw, paris}@illinois.edu

ABSTRACT

In this paper, we explore using deep recurrent neural networks for singing voice separation from monaural recordings in a supervised setting. We propose jointly optimizing the networks for multiple source signals by including the separation step as a nonlinear operation in the last layer. Discriminative training objectives are further explored to enhance the source to interference ratio. The algorithm has been tested against the MIREX 2014 singing voice separation task.

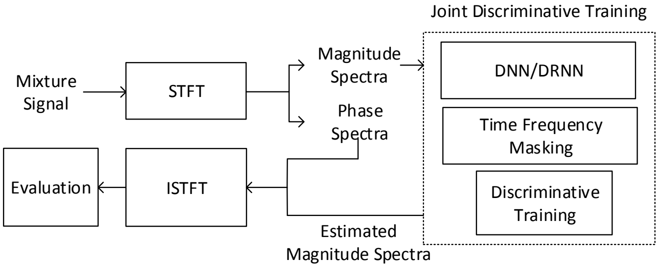

Figure 1. Proposed framework.

1. INTRODUCTION

Based on the work in [6], in this paper, we explore the use of deep recurrent neural networks for the MIREX 2014 singing voice separation task. We explore using a deep recurrent neural network architecture along with the joint optimization of the network and a soft masking function. The proposed framework is shown in Figure 1.

2. PROPOSED METHODS

2.1 Deep Recurrent Neural Networks

To capture the contextual information among audio signals, one way is to concatenate neighboring features together as input features to the deep neural network. However, the number of parameters increases rapidly according to the input dimension. Hence, the size of the concatenating window is limited. A recurrent neural network (RNN) can be considered as a DNN with indefinitely many layers, which introduce the memory from previous time steps. The potential weakness for RNNs is that RNNs lack hierarchical processing of the input at the current time step. To further provide the hierarchical information through multiple time scales, deep recurrent neural networks (DRNNs) are explored [2, 7].

Formally, we can define different schemes of DRNNs as follows. Suppose there is an L intermediate layer DRNN with the recurrent connection at the l-th layer, the l-th hidden activation at time t is defined as:

$$\begin{aligned}\mathbf{h}_t^l &= f_h(\mathbf{x}_t, \mathbf{h}_{t-1}^l) \\ &= \phi_l\left(\mathbf{U}^l\mathbf{h}_{t-1}^l + \mathbf{W}^l\phi_{l-1}\left(\mathbf{W}^{l-1}\left(\ldots\phi_1\left(\mathbf{W}^1\mathbf{x}_t\right)\right)\right)\right),\end{aligned} \tag{1}$$

and the output, $\mathbf{y}_t$, can be defined as:

$$\begin{aligned}\mathbf{y}_t &= f_o(\mathbf{h}_t^l) \\ &= \mathbf{W}^L\phi_{L-1}\left(\mathbf{W}^{L-1}\left(\ldots\phi_l\left(\mathbf{W}^l\mathbf{h}_t^l\right)\right)\right),\end{aligned} \tag{2}$$

where $\mathbf{x}_t$ is the input to the network at time t, ϕ_l is an element-wise nonlinear function, $\mathbf{W}^l$ is the weight matrix for the l-th layer, and $\mathbf{U}^l$ is the weight matrix for the recurrent connection at the l-th layer. The output layer is a linear layer.

Function $\phi_l(\cdot)$ is a nonlinear function, and we empirically found that using the rectified linear unit $f(\mathbf{x}) = \max(0, \mathbf{x})$ [1] performs better compared to using a sigmoid or tanh function. For a DNN, the temporal weight matrix $\mathbf{U}^l$ is a zero matrix.

2.2 Model Architecture

At time t, the training input, $\mathbf{x}_t$, of the network is the concatenation of features from a mixture within a window. We use magnitude spectra as features in this paper. The output targets, $\mathbf{y_1}_t$ and $\mathbf{y_2}_t$, and output predictions, $\hat{\mathbf{y}}_{\mathbf{1}_t}$ and $\hat{\mathbf{y}}_{\mathbf{2}_t}$, of the network are the magnitude spectra of different sources.

Since our goal is to separate one of the sources from a mixture, instead of learning one of the sources as the target, we adapt the framework from [5] to model all different

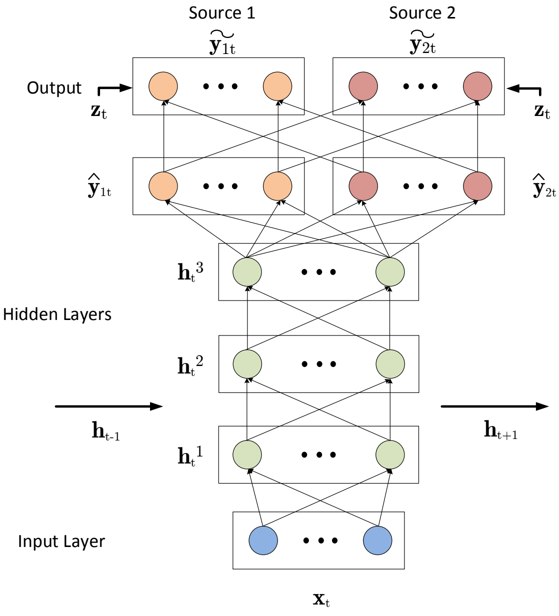

Figure 2. Proposed neural network architecture.

sources simultaneously. Figure 2 shows an example of the architecture.

Moreover, we find it useful to further smooth the source separation results with a time-frequency masking technique, for example, binary time-frequency masking or soft time-frequency masking [4, 5]. The time-frequency masking function enforces the constraint that the sum of the prediction results is equal to the original mixture.

Given the input features, $\mathbf{x}_t$, from the mixture, we obtain the output predictions $\hat{\mathbf{y}}_{\mathbf{1}_t}$ and $\hat{\mathbf{y}}_{\mathbf{2}_t}$ through the network. The soft time-frequency mask $\mathbf{m}_t$ is defined as follows:

$$\mathbf{m}_t(f) = \frac{|\hat{\mathbf{y}}_{\mathbf{1}_t}(f)|}{|\hat{\mathbf{y}}_{\mathbf{1}_t}(f)| + |\hat{\mathbf{y}}_{\mathbf{2}_t}(f)|}, \tag{3}$$

where $f \in \{1, \ldots, F\}$ represents different frequencies.

Once a time-frequency mask $\mathbf{m}_t$ is computed, it is applied to the magnitude spectra $\mathbf{z}_t$ of the mixture signals to obtain the estimated separation spectra $\hat{\mathbf{s}}_{\mathbf{1}_t}$ and $\hat{\mathbf{s}}_{\mathbf{2}_t}$, which correspond to sources 1 and 2, as follows:

$$\begin{aligned} \hat{\mathbf{s}}_{\mathbf{1}_t}(f) &= \mathbf{m}_t(f)\mathbf{z}_t(f) \\ \hat{\mathbf{s}}_{\mathbf{2}_t}(f) &= (1 - \mathbf{m}_t(f))\,\mathbf{z}_t(f), \end{aligned} \tag{4}$$

where $f \in \{1, \ldots, F\}$ represents different frequencies.

The time-frequency masking function can be viewed as a layer in the neural network as well. Instead of training the network and applying the time-frequency masking to the results separately, we can jointly train the deep learning models with the time-frequency masking functions. We add an extra layer to the original output of the neural network as follows:

$$\begin{aligned} \tilde{\mathbf{y}}_{\mathbf{1}_t} &= \frac{|\hat{\mathbf{y}}_{\mathbf{1}_t}|}{|\hat{\mathbf{y}}_{\mathbf{1}_t}| + |\hat{\mathbf{y}}_{\mathbf{2}_t}|} \odot \mathbf{z}_t \\ \tilde{\mathbf{y}}_{\mathbf{2}_t} &= \frac{|\hat{\mathbf{y}}_{\mathbf{2}_t}|}{|\hat{\mathbf{y}}_{\mathbf{1}_t}| + |\hat{\mathbf{y}}_{\mathbf{2}_t}|} \odot \mathbf{z}_t, \end{aligned} \tag{5}$$

where the operator $\odot$ is the element-wise multiplication (Hadamard product). In this way, we can integrate the constraints to the network and optimize the network with the masking function jointly. Note that although this extra layer is a deterministic layer, the network weights are optimized for the error metric between and among $\tilde{\mathbf{y}}_{\mathbf{1}_t}$, $\tilde{\mathbf{y}}_{\mathbf{2}_t}$ and $\mathbf{y}_{\mathbf{1}_t}$, $\mathbf{y}_{\mathbf{2}_t}$, using back-propagation. To further smooth the predictions, we can apply masking functions to $\tilde{\mathbf{y}}_{\mathbf{1}_t}$ and $\tilde{\mathbf{y}}_{\mathbf{2}_t}$, as in Eqs. (3) and (4), to get the estimated separation spectra $\tilde{\mathbf{s}}_{\mathbf{1}_t}$ and $\tilde{\mathbf{s}}_{\mathbf{2}_t}$. The time domain signals are reconstructed based on the inverse short time Fourier transform (ISTFT) of the estimated magnitude spectra along with the original mixture phase spectra.

2.3 Training Objectives

Given the output predictions $\hat{\mathbf{y}}_{\mathbf{1}_t}$ and $\hat{\mathbf{y}}_{\mathbf{2}_t}$ (or $\tilde{\mathbf{y}}_{\mathbf{1}_t}$ and $\tilde{\mathbf{y}}_{\mathbf{2}_t}$) of the original sources $\mathbf{y}_{\mathbf{1}_t}$ and $\mathbf{y}_{\mathbf{2}_t}$, we explore optimizing neural network parameters by minimizing the squared error, as follows:

$$J_{MSE} = ||\hat{\mathbf{y}}_{\mathbf{1}_t} - \mathbf{y}_{\mathbf{1}_t}||_2^2 + ||\hat{\mathbf{y}}_{\mathbf{2}_t} - \mathbf{y}_{\mathbf{2}_t}||_2^2. \tag{6}$$

Furthermore, minimizing Eq. (6) is for increasing the similarity between the predictions and the targets. Since one of the goals in source separation problems is to have high signal to interference ratio (SIR), we explore discriminative objective functions that not only increase the similarity between the prediction and its target, but also decrease the similarity between the prediction and the targets of other sources, as follows:

$$||\hat{\mathbf{y}}_{\mathbf{1}_t} - \mathbf{y}_{\mathbf{1}_t}||_2^2 - \gamma||\hat{\mathbf{y}}_{\mathbf{1}_t} - \mathbf{y}_{\mathbf{2}_t}||_2^2 + ||\hat{\mathbf{y}}_{\mathbf{2}_t} - \mathbf{y}_{\mathbf{2}_t}||_2^2 - \gamma||\hat{\mathbf{y}}_{\mathbf{2}_t} - \mathbf{y}_{\mathbf{1}_t}||_2^2. \tag{7}$$

where γ is a constant chosen by the performance on the development set.

3. EXPERIMENTS

3.1 Setting

Our system is trained using the MIR-1K dataset [3] [1] and the stereo audio dataset. [2] In the MIR-1K dataset, we randomly select 8 song clips as the development set and train on the remaining 992 song clips. In the stereo audio dataset, we mix 52 music clips with 20 randomly selected speech files to generate 1040 clips. We select 10 clips as the development set and the remaining 1030 clips for training.

For training the network, in order to increase the variety of training samples, we circularly shift (in the time domain) the singing voice signals and mix them with the background music.

In the experiments, we use magnitude spectra as input features to the neural network. The spectral representation is extracted using a 1024-point short time Fourier transform (STFT) with 50% overlap. Empirically, we found that using log-mel filterbank features or log power spectrum provide worse performance.

[1] https://sites.google.com/site/unvoicedsoundseparation/mir-1k

[2] http://www.isle.illinois.edu/sst/pubs/2014/chen14audiosources.txt

For our proposed neural networks, we optimize our models by back-propagating the gradients with respect to the training objectives. The limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm is used to train the models from random initialization. We set the maximum epoch to 600 and select the best model according to the development set.

We use a deep recurrent neural network with 3 hidden layers of 1000 hidden units, the recurrent connection at the 2nd hidden layer, the mean squared error criterion, joint masking training, and 25 K samples as the circular shift step size using features with a context window size of 3 frames. We select the models based on the GNSDR results on the development set.

4. REFERENCES

[1] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *JMLR W&CP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, 2011.

[2] M. Hermans and B. Schrauwen. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198, 2013.

[3] C.-L. Hsu and J.-S.R. Jang. On the improvement of singing voice separation for monaural recordings using the MIR-1K dataset. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(2):310 – 319, Feb. 2010.

[4] P.-S. Huang, S. D. Chen, P. Smaragdis, and M. Hasegawa-Johnson. Singing-voice separation from monaural recordings using robust principal component analysis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 57–60, 2012.

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

[6] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis. Singing-voice separation from monaural recordings using deep recurrent neural networks. In *International Society for Music Information Retrieval (ISMIR)*, 2014.

[7] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. In *International Conference on Learning Representations*, 2014.