2024 edition

Deep Learning for Music Analysis and Generation

Source Separation

(audio → audio)

Yi-Hsuan Yang Ph.D.
yhyangtw@ntu.edu.tw

Objectives

- To be familiar with how people **work on spectrograms**

- It's a type of *conditional* audio **generation** task (with *strong condition*)
  - **"audio (mixture) → audio (stem)"**
  - Will talk about other types of audio generation tasks in the forthcoming lectures

Reference 1: FMP Notebook

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8.html

Part	Title	Notions, Techniques & Algorithms	HTML	IPYNB
B	Basics	Basic information on Python, Jupyter notebooks, Anaconda package management system, Python environments, visualizations, and other topics	[html]	[ipynb]
0	Overview	Overview of the notebooks (https://www.audiolabs-erlangen.de/FMP)	[html]	[ipynb]
1	Music Representations	Music notation, MIDI, audio signal, waveform, pitch, loudness, timbre	[html]	[ipynb]
2	Fourier Analysis of Signals	Discrete/analog signal, sinusoid, exponential, Fourier transform, Fourier representation, DFT, FFT, STFT	[html]	[ipynb]
3	Music Synchronization	Chroma feature, dynamic programming, dynamic time warping (DTW), alignment, user interface	[html]	[ipynb]

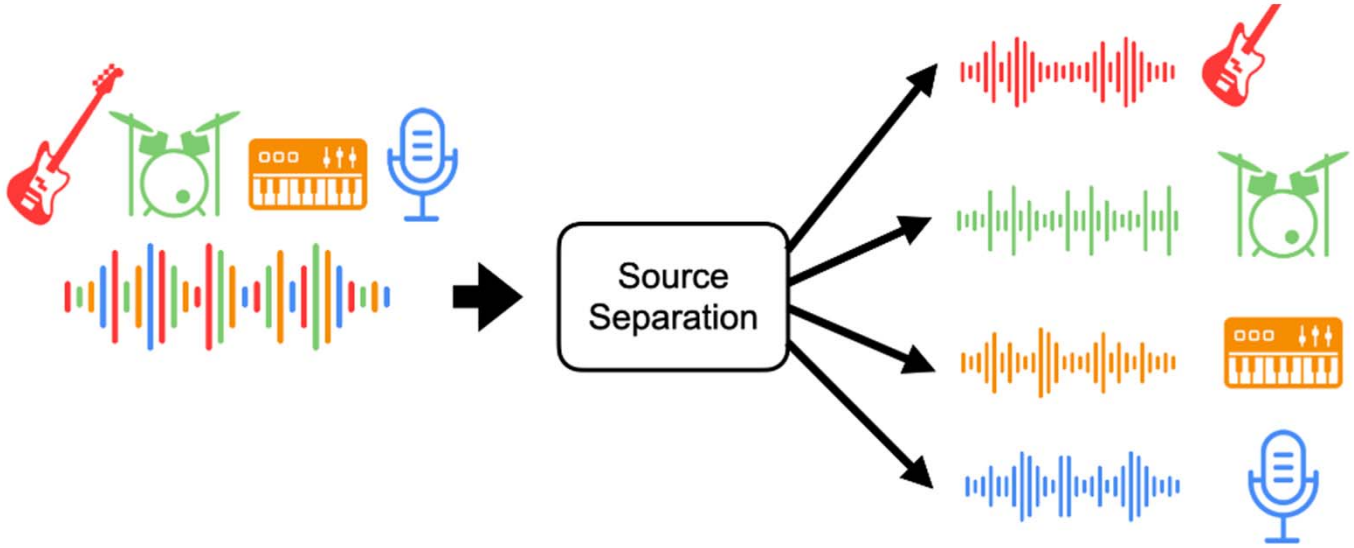
Part	Title	Notions, Techniques & Algorithms	HTML	IPYNB
4	Music Structure Analysis	Similarity matrix, repetition, thumbnail, homogeneity, novelty, evaluation, precision, recall, F-measure, visualization, scape plot	[html]	[ipynb]
5	Chord Recognition	Harmony, music theory, chords, scales, templates, hidden Markov model (HMM), evaluation	[html]	[ipynb]
6	Tempo and Beat Tracking	Onset, novelty, tempo, tempogram, beat, periodicity, Fourier analysis, autocorrelation	[html]	[ipynb]
7	Content-Based Audio Retrieval	Identification, fingerprint, indexing, inverted list, matching, version, cover song	[html]	[ipynb]
8	Musically Informed Audio Decomposition	signal reconstruction, instantaneous frequency, fundamental frequency (F0), trajectory, nonnegative matrix factorization (NMF)	[html]	[ipynb]

Reference 2: ISMIR 2020 Tutorial

https://source-separation.github.io/tutorial/landing.html

Open Source Tools & Data for Music Source Separation

By Ethan Manilow, Prem Seetharaman, and Justin Salamon



Outline

- **General idea**
- Traditional methods
- Deep learning based methods

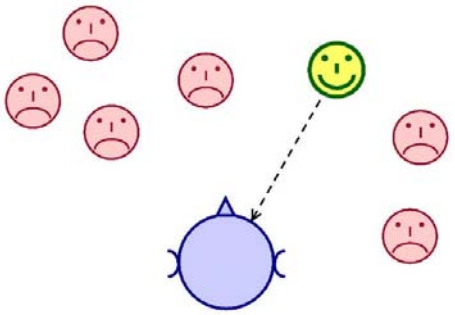
What is Source Separation?

https://source-separation.github.io/tutorial/intro/src_sep_101.html

- The process of isolating individual sounds (*sources*) in an auditory mixture of multiple sounds
- *Underdetermined* problem
  - Fewer observations $y(t)$ (1 or 2; mono or stereo) than the required outcomes $x_i(t)$ (e.g., 4)

$$y(t) = \sum_{i=1}^{N} x_i(t).$$

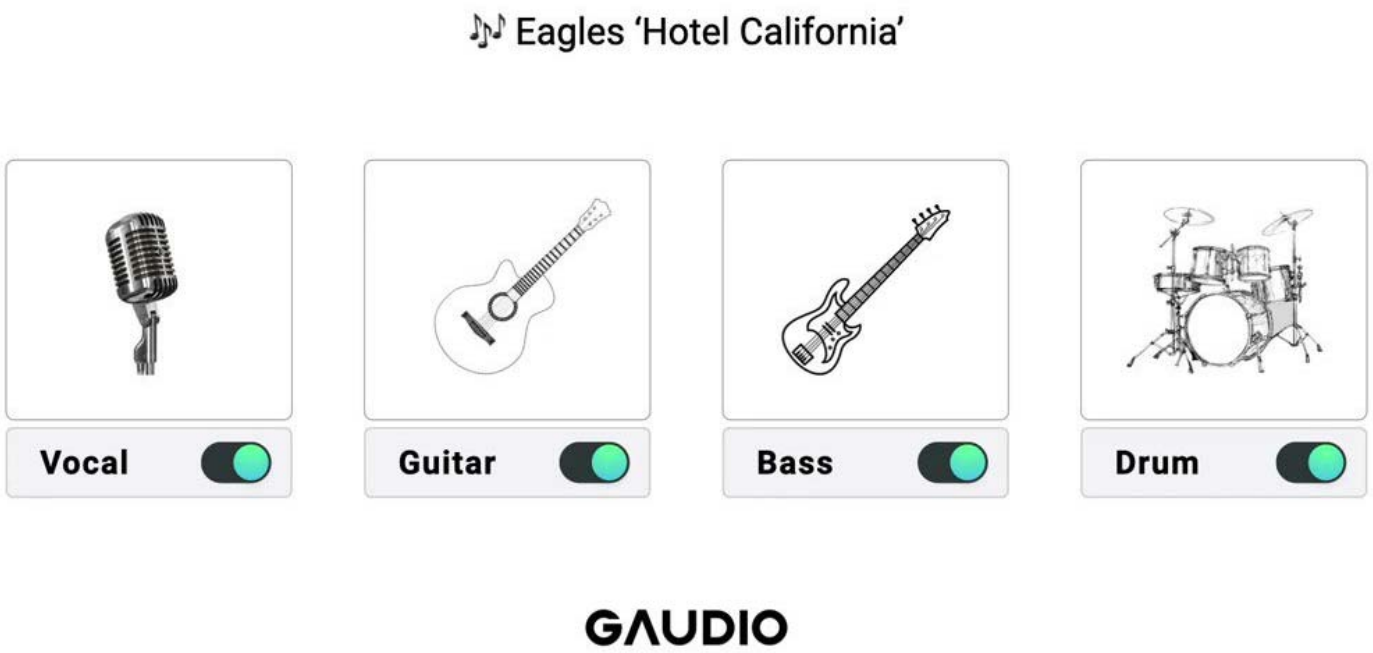
- In speech: cocktail party effect



Cocktail-Party-Effekt:
Herausfiltern einer Schallquelle bei Anwesenheit mehrere Schallquellen

Demo: Source Separation

https://www.gaudiolab.com/technology/source-separation



Why Source Separation?

https://source-separation.github.io/tutorial/intro/src_sep_101.html

- Benefit **downstream MIR** problems
  - automatic music transcription [PAB+02,MSP20],
  - lyric and music alignment [FGO+06],
  - musical instrument detection [HKV09],
  - lyric recognition [MV10],
  - automatic singer identification [WWollmerS11,HL15,SDL19],
  - vocal activity detection [SED18a],
  - fundamental frequency estimation [JBEW19], and
  - understanding the predictions of black-box audio models. [HMW20a,HMW20b]

Why Source Separation?

https://source-separation.github.io/tutorial/intro/src_sep_101.html

- Benefit **music generation**
  - Re-mix of the sources
  - Up-mix: stereo to 5.1-channel
  - Replacement of some of the sources
  - Audio editing
- **Active music listening**

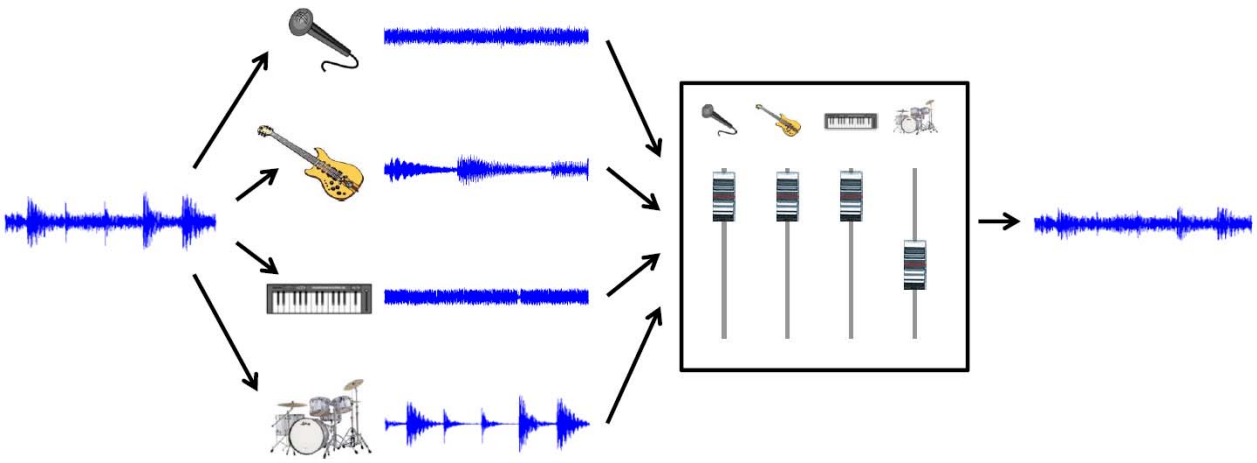
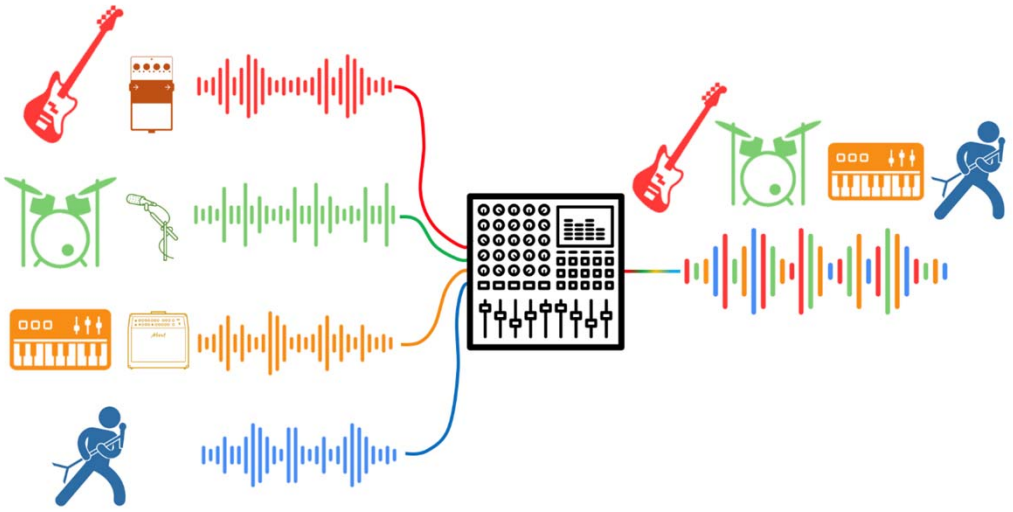


Figure from [Mueller, FPM, Chapter 8, Springer 2015]

Why Source Separation is Difficult?

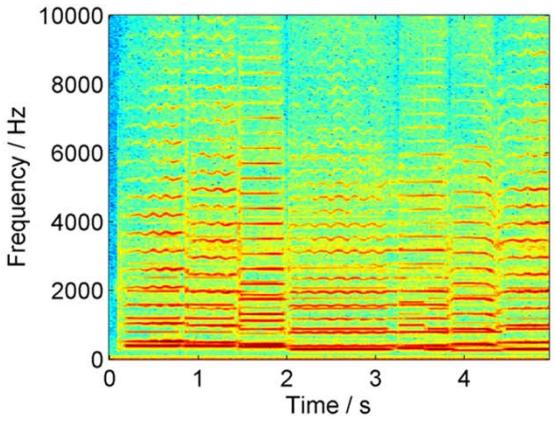
https://source-separation.github.io/tutorial/intro/src_sep_101.html

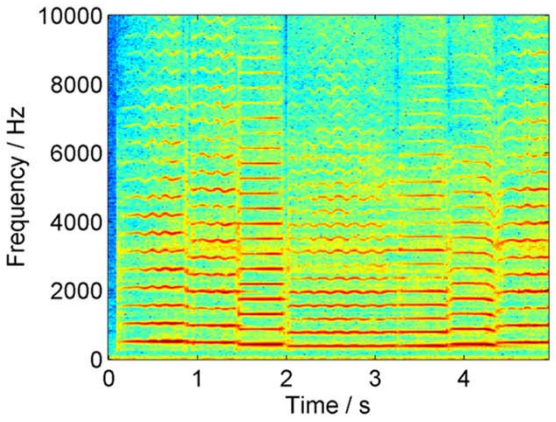
- Sources in music are **highly correlated** (harmonically and rhythmically)
- The **mixing** of music signals is **complex** and **non-linear**
  - Reverb, EQ, …
  - Don't know how the mixing was done
- It's actually an **audio-domain music generation** problem
  - Instrument recognition (*discriminative*) vs. instrument separation (*generative*)
  - The bar for quality can be high

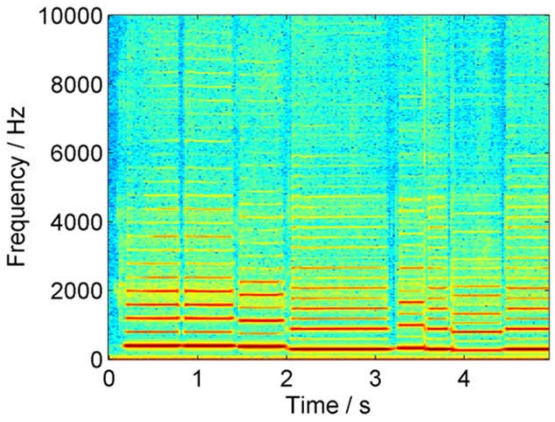


Why Source Separation is Difficult?

- Sources in music are **highly correlated** (harmonically and rhythmically)







mixture

violin

clarinet

Why Source Separation is Difficult?

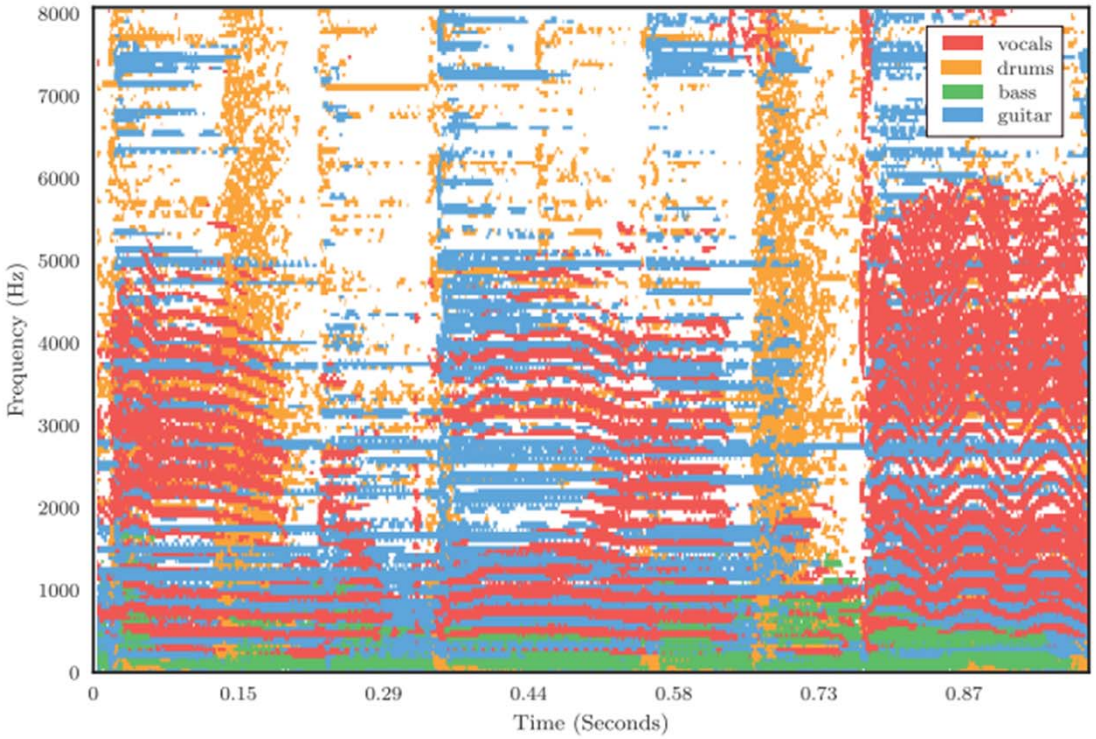
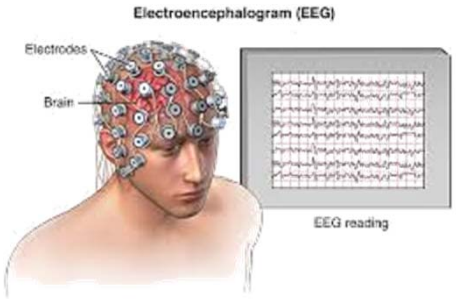


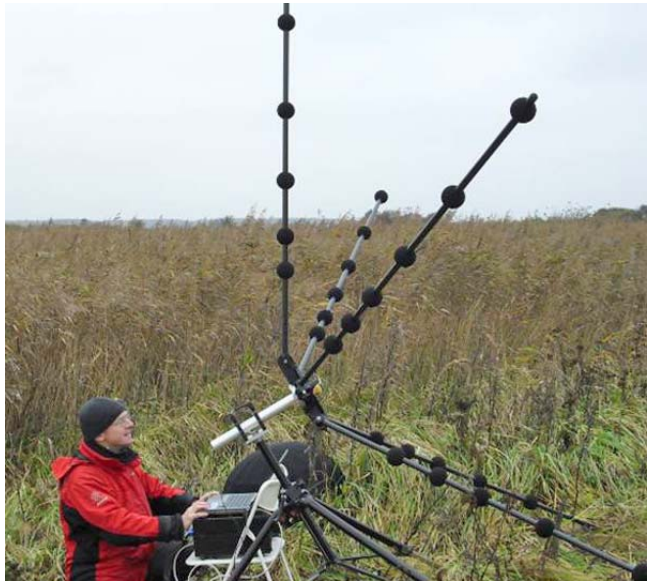
Figure 1: Representation of a music mixture in the time-frequency domain. The dominant musical source in each time-frequency bin is displayed with a different color.

Ref: Cano et al, "Musical source separation: An introduction," IEEE Signal Processing Magazine 2019

Types of Separation Problems

- #sources vs. #output channels
  - Overdetermined vs **underdetermined**
- Amount of side information
  - **Blind** source separation vs. **informed** source separation
    - *Score* informed
    - *Lyrics* informed
    - *Melody* informed
  - We mainly talk about blind & underdetermined source separation
- Online or offline





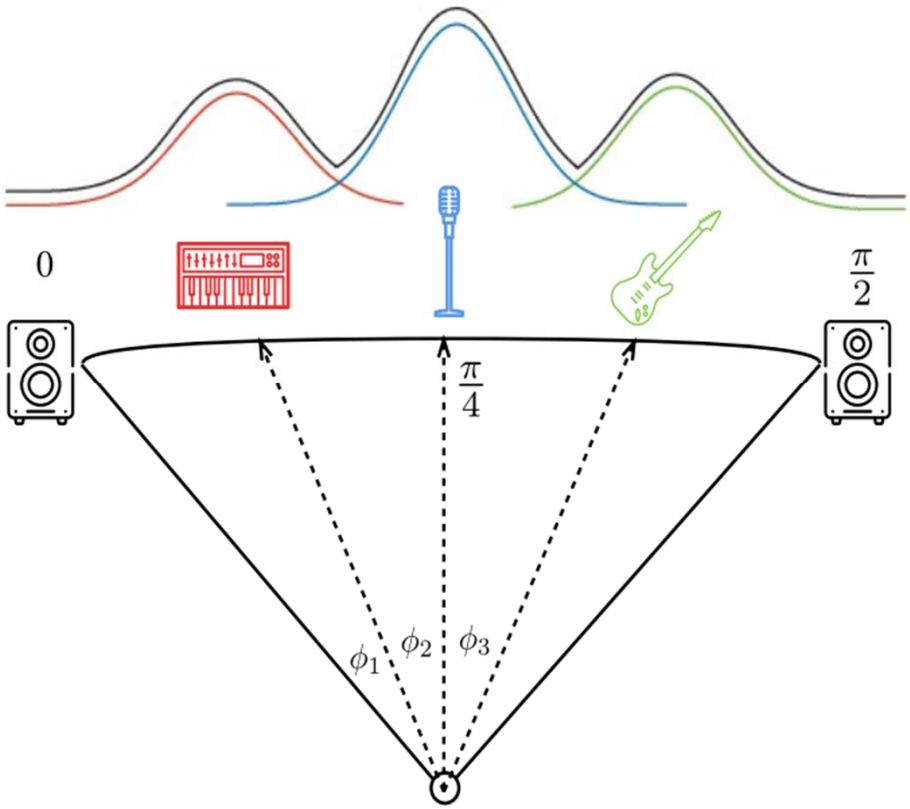
Types of Separation Problems

- What the #output channels are
  - **Two** stems: vocal vs. *non-vocal*
    - or: lead vs. *accompaniments*
  - **Four** stems: vocal, drums, bass, and *others*
  - Beyond four stems
  - Uncertain number and class of output channels

- Do different output channels correspond to different instruments?
  - Not always
    - Choral music separation (soprano, alto, tenor, and bass)
    - Speaker separation

Clues for Stereo Source Separation

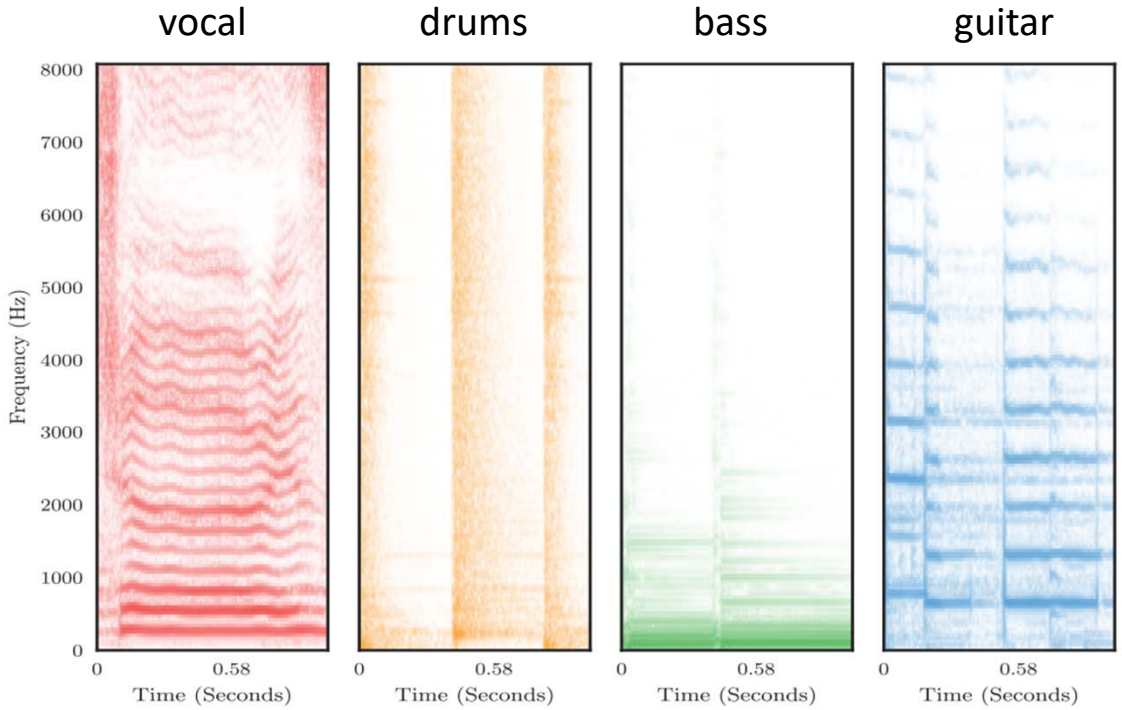
- Utilizing *spatial position* for separation
  - Vocal are usually in the middle



Ref: Cano et al, "Musical source separation: An introduction," IEEE Signal Processing Magazine 2019

Clues for Monaural Source Separation

- Different sound sources may have different *time-frequency characteristics* (timbre, pitch range, etc)



Ref: Cano et al, "Musical source separation: An introduction," IEEE Signal Processing Magazine 2019

Spectrogram-based and Waveform-based

https://paperswithcode.com/sota/music-source-separation-on-musdb18

- STFT = magnitude + phase
- People tend to use the **magnitude STFT**
  - provides rich info as a time-frequency representation
- Phase is hard to model
- Also hard model waveforms (in early days)

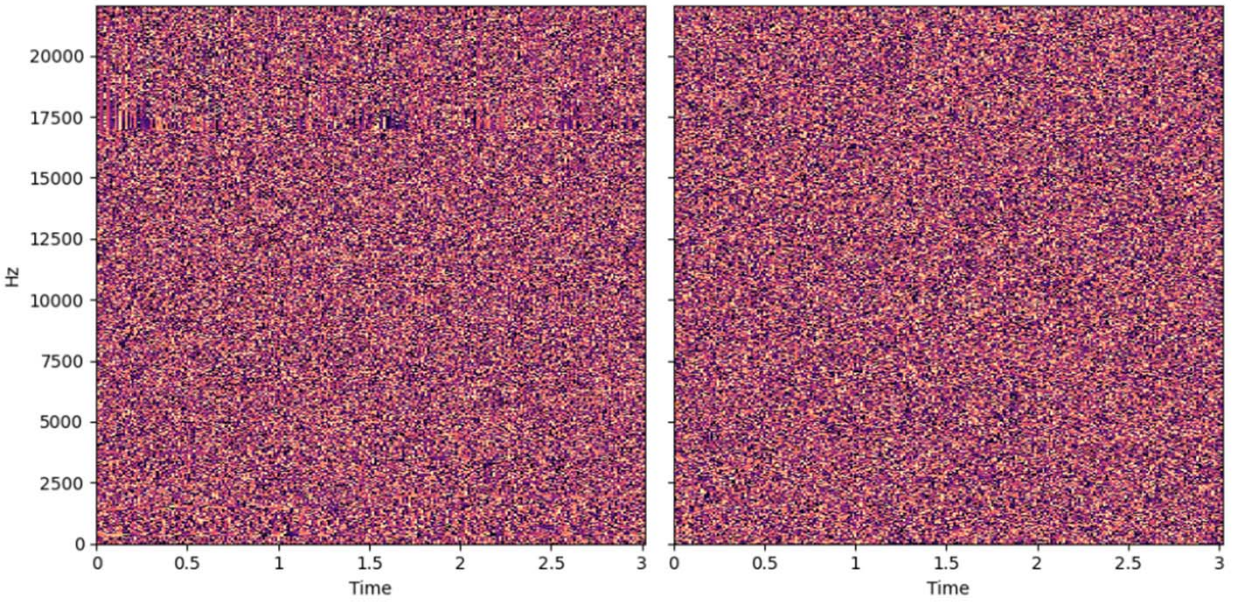


Fig. 17 The structure of phase within an STFT makes it hard to model. One of these two images shows the phase component of an STFT and another shows random noise. Can you guess which is which?

Approach

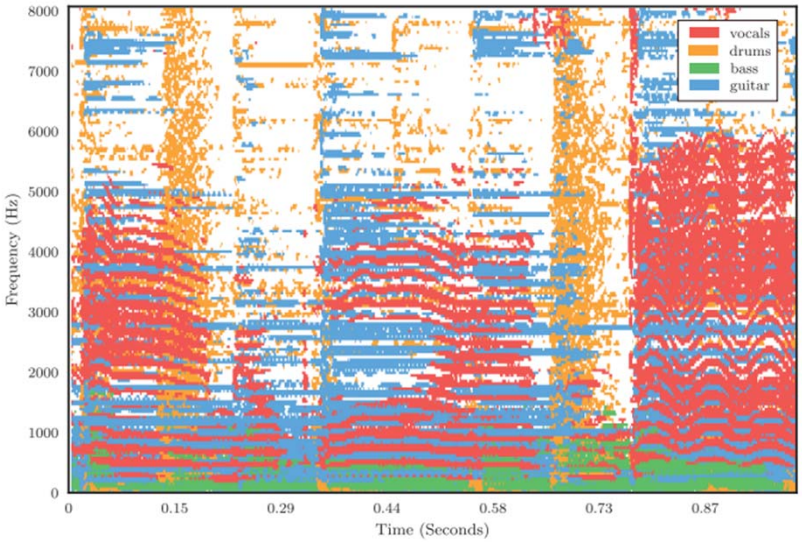
- **Traditional** methods
  - *Unsupervised*: rule-based, model-based
  - Faster, light-weight, but limited performance
  - Usually work on spectrograms

- **Deep learning** based methods
  - *Supervised*: learn from “clean sources”
  - Mixture in, clean sources out
  - Better result
  - Work on spectrograms, waveforms, or both

ML Viewpoint: Time-frequency Classification

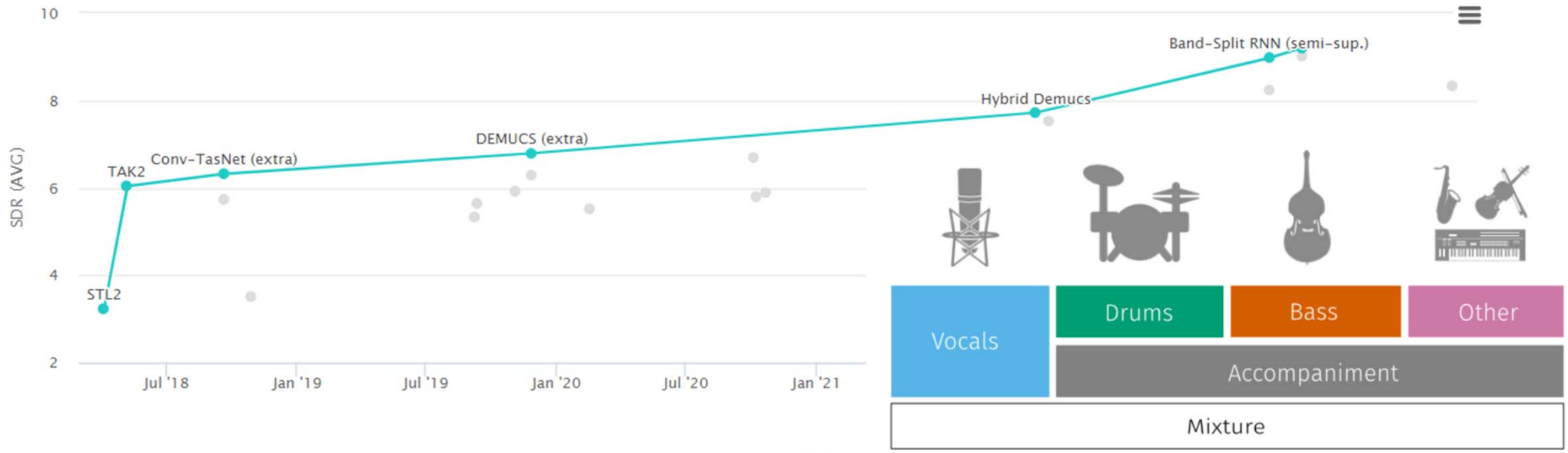
- **Per song:** genre classification
- **Per chunk:** instrument activity detection
- **Per time-frequency point:** f0 estimation, multi-pitch estimation, ***source separation***

- Input and output are of the same shape



Benchmark 1

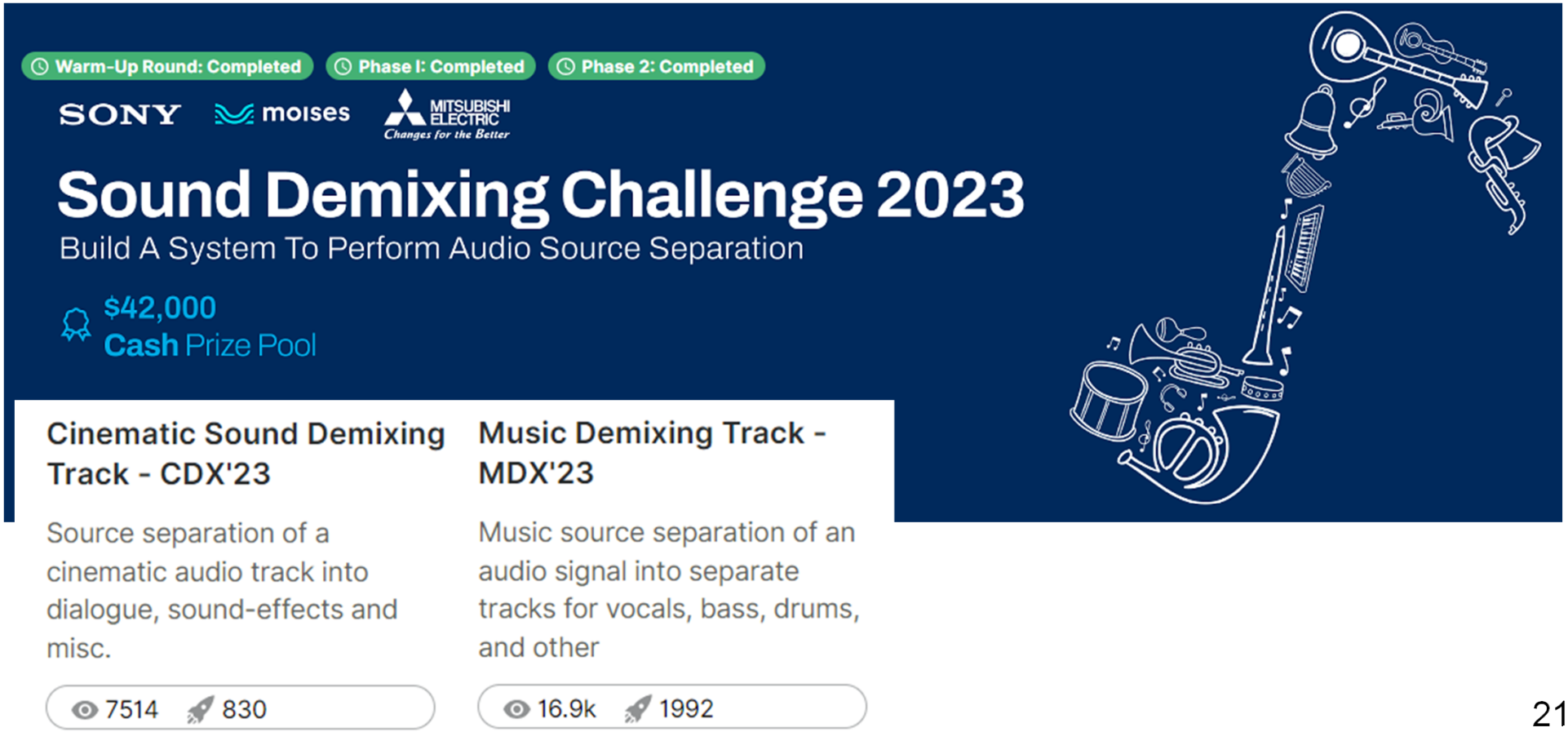
https://paperswithcode.com/sota/music-source-separation-on-musdb18



Rank	Model	SDR (avg)	SDR (vocals)	SDR (drums)	SDR (bass)	SDR (other)	Extra Training Data	Paper	Code	Result	Year	Tags
1	**Sparse HT Demucs** (fine tuned)	9.20	9.37	10.83	10.47	6.41	✓	Hybrid Transformers for Music Source Separation			2022	

Benchmark 2

https://www.aicrowd.com/challenges/sound-demixing-challenge-2023



Evaluation Metrics

https://source-separation.github.io/tutorial/basics/evaluation.html

- **Computed in the time-domain**
- **Source-to-Distortion Ratio** (**SDR**)
- Source-to-Interference Ratio (SIR)
- Source-to-Artifact Ratio (SAR)
  - true sources: **a**, **b**
  - estimated sources: **ae**, **be**
  - SDR(a): how **ae** is similar to **a**
  - SIR(a): how **ae** is similar to **b**
  - SAR(a): how **ae** is not similar to either **a or b**

Evaluation Metrics

https://docs.google.com/presentation/d/1XLC7SyGMRfOj3WwJaiyaYFOwCI69w4aXWLYl7UEsvXQ/

Source Separation Metrics: What are they really measuring?

Keynote presentation at the 2021 Music Demixing Workshop

$$\hat{s} = s_{target} + e_{interf} + e_{artif}$$

$$s_{target} = P_s \hat{s} = \frac{< \hat{s}, s >}{< s, s >} s$$

A rescaled s, which is as close as possible to s-hat

$$e_{interf} = P_n \hat{s} = \frac{< \hat{s}, n >}{< n, n >} n$$

A rescaled n, which is as close as possible to s-hat

$$e_{artif} = \hat{s} - s_{target} - e_{interf} = \hat{s} - \frac{< \hat{s}, s >}{< s, s >} s - \frac{< \hat{s}, n >}{< n, n >} n$$

What remains of s-hat

Evaluation Library: mir_eval

https://github.com/craffel/mir_eval

- mir_eval can be used in most MIR tasks as well (e.g., chord recognition, onset detection, segmentation)

Examples

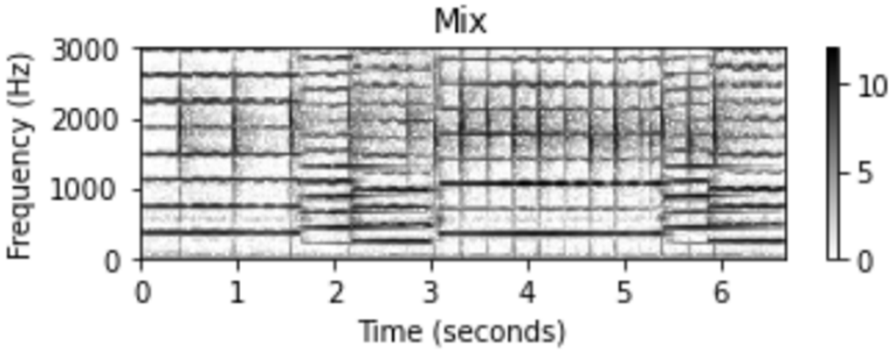
```
>>> # reference_sources[n] should be an ndarray of samples of the
>>> # n'th reference source
>>> # estimated_sources[n] should be the same for the n'th estimated
>>> # source
>>> (sdr, sir, sar,
...  perm) = mir_eval.separation.bss_eval_sources(reference_sources,
...                                               estimated_sources)
```

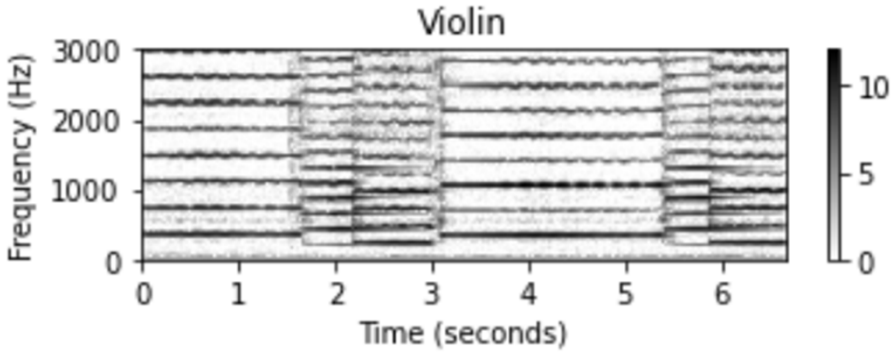
Outline

- General idea
- **Traditional methods** (*non-DL*, *unsupervised*)
  - **Median filtering**
  - Nonnegative matrix factorization (NMF)
  - Robust principal component analysis (RPCA)
- Deep learning based methods

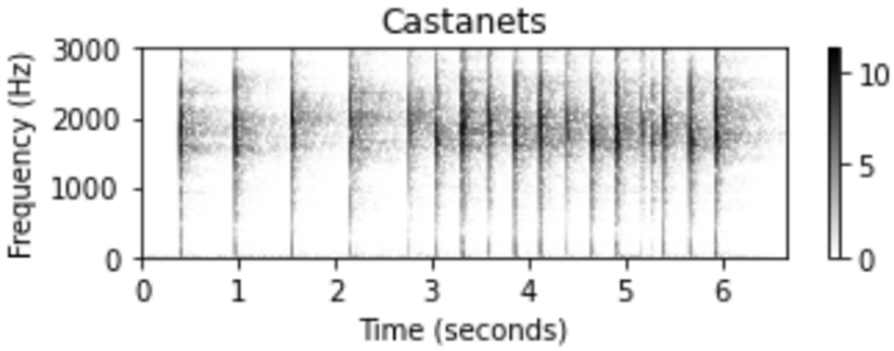
Harmonic Percussive Source Separation (HPSS)

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S1_HPS.html



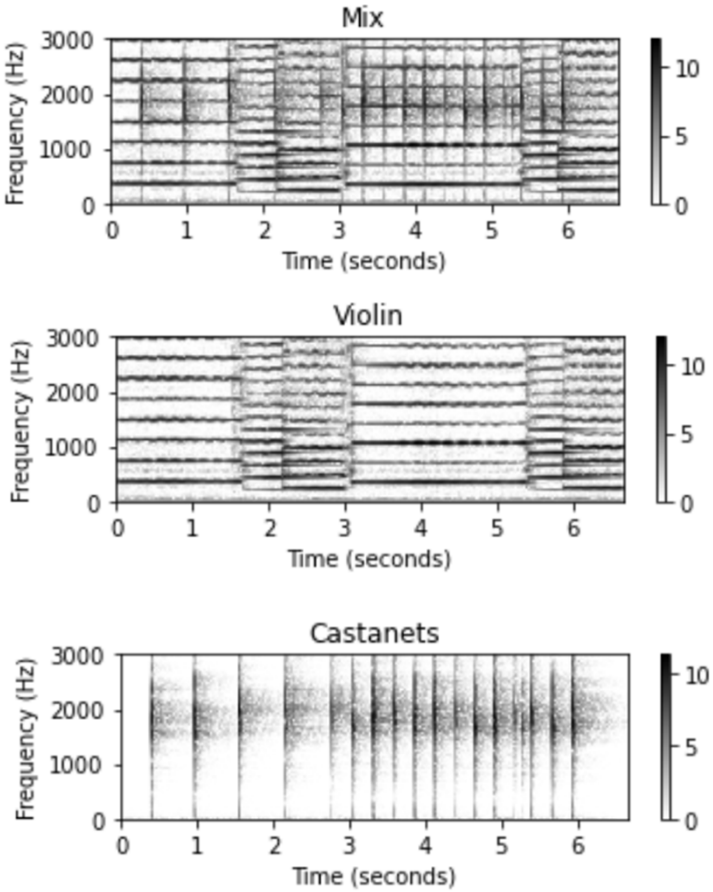


- Work on the **spectrogram**



Median-Filtering Approach

- Intuition
  - stable **harmonic** or stationary components form horizontal lines on the spectrogram
  - **percussive** components form vertical lines with a broadband frequency response
- Simple DSP techniques; no ML




```
A = np.array([5.,3,2,8,2])
filter_len = 3
A_result = signal.medfilt(A, kernel_size=filter_len)
print('A        = ', A)
print('A_result = ', A_result)
```

```
A        =  [5. 3. 2. 8. 2.]
A_result =  [3. 3. 3. 2. 2.]
```

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S1_HPS.html

Ref 1: Ono et al, "Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram," EUSIPCO 2008

Ref 2: FitzGerald, " Harmonic/percussive separation using median filtering," DAFx 2010

HPSS via Median Filtering & Masking

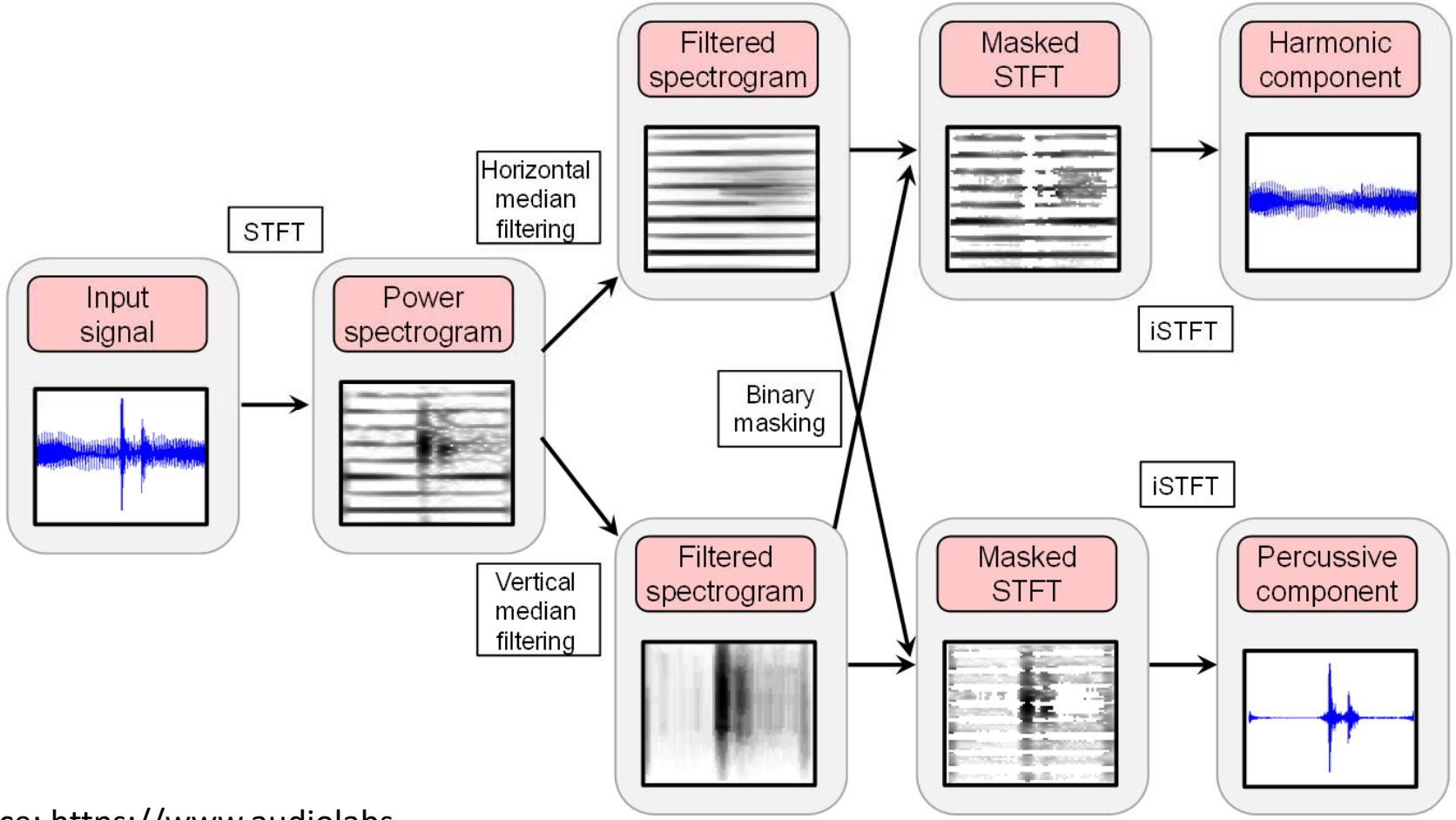
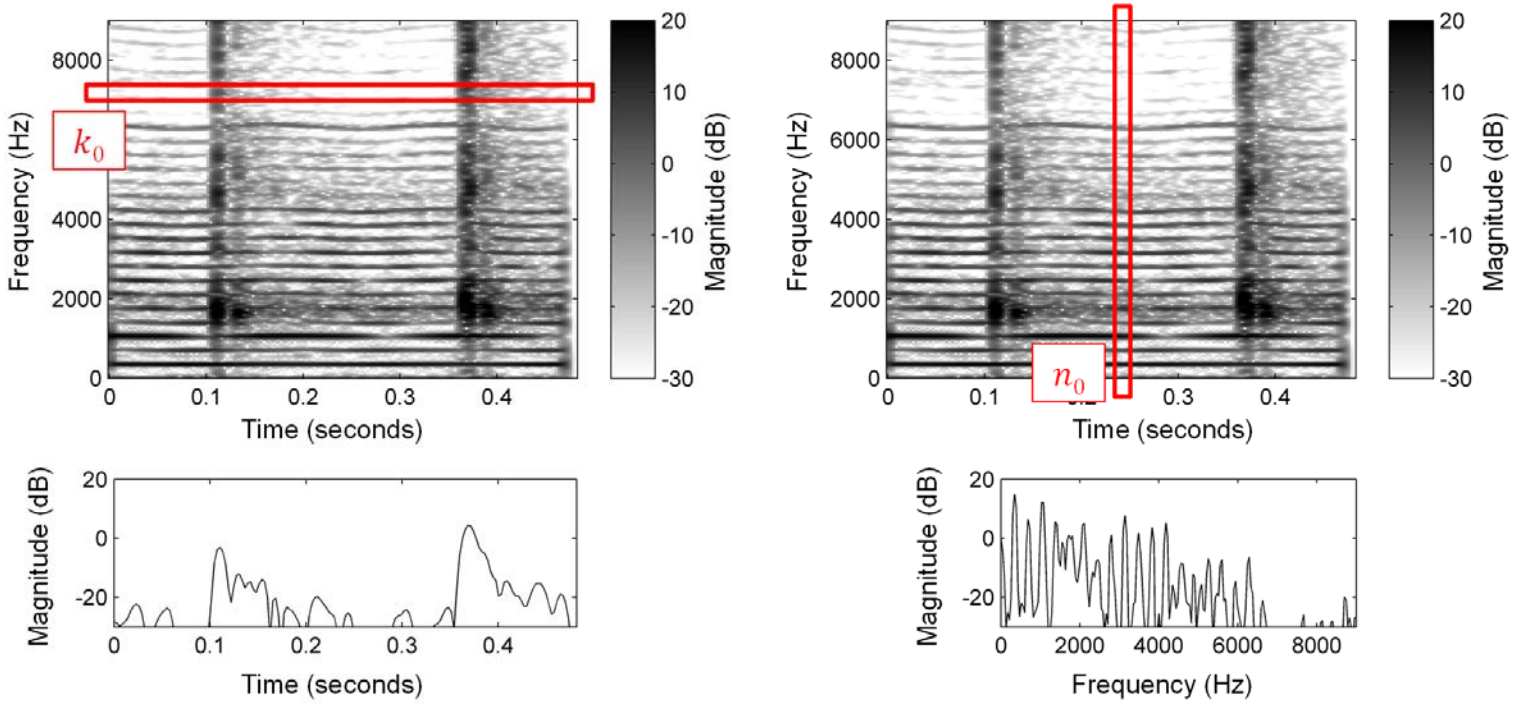


Figure source: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S1_HPS.html

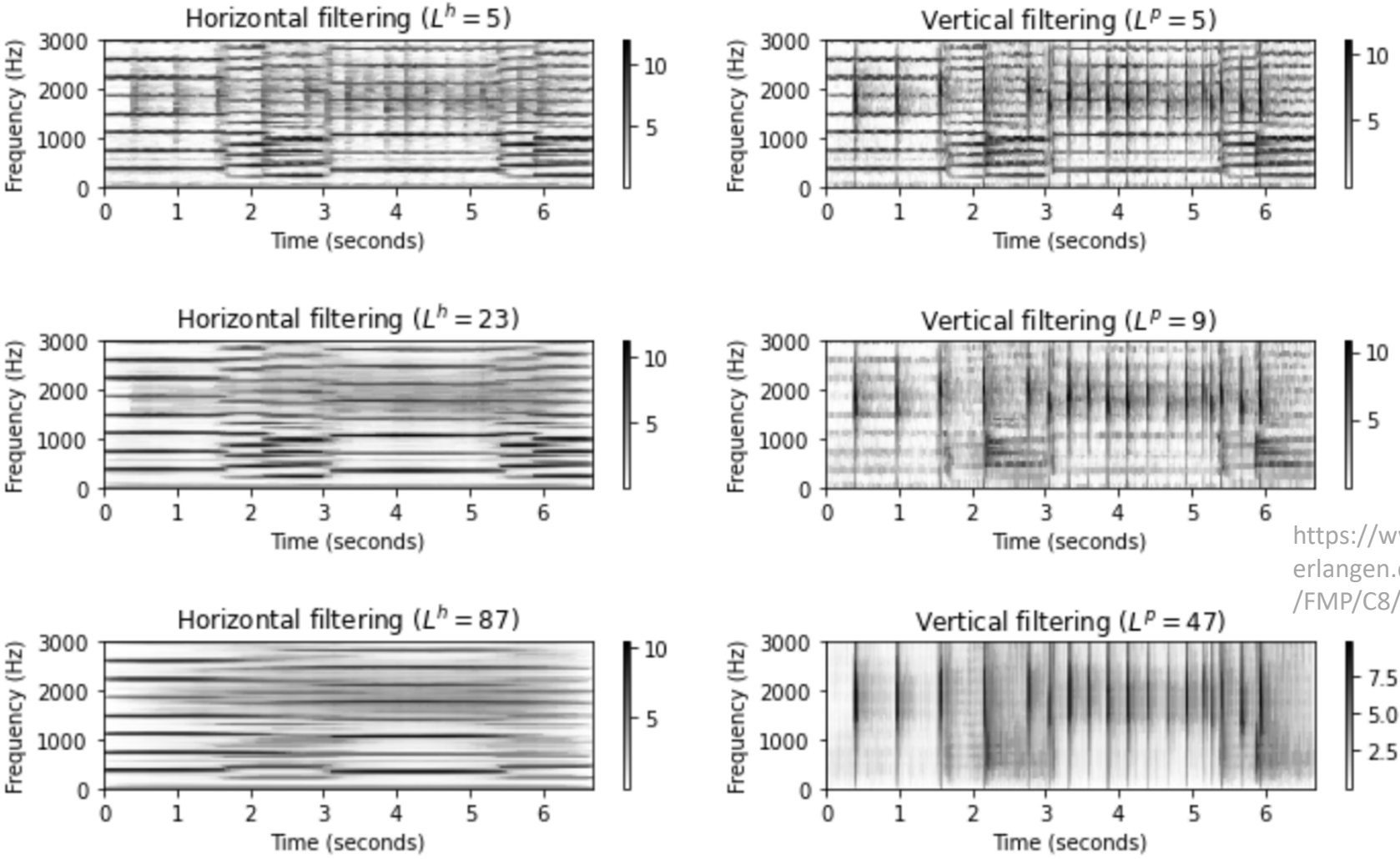
Median Filtering



$$\tilde{Y}_h(t,k) := \text{median}(Y(t-\ell_h,k),\ldots,Y(t+\ell_h,k))$$
$$\tilde{Y}_p(t,k) := \text{median}(Y(t,k-\ell_p),\ldots,Y(t,k+\ell_p))$$

Figure from [Mueller, FPM, Chapter 8, Springer 2015]

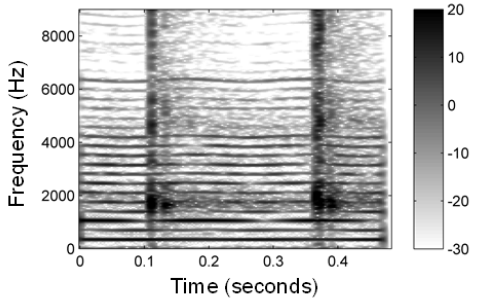
Effect of Filter Length



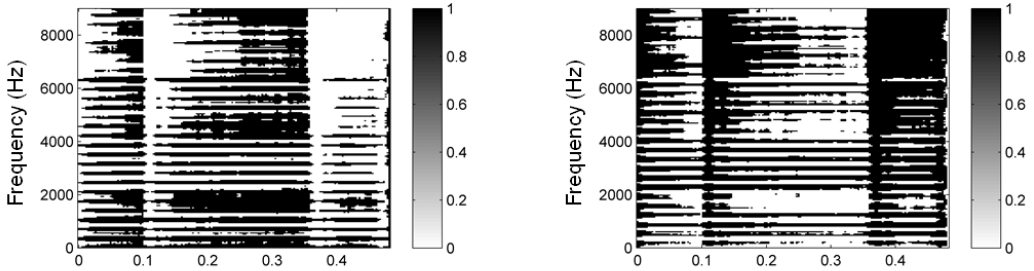
https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S1_HPS.html

Masking

violin + castanets



binary mask



soft mask

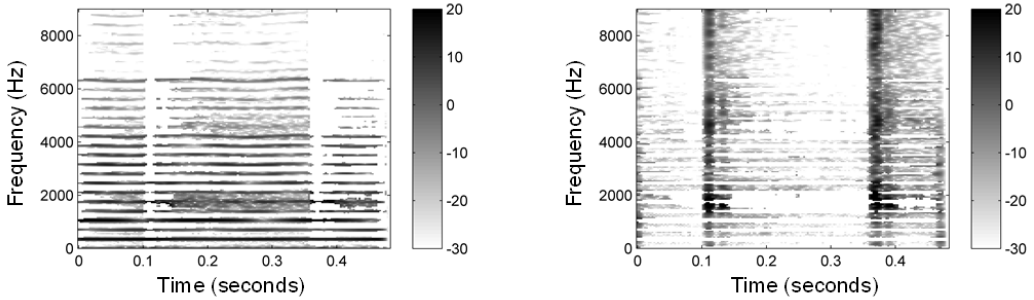


Figure from [Mueller, FPM, Chapter 8, Springer 2015]

Parameters

- Window size, hop size
- Filter length
- Masking method

Q1: Given sampling rate = 44.1 kHz, FFT window size = 4096 samples, hopsize = 1024 samples, what's the physical meaning of using a vertical (percussive) median filter length=17?

Q2: What's the physical meaning of using a horizontal (harmonic) median filter length=17?

Wait … How about Phase?

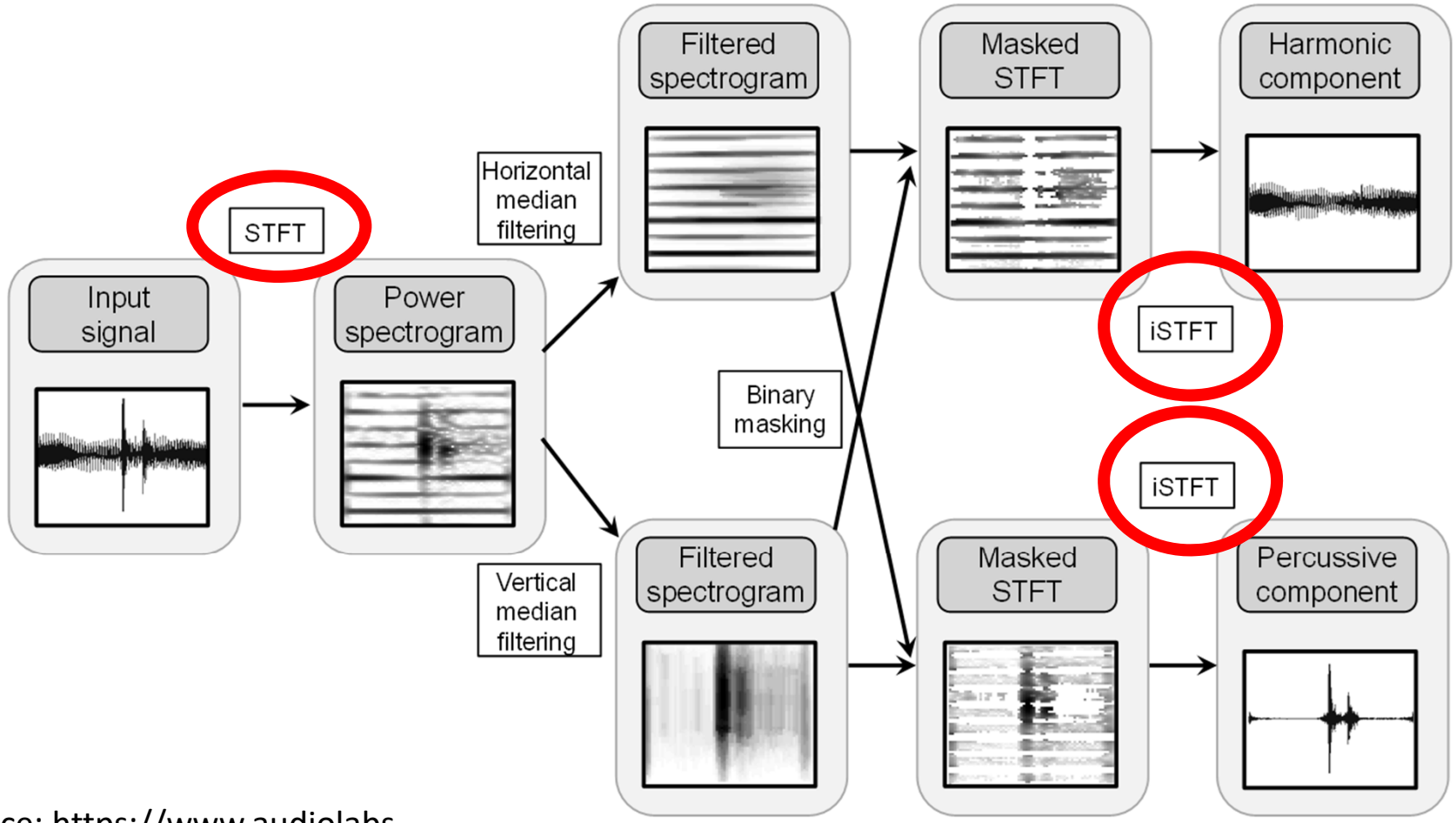


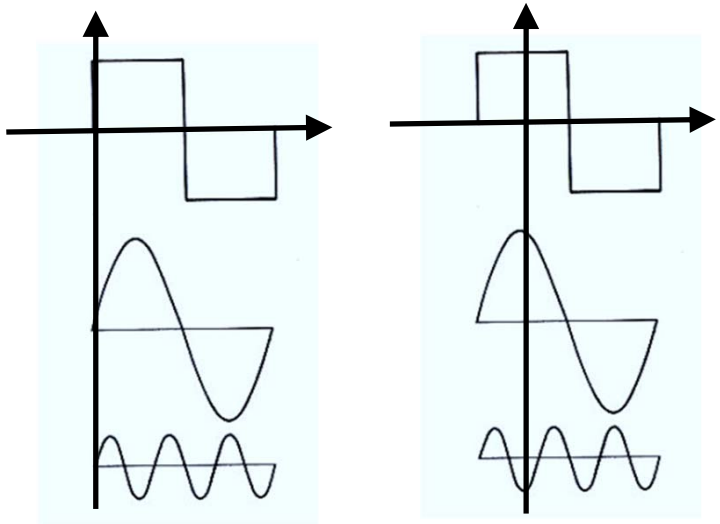
Figure source: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S1_HPS.html

Why Phase?

- STFT is the **inner product** between an input signal and a basis function carrying a specific frequency
- Allowing "time-shift" for computing the inner product
- "**Time-shift**" in the time domain → "**Phase-shift**" in the frequency domain

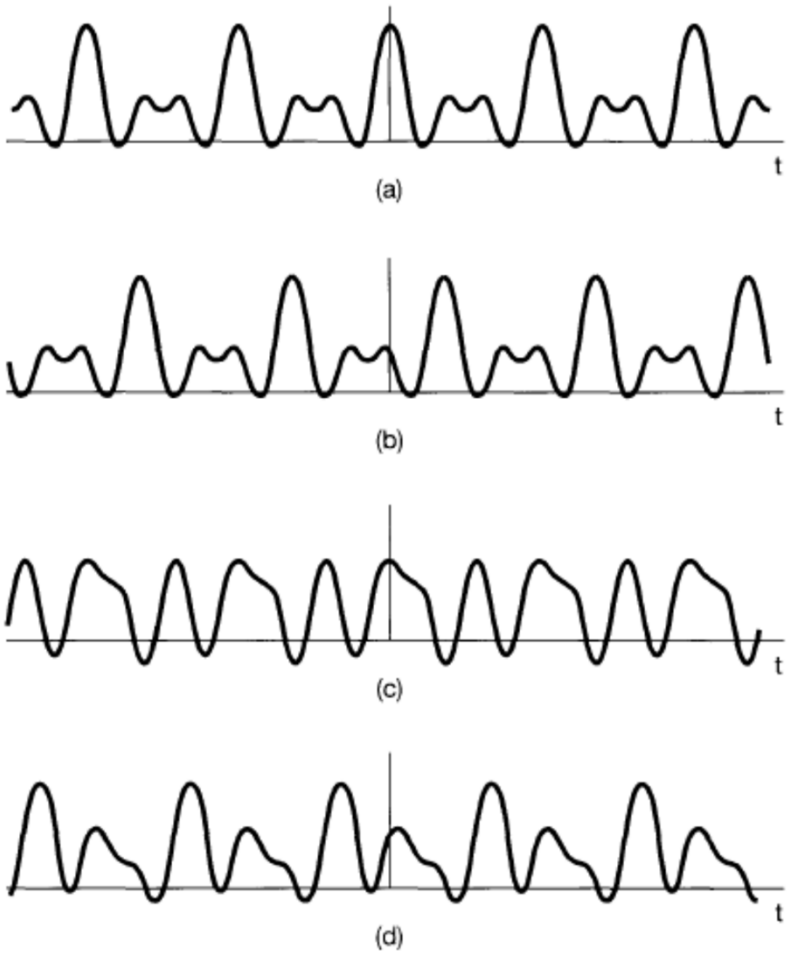
$$x[n] \overset{\mathcal{FS}}{\leftrightarrow} a_k$$

$$x[n - n_0] \overset{\mathcal{FS}}{\leftrightarrow} e^{-jk\omega_0 n_0} a_k$$



The amount of phase shift (in radians) is proportional to *k*

Why Phase?



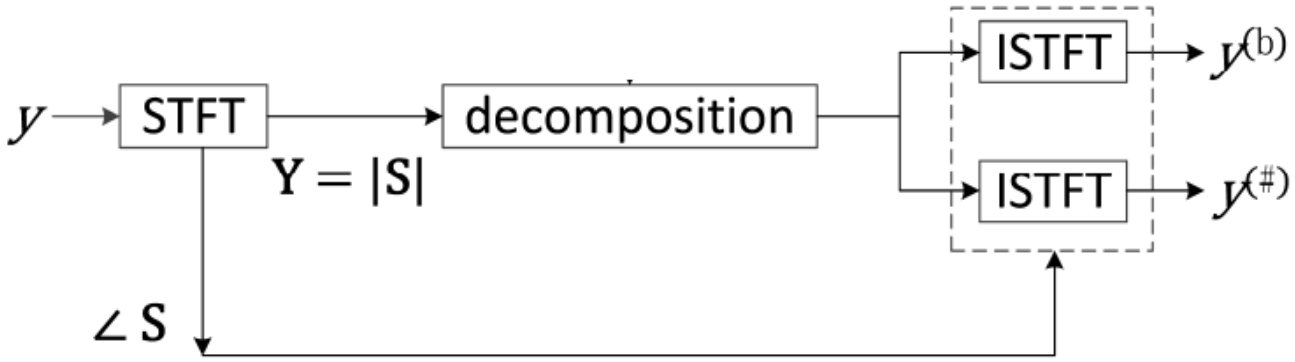
$$x(t) = 1 + \frac{1}{2}\cos(2\pi t + \phi_1) + \cos(4\pi t + \phi_2) + \frac{2}{3}\cos(6\pi t + \phi_3).$$

Figure 6.1 The signal $x(t)$ given in eq. (6.3) for several different choices of the phase angles ϕ_1, ϕ_2, and ϕ_3: (a) $\phi_1 = \phi_2 = \phi_3 = 0$; (b) $\phi_1 = 4$ rad, $\phi_2 = 8$ rad, $\phi_3 = 12$ rad; (c) $\phi_1 = 6$ rad, $\phi_2 = -2.7$ rad, $\phi_3 = 0.93$ rad; (d) $\phi_1 = 1.2$ rad, $\phi_2 = 4.1$ rad, $\phi_3 = -7.02$ rad.

Reconstruction: Use the Phase of the Mixture

(so that we only need to care about the magnitude)

1. Given a mixture *y*, compute the STFT ***Y***
2. Decompose the magnitude $|\boldsymbol{Y}|$ into two matrices **A** and **B** (which are also real-valued)
3. Make **A** and **B** complex-valued by adding the **phase of the mixture** $\measuredangle \boldsymbol{Y}$
4. Do inverse STFT (ISTFT)



Deal with Phase: Approaches

https://source-separation.github.io/tutorial/basics/phase.html

- **Copy the phase from the mixture**
  - Work fine for source separation (but maybe NOT for other audio generation problems)
- **Given the magnitude, estimate the phase** (see the lecture on "**vocoder**")
  - General-purpose solution → lecture 5
- **Work on audio waveforms, not spectrograms**
  - General-purpose solution

Implementation

https://librosa.org/doc/main/generated/librosa.effects.hpss.html

(previous page)
Q1: 183 Hz
Q2: 0.395 second

librosa.decompose.hpss

librosa.decompose.hpss(*S*, ***, *kernel_size=31*, *power=2.0*, *mask=False*, *margin=1.0*) [source]

Median-filtering harmonic percussive source separation (HPSS).

If `margin = 1.0`, decomposes an input spectrogram `S = H + P` where `H` contains the harmonic components, and `P` contains the percussive components.

If `margin > 1.0`, decomposes an input spectrogram `S = H + P + R` where `R` contains residual components not included in `H` or `P`.

This implementation is based upon the algorithm described by [1] and [2].

[1] Fitzgerald, Derry. "Harmonic/percussive separation using median filtering." 13th International Conference on Digital Audio Effects (DAFX10), Graz, Austria, 2010.

[2] Driedger, Müller, Disch. "Extending harmonic-percussive separation of audio." 15th International Society for Music Information Retrieval Conference (ISMIR 2014), Taipei, Taiwan, 2014.

Extension: Adding a Residual Component

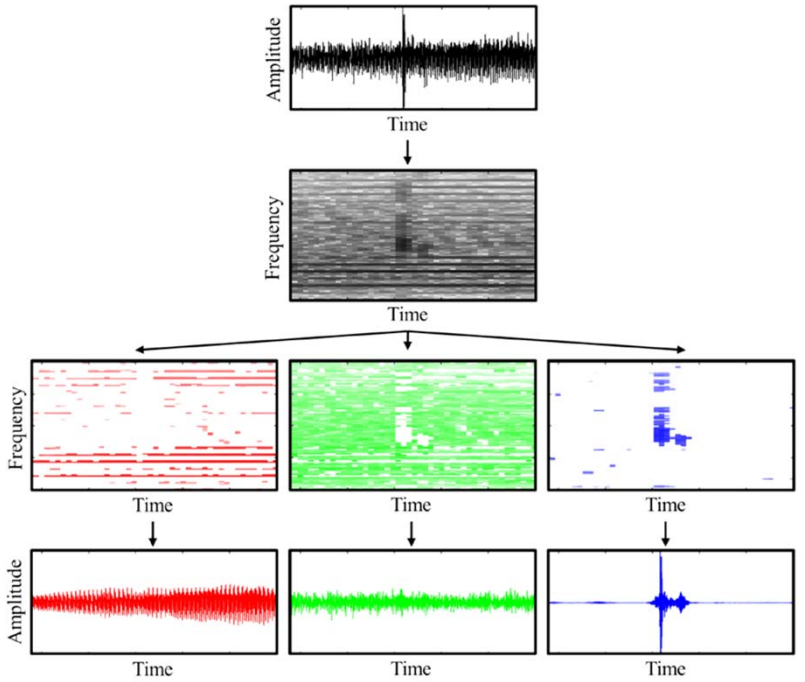
https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S1_HRPS.html

$$M_h(t,k) := \left(\tilde{Y}_h(t,k)/(\tilde{Y}_p(t,k)+\epsilon)\right) > \beta$$

$$M_p(t,k) := \left(\tilde{Y}_p(t,k)/(\tilde{Y}_h(t,k)+\epsilon)\right) \geq \beta$$

$$M_r(t,k) := 1 - \left(M_h(t,k) + M_p(t,k)\right)$$

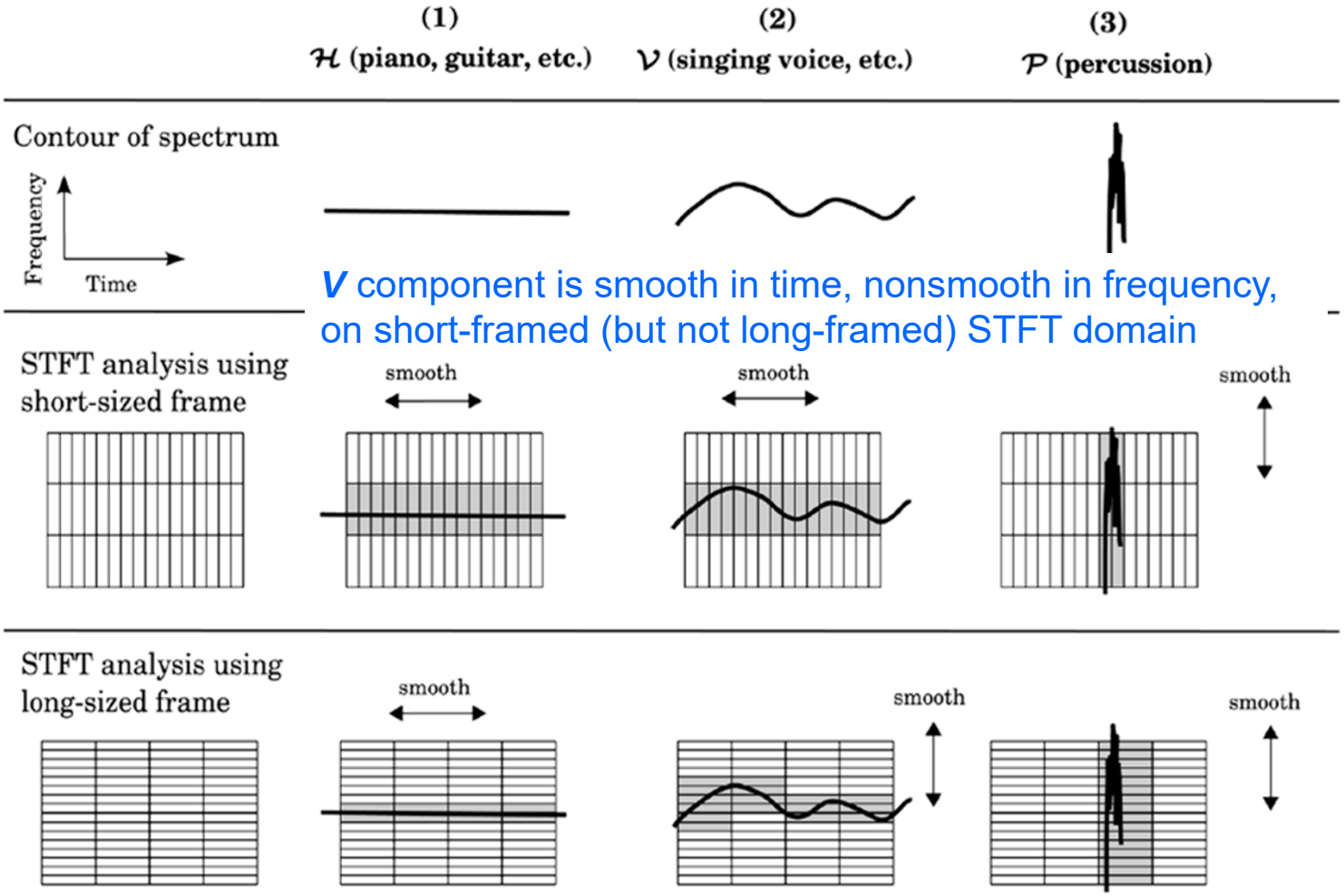
- The *harmonic* component contains the **violin**, the *percussive* component contains the **castanets**, and the *residual* contains the **applause**
- More demo: http://www.audiolabs-erlangen.de/resources/2014-ISMIR-ExtHPSep/



Harmonic Residual Percussive

Ref: Driedger et al, "Extending harmonic-percussive separation of audio signals," ISMIR 2014

Extension: Separating the Vocals

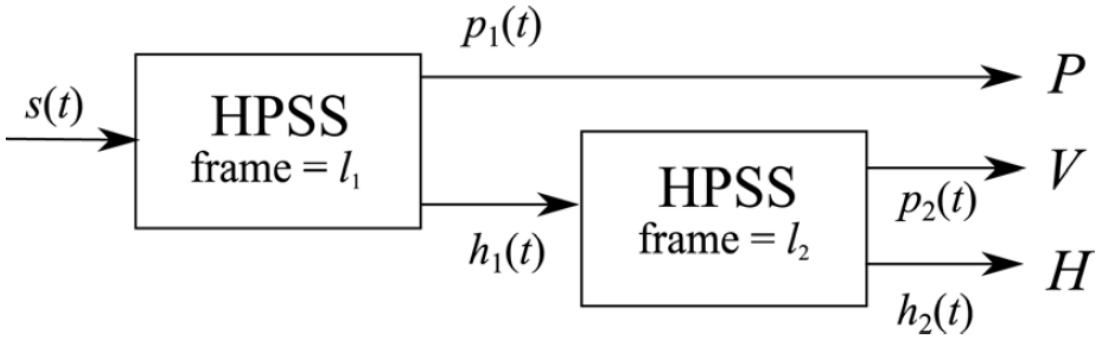


V component is smooth in time, nonsmooth in frequency, on short-framed (but not long-framed) STFT domain

Ref: Tachibana et al, "Singing voice enhancement in monaural music signals based on two-stage harmonic/percussive sound separation on multiple resolution spectrograms," TASLP 2014

Extension: Separating the Vocals

- Singing voice: intermediate component between ‘harmonic’ and ‘percussive’
- Perform the **two HPSS** on spectrograms with **two different** time-frequency **resolutions**



Ref: Tachibana et al, “Singing voice enhancement in monaural music signals based on two-stage harmonic/percussive sound separation on multiple resolution spectrograms,” TASLP 2014

Strong Assumption, OK but Limited Performance

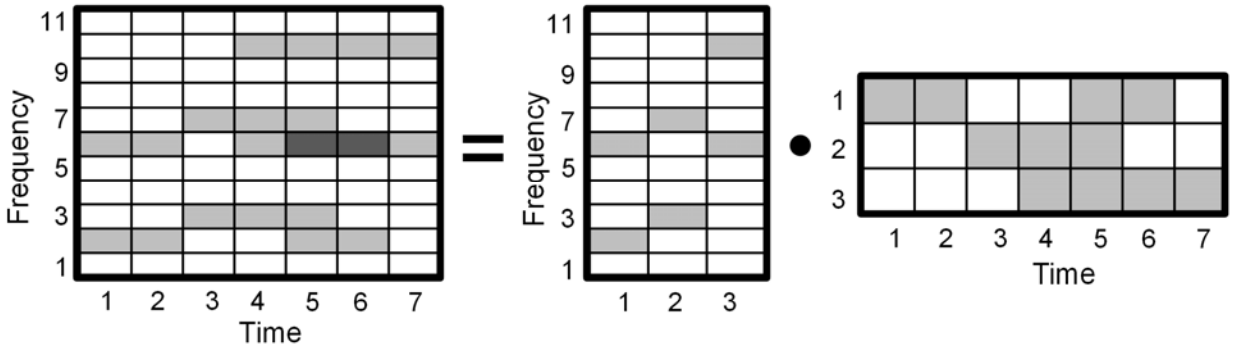
https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S1_HPS.html

Outline

- General idea
- **Traditional methods** (*non-DL*, *unsupervised*)
  - Median filtering
  - **Nonnegative matrix factorization (NMF)**
  - Robust principal component analysis (RPCA)
- Deep learning based methods

Nonnegative Matrix Factorization (NMF)

- Factorize a non-negative matrix (e.g., the **magnitude spectrogram**) into two
  - ***Templates***
  - ***Activations***



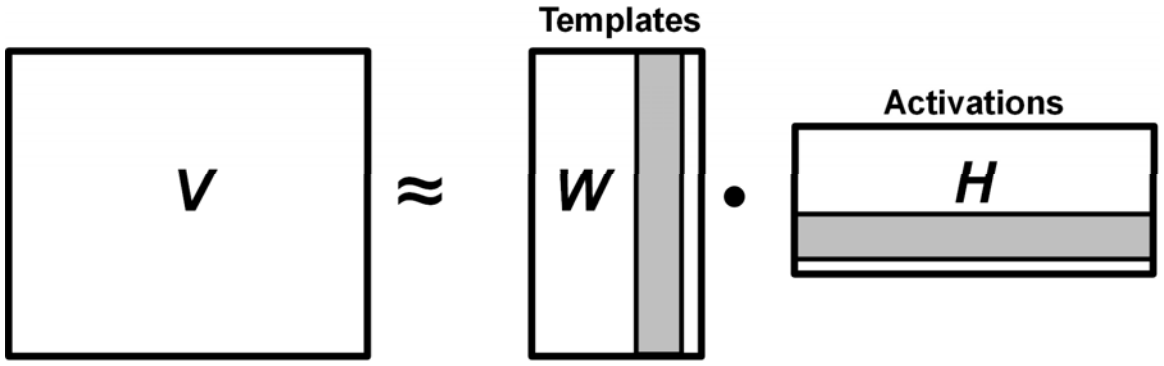


Figure from [Mueller, FPM, Chapter 8, Springer 2015]

NMF: Basic Idea

Given a *nonnegative* matrix $\mathbf{V}$ of dimensions $F \times N$, NMF is the problem of finding a factorization

$$\mathbf{V} \approx \mathbf{WH}$$

where $\mathbf{W}$ and $\mathbf{H}$ are *nonnegative* matrices of dimensions $F \times K$ and $K \times N$, respectively.

K is usually chosen such that $F\,K + K\,N << F\,N$, hence reducing the data dimension, but not always.

From Cédric Févotte's slides

NMF: Basic Idea

Along VQ, PCA or ICA, NMF provides an **unsupervised linear representation** of data

$\mathbf{v}_n$	$\approx$	$\mathbf{W}$	$\mathbf{h}_n$
data vector		"explanatory variables"	"regressors"
		"basis", "dictionary"	"expansion coefficients"
		"patterns"	"activation coefficients"

and $\mathbf{W}$ is learnt from the set of data vectors $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_N]$.

- **nonneg. of W** ensures *interpretability* of the dictionary (features $\mathbf{w}_k$ and data $\mathbf{v}_n$ belong to same space).
- **nonneg. of H** tends to produce *part-based* representations because subtractive combinations are forbidden.

From Cédric Févotte's slides

NMF: Algorithm

- **Cost function**: Euclidean distance

$$\|V - WH\|^2 = \sum_{ij} (V_{ij} - WH_{ij})^2$$

- Iterative update
  - Fix *W*, update *H*
  - Fix *H*, update *W*

$$H_{a\mu} \leftarrow H_{a\mu} + \eta_{a\mu} \left[(W^T V)_{a\mu} - (W^T W H)_{a\mu} \right].$$

- **Multiplicative** update
  - easily preserver nonnegativity
  - easy to implement
  - fast (O(FKN) per iteration)
  - zeros remain zeros!

$$\eta_{a\mu} = \frac{H_{a\mu}}{(W^T W H)_{a\mu}},$$

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T V)_{a\mu}}{(W^T W H)_{a\mu}}$$

Ref 1: Lee & Seung, “Algorithms for non-negative matrix factorization,” NIPS 2001
Ref 2: Lin, “On the convergence of multiplicative update algorithms for non-negative matrix factorization,” TNN 2007

NMF-based Music Transcription

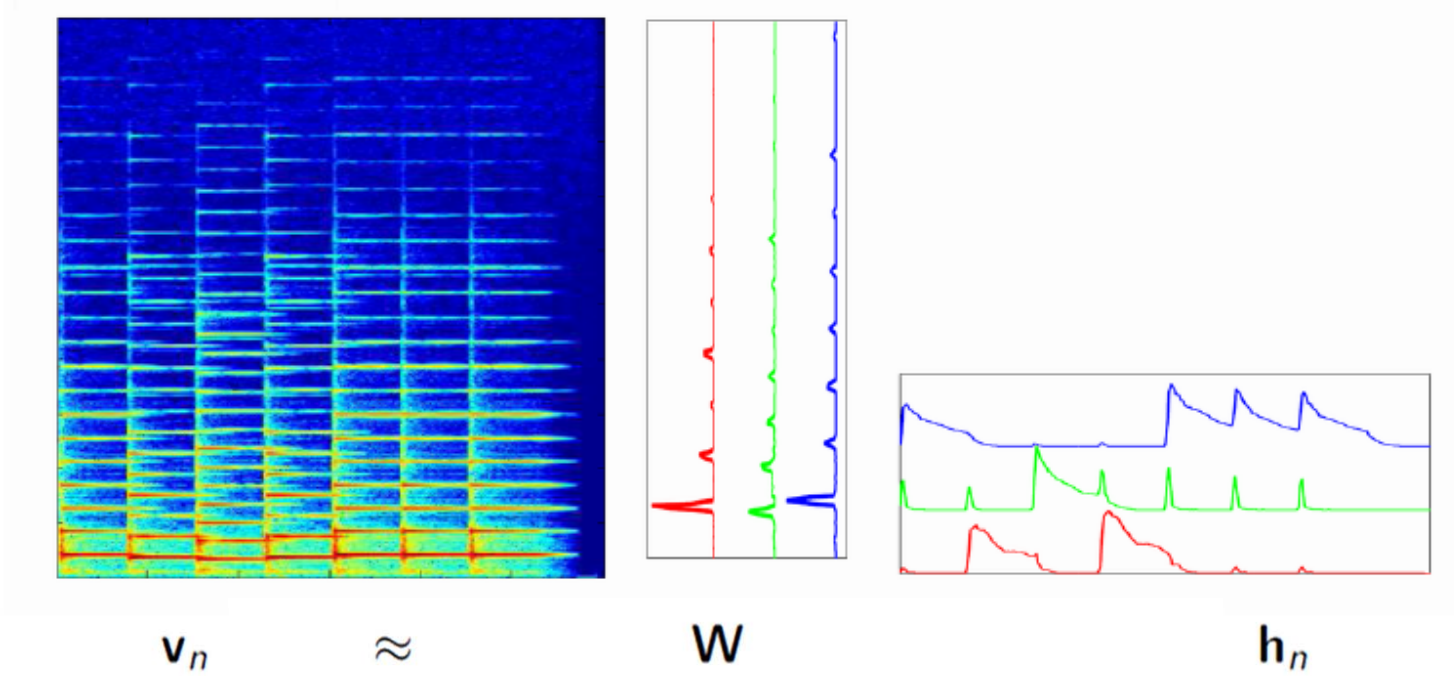
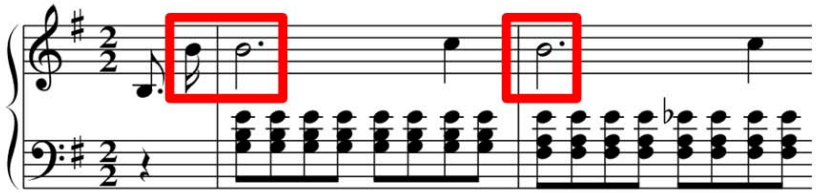


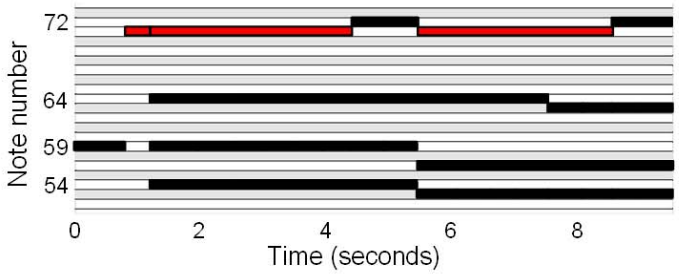
Figure from Byran & Sun's slides

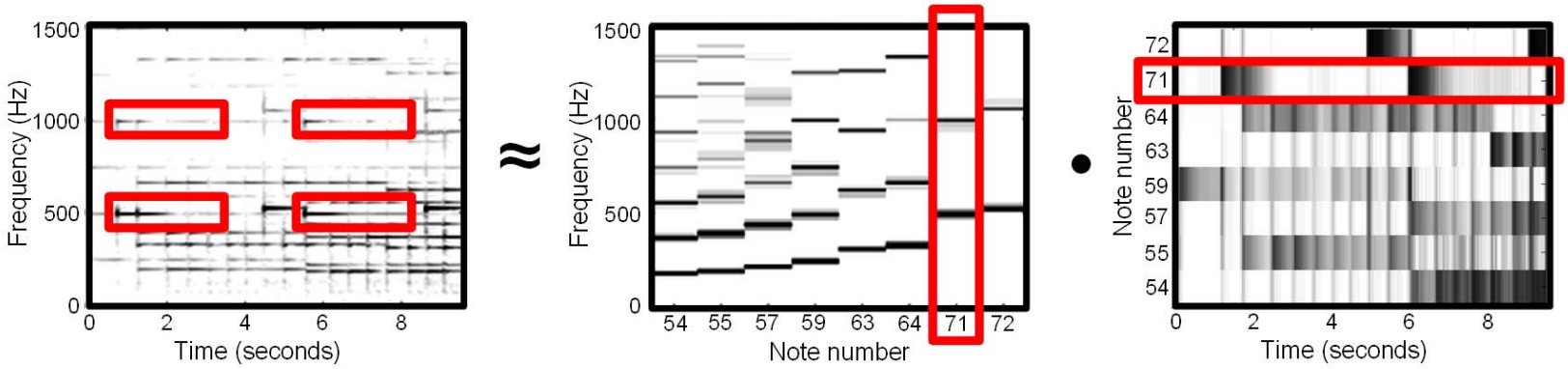
Figure from Byran & Sun's slides

NMF-based Music Transcription

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S3_NMFbasic.html



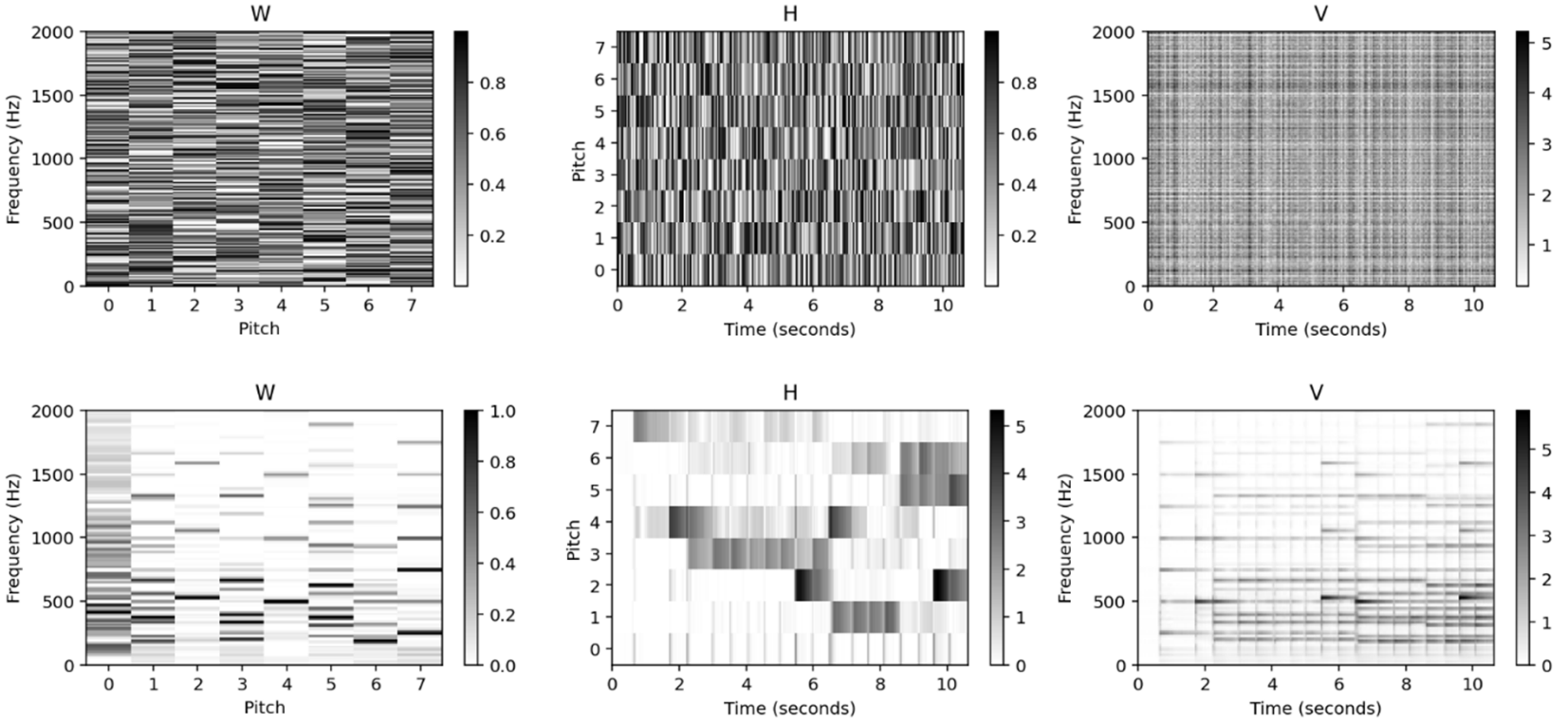




NMF with Random Initialization

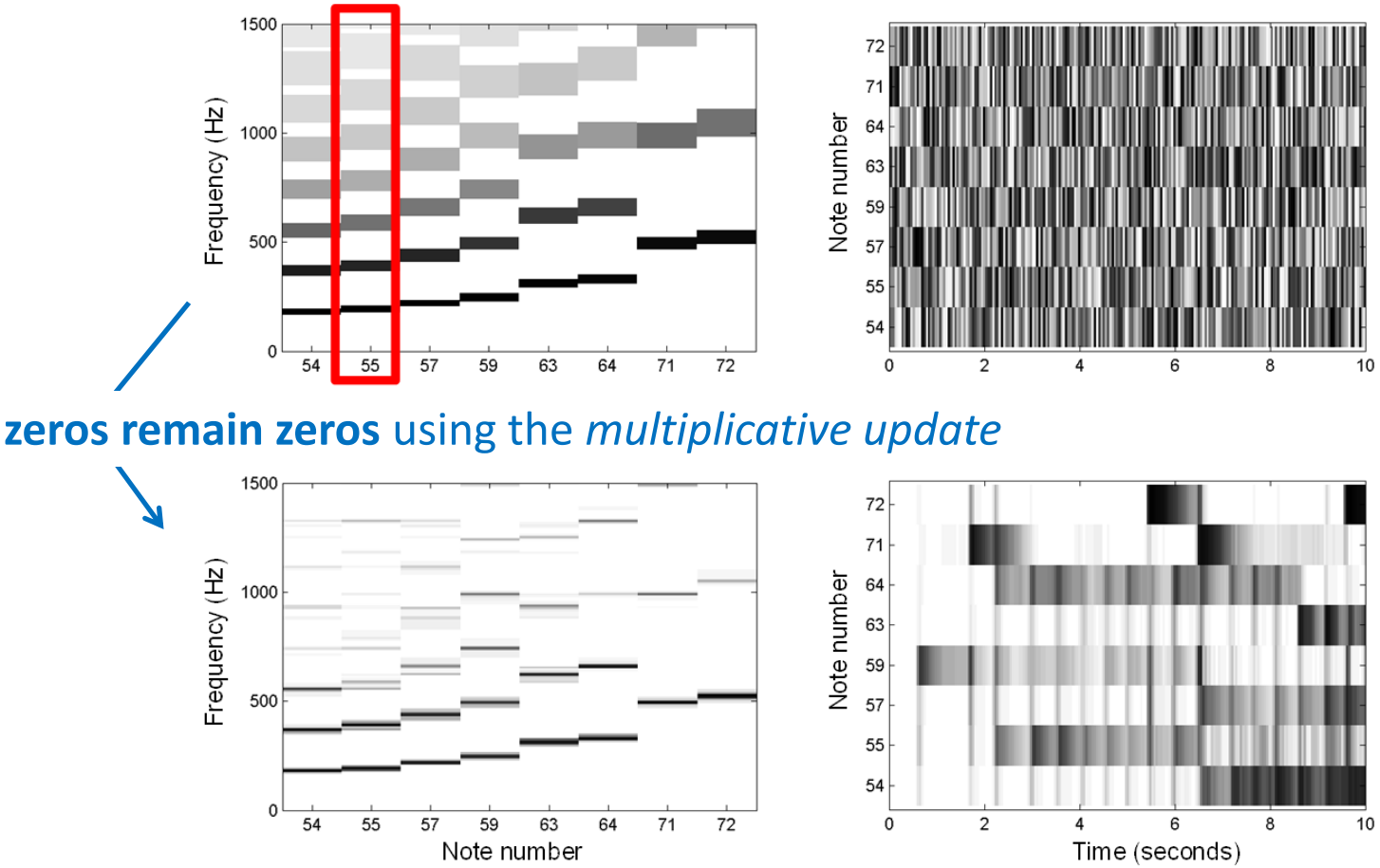
https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S3_NMFbasic.html

- OK reconstruction, but lacks clear musical semantics



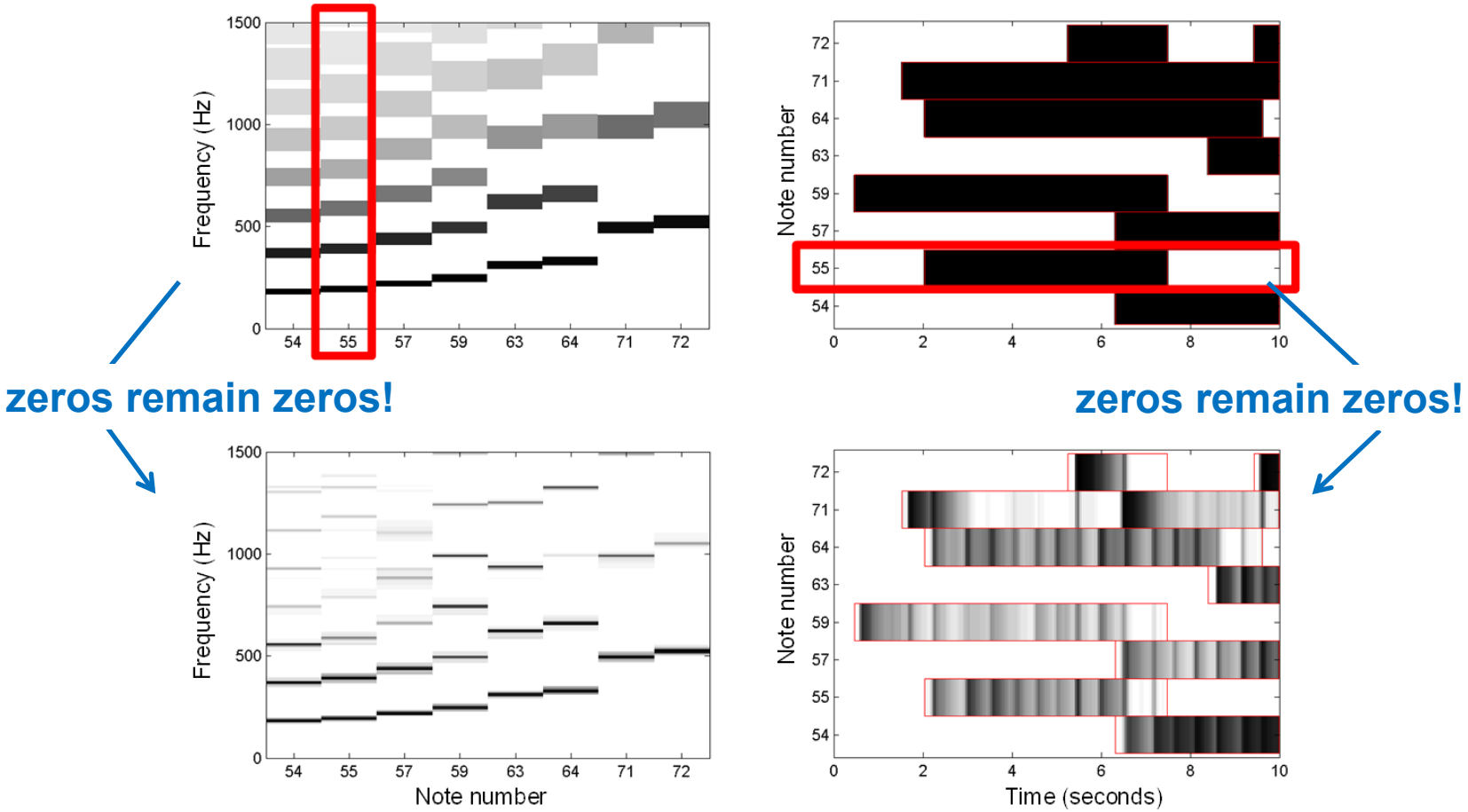
NMF with Harmonic Template Initialization (on W)

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S3_NMFbasic.html



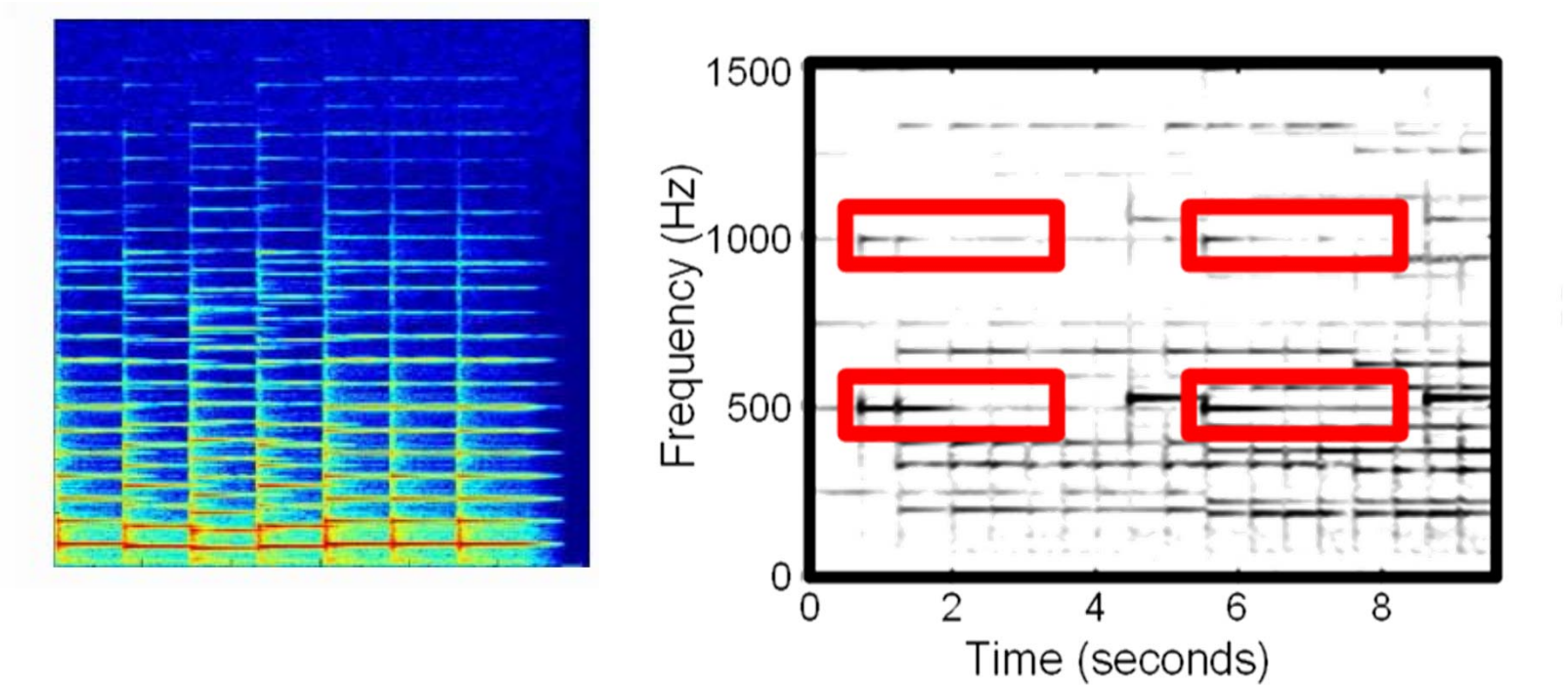
NMF with Score-Informed Initialization (on W and H)

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S3_NMFbasic.html



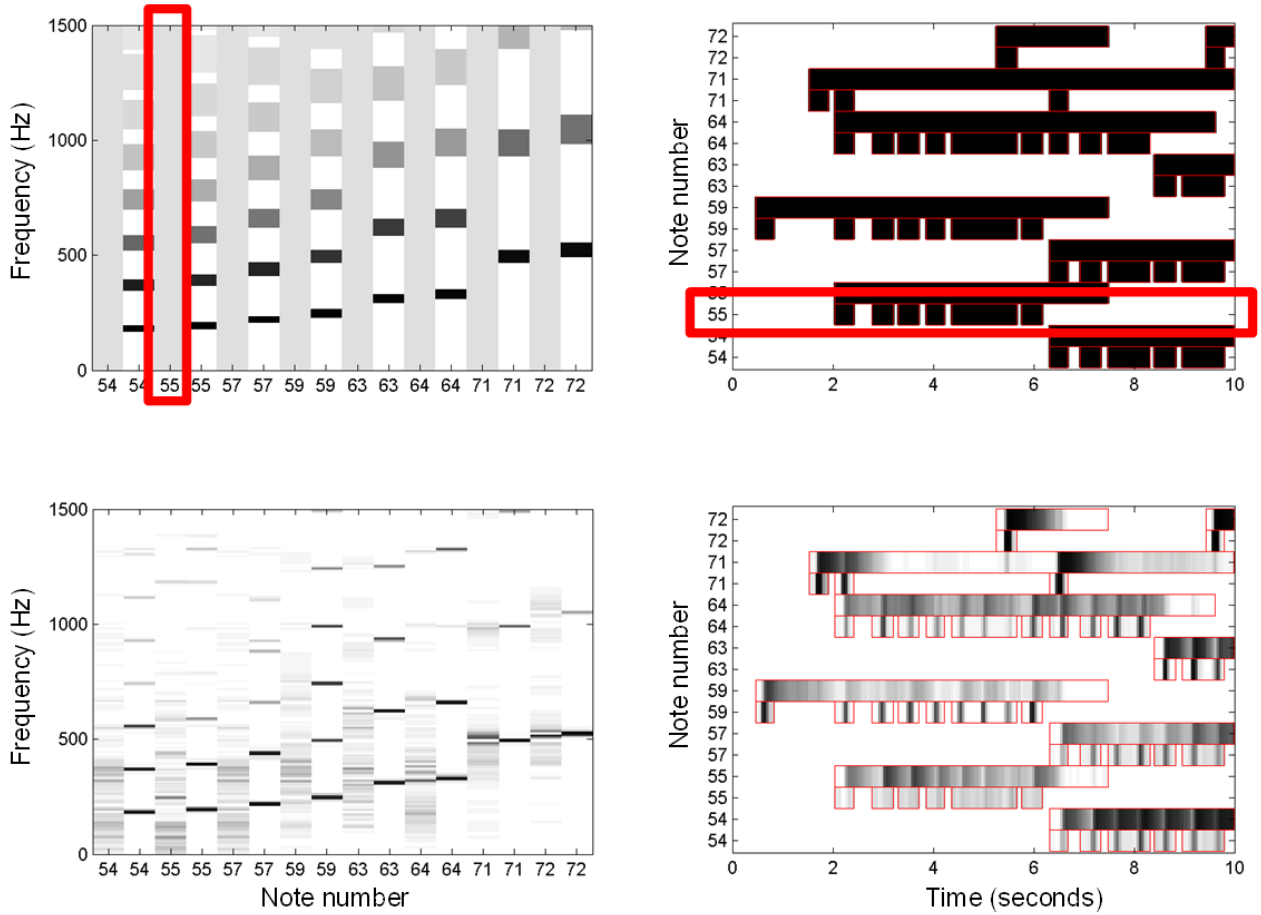
Dealing with Transients

- **Transient**: a high amplitude, short-duration sound at the beginning of a waveform that occurs in phenomena such as musical sounds



NMF with Score-Informed Initialization + Onset

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C8/C8S3_NMFbasic.html



NMF-based Source Separation

- **NMF-based factorization**
- **Template clustering/classification**
  - For example, by reusing a vocal/non-vocal classifier (on the templates) with MFCC, LFPC, and PLP coefficients as features

- $\min_{H} \| V_{\text{mix}} - [W_A, W_B]H \|_F$
  - $\widehat{V_{\text{A}}} = W_A H_A$
  - $\widehat{V_{\text{B}}} = W_B H_B$

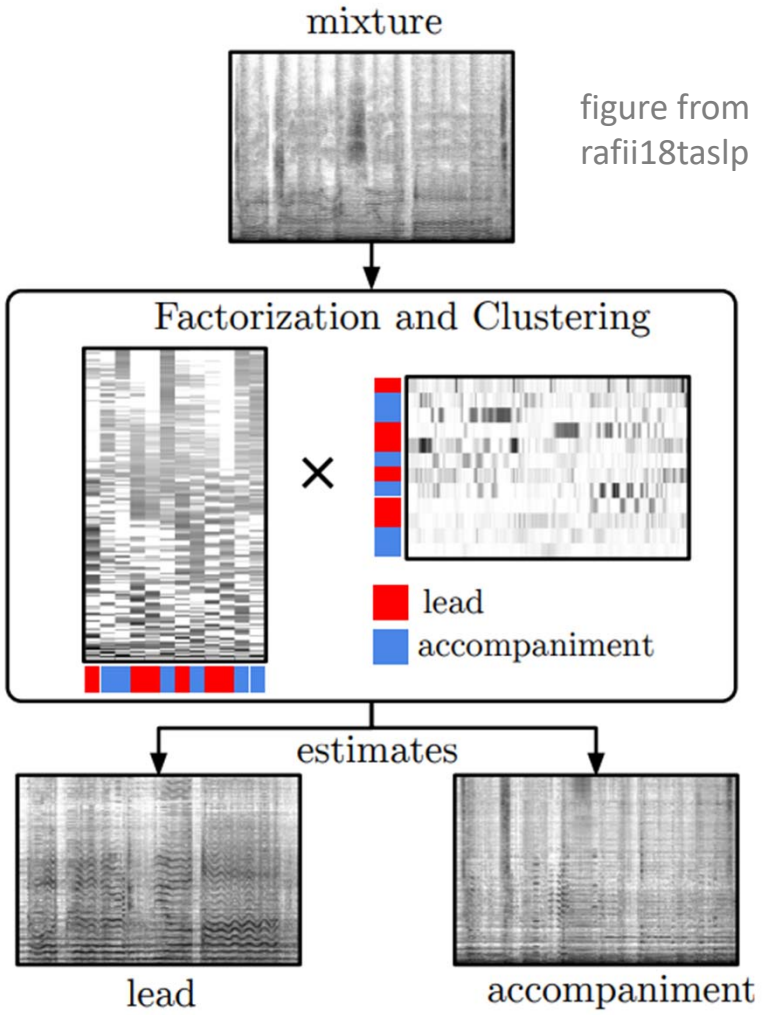


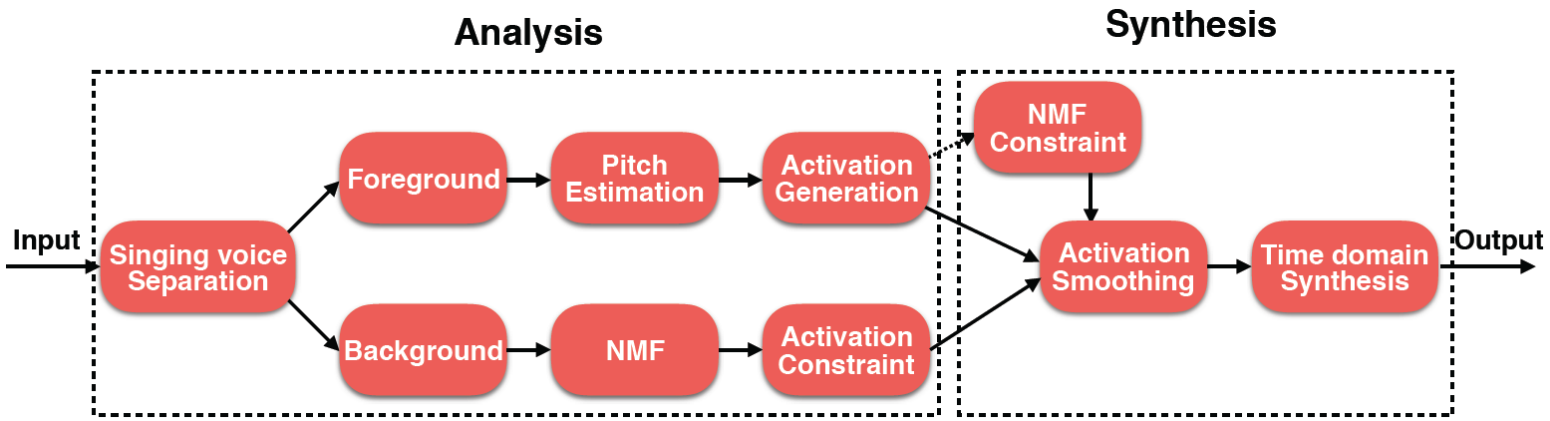
figure from rafii18taslp

Ref 1: Vembu & Baumann, "Separation of vocals from polyphonic audio recordings," ISMIR 2005
Ref 2: Rafii et al, "An overview of lead and accompaniment separation in music," TASLP 2018

Application of NMF: Pop→8bit Style Transfer

https://lemonatsu.github.io/py8bit_web/

8bit music, also known as “chip-tunes”



Adele - someone like you

Pop **8**-bit

Christina Perri - Jar of Hearts

Pop **8**-bit

John Legend - All of Me

Pop **8**-bit

Ref: Su et al, “Automatic conversion of pop music into chiptunes for 8-bit pixel art,” ICASSP 2017

Outline

- General idea
- **Traditional methods** (*non-DL*, *unsupervised*)
  - Median filtering
  - Nonnegative matrix factorization (NMF)
  - **Robust principal component analysis (RPCA)**
- Deep learning based methods

RPCA-based Source Separation

- Decompose a matrix into a **sparse** matrix (*assumed* to correspond to *vocal*) and a **low-rank** matrix (*instrumental background*) and

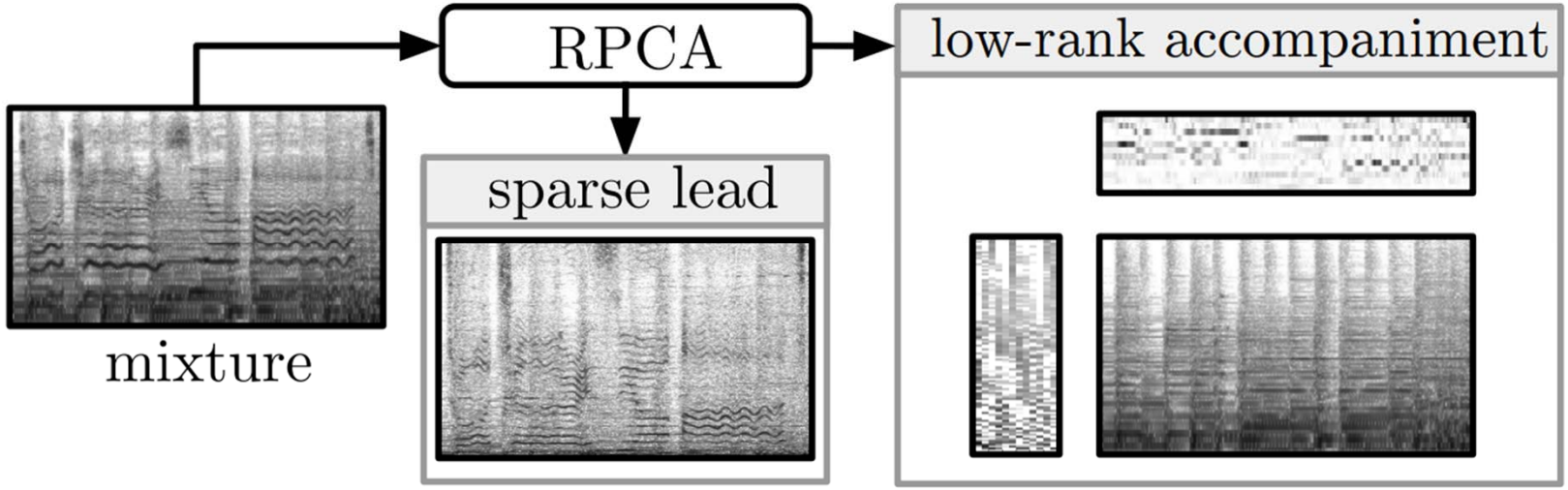


figure from rafii18taslp

Ref 1: Huang et al, "Singing-voice separation from monaural recordings using robust principal component analysis," ICASSP 2014
Ref 2: Rafii et al, "An overview of lead and accompaniment separation in music," TASLP 2018

Robust Principal Component Analysis (RPCA)

- Decomposes M into a **low-rank** matrix L plus a **sparse** matrix S

$$\min \ \|L\|_* + \lambda\|S\|_1$$

$$\text{s.t.} \quad L + S = M$$

- $\boldsymbol{l_1}$ **norm**: $\|S\|_1 = \sum_{ij} |S_{ij}|$ (sum of absolute values; a convex relaxation of l_0 norm)
- **Nuclear norm**: $\|L\|_* = \sum_i \sigma_i\ (L)$ (sum of singular values; a convex relaxation of the rank of the matrix)

Ref: Lin et al, "The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices," 2009

Algorithm: Augmented Lagrangian Approach

$$\begin{aligned} &\text{minimize} && \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\tau}\|M - L - S\|_F^2 \\ &\text{subject to} && L + S = M \end{aligned}$$

Lagrangian

$$\mathcal{L}(L, S; Y) = \|L\|_* + \lambda\|S\|_1 + \frac{1}{\tau}\langle Y, M - L - S\rangle + \frac{1}{2\tau}\|M - L - S\|_F^2$$

Easy to minimize over L and S separately

$$\arg\min_L \mathcal{L}(L, S, Y) = \mathcal{D}_\tau(M - S + Y)$$
$$\arg\min_S \mathcal{L}(L, S, Y) = \mathcal{S}_{\lambda\tau}(M - L + Y)$$

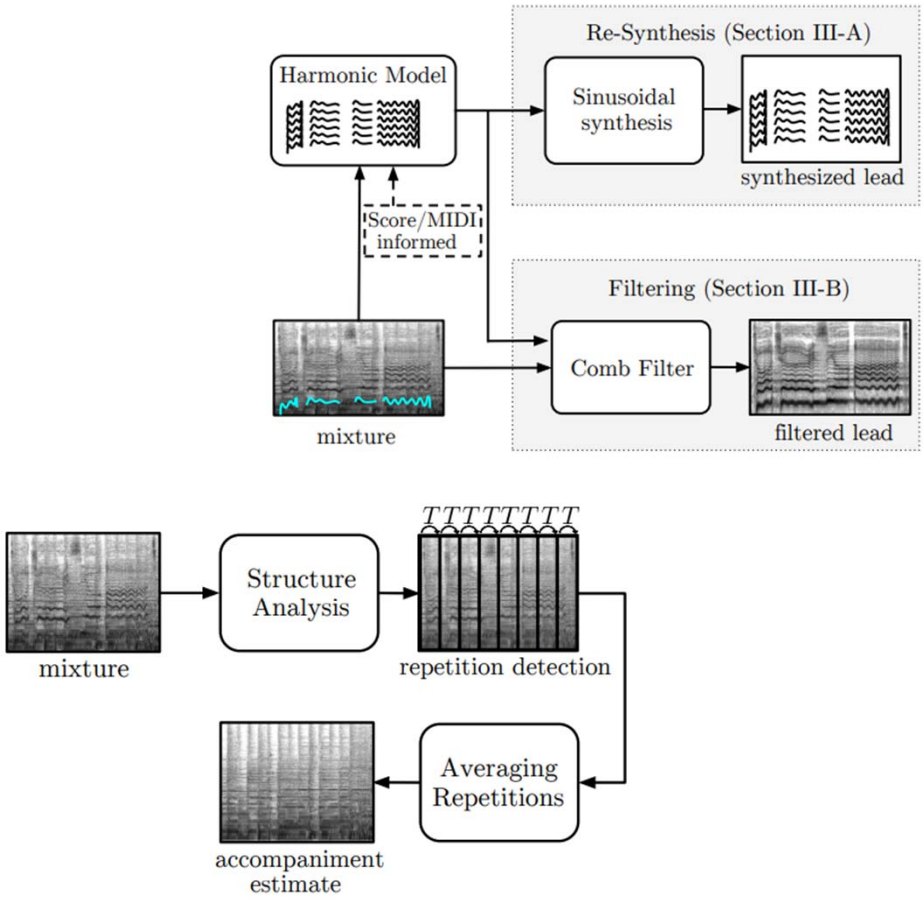
Scalar shrinkage: $\mathcal{S}_\tau[x] = \text{sgn}(x)\max(|x| - \tau, 0)$

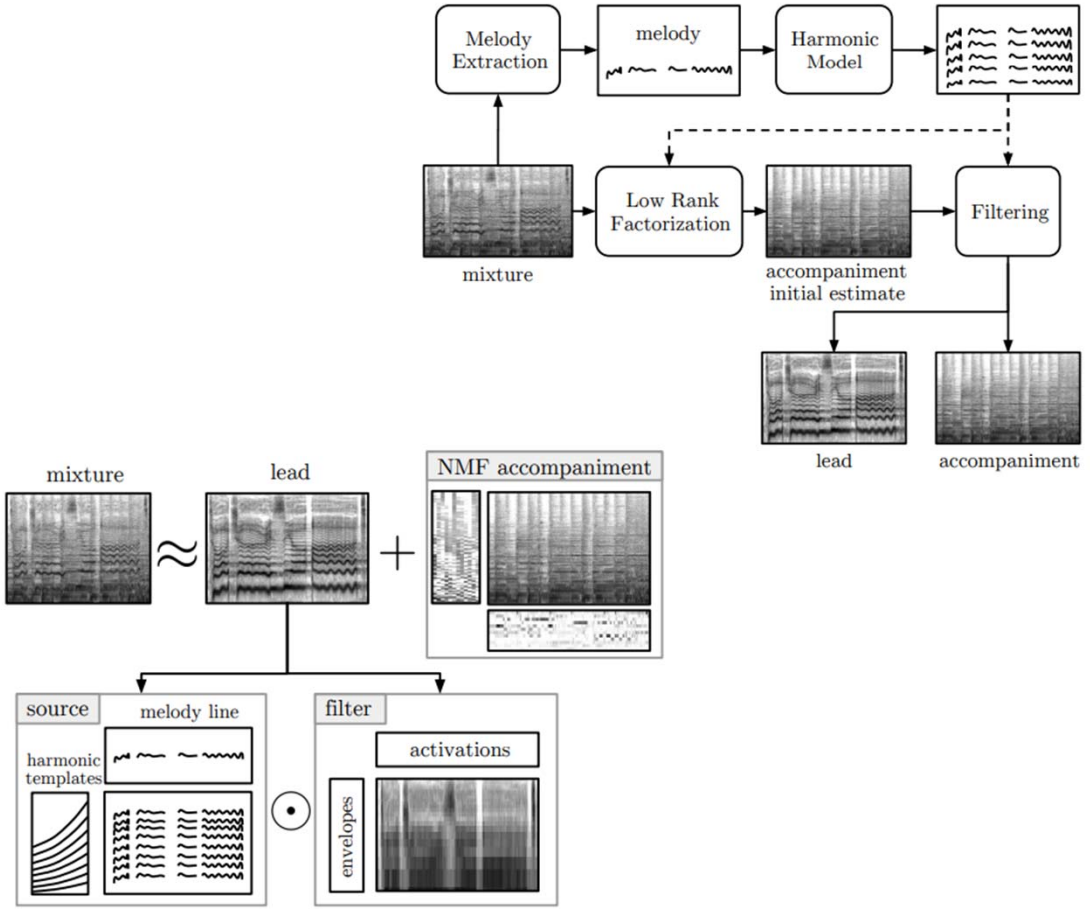
- Componentwise thresholding $\mathcal{S}_\tau(X)$
- Singular value thresholding $\mathcal{D}_\tau(X)$

$$\mathcal{D}_\tau(X) = U\mathcal{S}_\tau(\Sigma)V^* \qquad X = U\Sigma V^*$$

Ref: Lin et al, “The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices,” 2009

Many Other Unsupervised Methods





Ref: Rafii et al, "An overview of lead and accompaniment separation in music," TASLP 2018

Outline

- General idea
- Traditional methods
- **Deep learning based methods**

Supervised Approach

- Learn from **paired data** of {mixture, stems}

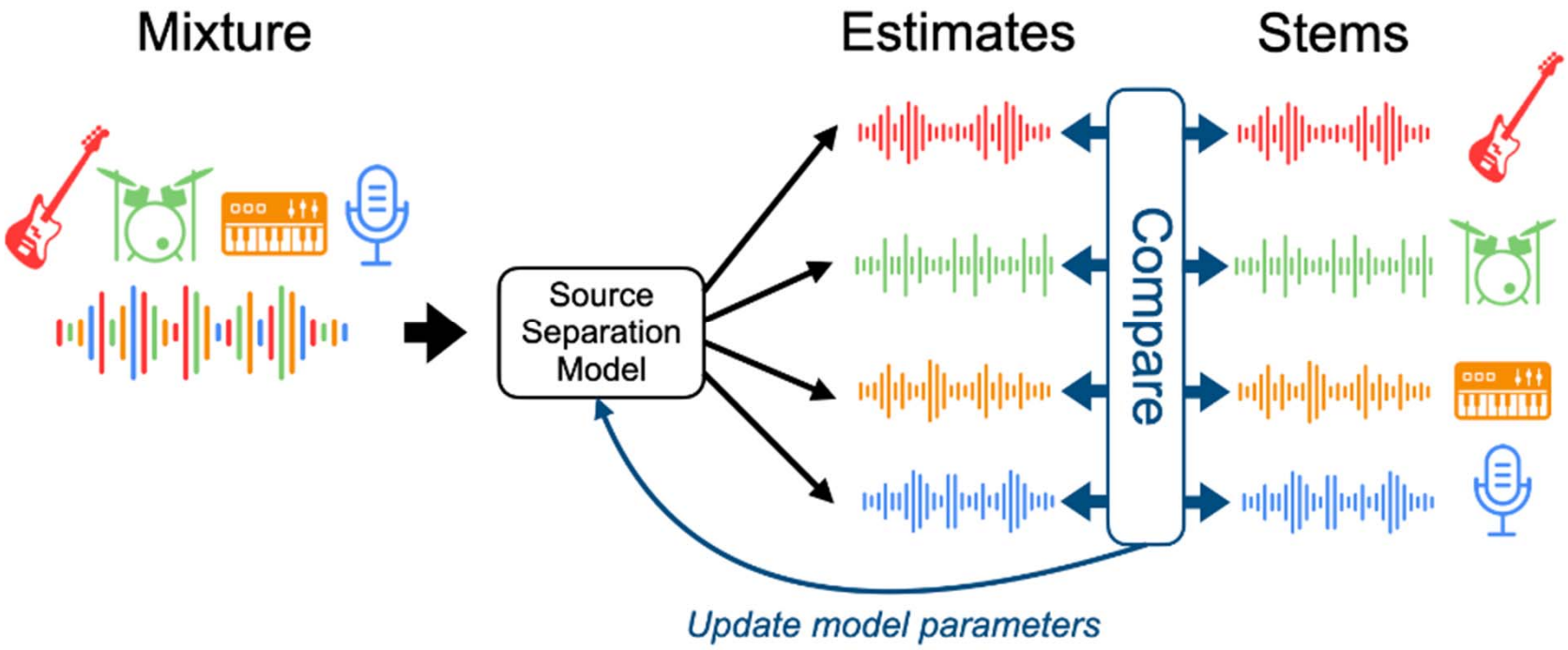


Figure source: https://source-separation.github.io/tutorial/approaches/deep/introduction.html

Why Not Using a Supervised Approach?

- **Challenge 1: Scarcity of paired data** in the early days
- **Challenge 2:** It involves **audio-domain music generation**, which is hard

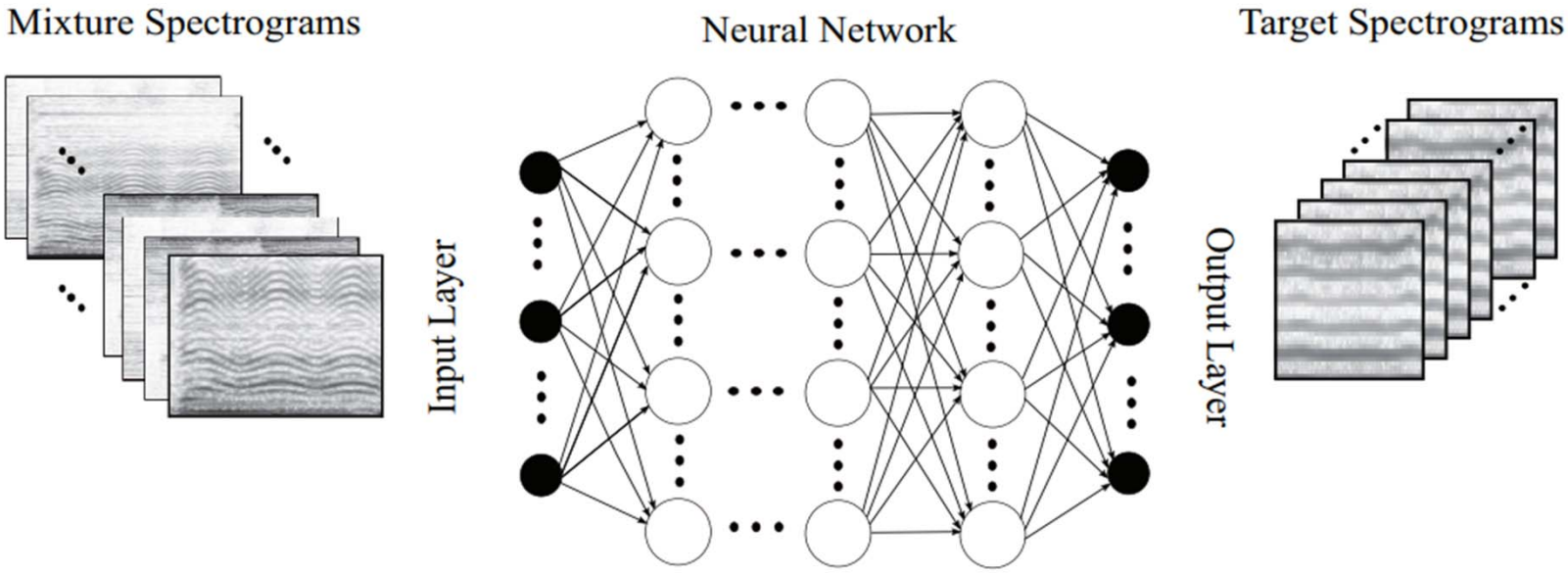


Figure source: Cano et al, "Musical source separation: An introduction," IEEE Signal Processing Magazine 2019

Solutions

- **Challenge 1: Scarce of paired data** in the early days

✓ Solution

  - Creation of **new datasets** for source separation
  - **Data augmentation** techniques

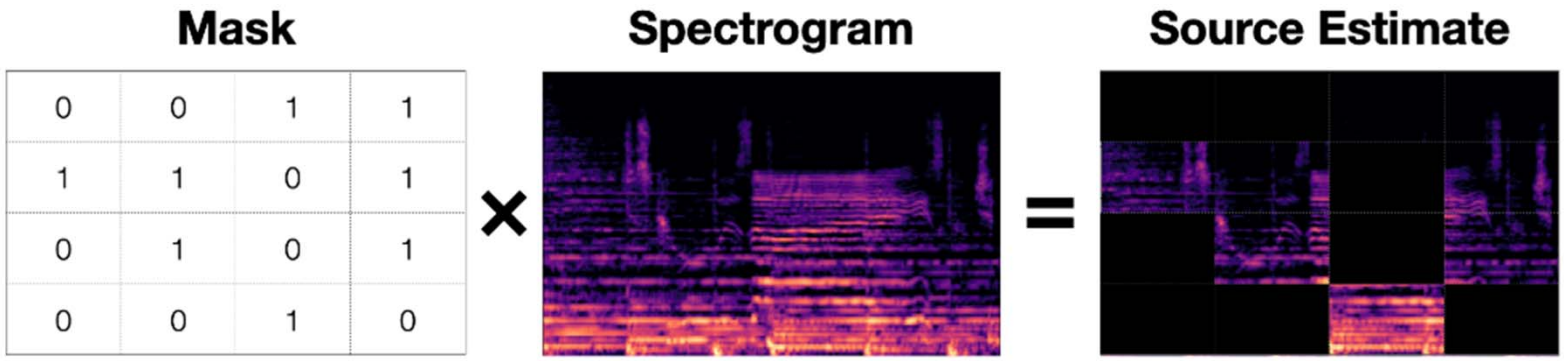
- **Challenge 2:** It involves **audio-domain music generation**, which is hard

✓ Solution

  - Predict "**masks**", rather than the actual stem
  - Or, **directly predict the stem anyway**

Breakthrough 1: Predicting the “Mask”

https://source-separation.github.io/tutorial/basics/tf_and_masking.html



- Bypass the difficulty of generating the audio directly
- Can be either **binary** mask or **soft** mask
- Can be considered as a *classification* problem
  - For example: whether a **time-frequency point** is vocal or not

RNN-based Approach

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:

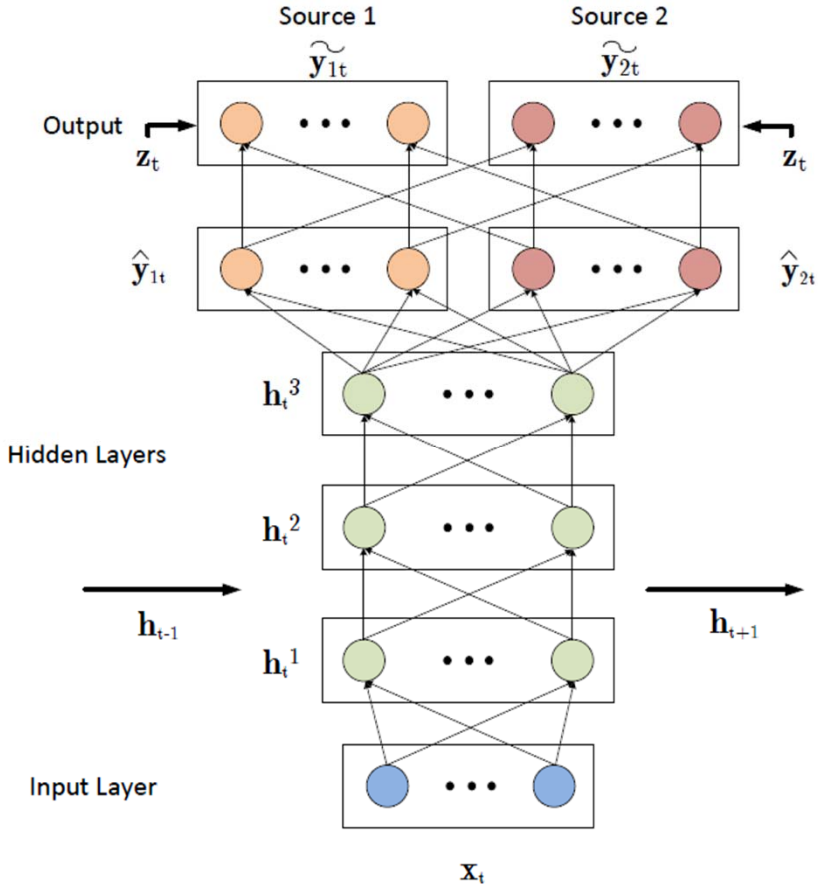
$$\mathbf{m}_t(f) = \frac{|\hat{\mathbf{y}}_{\mathbf{1}_t}(f)|}{|\hat{\mathbf{y}}_{\mathbf{1}_t}(f)| + |\hat{\mathbf{y}}_{\mathbf{2}_t}(f)|}, \tag{4}$$

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:

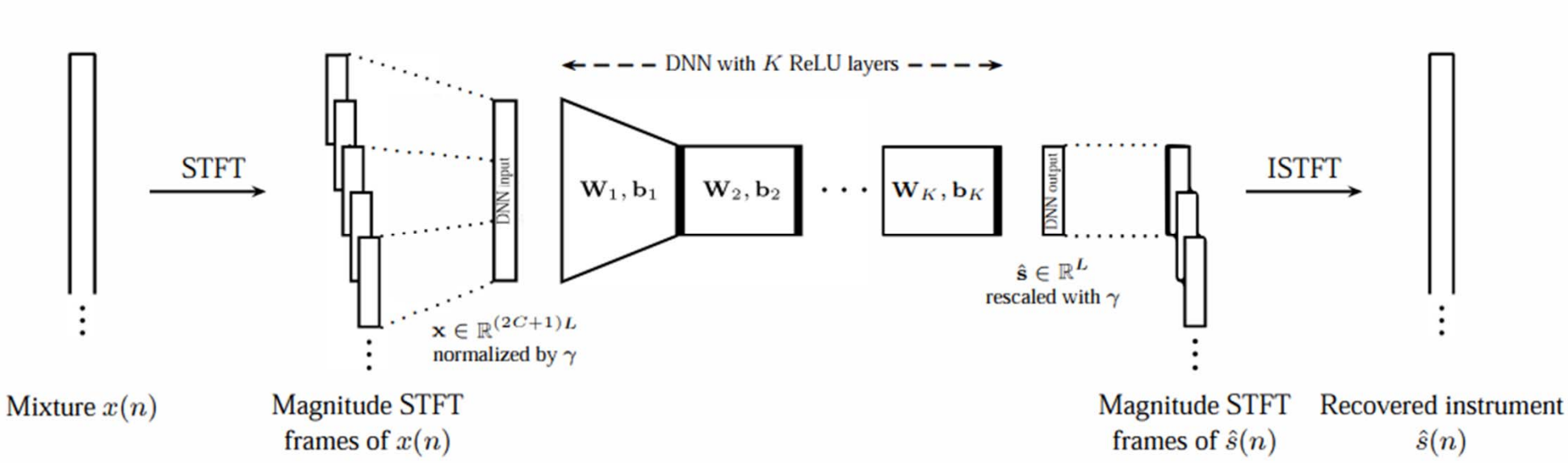
$$\begin{aligned} \hat{\mathbf{s}}_{\mathbf{1}_t}(f) &= \mathbf{m}_t(f)\mathbf{z}_t(f) \\ \hat{\mathbf{s}}_{\mathbf{2}_t}(f) &= (1 - \mathbf{m}_t(f))\,\mathbf{z}_t(f), \end{aligned} \tag{5}$$

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



Ref: Huang et al, “Singing-voice separation from monaural recordings using deep recurrent neural networks,” ISMIR 2014

DNN-based Approach



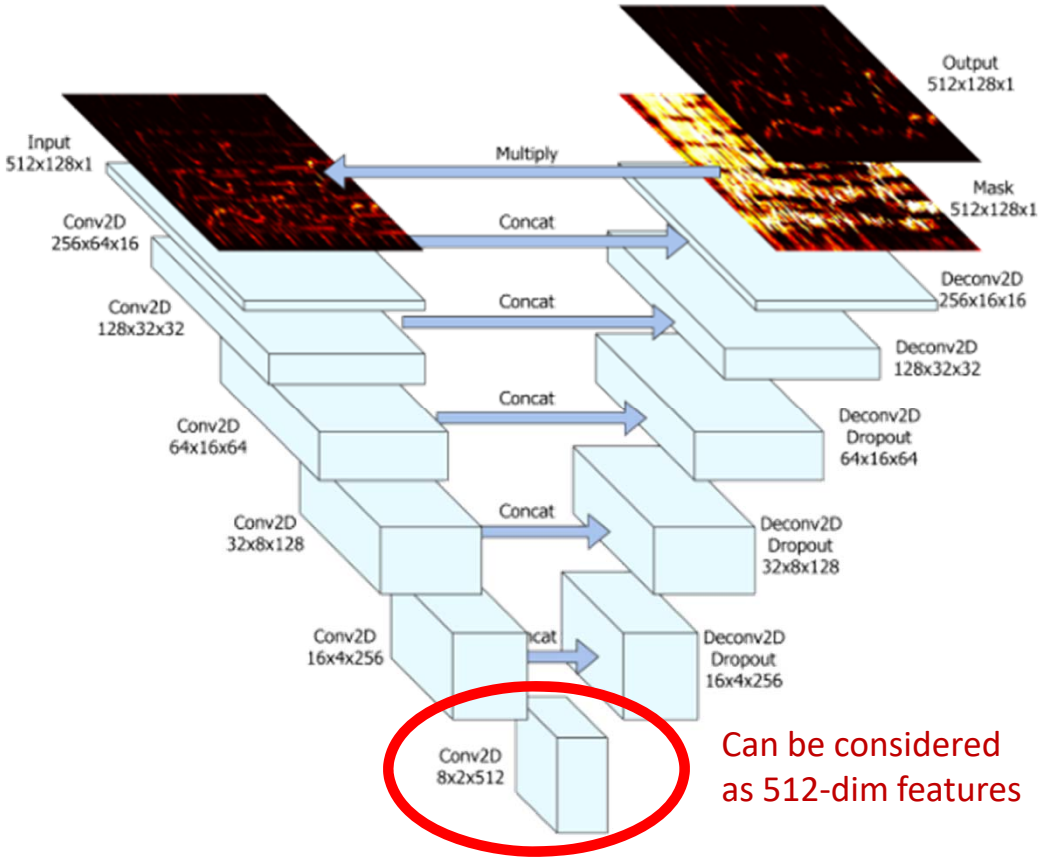
- Like the previous approach; operate *frame-by-frame*
- Connection between frames handled by recurrent layers or by stacking frames

Ref: Uhlich et al, "Deep neural network based instrument extraction from music," ICASSP 2015

U-Net-based Approach

https://source-separation.github.io/tutorial/approaches/deep/architectures.html

- Input a spectrogram (***multiple* frames**), output a mask
  - Exploit 2D time-frequency patterns
  - Fixed-size input/output
    - An audio signal must be broken up into spectrograms with fixed number of time and frequency dimensions
- U-Net: **encoder/decoder** with symmetric skip connections
  - Down-sample and then up-sample



Ref: Jansson et al, "Singing voice separation with deep u-net convolutional networks," ISMIR 2017

Library: Spleeter

(It uses the U-Net approach)

https://github.com/deezer/spleeter

Spleeter is Deezer source separation library with pretrained models written in Python and uses Tensorflow. It makes it easy to train source separation model (assuming you have a dataset of isolated sources), and provides already trained state of the art model for performing various flavour of separation :

- Vocals (singing voice) / accompaniment separation (2 stems)
- Vocals / drums / bass / other separation (4 stems)
- Vocals / drums / bass / piano / other separation (5 stems)

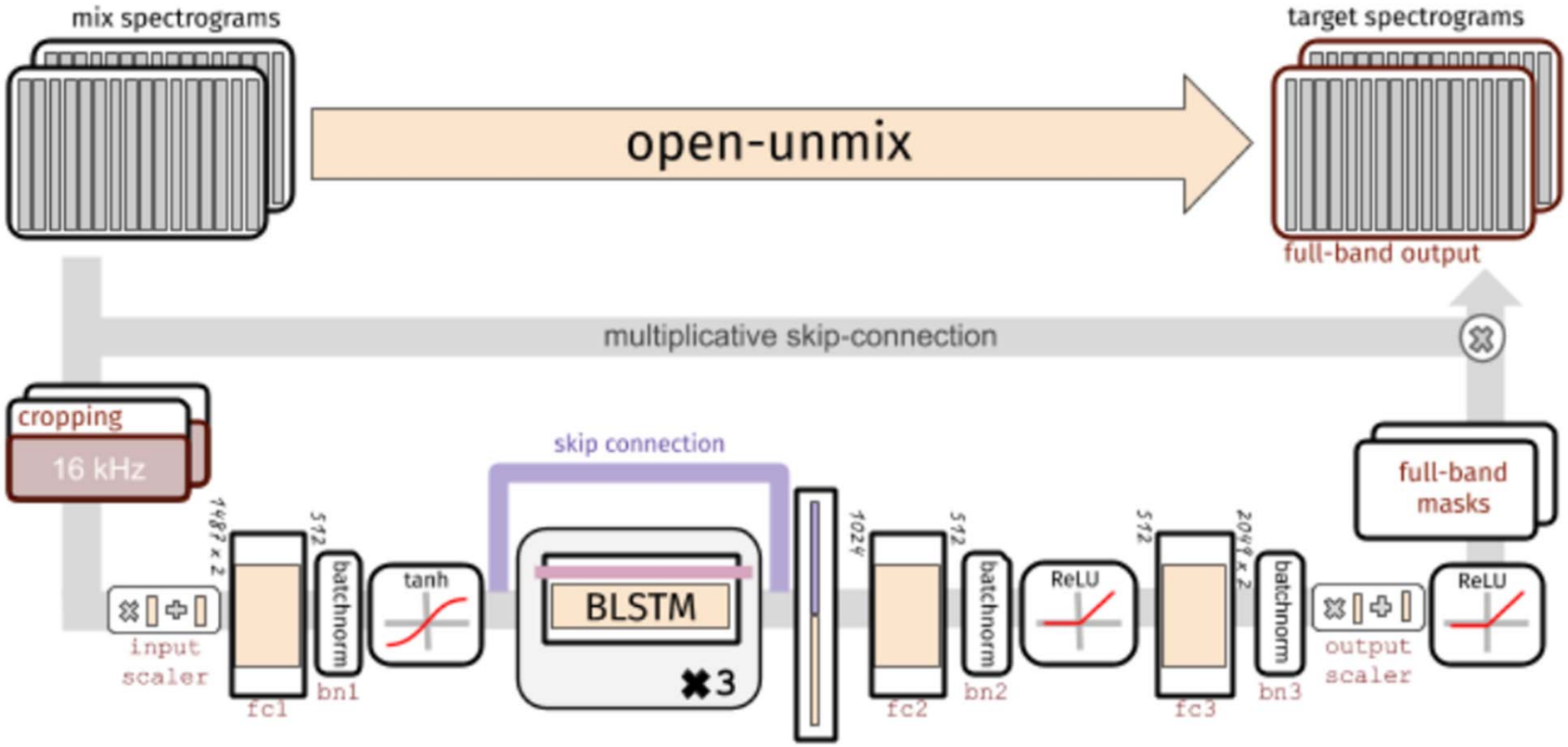
2 stems and 4 stems models have high performances on the musdb dataset. **Spleeter** is also very fast as it can perform separation of audio files to 4 stems 100x faster than real-time when run on a GPU.

We designed **Spleeter** so you can use it straight from command line as well as directly in your own development pipeline as a Python library. It can be installed with pip or be used with Docker.

Ref: Hennequin et al, "Spleeter: a fast and efficient music source separation tool with pre-trained models," J. Open Source Software 2020

Library: Open-Unmix

https://sigsep.github.io/open-unmix/



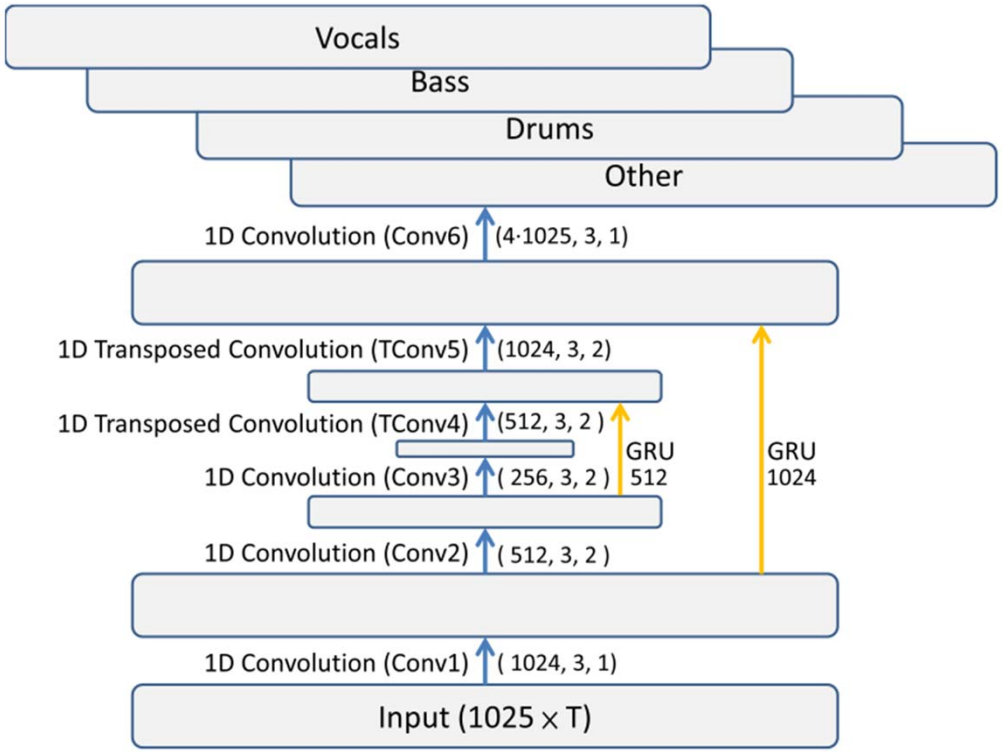
Ref: Stöter et al, “Open-Unmix - A reference implementation for music source separation,” J. Open Source Software 2019

Breakthrough 2: Advance in Neural Audio Generation

- Generative models become more and more powerful, for not only images but also audio
  - More on the details of such models in the next lectures
- **Let's predict (synthesize) the stem directly**
  - *Discriminative → generative*

Let's Synthesize the Stems: the ARC Model

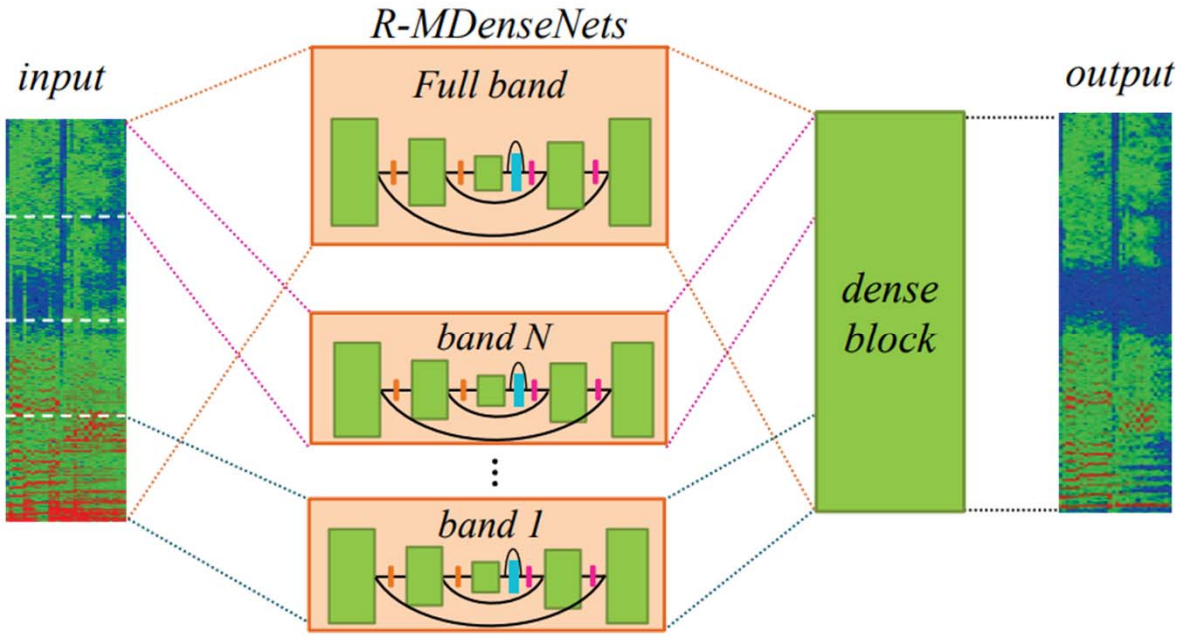
- ARC model [liu18icmla]
  - **Leakly ReLU** instead of **softmax** at output
    - "In our pilot experiments, we also tried to apply a softmax function to the output layer so that the network predicts masks for different sources and enforces the condition that the summation of the predicted source spectrograms is equal to the mixture spectrogram. We found that this setting ***largely speeds up*** the training process, but ***the result becomes much worse***. Therefore, we decided to use a leakly ReLU as the nonlinearity function to the output layer to directly estimate the source spectrograms".
  - 1D convolution instead of 2D (so that we can use *recurrent* skip connections)

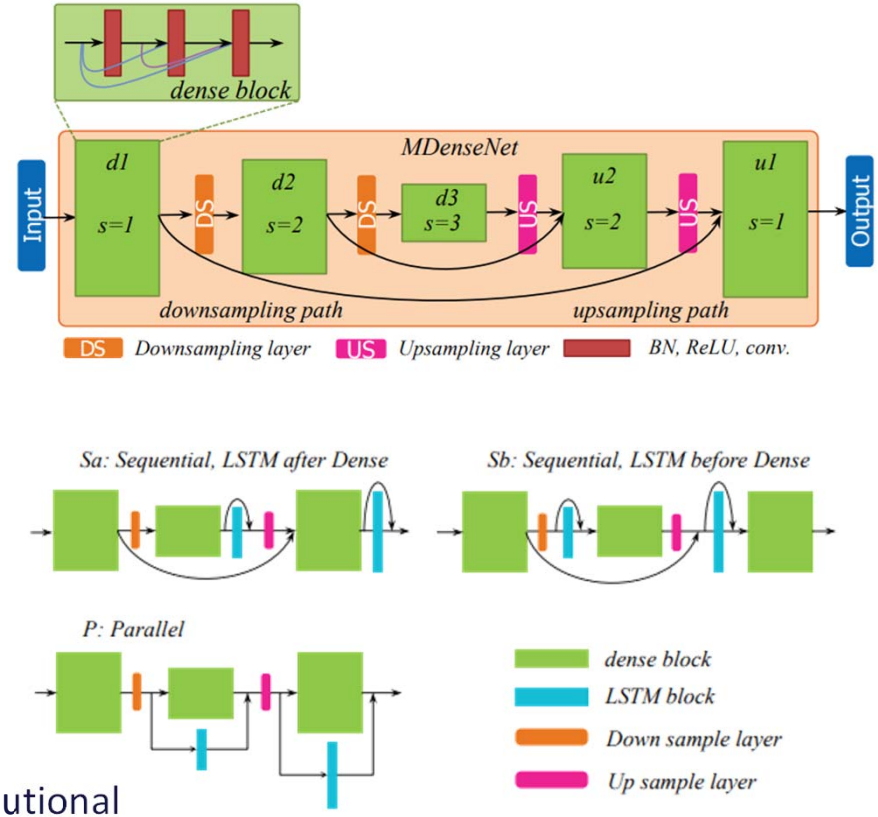


Ref: Liu and Yang, "Denoising auto-encoder with recurrent skip connections and residual regression for music source separation," ICMLA 2018

MMDenseLSTM

- DenseNet with multi-scale and multi-band structure
- LSTM for long-term dependencies

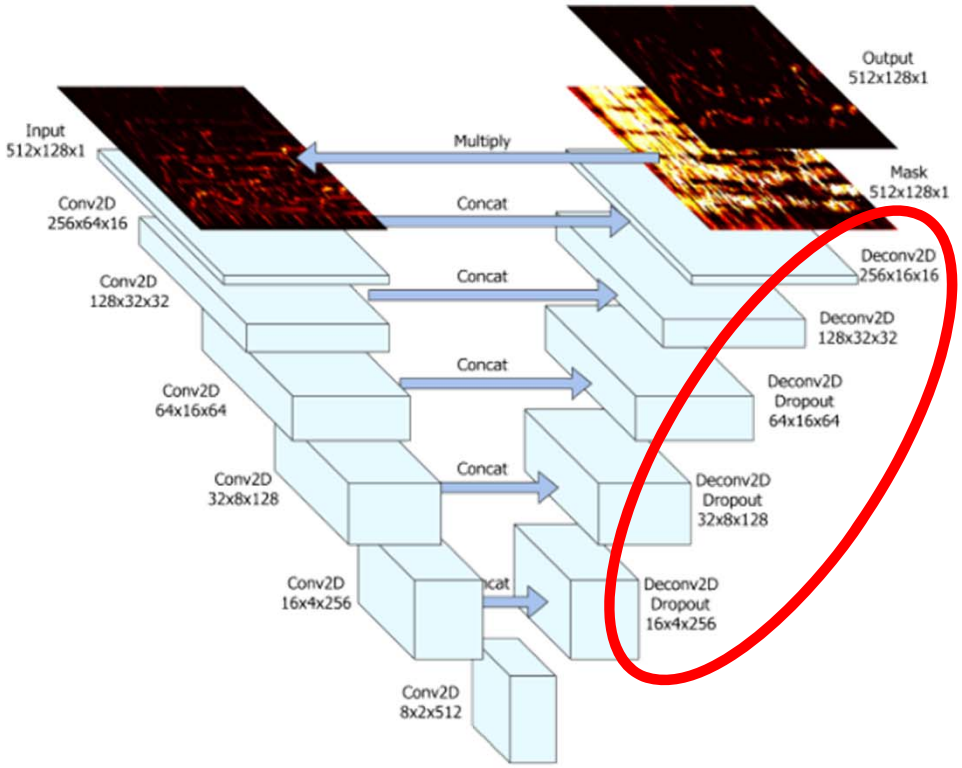




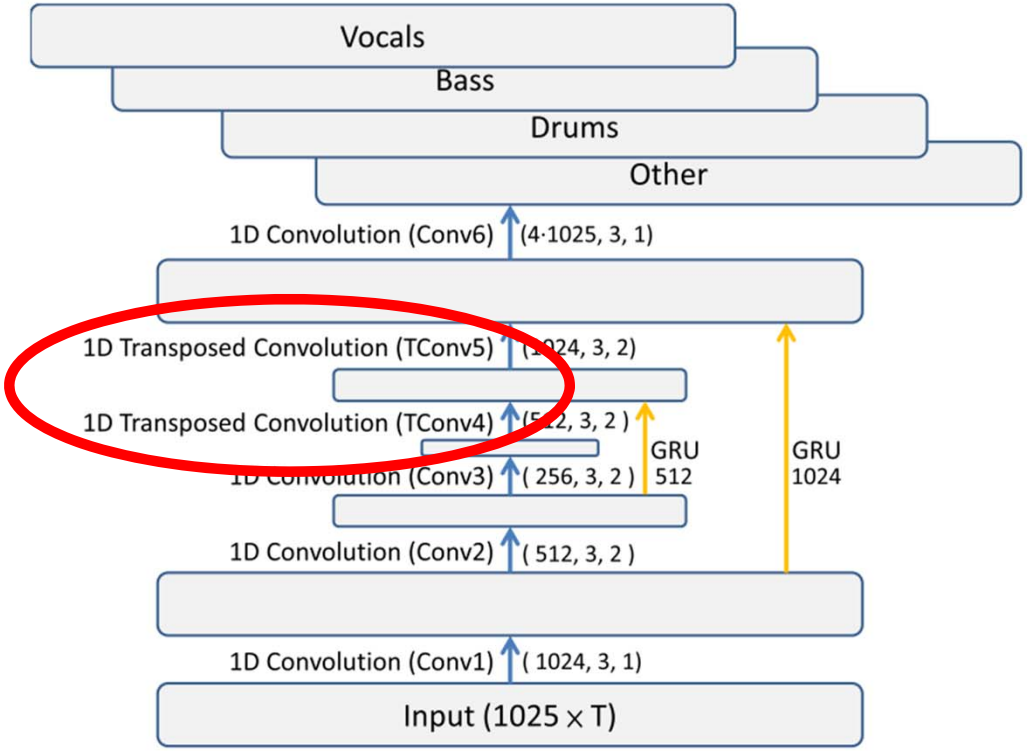
Ref: Takahashi et al, "MMDenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation," IWAENC 2018

Upsampling Layers: Transposed Convolution

- Also known as "**deconvolution** layers" → lecture 5



U-Net-based source separation model [jansson17ismir]



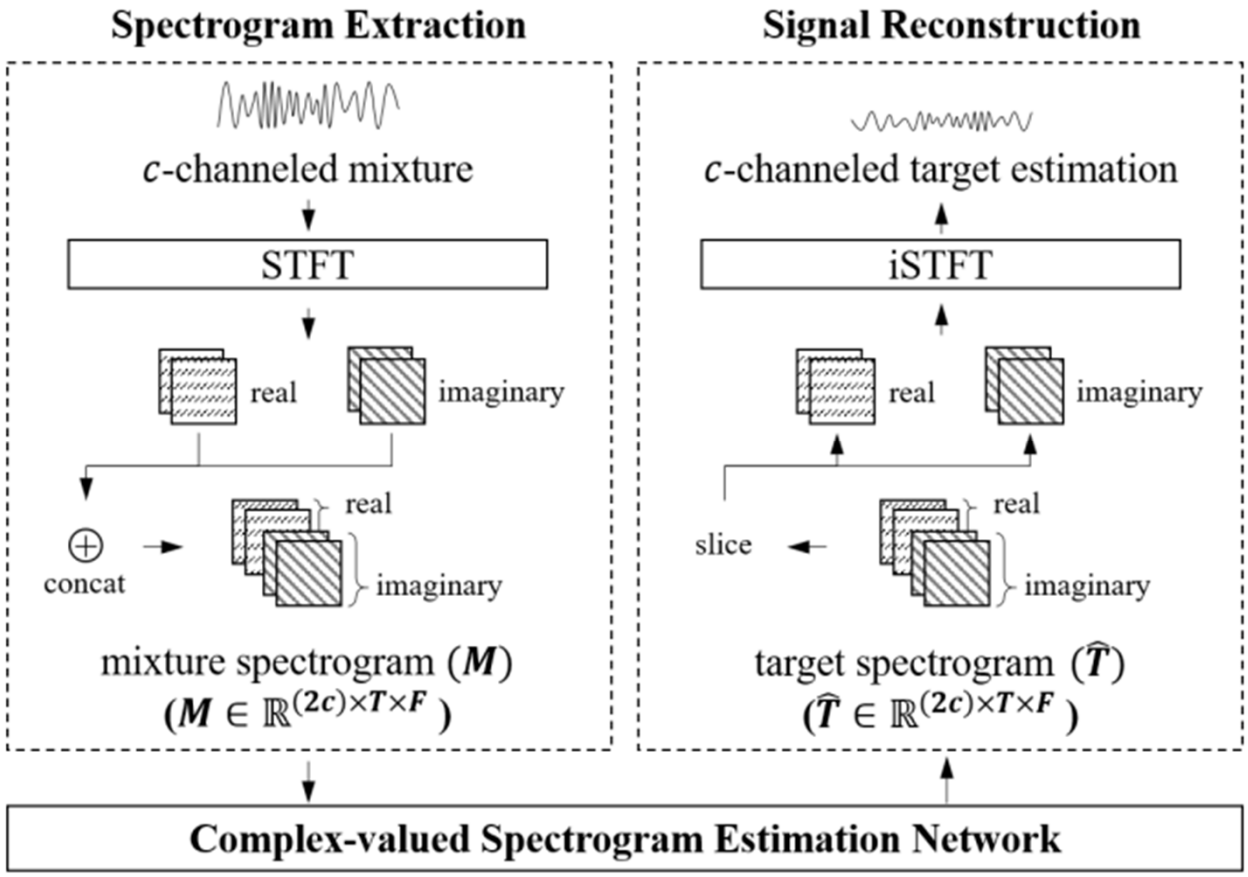
ARC [liu18icmla]

Deal with Phase: Approaches

https://source-separation.github.io/tutorial/basics/phase.html

- **Copy the phase from the mixture**
  - Work fine for source separation (but maybe NOT for other audio generation problems)
- **Given the magnitude, estimate the phase** (see the lecture on "**vocoder**")
  - General-purpose solution → lecture 5
- **Work on audio waveforms, not spectrograms**
  - General-purpose solution

TFC-TDF: Work on Complex-Valued Spectrogram



Choi et al, “Investigating U-Nets with various intermediate blocks for spectrogram-based singing voice separation,” ISMIR 2020

How about Phase?

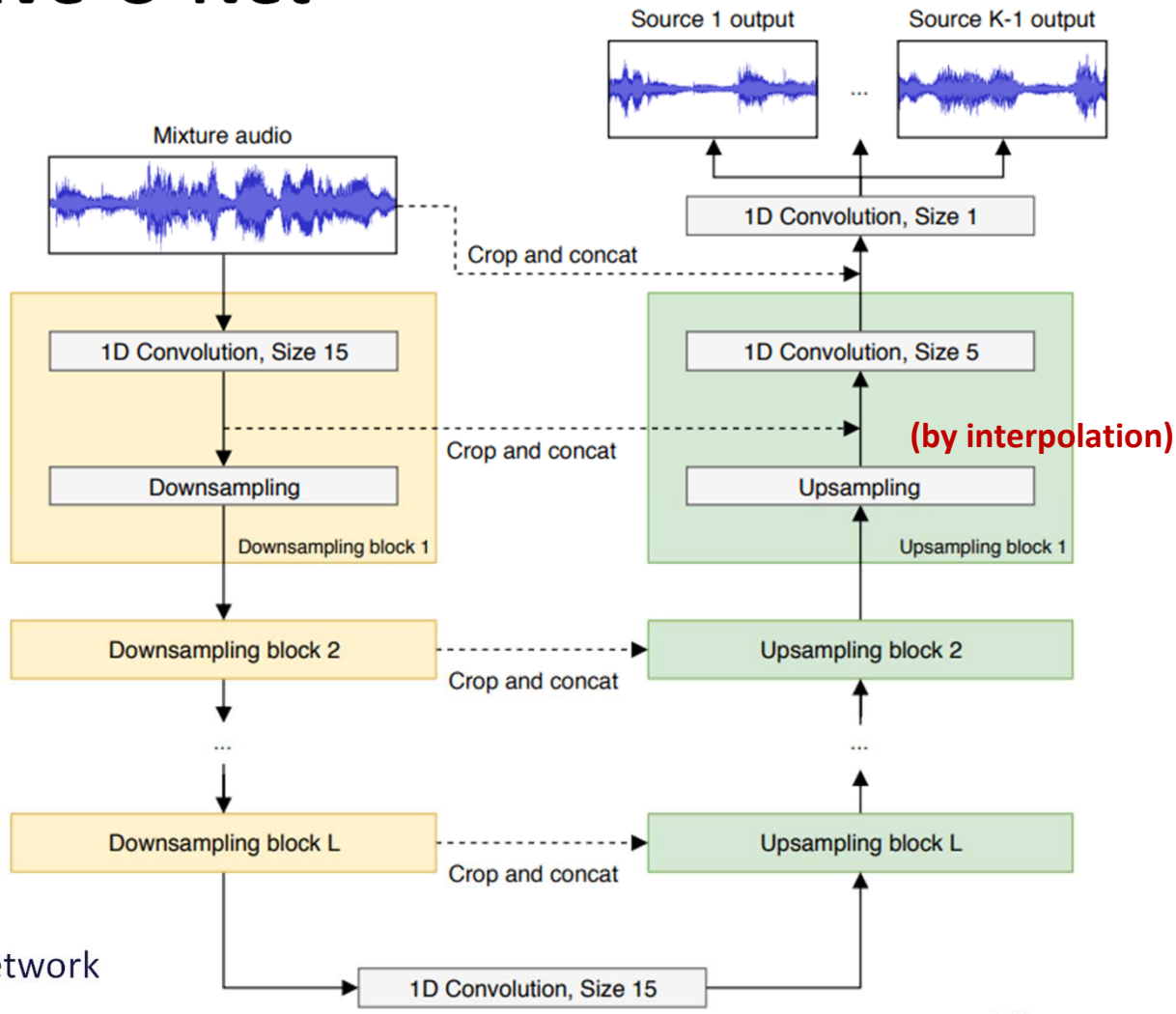
https://source-separation.github.io/tutorial/basics/phase.html

- Copy the phase from the mixture
  - Work fine for source separation (but maybe NOT for other audio generation problems)
- Given the magnitude, estimate the phase (see the lecture on “vocoder”)
  - General-purpose solution
- **Work on audio waveforms, not spectrograms**
  - General-purpose solution

Wave-U-Net

(waveform-domain loss)

- Work on audio waveforms
- The "claimed" limits of the STFT approach
  - "the STFT output depends on many parameters, such as the size and overlap of audio frames, [… which are not optimized along with the separation model]"
  - "does not estimate the source phase"
  - "the mixture phase is ignored in the estimation of sources"
- However, poor performance



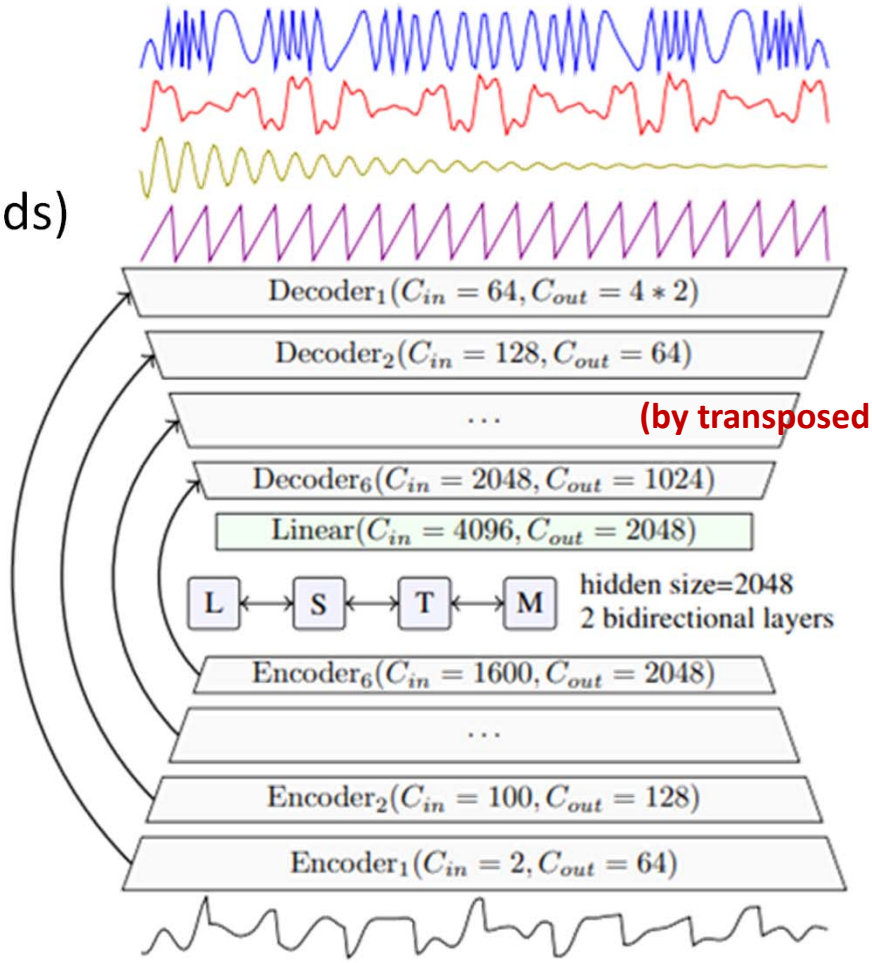
Ref: Stoller et al, "Wave-U-Net: A multi-scale neural network for end-to-end audio source separation," ISMIR 2018

Demucs (v2)

(waveform-domain loss)

- Several novel elements inspired by latest work on audio synthesis at that time
- Great result (but *slower* than STFT-based methods)

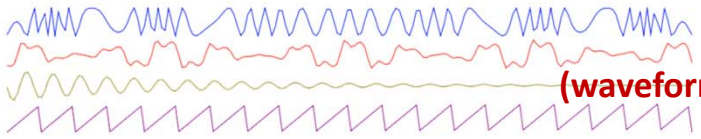
Difference	Valid set L1 loss	Test set SDR
no BiLSTM	0.171	5.40
no pitch/tempo aug.	0.156	5.80
no initial weight rescaling	0.156	5.88
no 1x1 convolutions in encoder	0.154	5.89
ReLU instead of GLU	0.157	5.92
no BiLSTM, depth=7	0.160	5.94
MSE loss	N/A	5.99
no resampling	0.153	6.03
no shift trick	N/A	6.05
kernel size of 1 in decoder convolutions	0.153	6.11
Reference	0.150	6.28



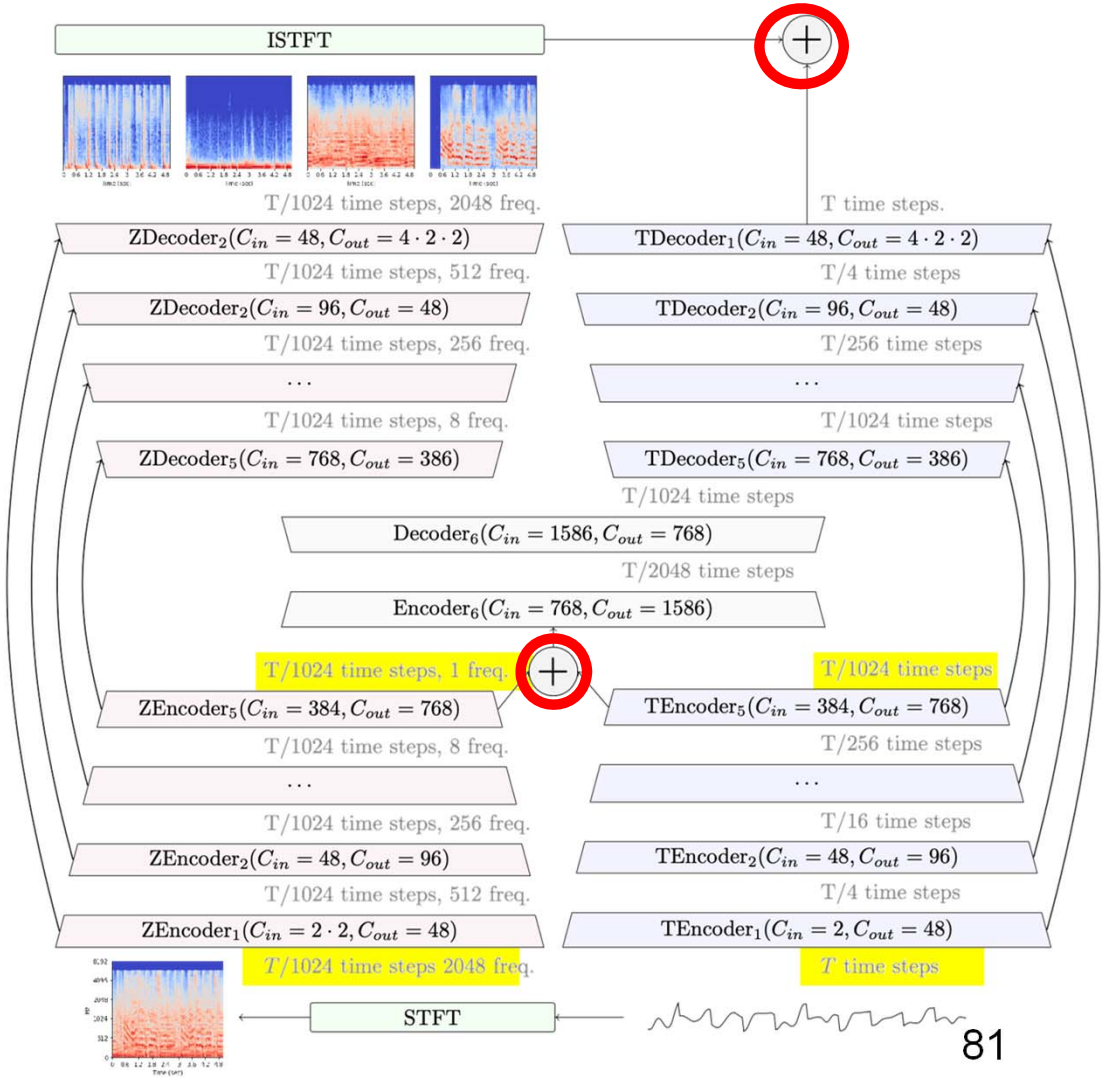
(by transposed conv)

Ref: Défossez et al, "Music source separation in the waveform domain," arXiv 2019

Demucs (v3)

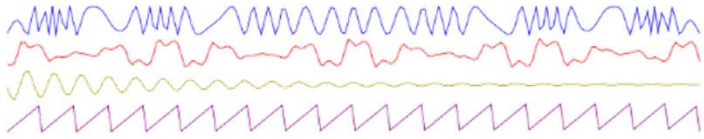


- Conv-2D on spectrograms still has some advantages
- Temporal branch, spectral branch, and shared layers
  - Aligned temporal resolution
  - “The output of the spectral branch is inversed with the ISTFT, and summed with the temporal branch output, giving the final model prediction.”
  - “The model is free to use whichever representation is most convenient for different parts of the signal, even within one source, and can freely share information between the two representations.”

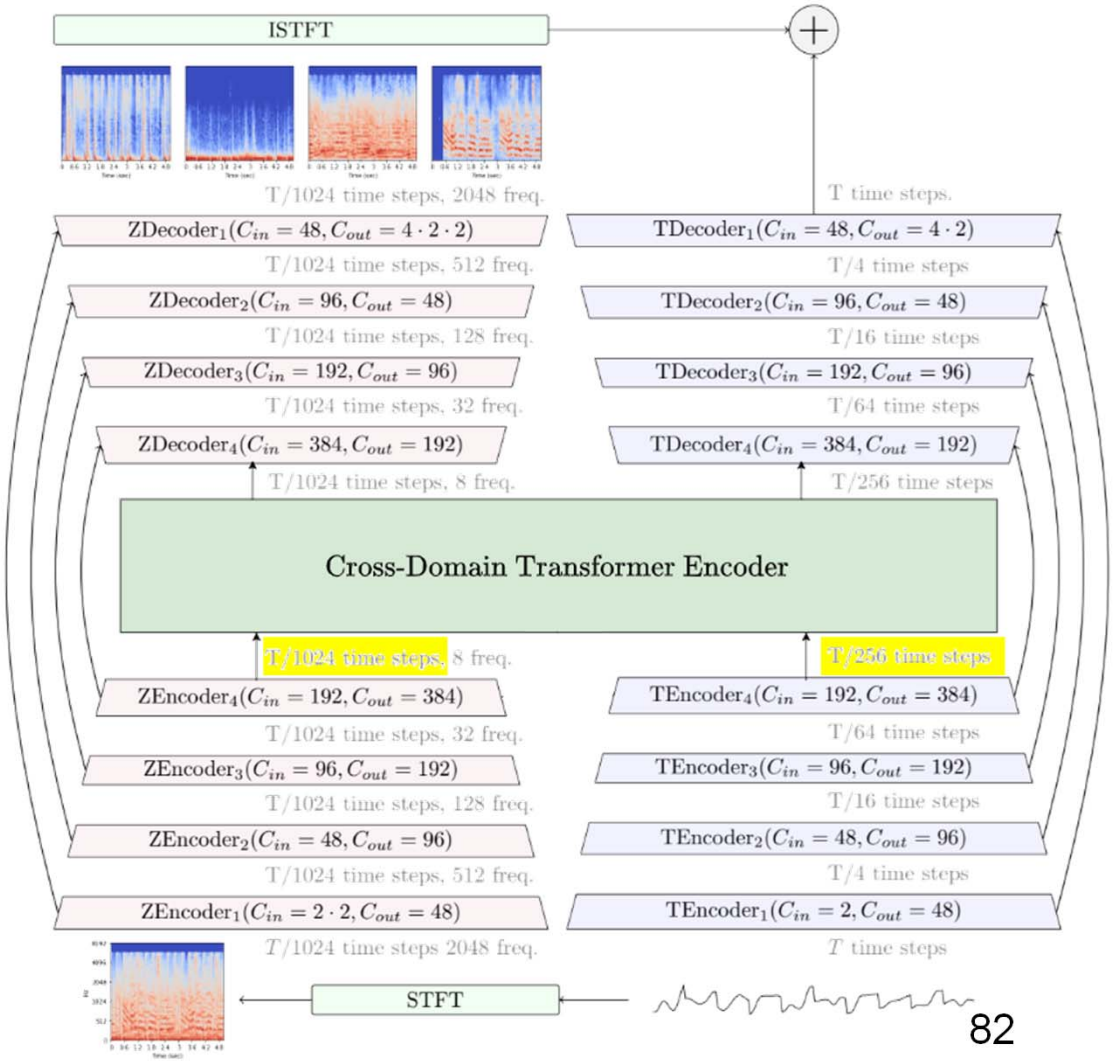


Ref: Défossez, “Hybrid spectrogram and waveform source separation,” ISMIR MDX Workshop 2021

Demucs (v4)



- Use **Transformer** to learn long range contextual information
  - "Innermost layers are replaced by a cross-domain Transformer Encoder, using self-attention within one domain, and cross-attention across domains."
- Need larger data to benefit from the Transformer
  - "Highly benefited from using extra training data"



Ref: Rouard et al, "Hybrid Transformers for music source separation," ICASSP 2023

Library: Demucs

https://github.com/facebookresearch/demucs

Model	Domain	Extra data?	Overall SDR
Wave-U-Net	waveform	no	3.2
Open-Unmix	spectrogram	no	5.3
Demucs (v2)	waveform	no	6.3
Band-Spit RNN	spectrogram	no	**8.2**
Hybrid Demucs (v3)	hybrid	no	7.7
MMDenseLSTM	spectrogram	804 songs	6.0
Spleeter	spectrogram	25k songs	5.9
HT Demucs f.t. (v4)	hybrid	800 songs	**9.0**

Audiostrip is providing free online separation with Demucs on their website https://audiostrip.co.uk/.

Neutone provides a realtime Demucs model in their free VST/AU plugin that can be used in your favorite DAW.

Other pre-trained models can be selected with the `-n` flag. The list of pre-trained models is:

- `htdemucs_6s`: 6 sources version of `htdemucs`, with `piano` and `guitar` being added as sources. Note that the `piano` source is not working great at the moment.

Note: Time-Domain Models Need More Compute

- Demucs v2: 8 V100 GPUs with 16GB of RAM
- Demucs v4: 8 V100 GPUs with 32GB of RAM
- And the training may take days or even weeks

	Music source separation (MUSDB [27] benchmark)	SDR ↑	epoch	#parm
	WaveUnet [2]: linear intepolation	3.23	-	10M
v2 →	Demucs [3]: transposed convolution (full-overlap)	5.34	-	648M
	OpenUnmix [30]: spectrogram-based	5.36	-	8.9M
	Sams-net [31]: spectrogram-based	5.65	-	3.7M

Table source: Pons et al, “Upsampling artifacts in neural audio synthesis,” ICASSP 2021

Overlap-and-Add

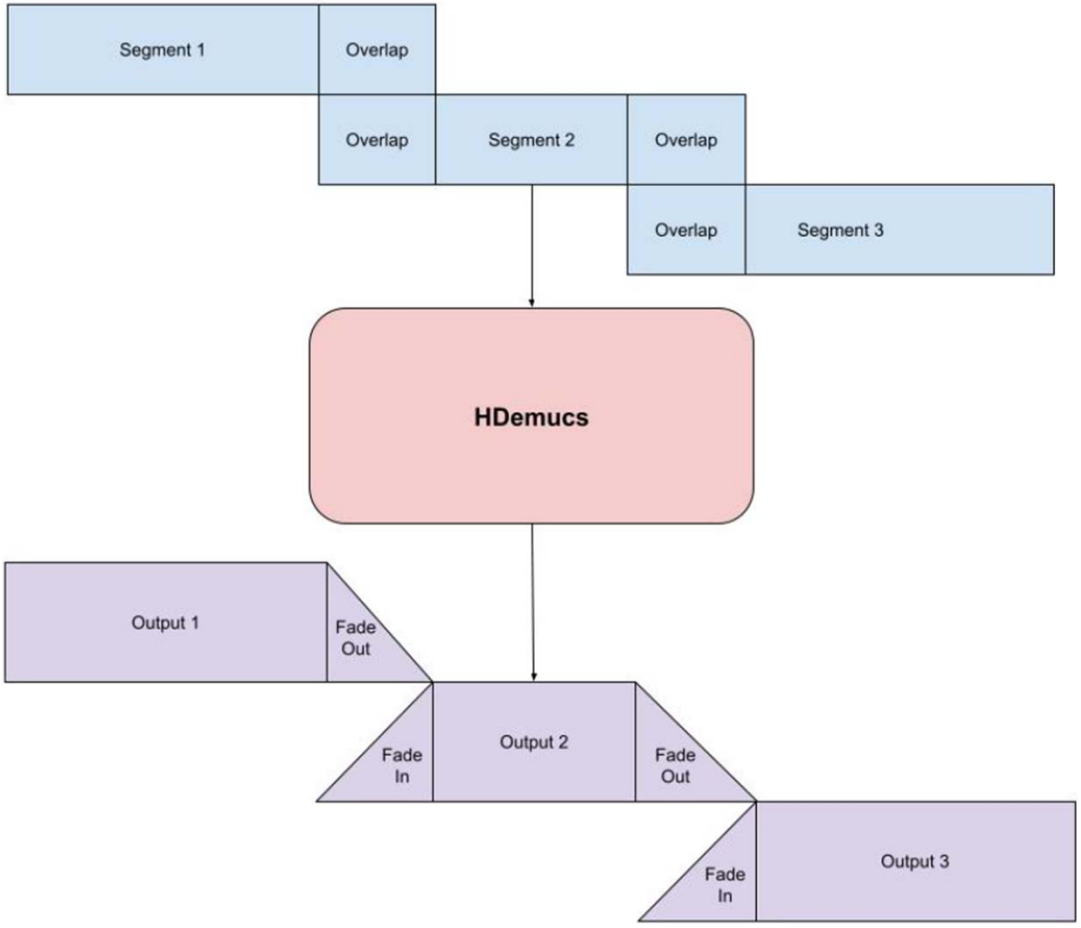
https://pytorch.org/audio/main/tutorials/hybrid_demucs_tutorial.html

https://pytorch.org/audio/main/pipelines.html

```
bundle = HDEMUCS_HIGH_MUSDB_PLUS

model = bundle.get_model()
```

```
while start < length - overlap_frames:
    chunk = mix[:, :, start:end]
    with torch.no_grad():
        out = model.forward(chunk)
    out = fade(out)
    final[:, :, :, start:end] += out
    if start == 0:
        fade.fade_in_len = int(overlap_frames)
        start += int(chunk_len - overlap_frames)
    else:
        start += chunk_len
    end += chunk_len
    if end >= length:
        fade.fade_out_len = 0
return final
```



More and More Datasets

- **MIR-1k** (https://zenodo.org/record/3532216) /2010
  - 1,000 song clips which the music accompaniment and the singing voice are recorded at left and right channels (i.e., **vocal/others**)
- **MedleyDB** (https://medleydb.weebly.com/downloads.html) / 2014, 2016
  - 194 songs, with a diverse set of stems (piano, flute, violin, viola, bassoon, etc)
- **MUSDB** (https://sigsep.github.io/datasets/musdb.html) /2017, 2019
  - 150 full-length songs (~10h duration) with **drums/bass/vocals/others** stems
- **Moisesdb** (https://github.com/moises-ai/moises-db) / 2023
  - 240 tracks; for source separation beyond 4 stems (e.g., 6 stems including **piano/guitar**)

Extra (Private) Datasets

Rank	Model	SDR (avg)	SDR (vocals)	SDR (drums)	SDR (bass)	SDR (other)	Extra Training Data	Paper	Code	Result	Year
1	**Sparse HT Demucs** (fine tuned)	9.20	9.37	10.83	10.47	6.41	✓	Hybrid Transformers for Music Source Separation			2022
2	**Hybrid Transformer Demucs** (f.t.)	9.00	9.20	10.08	9.78	6.42	✓	Hybrid Transformers for Music Source Separation			2022
3	**Band-Split RNN** (semi-sup.)	8.97	10.47	10.15	8.16	7.08	✓	Music Source Separation with Band-split RNN			2022
4	**TFC-TDF-UNet** (v3)	8.34	9.59	8.44	8.45	6.86	×	Sound Demixing Challenge 2023 Music Demixing Track Technical Report: TFC-TDF-UNet v3			2023
5	**Band-Split RNN**	8.23	10.21	8.58	7.51	6.62	×	Music Source Separation with Band-split RNN			2022
6	**Hybrid Demucs**	7.72	8.04	8.58	8.67	5.59	×	Hybrid Spectrogram and Waveform Source Separation			2021
7	**KUIELab-MDX-Net**	7.54	9.00	7.33	7.86	5.95	×	KUIELab-MDX-Net: A Two-Stream Neural Network for Music Demixing			2021

Breakthrough 3: Data Augmentation

- "We use the following data modifications on-the-fly when we construct a training mini-batch sequence"
  - Random swapping left/right channel for each instrument
  - Random scaling with uniform amplitudes from [0.25,1.25]
  - Random chunking into sequences for each instrument
  - **Random mixing of instruments from different songs**

Ref: Uhlich et al, "Improving music source separation based on deep neural networks through data augmentation and network blending," ICASSP 2017

Incoherent (Random) Mixing vs Coherent Mixing

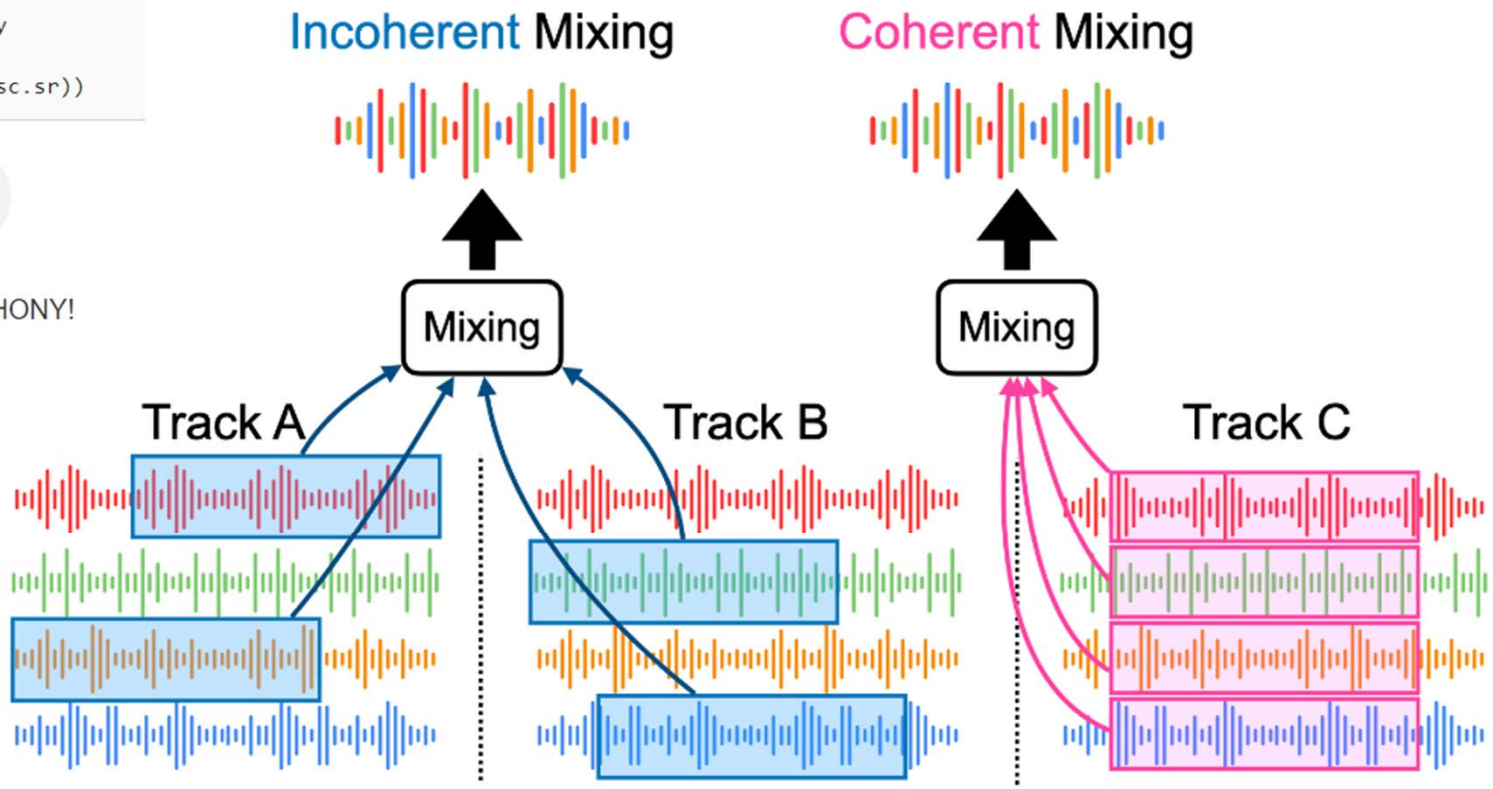
https://source-separation.github.io/tutorial/data/scaper.html

```
from IPython.display import Audio, display

display(Audio(data=mixture_audio.T, rate=sc.sr))
```

0:03 / 0:05

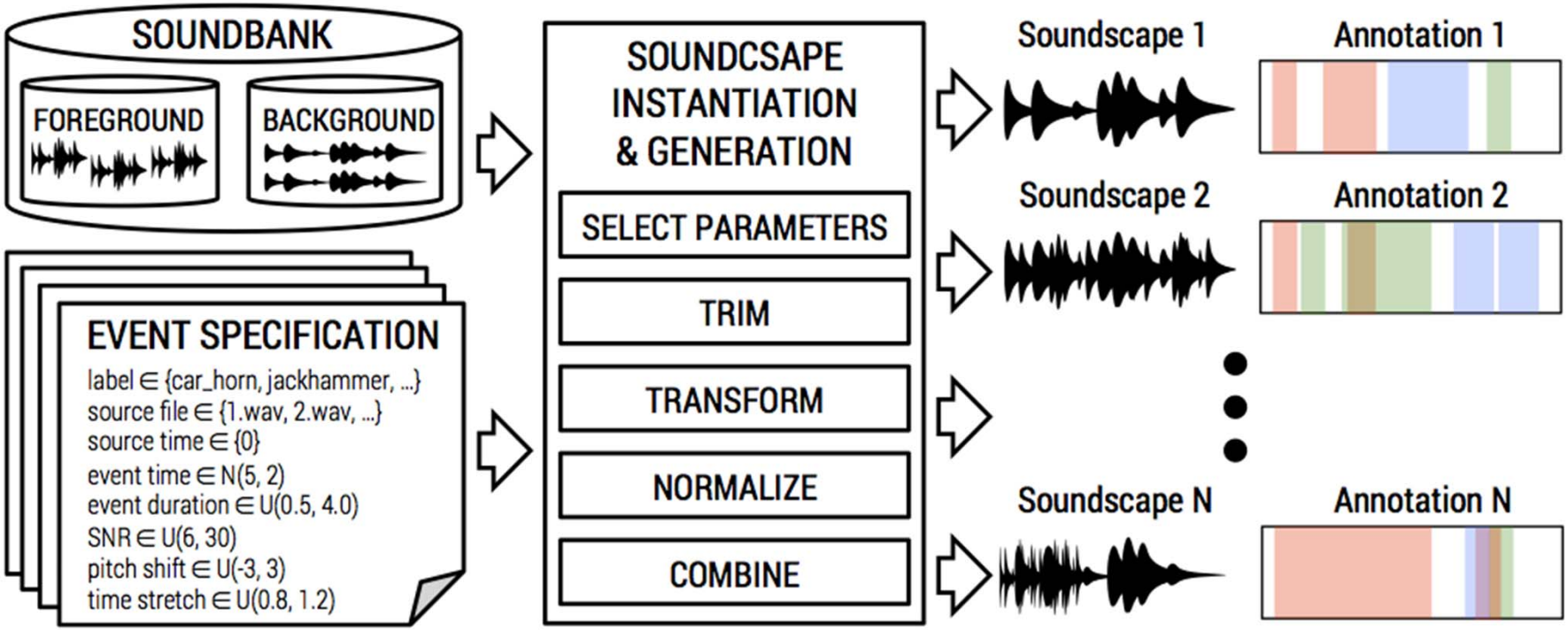
...Wait! WHAT WAS THAT? CHAOS! CACOPHONY!



Ref: Jeon et al, "Why does music source separation benefit from cacophony?," ICASSP Workshop on Explainable AI for Speech and Audio 2024

Library: Scaper

https://source-separation.github.io/tutorial/data/scaper.html



Coherent (Musically-Realistic) Mixing

https://github.com/facebookresearch/demucs/blob/main/tools/automix.py

https://github.com/facebookresearch/demucs/blob/main/docs/training.md

- Demucs **auto-mix**: Beat matching and pitch compatibility

> We use `librosa` (McFee et al., 2015) for both beat tracking and tempo estimation, as well as chromagram estimation. Beat tracking is applied only on the drum source, while chromagram estimation is applied on the bass line. We aggregate the chromagram over time to a single chroma distribution and find the optimal pitch shift between two stems to maximize overlap (as measured by the L1 distance). We assume that the optimal shift for the bass line is the same for the vocals and accompaniments. Similarly, we align the tempo and first beat. In order to limit artifacts, we only allow two stems to blend if they require less than 3 semi-tones of shift and 15% of tempo change.

Ref: Défossez, "Hybrid spectrogram and waveform source separation," ISMIR MDX Workshop 2021

Recap

- How people work on spectrograms?
  - Copy the phase of the mixture (for source separation)
  - Vocoder → lecture 5
  - Work on waveforms instead of spectrograms (or both)

- It's a type of audio generation task (with *strong condition*)
  - "audio (mixture) → audio (stem)"
  - Earlier deep learning approaches predict mask: *discriminative* setting
  - SOTA approaches synthesize the stems from the input mixture: *generative* setting
  - But, we haven't talked about the details of the up-sampling layers → lecture 5