

GCT634/AI613: Musical Applications of Machine Learning

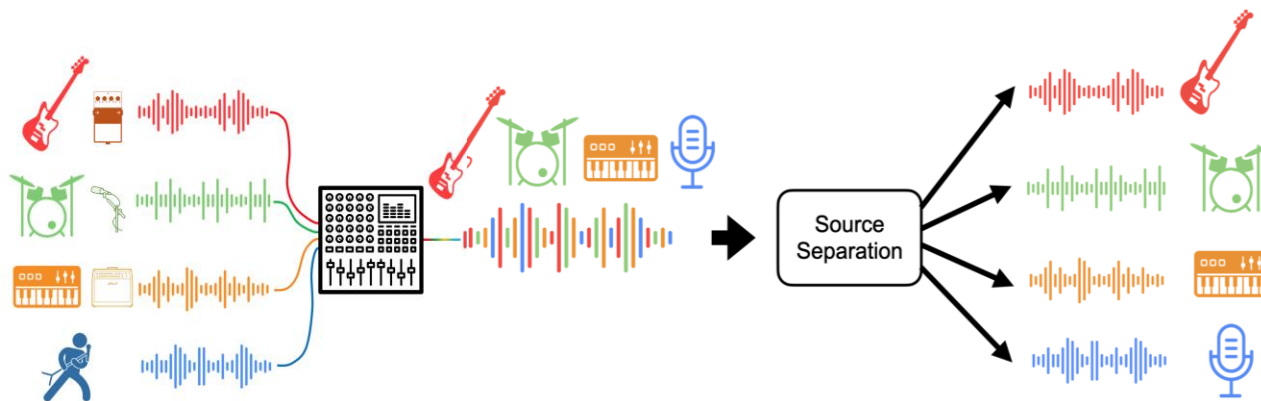
Music Source Separation



Juhan Nam

Introduction

- Music is composed of many instruments or vocals and they are all mixed in the waveform domain
- Music source separation is a task that isolate individual sources from the mixture
 - It is also called **demixing** as an inverse process of mixing

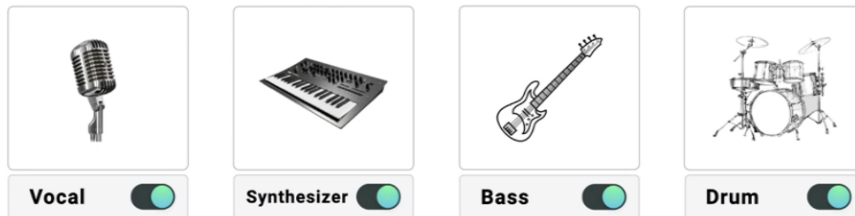


Introduction

- The output of music source separation is based on “stem” units
 - Two stems: vocal + accompaniment
 - Multiple stems: vocal, drum, bass, keyboards, and more

GSEP™ - Instrument Extraction Demo

🎵 BTS 'Dynamite'



GAUDIO

Applications

- Downstream tasks
 - Automatic music transcription: pitch, note
 - Lyrics recognition and alignment
 - Multiple instrument recognition
 - Dataset for other tasks (e.g. singing voice synthesis)
- Entertainment / Education
 - Karaoke
 - Performance practice
- New music content
 - Remixing: reuse the separated sources for different versions of the song
 - Upmixing: locate the separated sources in different spatial positions

How can music source separation be possible?

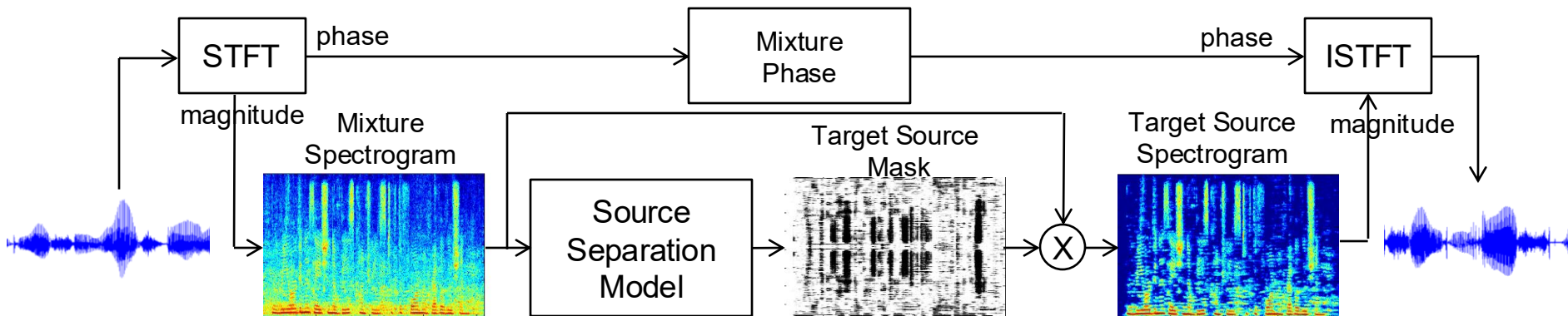
- It is an ill-posed (or underdetermined) problem because there are more unknown source outputs than the given mixed input

$$m(t) = \sum_{i=1}^N s_i(t) \longrightarrow s_i(t) \quad i = 1, 2, \dots, N$$

- However, human can recognize a particular sound source by the acoustic patterns (e.g., spectral shape, pitch range/contours) and **pay attentions to it**
 - We can repeat the **source-specific attention** for other instruments
- Machine can also separate the sources with a similar principle if the learning model is trained with a pair of mixed audio and its individual sources (supervised learning)

Basic pipeline

1. (Representation) use spectrogram as an audio representation
2. (mask) the source separation model estimates **2D time-frequency mask**
3. (magnitude) The **target source spectrogram** is estimated by **element-wise multiplication between the mixture spectrogram and the masks**
4. (phase) the mixture phase is copied to the target source phase



Evaluation Metrics

- Subjective evaluation: listening test
- Objective evaluation

- Estimated source: $\hat{s} = s_{target} + e_{interface} + e_{noise} + e_{artifact}$

- Source-to-Artifact Ratio (SAR): $10 \cdot \log_{10} \frac{\|s_{target} + e_{interface} + e_{noise}\|^2}{\|e_{artifact}\|^2}$

- Source-to-Interference Ratio (SIR): $10 \cdot \log_{10} \frac{\|s_{target}\|^2}{\|e_{interface}\|^2}$

- Source-to-Distortion Ratio (SDR): $10 \cdot \log_{10} \frac{\|s_{target}\|^2}{\|e_{interface} + e_{noise} + e_{artifact}\|^2}$

Music Source Separation Methods

- Traditional methods

- NMF

$$V_m = [W_{s1} \quad W_{s2}] \cdot \begin{bmatrix} H_{s1} \\ H_{s2} \end{bmatrix}$$

← Temporal weight of source #1 (s_1)
← Temporal weight of source #2 (s_2)

Fixed spectral components of source #1 (s_1)

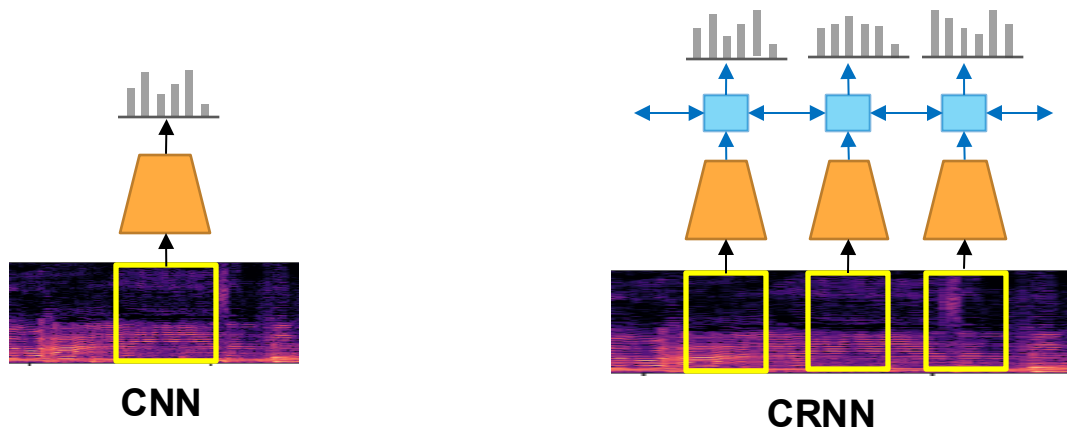
Fixed spectral components of source #2 (s_2)

- Deep learning methods

- U-Net: encoder/decoder architecture

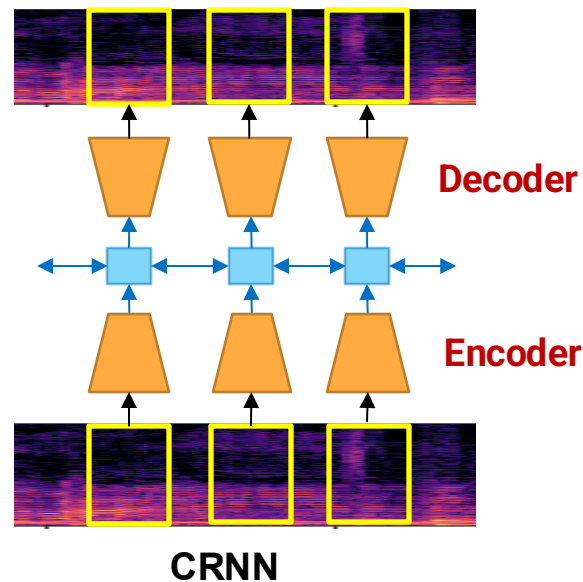
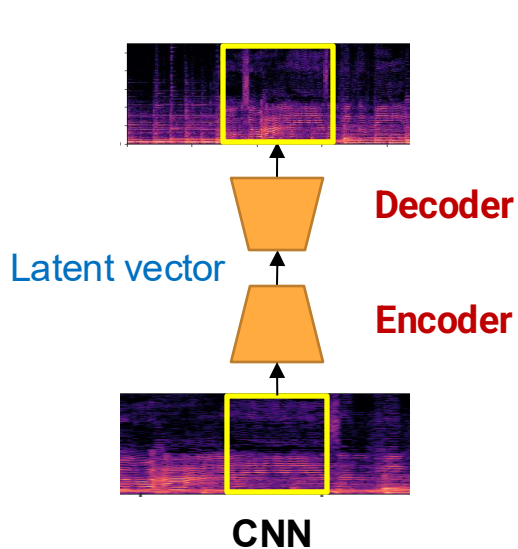
Encoder-Decoder Architecture

- So far, we have handled CNN and RNN in the classification framework
 - The output is either the softmax or sigmoid function
 - The loss function is the cross-entropy between the prediction output and the one-hot style ground-truth vectors
 - Audio-to-label (genre, mood, inst.) or audio-to-score (MIDI, beat, chord) tasks



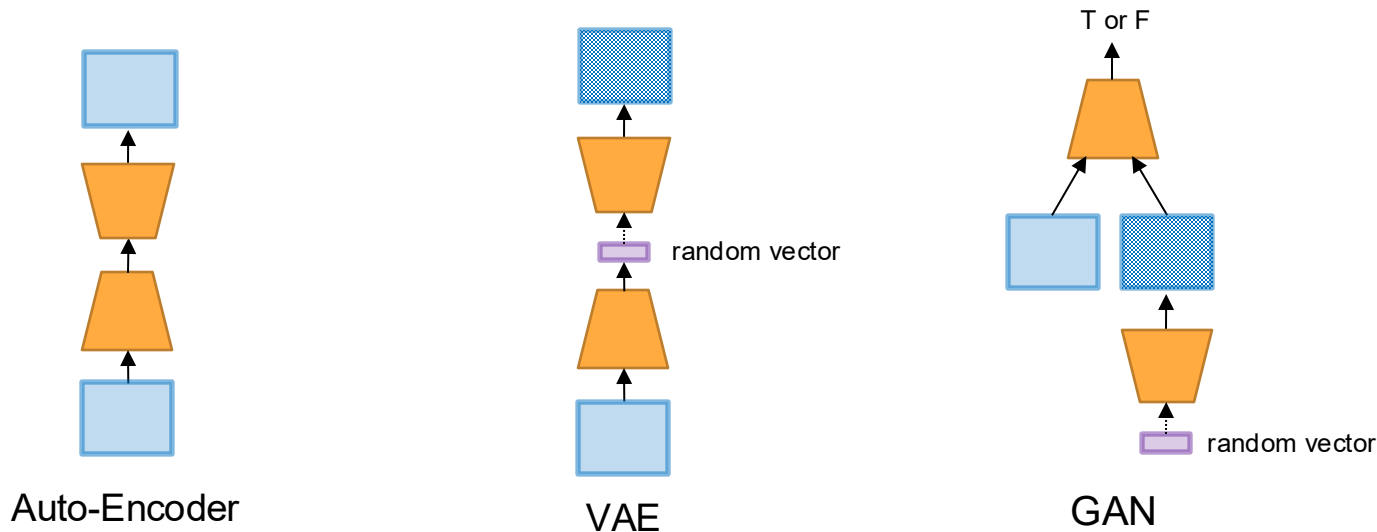
Encoder-Decoder Architecture

- What if the output is also audio?
 - For example, audio source separation or audio style transfer
 - Need an **inverse architecture (or decoder)** that converts the feature space back to the audio representation through upsampling



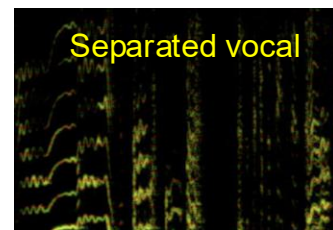
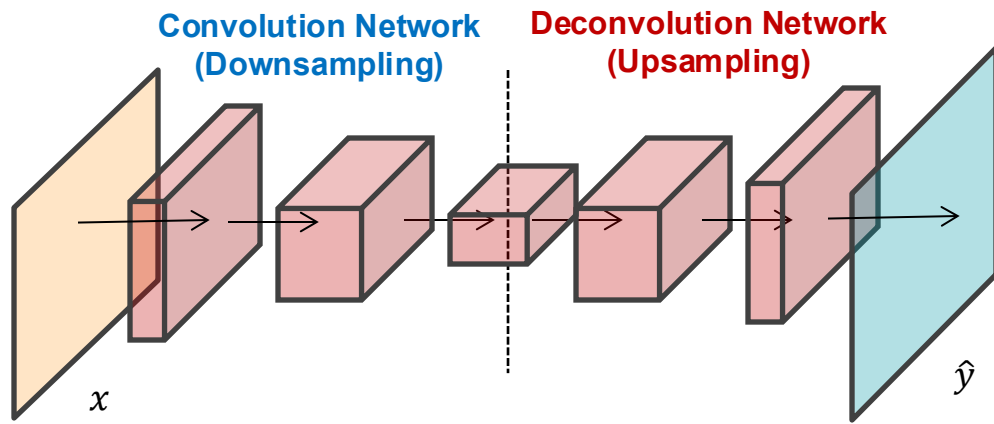
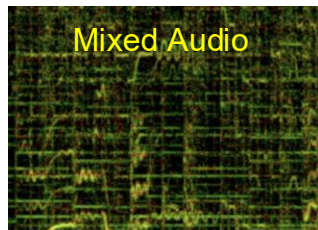
Encoder-Decoder Architecture

- The encoder-decoder architecture appears in
 - Auto-encoder (AE)
 - Variational auto-encoder (VAE)
 - Generative adversarial network (GAN)



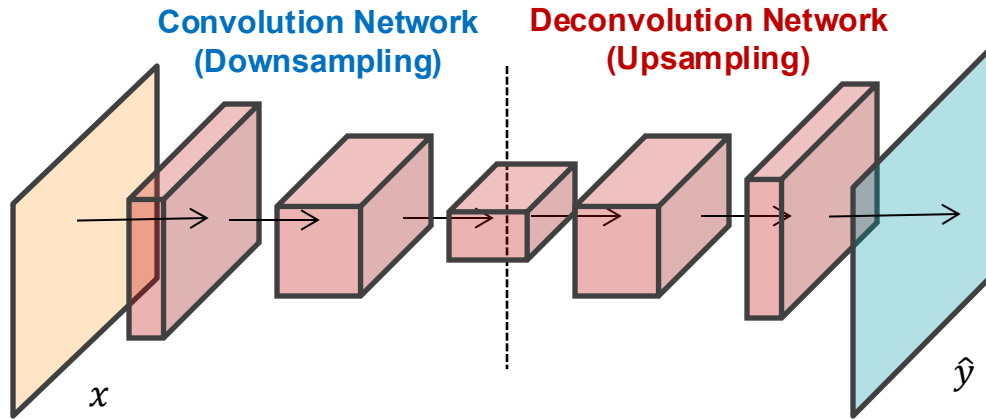
Encoder-decoder Architecture in Supervised Learning

- The encoder-decoder architecture is popularly used in supervised settings
 - Image segmentation
 - Music source separation



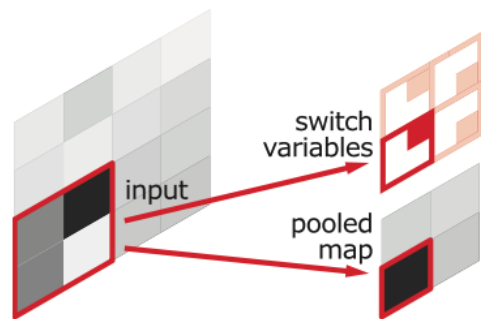
Encoder-decoder Architecture in Supervised Learning

- The encoder-decoder architecture is usually set to be **fully convolutional**
 - Encoder: **convolution and pooling** layers (down-sampling)
 - Decoder: **transposed convolution and unpooling** layers (up-sampling)



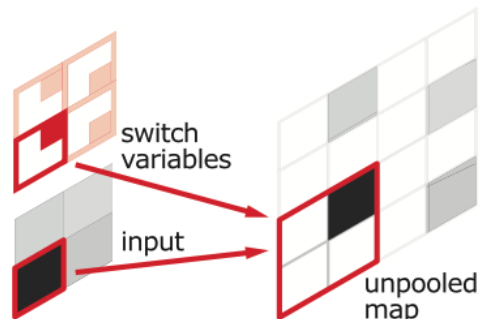
Unpooling

- Undo the pooling
 - Max-unpooling is a commonly used method
 - Store the position of maximum value in the encoder and use it to locate the input value on the unpooled map in the decoder; set the rest to zero



Pooling

2 x 2 max pooling
with a stride of 2

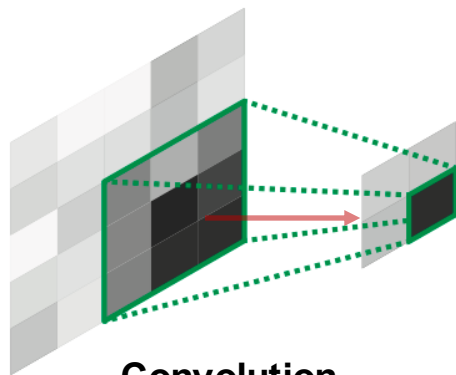


Unpooling

2 x 2 max unpooling
with a stride of 2

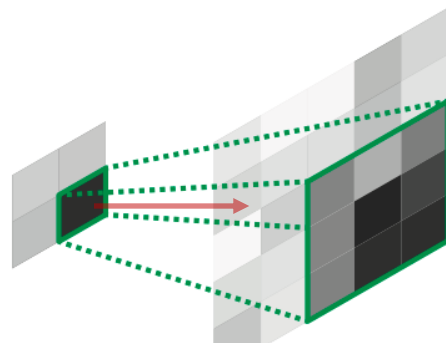
Transposed Convolution

- Map a single input element to the filter-size output
 - Input gives a weight for the filter
 - **Striding** is applied to the output: this **determines the upsampling factor**
 - The overlap between the filter-size output are summed



Convolution

3 x 3 filter
stride=2, no padding



Transposed Convolution

3 x 3 filter
stride=2, no padding

Transposed Convolution

- Convolution as matrix multiplication (1D example)

$$\begin{array}{c} \text{Filter} \quad \text{Input} \\ \vec{x} * \vec{a} = X\vec{a} \end{array}$$

$$\begin{array}{c} \text{Sliding filter} \\ \begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \end{array}$$

$$\begin{array}{c} \text{Input} \\ \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} \end{array}$$

$$= \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

1 x 3 filter
stride=1, padding=1

$$\begin{array}{c} \text{Filter} \quad \text{Input} \\ \vec{x} *^T \vec{a} = X^T \vec{a} \end{array}$$

$$\begin{array}{c} \text{Sliding filter} \\ \begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \end{array}$$

$$\begin{array}{c} \text{Input} \\ \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \end{array}$$

$$= \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

1 x 3 filter
stride=1, no padding

Transposed Convolution

- Convolution as matrix multiplication (1D example)

$$\begin{array}{c}
 \text{Filter} \quad \text{Input} \\
 \vec{x} * \vec{a} = X \vec{a}
 \end{array}$$

$$\begin{array}{c}
 \text{Sliding filter} \\
 \begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix}
 \end{array}
 \begin{array}{c}
 \text{Input} \\
 \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix}
 \end{array}
 =
 \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

1 x 3 filter
stride=2, padding=1

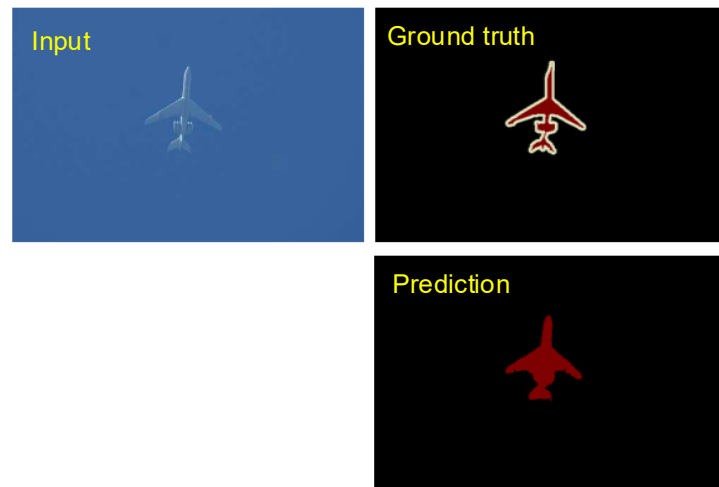
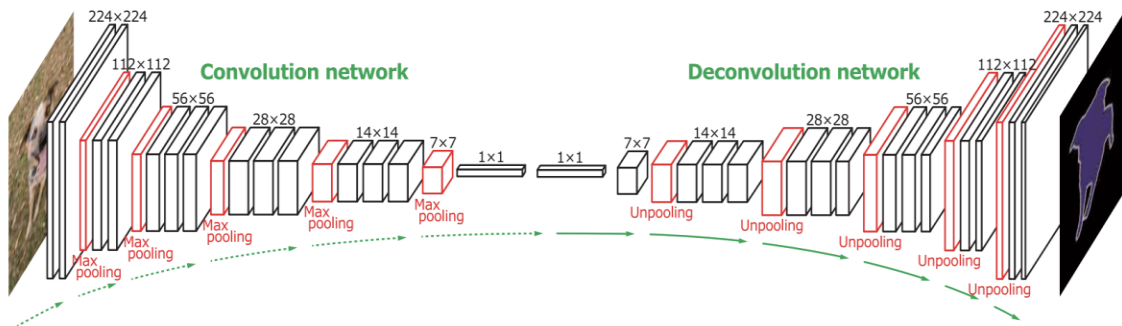
$$\begin{array}{c}
 \text{Filter} \quad \text{Input} \\
 \vec{x} *^T \vec{a} = X^T \vec{a}
 \end{array}$$

$$\begin{array}{c}
 \text{Sliding filter} \\
 \begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix}
 \end{array}
 \begin{array}{c}
 \text{Input} \\
 \begin{bmatrix} a \\ b \end{bmatrix}
 \end{array}
 =
 \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

1 x 3 filter
stride=2, no padding

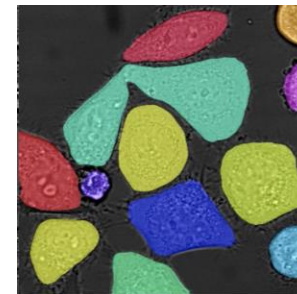
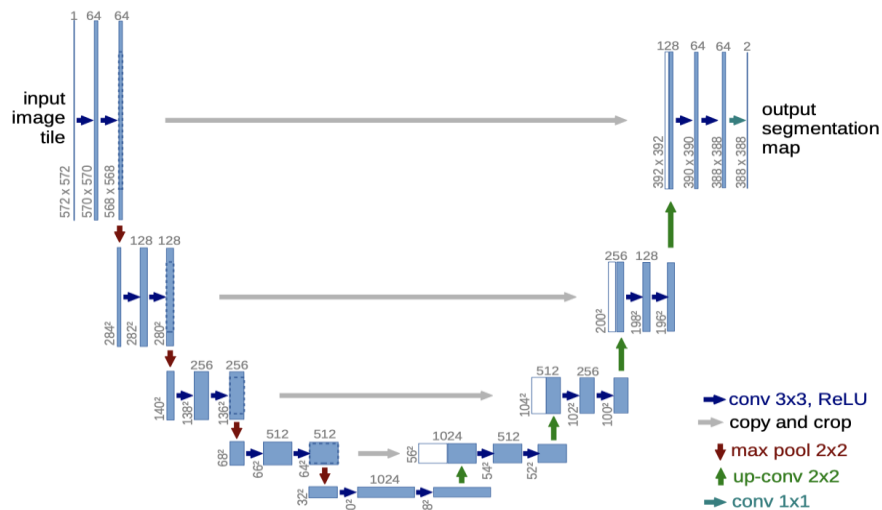
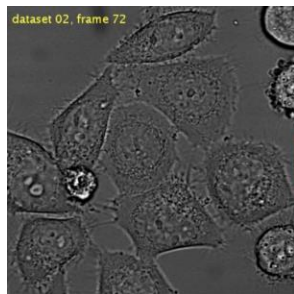
Example: Image Segmentation

- VGG-16-based CNN architecture
 - The segmentation boundary may be not precise



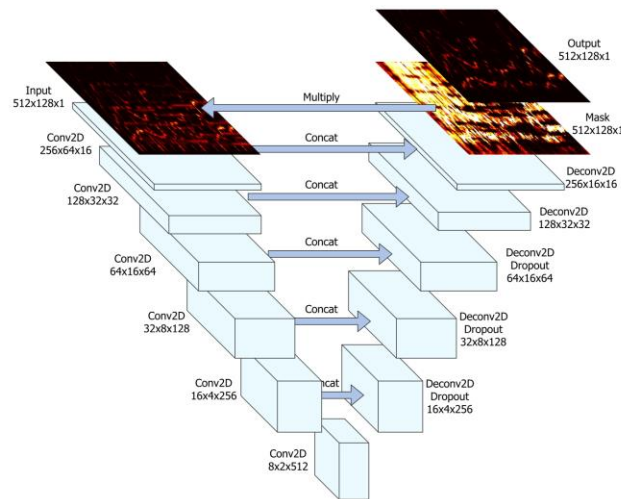
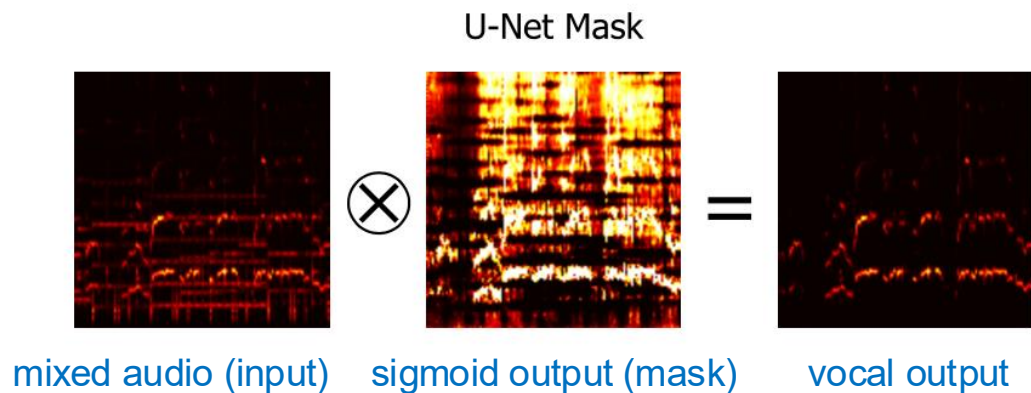
U-Net

- Skip connections between the same level of encoder and decoder layers
 - The direct feature maps from the corresponding encoder layer ensure **high precision of localization**
 - First introduced for medical image segmentation



Singing Voice Separation Using U-Net

- The precise segmentation by U-Net is effective in source separation
 - The input of U-Net is mixture spectrogram and the output is mask
 - Audio quality is very sensitive to small displacement of the reconstructed sound on the time-frequency representation



Singing Voice Separation Using U-Net

- Details of the U-Net architecture

- The waveform input is resampled to 8kHz and the spectrogram is normalized to the range [0,1]
- 5x5 Filter with a striding size of 2 in both conv and transposed conv layers
- No pooling
- The output is the sigmoid function ($\in [0.0, 1.0]$)
- Use L1 norm as a loss function: $L(X, Y; \Theta) = \|f(X, \Theta) \odot X - Y\|_{1,1}$

vocal source spectrogram (output)

mask (model output) mixture spectrogram (input)

Singing Voice Separation Using U-Net

- Datasets

- Collect 20k pairs of music tracks and its instrumental version
- The vocal spectrogram was obtained by $\max(X - Y_i, 0)$ at each T-F bin

- Results

- Baseline: the same model without skip connection
- Chimera: based on deep clustering

Mixture spectrogram

instrumental track spectrogram

	U-Net	Baseline	Chimera
NSDR Vocal	11.094	8.549	8.749
NSDR Instrumental	14.435	10.906	11.626
SIR Vocal	23.960	20.402	21.301
SIR Instrumental	21.832	14.304	20.481
SAR Vocal	17.715	15.481	15.642
SAR Instrumental	14.120	12.002	11.539

iKala mean scores

	U-Net	Baseline	Chimera
NSDR Vocal	8.681	7.877	6.793
NSDR Instrumental	7.945	6.370	5.477
SIR Vocal	15.308	14.336	12.382
SIR Instrumental	21.975	16.928	20.880
SAR Vocal	11.301	10.632	10.033
SAR Instrumental	15.462	15.332	12.530

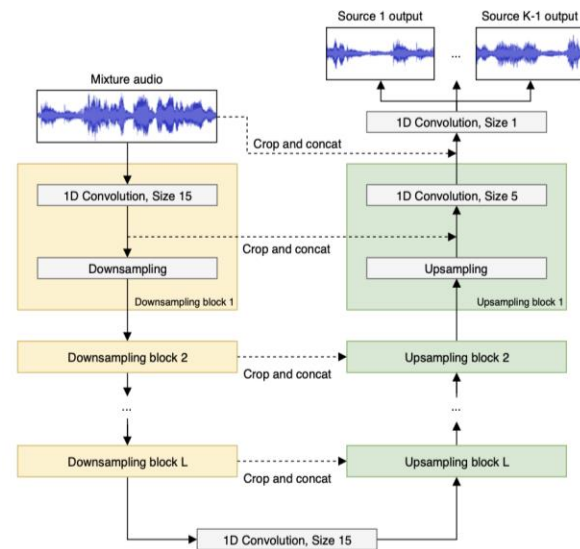
MedleyDB mean scores

Issues with Spectrogram

- STFT depends on many parameters: window size, hop size, window type
 - It is not trivial to find the optimal parameter setup
- Using the mixture phase for the output may cause artifacts
 - In particular when harmonic partials from different sources are overlapping
 - Alternatively, we can estimate the phase from the magnitude
 - The **Griffin-Lim algorithm** is one of the most popular methods (<https://librosa.org/doc/latest/generated/librosa.griffinlim.html>)
 - But, the iterative algorithm is slow
- Finally, ignoring the mixture phase in the estimation process can miss some information about the source

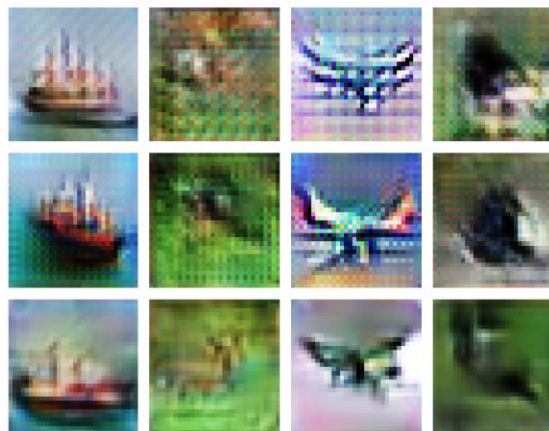
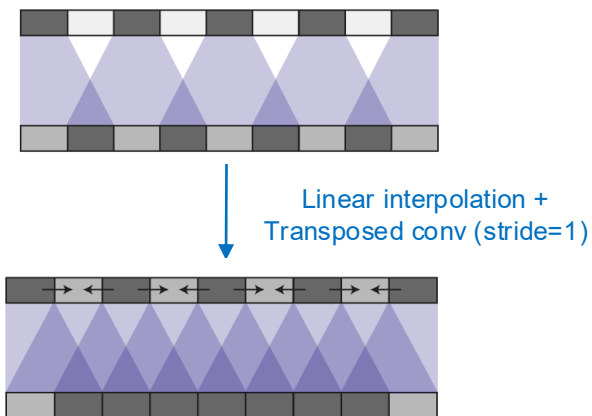
Source Separation Using Wave U-Net

- Use raw waveforms as input
 - 1D convolution and deeper networks
 - Multiple source separation outputs
 - Direct source estimation as waveforms
 - Use a tanh non-linearity in the last layer to ensure to the range $[-1,1]$
- Add the **additivity constraint** to the loss
 - The sum of the separated sources to be the same as the original mixture
- Use a larger input size than the output size to learn a wider input context
- Do **interpolation** on the input feature map to avoid aliasing noise



Transposed Convolution and Artifacts

- Transposed convolution with a striding size of 2 or more has “uneven overlap”
 - This causes the checkerboard artifacts
 - A solution: do **linear interpolation** on input and transposed convolution with a striding size of 1



Deconv in last two layers.
Other layers use resize-convolution.
Artifacts of frequency 2 and 4.

Deconv only in last layer.
Other layers use resize-convolution.
Artifacts of frequency 2.

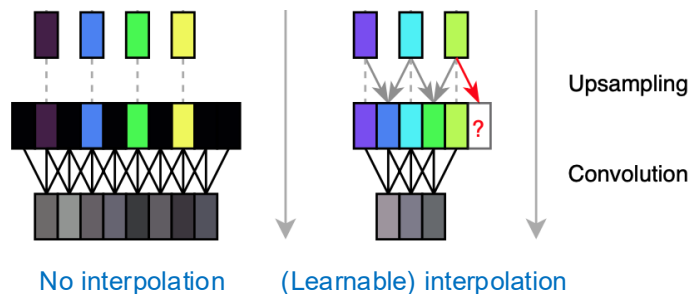
All layers use resize-convolution.
No artifacts.

Source Separation Using Wave U-Net

- Linear interpolation is simple and effective to avoid the aliasing noise
 - The interpolation takes the average from the two adjacent positions
 - But it is not optimized
- Learnable upsampling
 - Every input element can have a different weight

$$f_{t+0.5} = \sigma(w) \odot f_t + (1 - \sigma(w)) \odot f_{t+1}$$

$$f_t, f_{t+1} \in \mathbb{R}^F \quad w \in \mathbb{R}^F$$



Source Separation Using Wave U-Net

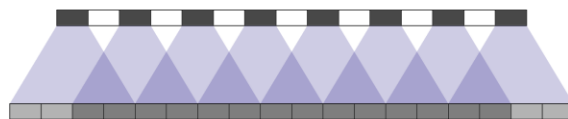
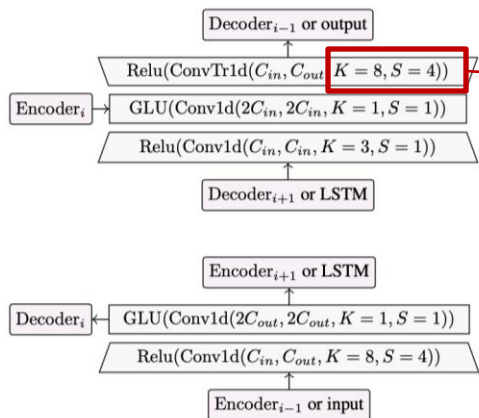
- Results

- Trained and tested with the MUSDB multi-track dataset
- Used the SDR metric only

		M1	M2	M3	M4	M5	M7	U7	U7a
Voc.	Med.	3.90	3.92	3.96	4.46	4.58	3.49	2.76	2.74
	MAD	3.04	3.01	3.00	3.21	3.28	2.71	2.46	2.54
	Mean	-0.12	0.05	0.31	0.65	0.55	-0.23	-0.66	0.51
	SD	14.00	13.63	13.25	13.67	13.84	13.00	12.38	10.82
Acc.	Med.	7.45	7.46	7.53	10.69	10.66	7.12	6.76	6.68
	MAD	2.08	2.10	2.11	3.15	3.10	2.04	2.00	2.04
	Mean	7.62	7.68	7.66	11.85	11.74	7.15	6.90	6.85
	SD	3.93	3.84	3.90	7.03	7.05	4.10	3.67	3.60

Source Separation Using Wave U-Net and RNN

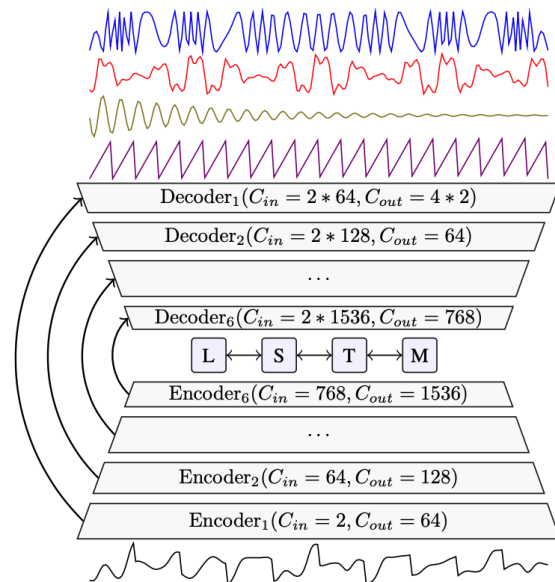
- Similar to Wave U-Net but have a recurrent layer
 - 1x1 convolution and gated linear unit (GLU)
 - LSTM between the encoder and decoder
 - Set the filter size to be twice the stride size in the transposed convolution: "even overlap"



Even overlap

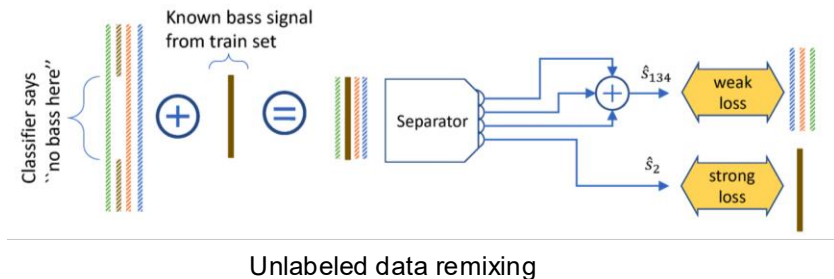
$$h_l(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c})$$

GLU (a trainable and smoother version of ReLU)



Source Separation Using Wave U-Net and RNN

- Unlabeled data remixing
 - Trained a classifier to detect the presence of each stem
 - Built a new dataset by mixing an absent source from the training set
 - The manually remixed source is “strongly label data”
- Better results than Wave U-Net



Architecture	Wav?	Extra data		Test SDR in dB				
		labeled	unlabeled	All	Drums	Bass	Other	Vocals
MMDenseNet	✗	✗	✗	5.34	6.40	5.14	4.13	6.57
Wave-U-Net	✓	✗	✗	3.17	4.16	3.17	2.24	3.05
Demucs	✓	✗	✗	4.81	5.38	5.07	3.01	5.44
Demucs	✓	✗	2,000	5.09	5.79	6.23	3.45	5.51
Demucs	✓	100	✗	5.41	5.99	5.72	3.65	6.17
Demucs	✓	100	2,000	5.67	6.50	6.21	3.80	6.21
MMDenseLSTM	✗	804	✗	5.97	6.75	5.28	4.72	7.15
MMDenseNet	✗	804	✗	5.85	6.81	5.21	4.37	6.74

Resources

- SigSep: open resources for music source separation
 - <https://sigsep.github.io/>
- ISMIR 2020 Tutorial
 - <https://source-separation.github.io/tutorial>
- Pretrained models
 - Demucs: <https://github.com/facebookresearch/demucs>
 - Spleeter: <https://github.com/deezer/spleeter>
 - ByteDance: https://github.com/bytedance/music_source_separation
- PyTorch Audio
 - https://pytorch.org/audio/main/tutorials/hybrid_demucs_tutorial.html