Music demixing with the sliCQ transform

Sevag Hanssian

February 18, 2022

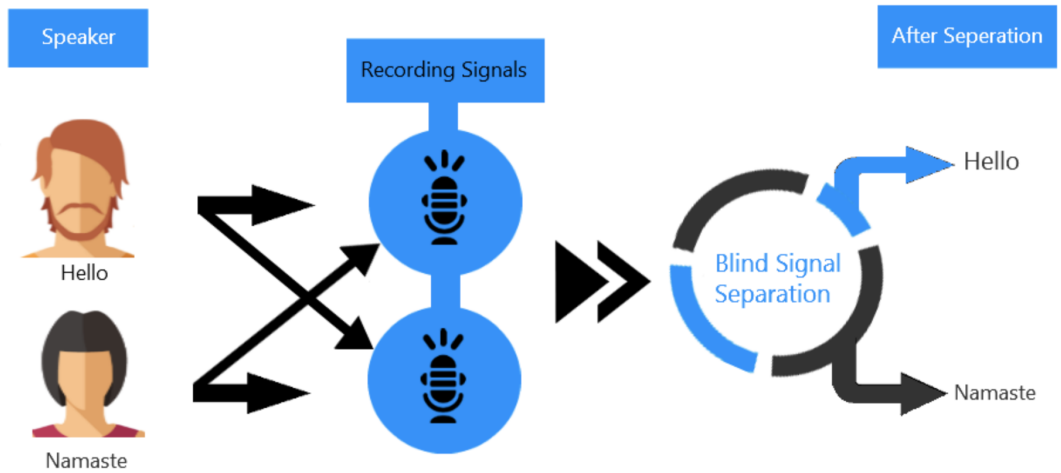
Intro to myself

1. Electrical engineering background, McGill 2014
2. Linux infrastructure engineer at NVIDIA (formerly Pandora, the U.S. music streaming service)
3. Open-source enthusiast: `https://github.com/sevagh`, `https://gitlab.com/sevagh`
4. Student in the Master of Arts, Music Tech (thesis) program at McGill
5. Member of the Distributed Digital Music Archives & Libraries lab led by Prof. Ichiro Fujinaga

Music source separation is the task of extracting an estimate of one or more isolated sources or instruments (for example, drums or vocals) from musical audio. The task of music demixing or unmixing considers the case where the musical audio is separated into an estimate of all of its constituent sources that can be summed back to the original mixture. The Music Demixing Challenge was created to inspire new demixing research. Open-Unmix (UMX), and the improved variant CrossNet-Open-Unmix (X-UMX), were included in the challenge as the baselines. Both models use the Short-Time Fourier Transform (STFT) as the representation of music signals. The time-frequency uncertainty principle states that the STFT of a signal cannot have maximal resolution in both time and frequency. The tradeoff in time-frequency resolution can significantly affect music demixing results. Our proposed adaptation of UMX replaced the STFT with the sliCQT, a time-frequency transform with varying time-frequency resolution. Unfortunately, our model xumx-sliCQ achieved lower demixing scores than UMX.

Audio source separation

1. Audio source separation is the task of extracting an estimate of an isolated source from audio, e.g., cocktail party in speech



[1]

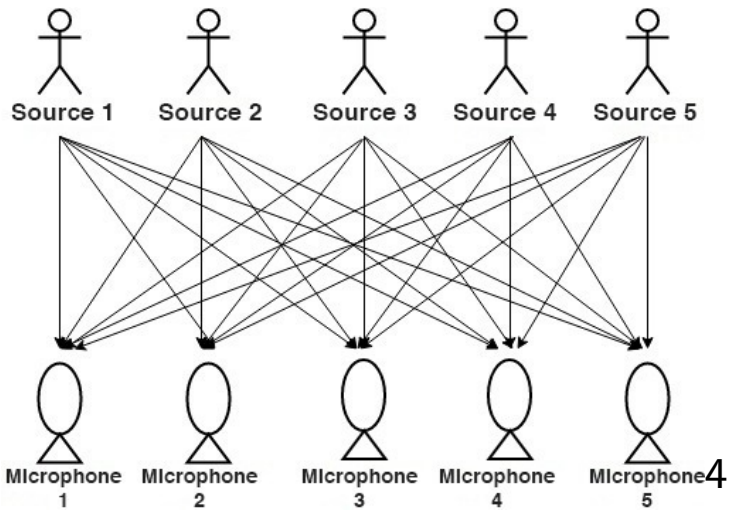
2. Computational source separation has a history of at least 50 years.[2] In computational auditory scene analysis (CASA) and blind source separation (BSS), separate unknown sources

[1] https://gowrishankar.info/blog/cocktail-party-problem-eigentheory-and-blind-source-separation-using-ica/

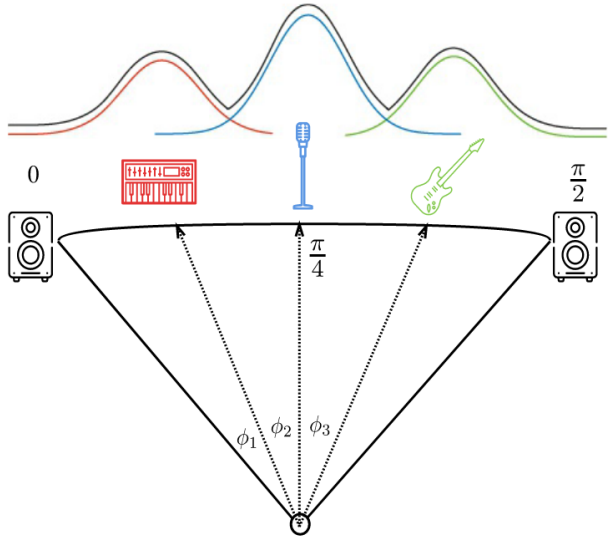
[2] Liu and Li, 2009; Rafii, Liutkus, Stöter, Mimilakis, FitzGerald, et al., 2018.

Why speech techniques don't work for music

Speech separation algorithm: Independent Component Analysis (ICA)[3]



[4]



ICA in speech applications uses spatial information, requires as many channels as the number of sources, assumes independent sources, and assumes the background is stationary

In music, there are typically more instruments than channels, musical sources are highly dependent, and music is nonstationary and synchronous

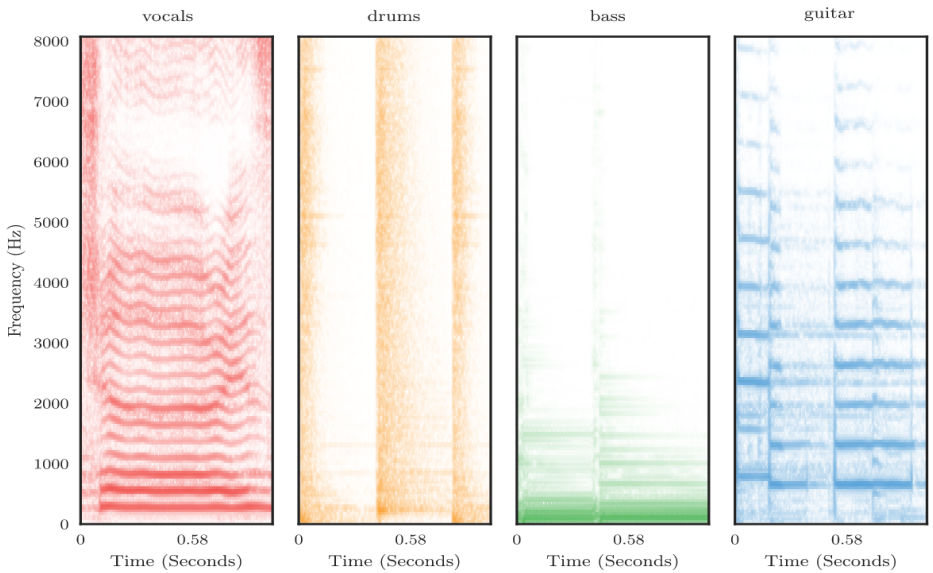
[3] Rafii, Liutkus, Stöter, Mimilakis, FitzGerald, et al., 2018.

[4] `https://medium.com/appengine-ai/independent-component-analysis-machine-learning-b62ff260c022`

Music source separation

... is the task of extracting an estimate of one or more isolated sources or instruments from musical audio

- Music source separation: extract an estimate of an isolated *known* source from the mix (e.g., harmonic/percussive, vocals, drums), or a source with *known characteristics*, that are dependent[5]
- Popular approach: musical source models, which are "model-based approaches that attempt to capture the spectral characteristics of the target source"[6] with time-frequency masks

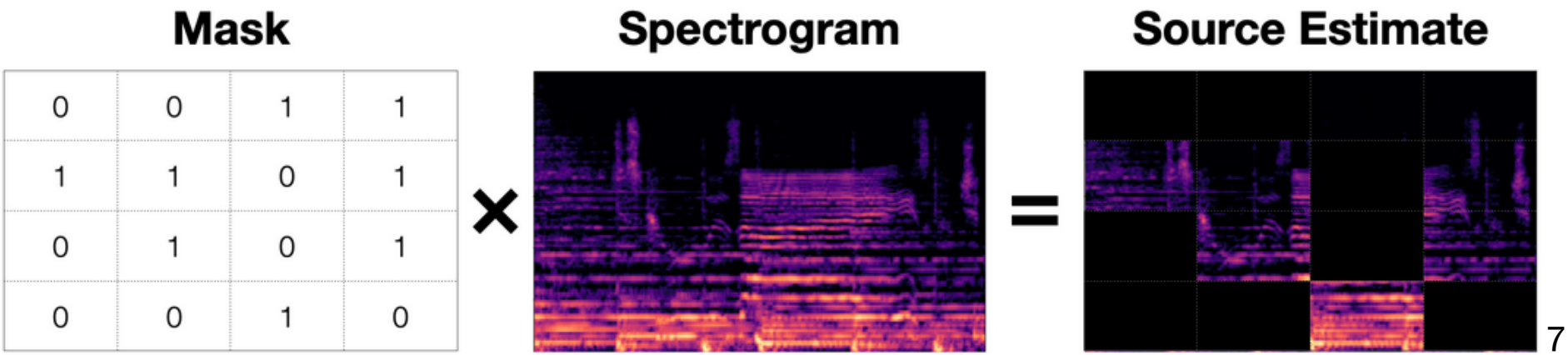


[5] Rafii, Liutkus, Stöter, Mimilakis, FitzGerald, et al., 2018.

[6] Cano, Fitzgerald, et al., 2018, p. 36.

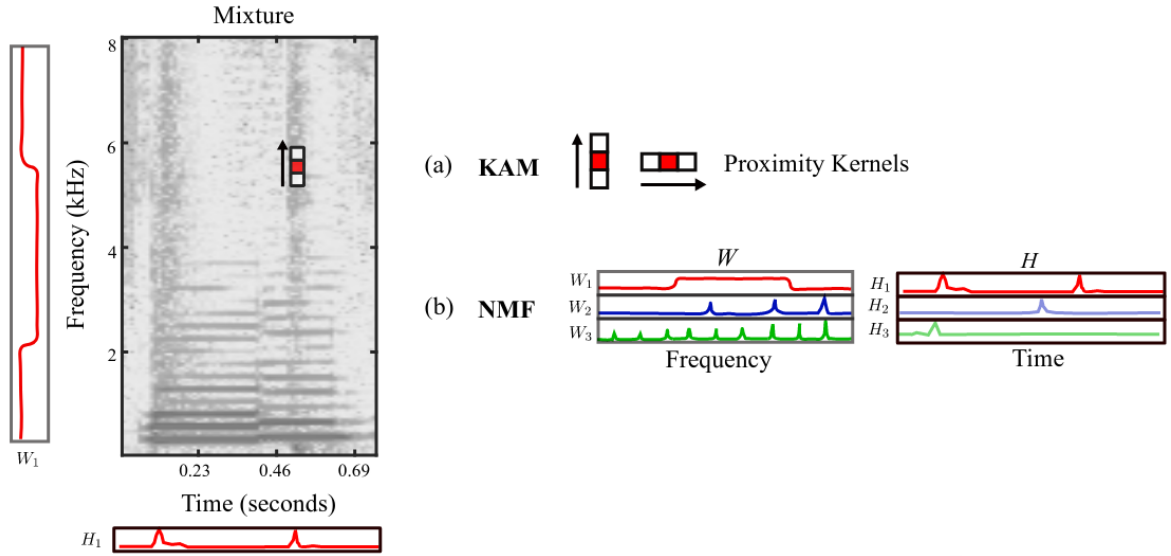
Time-frequency masks

Multiply the time-frequency (TF) transform (typically, a Short-Time Fourier Transform) with a mask, or a matrix of the same size as the TF transform with values $\in [0, 1]$

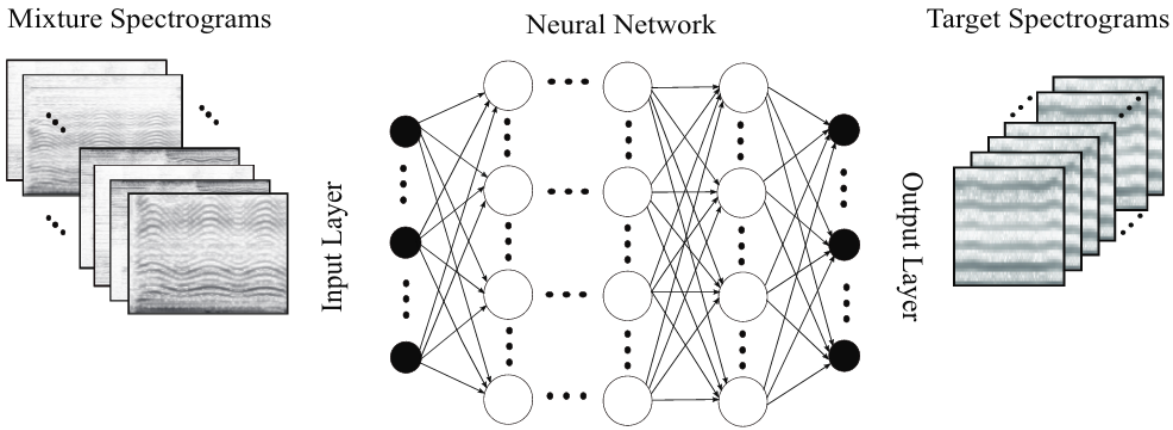


[7] https://source-separation.github.io/tutorial/basics/tf_and_masking.html

Estimating time-frequency masks



(a) Kernel Additive Modeling (KAM) and Nonnegative Matrix Factorization (NMF)

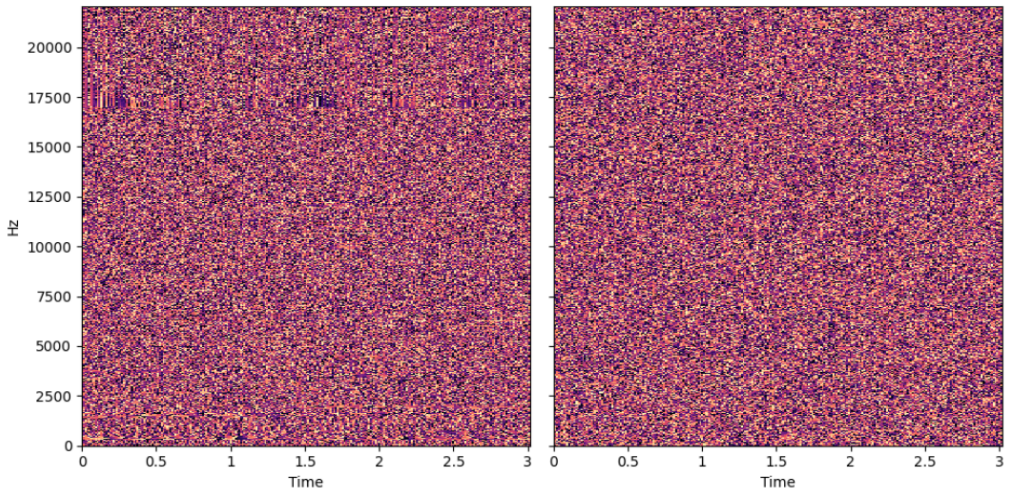


(b) Deep neural networks

Magnitude and phase spectrogram

Common approaches to MSS discard the phase; it's difficult to learn relationships from phase

1. Simplifying assumption: estimate magnitude spectrograms, use the phase of the original mixed audio. Called "noisy phase"[8]. Done by Open-Unmix (UMX), CrossNet-Open-Unmix (X-UMX)[9], and many other popular & near-SOTA models
2. Why? Phase is hard to model![10]



[8] Wichern, Antognini, et al., 2019.

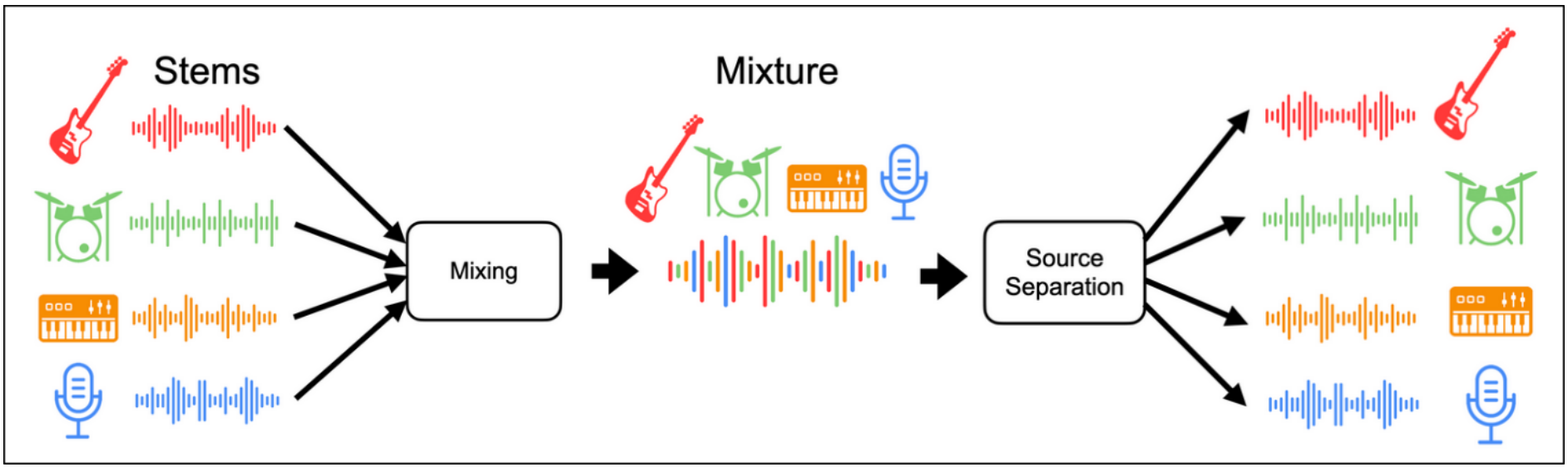
[9] Stöter, Uhlich, et al., 2019; Sawata, Uhlich, et al., 2021.

[10] `https://source-separation.github.io/tutorial/basics/phase.html#why-we-don-t-model-phase`

Music demixing (MDX) or unmixing

... considers the case where the musical audio is separated into an estimate of all of its constituent sources that can be summed back to the original mixture

Music demixing (or unmixing): estimate multiple sources (vocals, drums, bass, other[11]) that can be summed back to the original mix. Multiple MSS subproblems, reversing the linear mixing of stems in the recording studio (stem datasets can be used for mixing and demixing)



[11] Rafii, Liutkus, Stöter, Mimilakis, and Bittner, 2019.

MDX ecosystem

Evaluation measure: BSS (Blind Source Separation) metrics[12]

- **SDR:** Signal to Distortion Ratio
- **SIR:** Signal to Interference Ratio
- **SAR:** Signal to Artifacts Ratio
- **ISR:** source Image to Spatial distortion Ratio

Are these good metrics?[13], [14]
Datasets: MUSDB18-HQ[15]; stems: vocals, drums, bass, other
Campaigns: Signal Source Separation Evaluation Campaign (SiSEC) 2016, 2018[16]

---

[12] Vincent, Gribonval, et al., 2006; Vincent, Sawada, et al., 2007.

[13] Le Roux, Wisdom, et al., 2018.

[14] MDX @ ISMIR 2021 keynote: Rachel Bittner, "Source Separation Metrics: What are they measuring?"

[15] Rafii, Liutkus, Stöter, et al., 2017; Rafii, Liutkus, Stöter, Mimilakis, and Bittner, 2019.

[16] Liutkus, Stöter, et al., 2017; Stöter, Liutkus, et al., 2018.

MDX, UMX, and X-UMX

The Music Demixing Challenge was created to inspire new demixing research. Open-Unmix (UMX), and the improved variant CrossNet-Open-Unmix (X-UMX), were included in the challenge as the baselines.

MDX: Sony MDX (Music Demixing Challenge) on AICrowd for ISMIR 2021[17], used SDR to rank systems, introduced a new hidden dataset MXDB21

UMX: Open-Unmix, a near-SOTA music demixing system based on the STFT with a Bi-LSTM neural network[18]. There are four provided pre-trained models for vocals, drums, bass, other

X-UMX: CrossNet-Open-Unmix, combining four UMX networks for vocals, bass, drums, other with mixed loss functions[19]

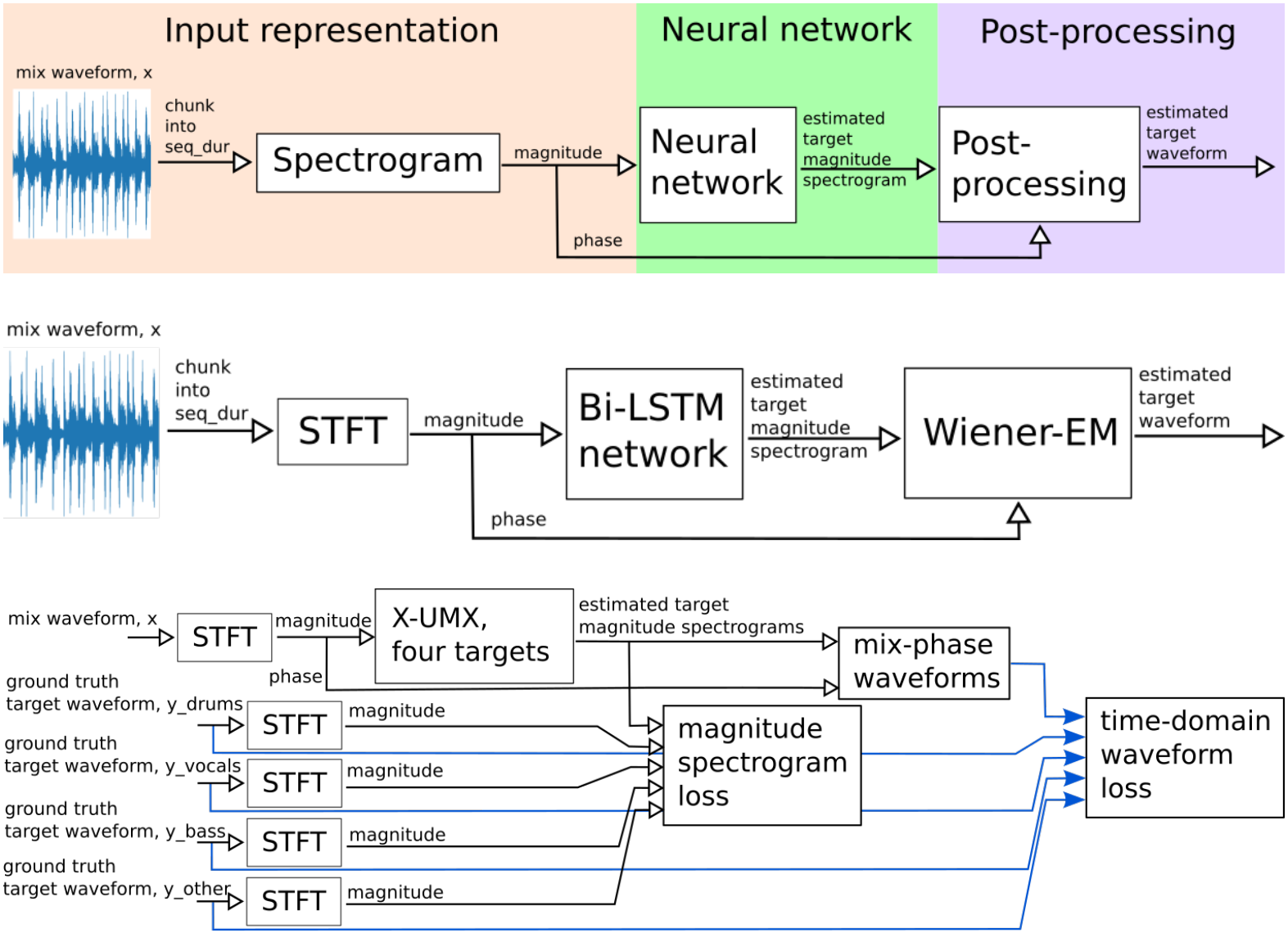
---

[17] `https://www.aicrowd.com/challenges/music-demixing-challenge-ismir-2021`

[18] Stöter, Uhlich, et al., 2019.

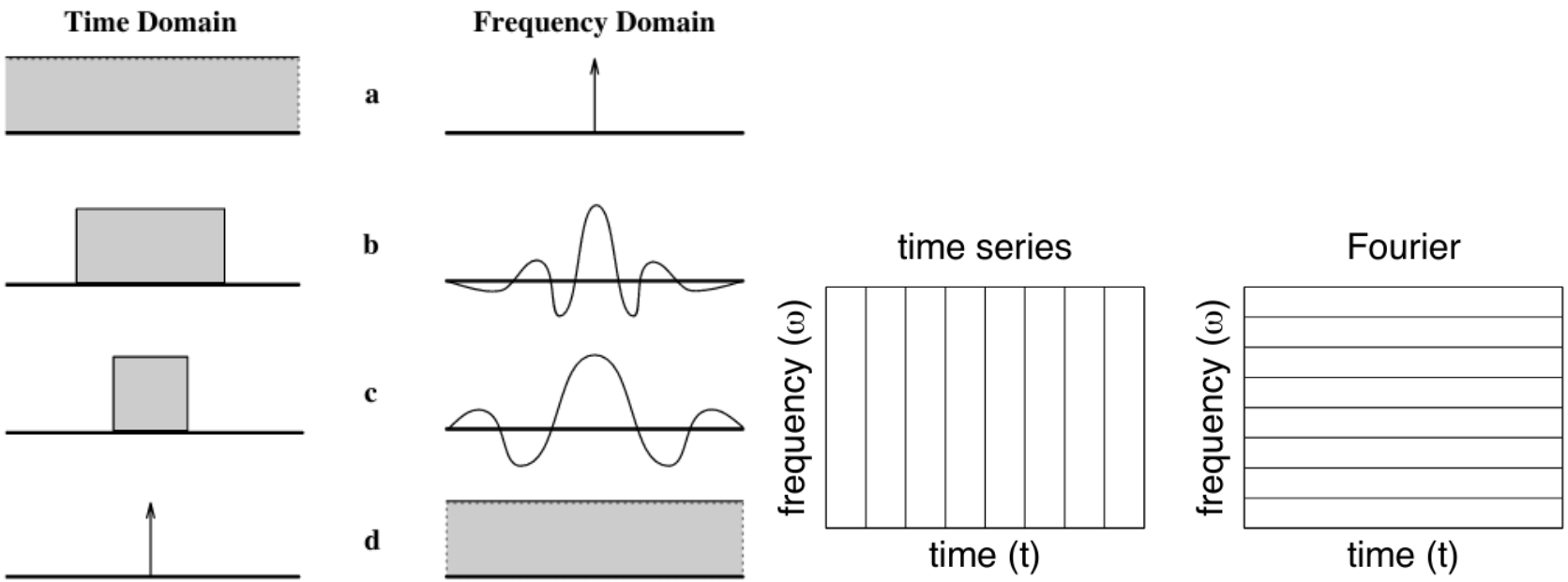
[19] Sawata, Uhlich, et al., 2021.

UMX and X-UMX


Input representation
Neural network
Post-processing
mix waveform, x
chunk into seq_dur
Spectrogram
magnitude
phase
Neural network
estimated target magnitude spectrogram
Post-processing
estimated target waveform
mix waveform, x
chunk into seq_dur
STFT
magnitude
phase
Bi-LSTM network
estimated target magnitude spectrogram
Wiener-EM
estimated target waveform
mix waveform, x
STFT
magnitude
X-UMX, four targets
phase
estimated target magnitude spectrograms
mix-phase waveforms
ground truth target waveform, y_drums
STFT
magnitude
ground truth target waveform, y_vocals
STFT
magnitude
ground truth target waveform, y_bass
STFT
magnitude
ground truth target waveform, y_other
STFT
magnitude
magnitude spectrogram loss
time-domain waveform loss

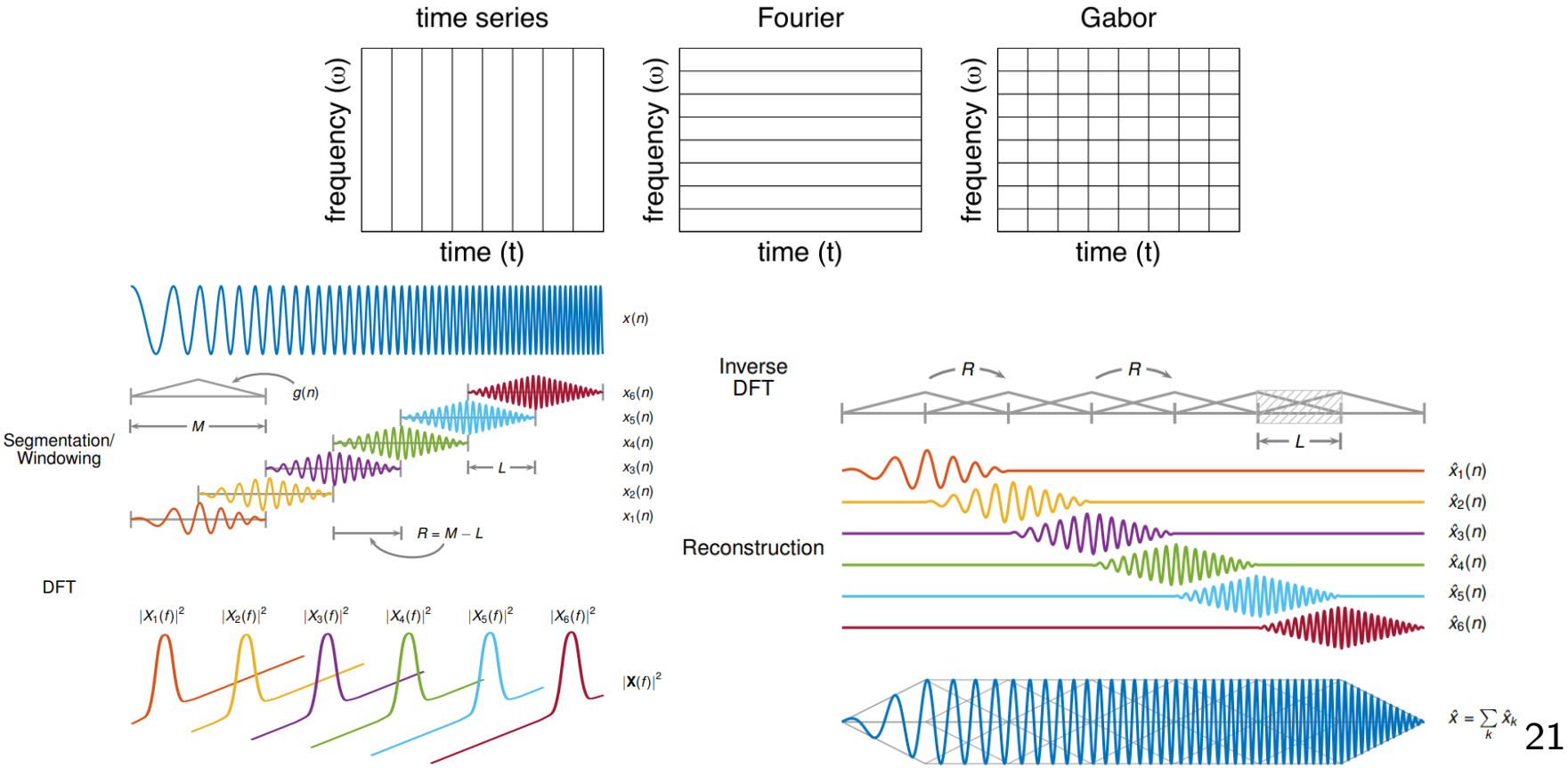
Time-frequency uncertainty principle

The Fourier transform decomposes a signal into a sum of **infinite** sinusoids: no temporal information
Time and frequency are orthogonal domains (opposites of the Fourier transform), like the position and momentum of an electron



Joint time-frequency analysis, the STFT, and the spectrogram

Joint time-frequency analysis is important for signals whose frequencies change with time[20] Take Fourier transform of local windows of the signal



21

[20] Gabor, 1946.

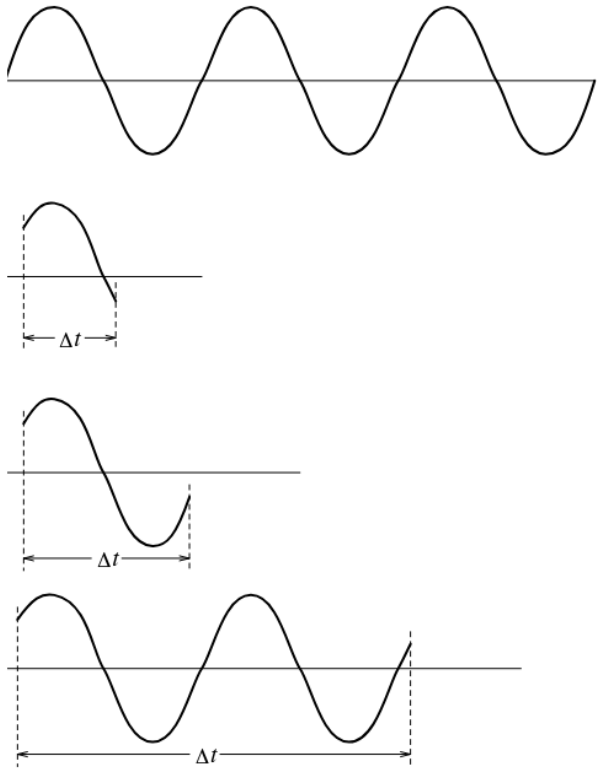
[21] https://www.mathworks.com/help/signal/ref/iscola.html

The time-frequency uncertainty principle

... states that the STFT of a signal cannot have maximal resolution in both time and frequency

Time-frequency uncertainty[22]:

> *although we can carry out the analysis with any degree of accuracy in the time direction or frequency direction, we cannot carry it out simultaneously in both beyond a certain limit*



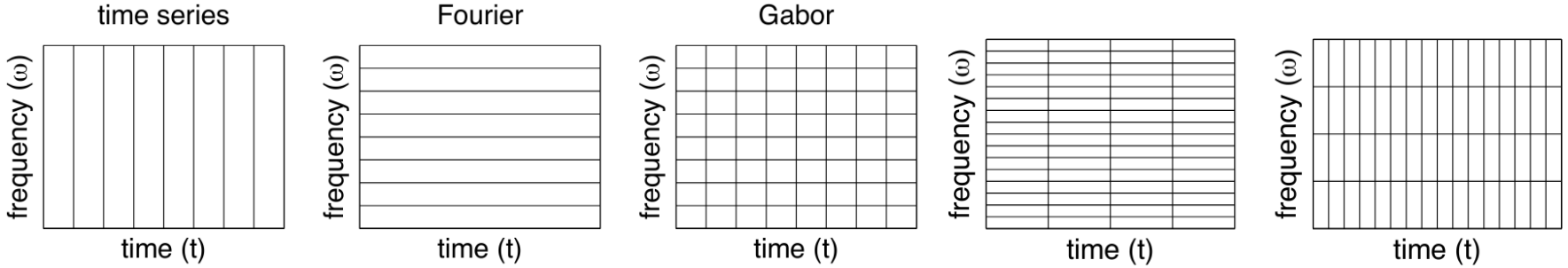
[22] Gabor, 1946.

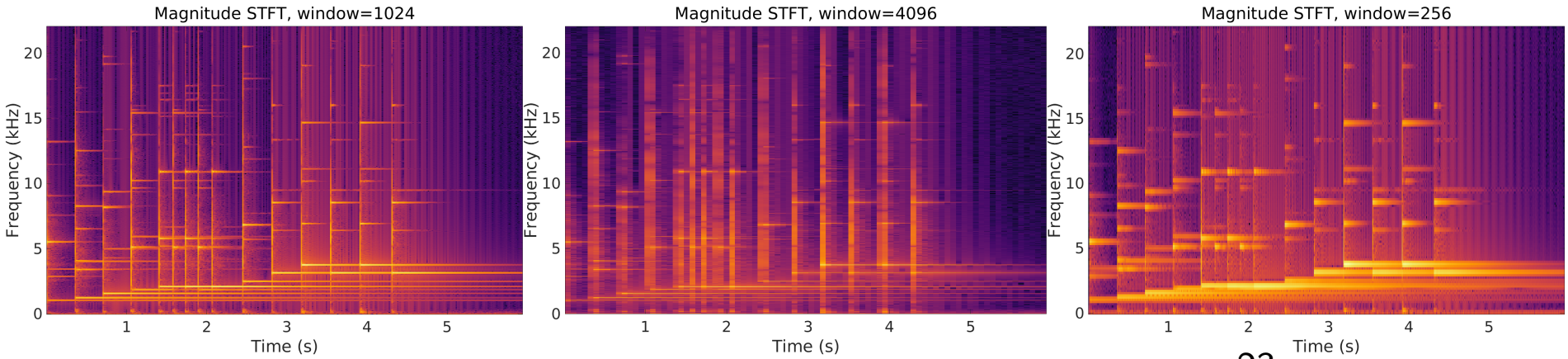
Time-frequency tradeoff

... can significantly affect music demixing results

Time-frequency tiles are constrained to a minimum area: $\Delta t \Delta f \geq 1$

Change window size to trade off time and frequency:





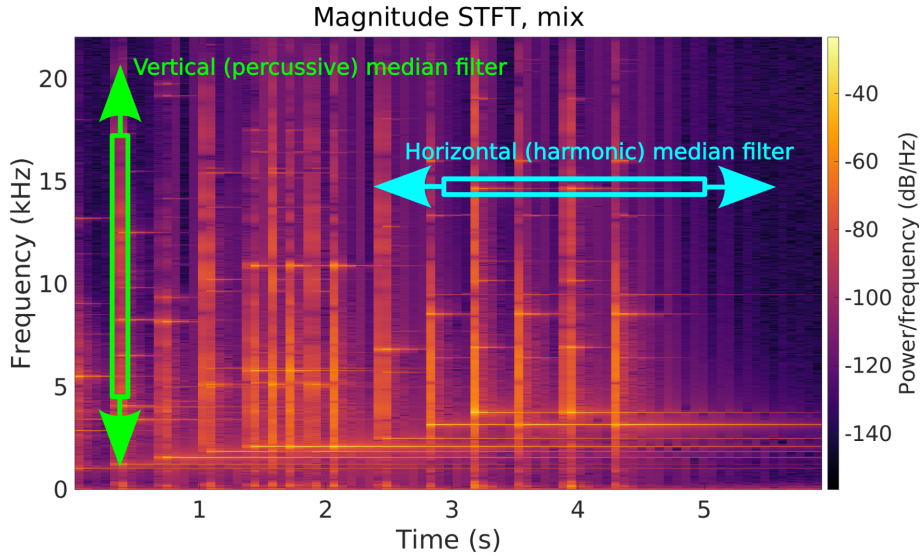
In music source separation, window size matters per-target.[23]

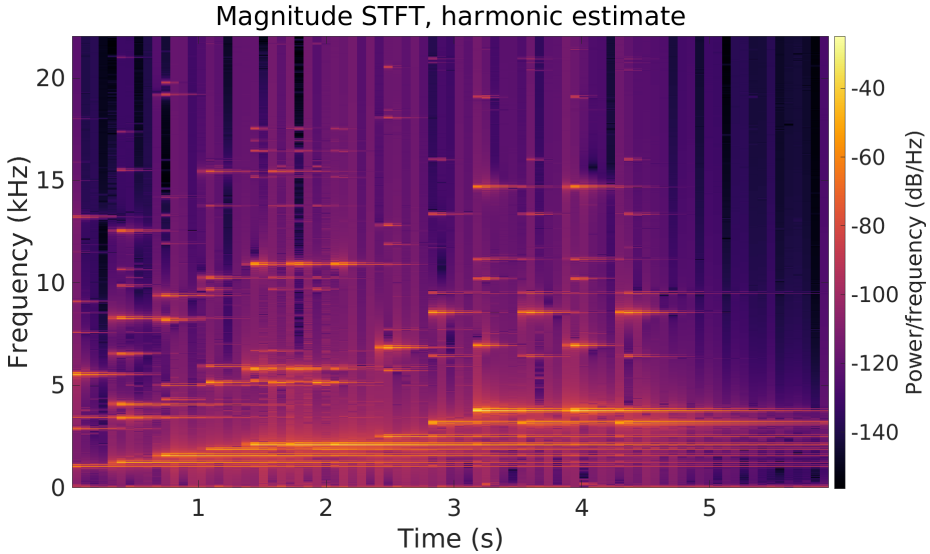
Short-window for percussion, long-window for harmonic

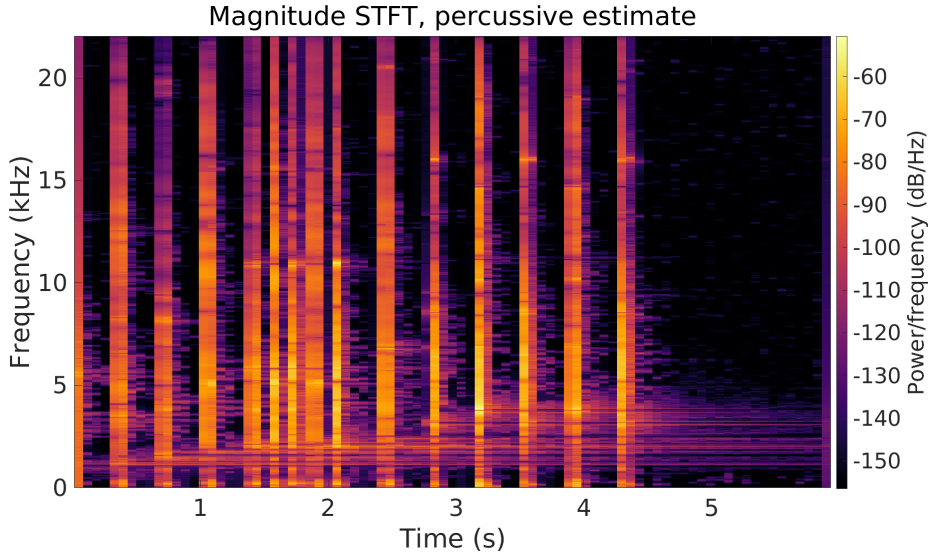
[23] Simpson, 2015; Kavalerov, Wisdom, et al., 2019.

Time-frequency tradeoff: HPSS case study

Harmonic/percussive source separation with median filters[24]







[24] Fitzgerald, 2010.

Improving harmonic/percussive source separation (HPSS)

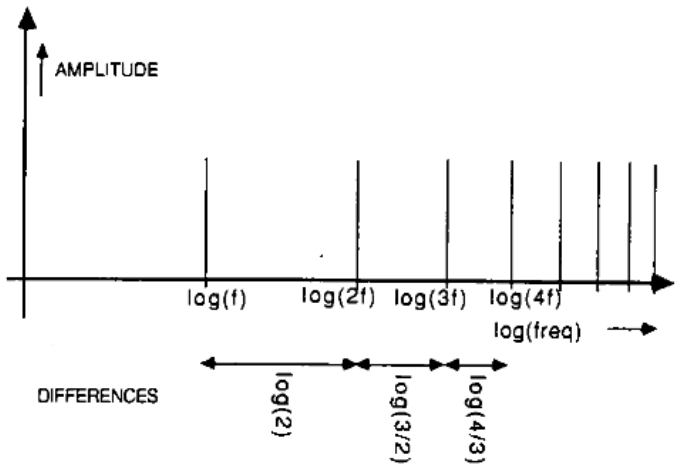
1. From musical and auditory aspects, frequency resolution should increase from high to low frequencies (vice-versa for time resolution)[25]
2. Use long windows/$\uparrow \Delta f$ in low frequencies, and short windows/$\uparrow \Delta t$ in high frequencies to analyze music (harmonic basis and transients)[26]
3. window=4096 for harmonic, window=256 for percussive in HPSS[27]
4. Constant-Q Transform (CQT) and multiple STFTs (16384 for harmonic, 1024 for percussive) in HPSS[28]
5. CQT[29] uses long windows in low frequencies and short windows in high frequencies for the 12-tone Western pitch scale

[25] Schörkhuber, Klapuri, et al., 2012.
[26] Dörfler, 2002.
[27] Driedger, Müller, et al., 2014.
[28] Fitzgerald and Gainza, 2010.
[29] Brown, 1991; Schörkhuber and Klapuri, 2010.

Constant-Q Transform

Constant-Q transform for music analysis[30]:

1. Harmonics of the fundamental have consistent spacing in the log scale – the constant pattern



2. Log-frequency spectra, demonstrating the constant pattern for harmonics, would be more useful in musical tasks

[30] Brown, 1991; Brown and Puckette, 1992.

Constant-Q Transform

"Constant ratio of frequency to frequency resolution": $\frac{f}{\delta f} = Q$

	Constant Q	DFT
Frequency	$(2^{1/24})^k \cdot f\min$ exponential in k	$k\,\Delta f$ linear in k
Window	variable $= \mathrm{N[k]} = \frac{SR \cdot Q}{f_k}$	constant $= N$
Resolution		
Δf	variable $= f_k/Q$	constant $= \mathrm{SR/N}$
$\frac{f_k}{\Delta f_k}$	constant $= Q$	variable $= k$
Cycles in Window	constant $= Q$	variable $= \mathrm{k}$

(a) Properties of DFT, CQT

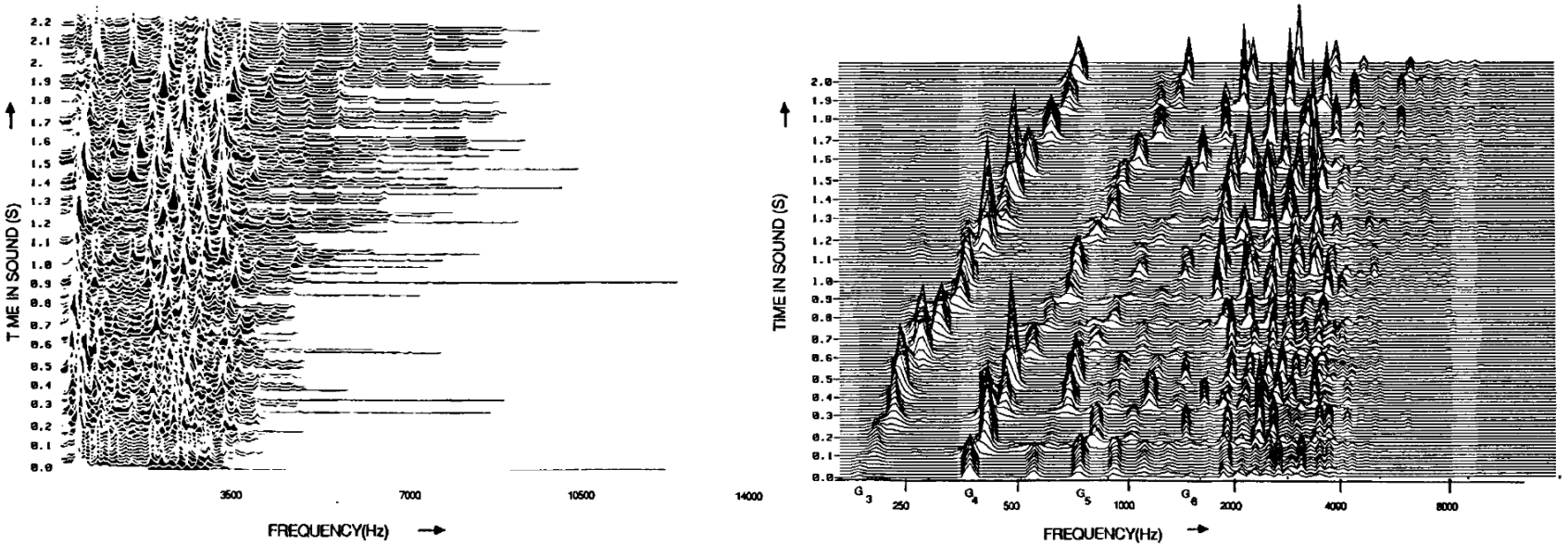
Channel	Midinote	Frequency (Hz)	Window (Samples)	(ms)
0	53	175	6231	195
6	56	208	5239	164
12	59	247	4406	138
18	62	294	3705	116
24	65	349	3115	97
30	68	415	2619	82
36	71	494	2203	69
42	74	587	1852	58
48	77	699	1557	49
54	80	831	1309	41
60	83	988	1101	34
66	86	1175	926	29
72	89	1398	778	24
78	92	1664	1308	41
84	95	1978	1100	34
90	98	2350	926	29
96	101	2797	778	24
102	104	3327	654	20
108	107	3956	550	17
114	110	4710	462	14
120	113	5608	388	12
126	116	6675	326	10
132	119	7942	274	9
138	122	9461	230	7
144	125	11 216	194	6
150	128	13 432	162	5

(b) Window sizes for CQT

Non-invertible; approximate inverse introduced[31]

[31] Schörkhuber and Klapuri, 2010.

Constant-Q Transform



(a) Discrete Fourier Transform (b) Constant Q transform

Figure: Violin playing diatonic scale, $G_3(196\text{Hz}) - G_5(784\text{Hz})$[32]

[32] Brown, 1991.

NSGT and sliCQT

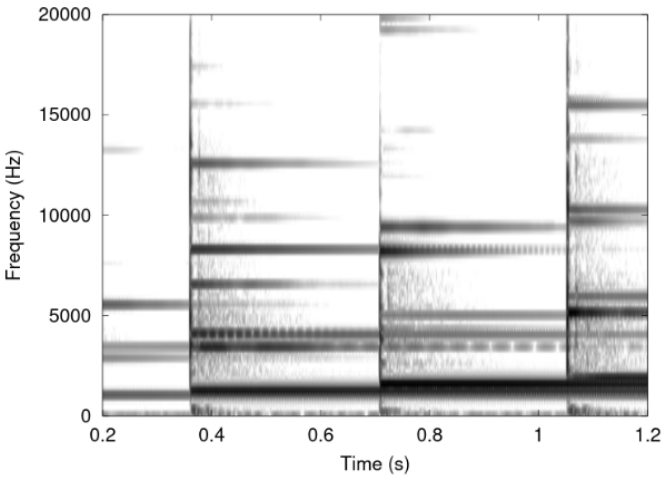
STFT = "stationary" Gabor transform: use the same window, suffering from a fixed time-frequency resolution
Nonstationary Gabor Transform (NSGT)[33], realtime sliCQ Transform[34] = "nonstationary" Gabor transform: use different window sizes to vary the time-frequency resolution

1. STFT-like transforms with windows that vary with time
2. CQT motivates the NSGT/sliCQ, but can use any monotonically increasing frequency scale (log/cq, mel, Bark, etc.)
3. Outputs the familiar Fourier coefficients with **perfect inverse**
4. CQT implemented with NSGT/sliCQT = CQ-NSGT

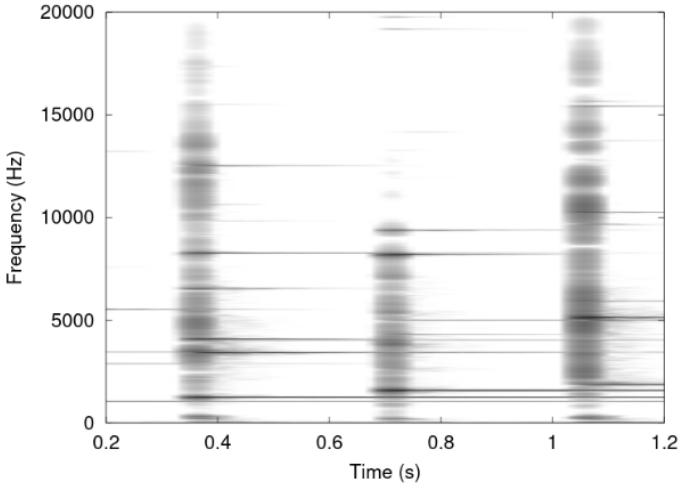
[33] Balazs, Doerfler, et al., 2011.

[34] Velasco, Holighaus, et al., 2011; Holighaus, Dörfler, et al., 2013; Schörkhuber, Klapuri, et al., 2014.

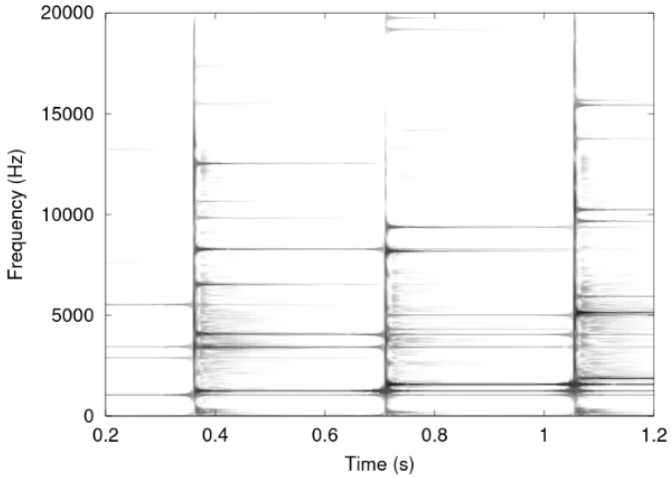
NSGT demo



(a) 6ms STFT

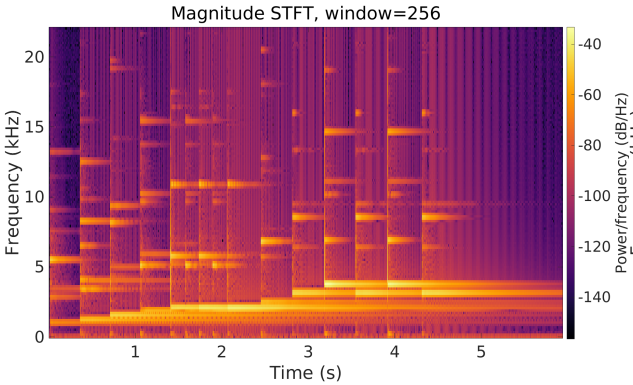


(b) 93ms STFT

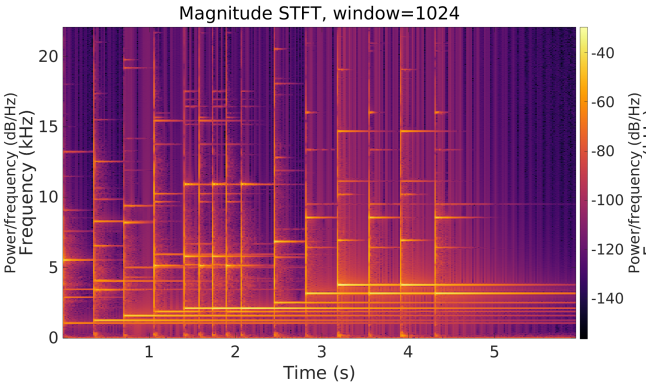


(c) 6–93ms NSGT

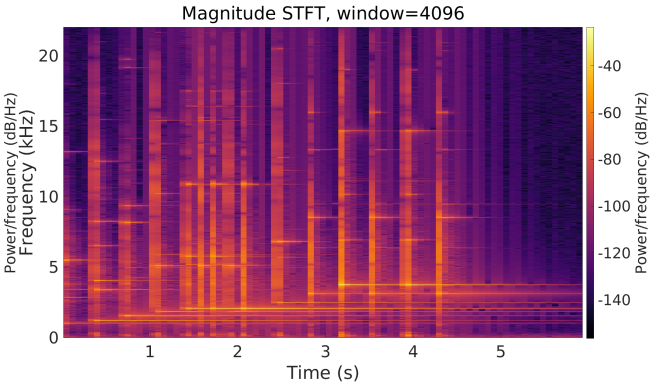
STFT vs. CQ-NSGT



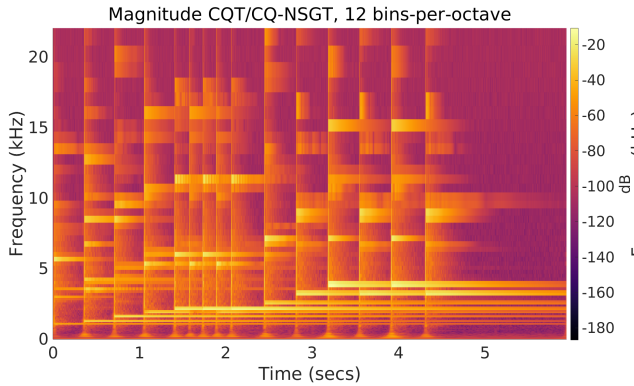
(a) STFT, window = 256



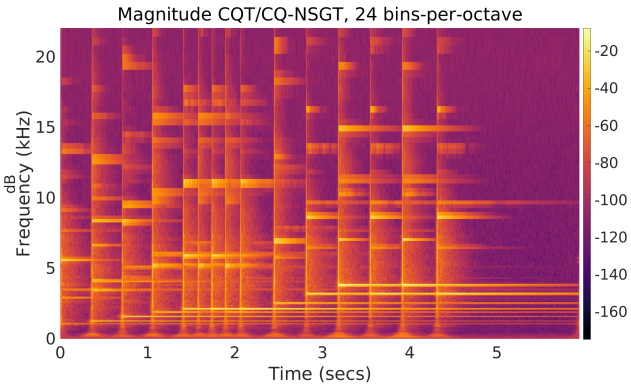
(b) STFT, window = 1024



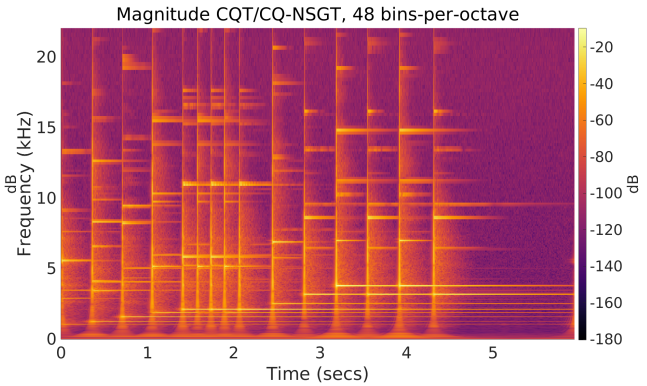
(c) STFT, window = 4096



(d) CQT, 12 bins/octave



(e) CQT, 24 bins/octave



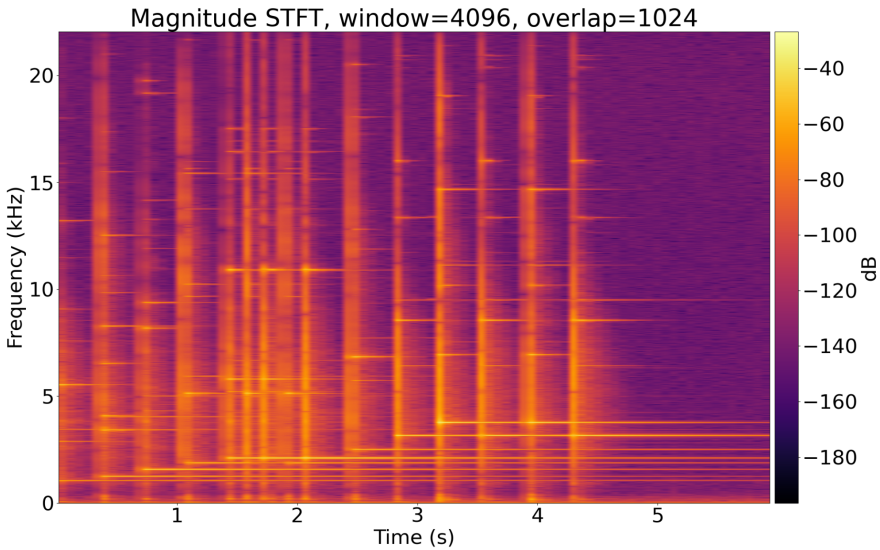
(f) CQT, 48 bins/octave

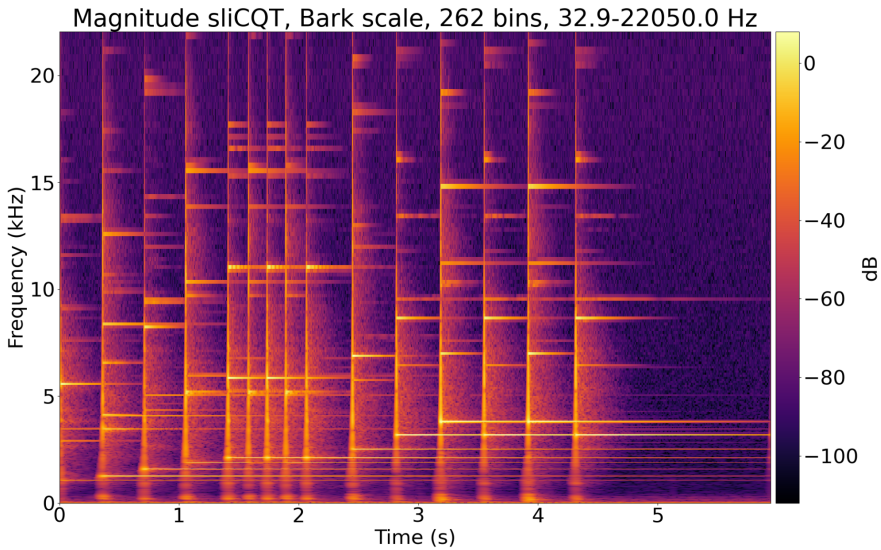
xumx-sliCQ: spectral transform

Our proposed adaptation of UMX replaced the STFT with the sliCQT, a time-frequency transform with varying time-frequency resolution

Hypothesis: improve Open-Unmix by using sliCQT with varying time-frequency resolution. sliCQT demonstrates good tonal/transient representation, and displays more musical information than the STFT

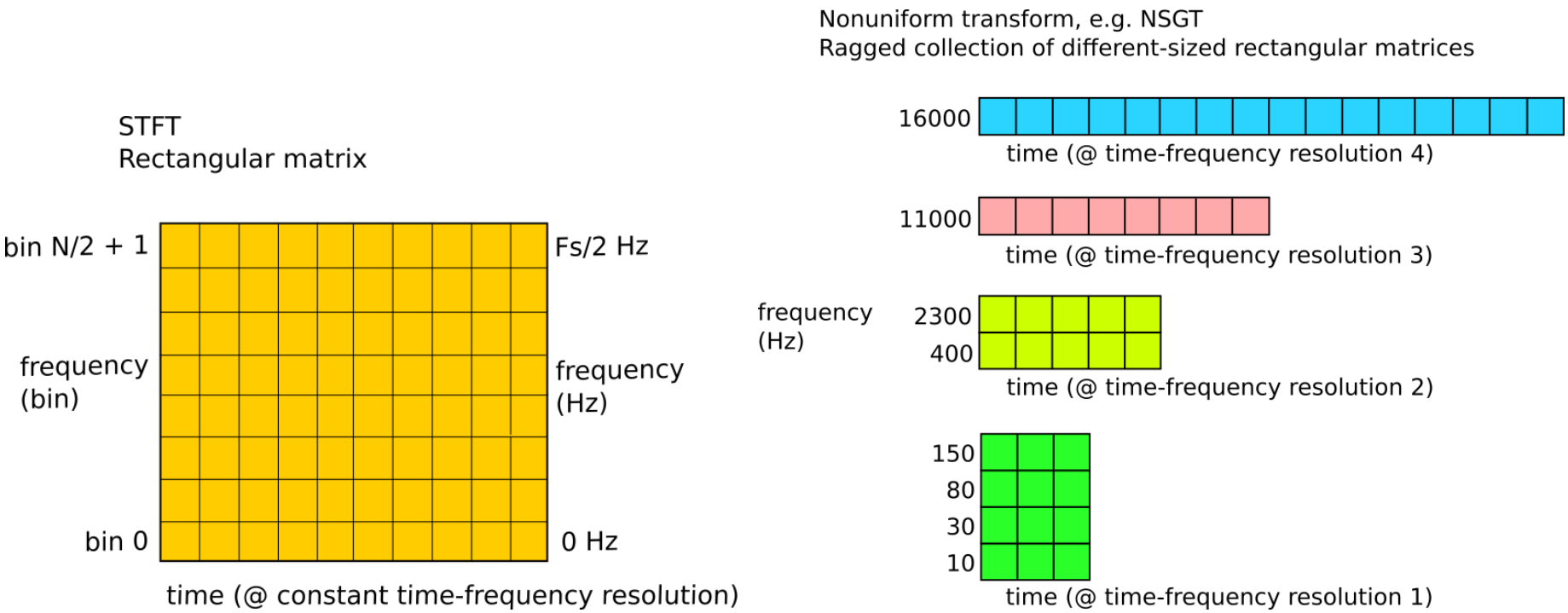
Choose sliCQT params by maximizing SDR of "noisy phase" oracle: $\hat{X}_{\text{target}} = |X_{\text{target}}| \cdot \angle X_{\text{mix}}$; **7.42 dB** vs. 6.23 dB of STFT-4096 on MUSDB18-HQ validation set





Ragged shape of the sliCQT

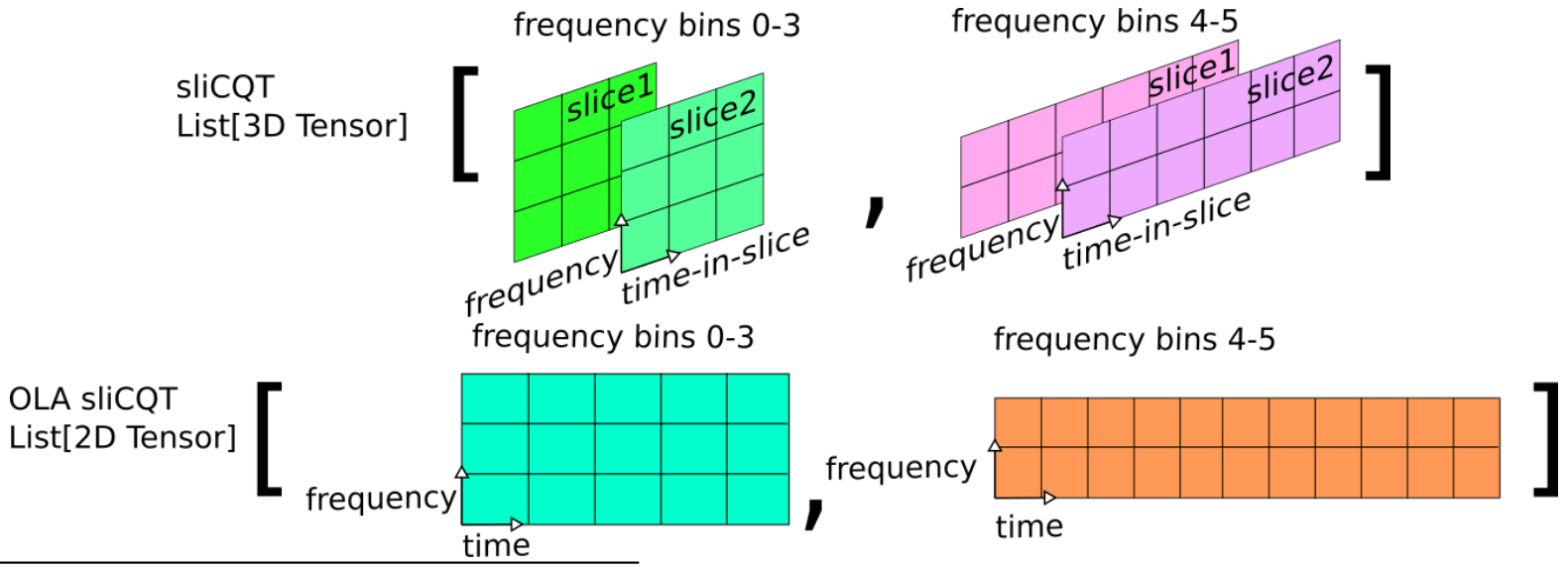
sliCQT output: list of complex 2D Time $\times$ Frequency tensors of Fourier coefficients, bucketed by time resolution. Different temporal frame rate per bucket



3D shape of the sliCQT

sliCQT returns coefficients for sliced input signal: "slicing windows are symmetrically zero-padded to length 2N , reducing time-aliasing significantly"[35]. Overlap-add each slice by 50%:

> *Displaying the framewise transform is slightly more tricky as we have to overlap-add the spectrograms obtained for each frame... Note that it is not possible to synthesize the audio from this overlapped version as we cannot retrieve the analysis frames from it.*[36]

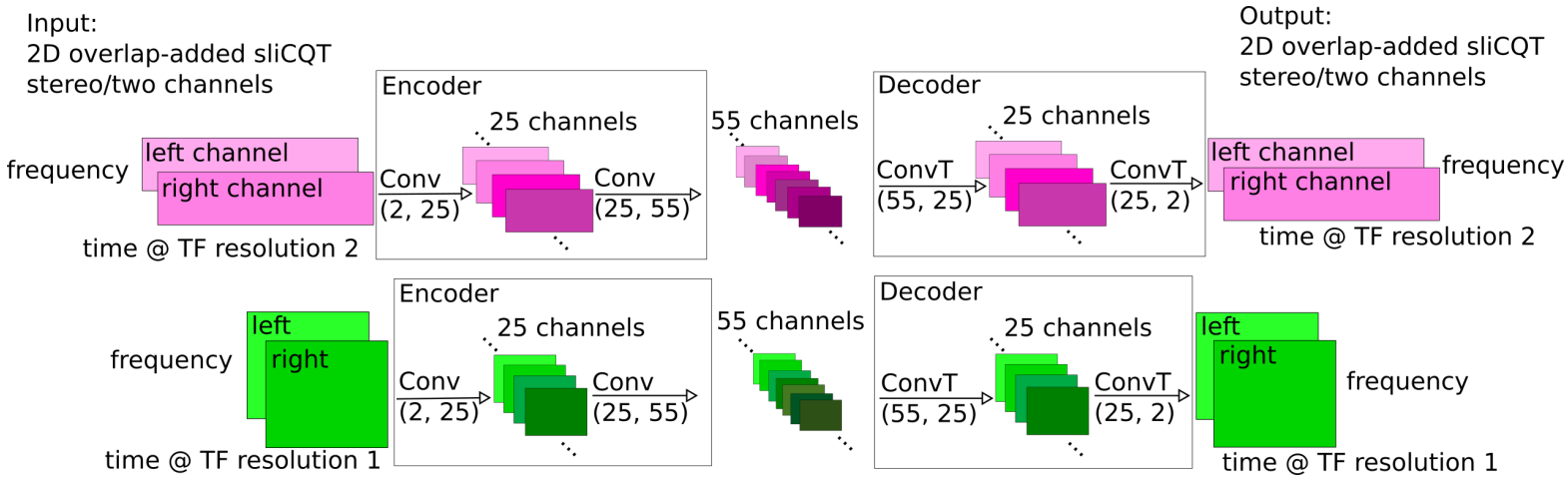


[35] Holighaus, Dörfler, et al., 2013, p. 10.

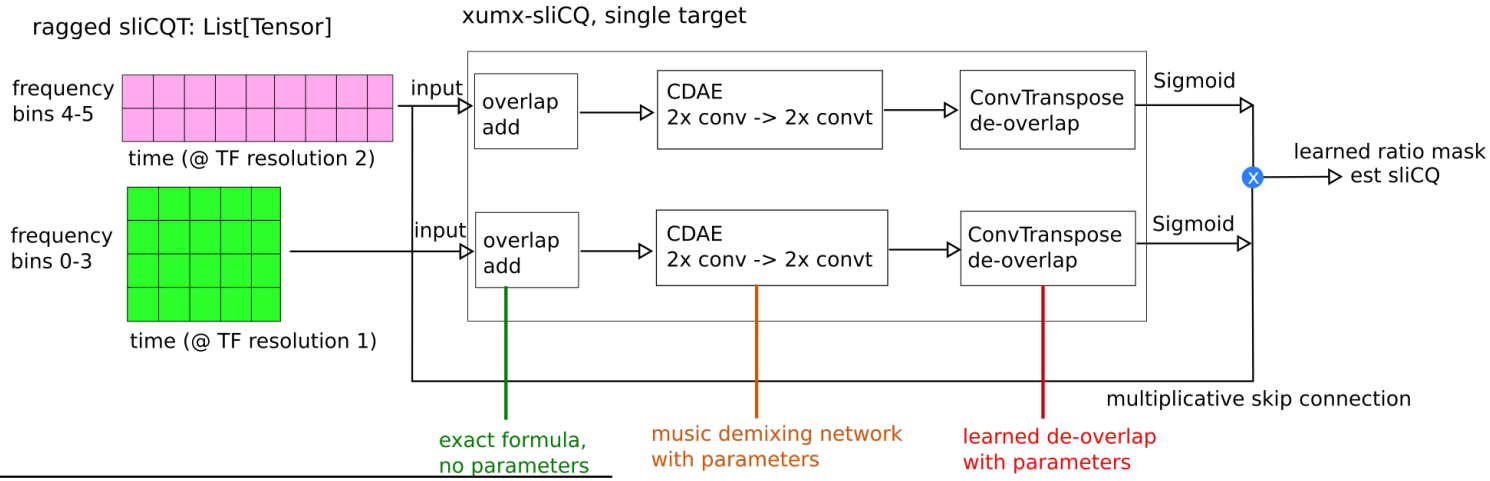
[36] `https://mtg.github.io/essentia-labs/news/2019/02/07/invertible-constant-q/`

xumx-sliCQ: CDAE network architecture

Use Convolutional Denoising Autoencoder (CDAE)[37] neural architecture, applied to each matrix of the ragged overlap-added sliCQT separately



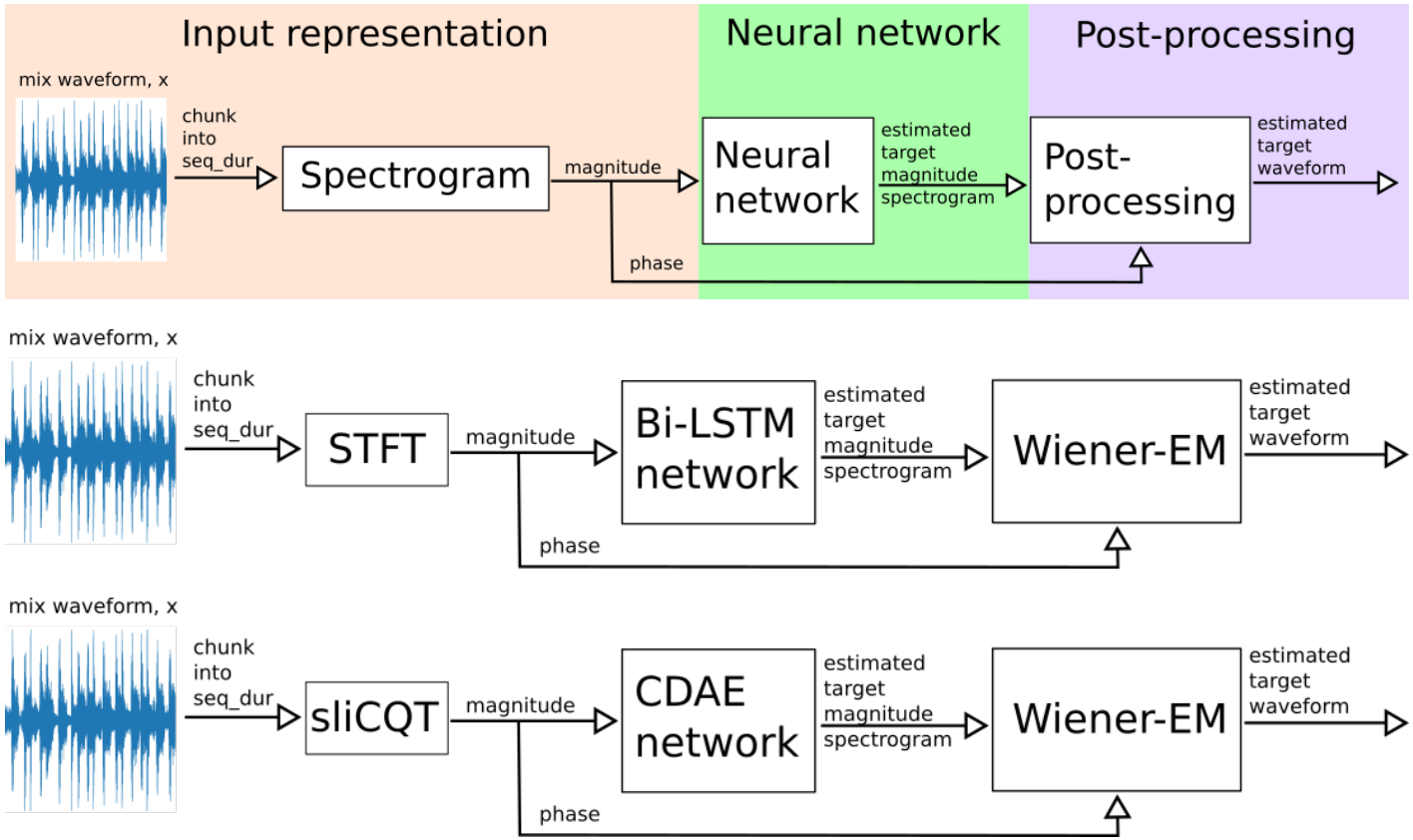
Introduce an extra layer for "de-overlap"



[37] Grais and Plumbley, 2017; Grais, Zhao, et al., 2021.

xumx-sliCQ: block diagrams, results

1. My goal: improve Open-Unmix by replacing STFT with sliCQT
2. Scored 3.6 dB vs. 4.6 dB (UMX) and 5.54 dB (X-UMX); there is still room for improvement



MDX 21 winners, current trends in demixing

1. Previously, music demixing systems were submitted to and evaluated at SiSEC (Signal Separation Evaluation Campaign). This year: MDX (Music Demixing Challenge) ISMIR 2021 @ AICrowd, follow-up MDX21 workshop, satellite @ ISMIR 2021
2. **ISMIR 2021**: Model that uses the complex spectrogram (i.e. includes phase) and uses complex masks[38]
3. **MDX21**: 1: Demucs++[39] (waveforms + complex spectrogram), 2: KUIELAB-MDX-Net[40] (waveforms + magnitude spectrogram), 3: Danna-Sep[41] (waveform + magnitude spectrogram, use complex spectrogram in loss function)

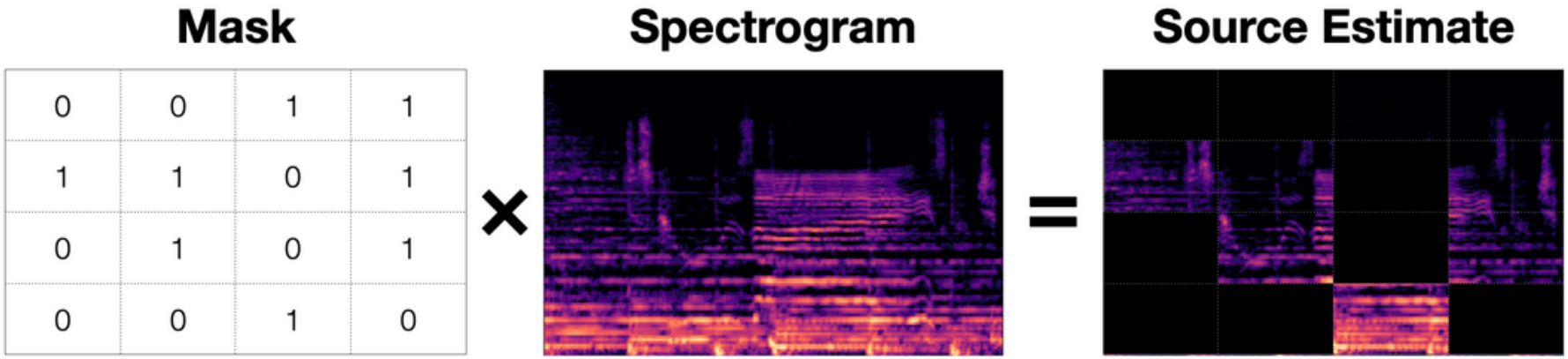
Properties in common: blending networks, waveforms (implicitly includes phase), complex spectrograms/masks, mixing spectrogram and waveform models

[38] Kong, Cao, et al., 2021.
[39] Défossez, 2021.
[40] Kim, Choi, et al., 2021.
[41] Yu and Cheuk, 2021.

Magnitude mask above 1

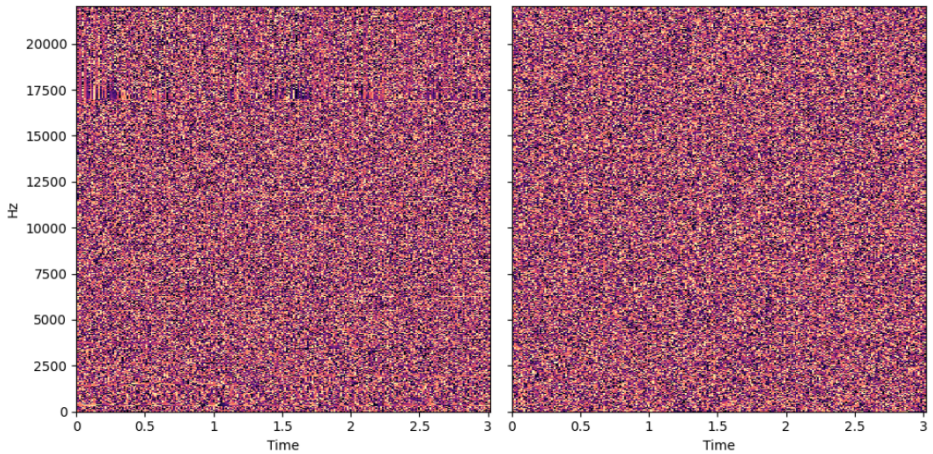
Common approaches to music source separation (MSS):

1. Get spectrogram of mix
2. Take magnitude
3. Multiply by a mask $\in [0, 1]$ to get source estimate
4. Why $[0, 1]$? DFT/STFT is a linear operation:
   $x_a = x_b + x_c, |X_a| = |X_b| + |X_c|$
   $|X_b| = M_b (\in [0, 1]) \times |X_a|$
   if M_b (i.e., Mask of source b) > 1, then $|X_b| > |X_a|$?



Phase!

Common approaches to MSS discard the phase; it's difficult to learn relationships from phase



This paper considers the phase, and uses a complex mask to estimate the magnitude and phase of the spectrogram

$|X_b| = M_b(\in [0,1]) \times |X_a|$

if M_b (i.e., Mask of source b) > 1, then $|X_b| > |X_a|$? **Yes!**

|M(t,f)| can be larger than 1... this may happen when S(t,f) and N(t,f) are out of phase, since that makes the magnitude of mixture to be smaller than that of (individual) signal

Sound samples

"Winners" of MDX21 (didn't release their code/data, so they didn't get a prize; proprietary company):
`https://www.youtube.com/watch?v=fNgIXBErUMI`