M2 Data Science

DS-telecom-15 "Audio and Music Information Retrieval"

Geoffroy Peeters

contact: geoffroy.peeters@telecom-paris.fr
Télécom-Paris, IP-Paris, France

Deep Learning For Audio: **which input representations ?**

Deep Learning For Audio: **T/F input representations**

SPEECH: T/F input representations

- **1990 →Time-Delay Neural Network (TDNN)**
  - similar to a 1-D convolution operating only over time, no convolution are performed over the frequency axis.
  - convolution is applied to a Mel-gram (16 normalized Mel-scale spectral coefficients)
- **2009 →Convolutional Deep Belief Networks (CDBN)**
  - input = 160 dimensional spectrogram which is then PCA-whitened to 80 dimensions
  - filters (named bases in [LPLN09]) of the first and second layers are of length 6 and are convolved over the PCA-whitened spectrogram.
  - visual comparison: the learned filters (bases) are related to the different phonemes of speech.
- **2012 →Seminal paper**
  - new baseline for speech recognition system = DNN-HMM model
  - acoustic model part of the system is defined as a DNN model (stacked RBMs).

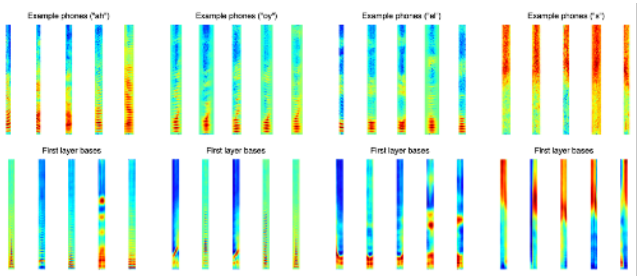
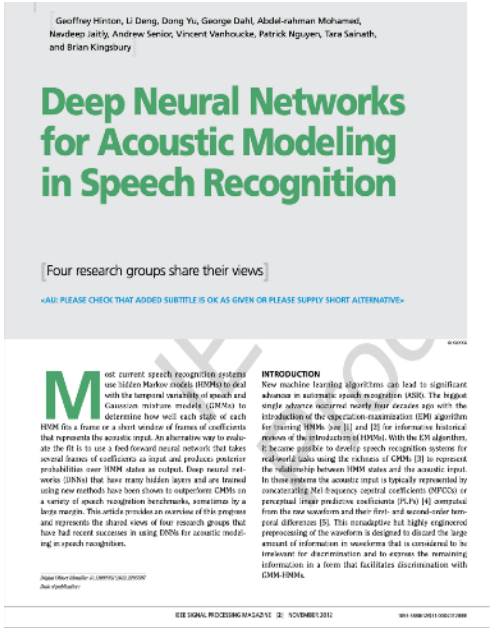


Figure 2: Visualization of the four different phonemes and their corresponding first-layer CDBN bases. For each phoneme: (top) the spectrograms of the five randomly selected phones; (bottom) five first-layer bases with the highest average activations on the given phoneme.

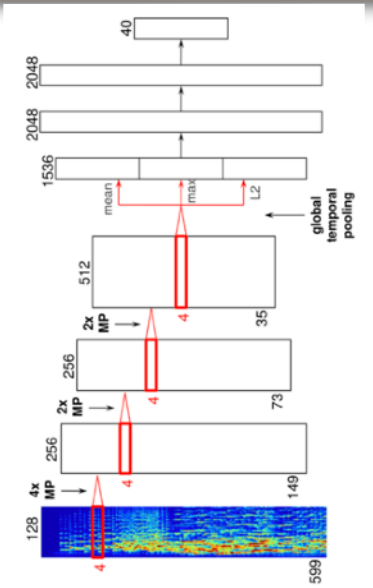


[Alexander Waibel et al. "Phoneme recognition using time-delay neural networks." In Readings in speech recognition, 1990]
[Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. "Unsupervised feature learning for audio classification using Conv DBN". NeurIPS, 2009]
[Geoffrey Hinton et al. "Deep neural networks for acoustic modeling in speech recognition". IEEE Signal processing magazine, 2012]

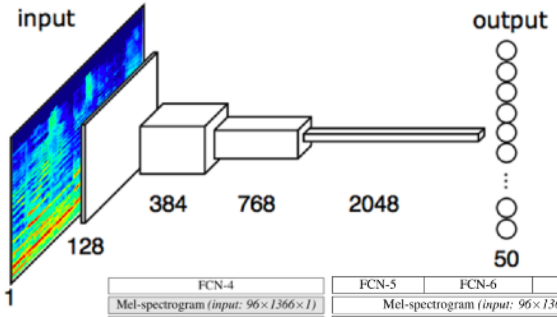
Deep Learning For Audio: **T/F input representations**

MUSIC: T/F input representations

- **2014 → Dieleman**
  - Task: predicting latent representation of music tracks (a regression problem)
  - Model: 1D-convolution operating only over time.
  - Input: Mel-Spectrogram (MS) of 128 frequency bins



- **2016 → Choi**
  - Task: music auto-tagging
  - Model: VGG- Net [SZ15], i.e. deep stack of layers with small (3,3) filters convolved over time and freq.
    - consider time/frequency representation as natural images and apply a computer vision CNN to it.



FCN-4
Mel-spectrogram *(input: 96×1366×1)*
Conv 3×3×128
MP (2, 4) *(output: 48×341×128)*
Conv 3×3×384
MP (4, 5) *(output: 24×85×384)*
Conv 3×3×768
MP (3, 8) *(output: 12×21×768)*
Conv 3×3×2048
MP (4, 8) *(output: 1×1×2048)*
Output 50×1 (sigmoid)

FCN-5	FCN-6	FCN-7
Mel-spectrogram *(input: 96×1366×1)*		
Conv 3×3×128		
MP (2, 4) *(output: 48×341×128)*		
Conv 3×3×256		
MP (2, 4) *(output: 24×85×256)*		
Conv 3×3×512		
MP (2, 4) *(output: 12×21×512)*		
Conv 3×3×1024		
MP (3, 5) *(output: 4×4×1024)*		
Conv 3×3×2048		
MP (4, 4) *(output: 1×1×2048)*		
	Conv 1×1×1024	Conv 1×1×1024
		Conv 1×1×1024
Output 50×1 (sigmoid)		

- **WARNING: time/frequency representations cannot be considered as a natural image**

[Sander Dieleman. "Recommending music on spotify with deep learning. Technical report", http://benanne.github.io/2014/08/05/spotify-cnns.html, 2014]
[Keunwoo Choi, Gyorgy Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. In Proc. of ISMIR, 2016]

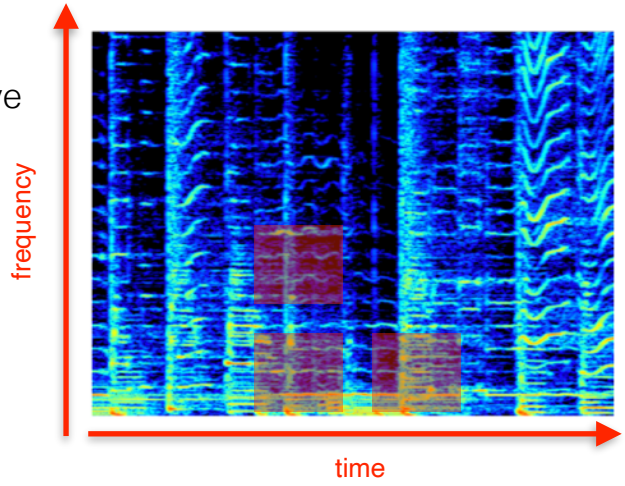
Input representation: 2D (time/frequency)

T/F images ≠ natural images ⇒ need to adapt the ConvNet architecture

– **Natural images**
- (a) the two axis x and y
  - represent the same concept (spatial position)
- (b) the elements of an image have
  - the same meaning independently of their positions over x and y.
- (c) neighboring pixels:
  - usually highly correlated,
  - often belong to the same object

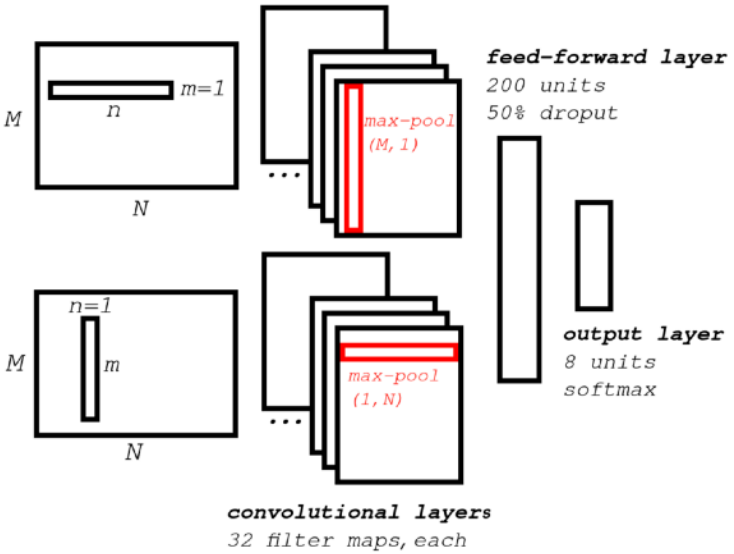
– **Time-frequency audio representations**
- (a) the two axis x and y
  - represent profoundly different concepts (time and frequency).
- (b) the elements of spectrogram (such as the T/F area of a source) have
  - the same meaning independently of its position over time
  - but not over frequency

    ⇒ no invariance over y, even in the case of log-frequencies
- (c) neighboring pixels:
  - are not necessarily correlated
  - a given sound source (such has an harmonic sound) can be distributed over the whole frequency in a sparse way (the harmonics of a given sound can be spread over the whole



MUSIC: T/F input representations

- **2013 → Schluter**
  - task of onset detection (detecting the start of a musical events)
  - model: CNN with carefully design filters to highlight mid-duration variations over small-frequency ranges
  - input: multi-scale analysis, i.e. STFT computed using various window durations (23 ms, 46 ms and 93 ms) →use the depth of the input layer
- **2016 → Pons**
  - musically- motivated filter design
    - shapes of the CNN filters carefully chosen to allow representing the timbre (vertical filters extending over the frequency axis) or the rhythm (horizontal filters extending over the time axis) content of a music track.
    - equivalent performances than the CV-based approach ("black-box") but with much less parameters.



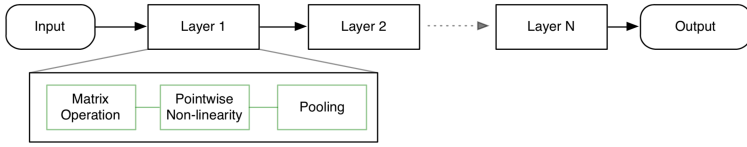
[Jan Schluter and Sebastian Böck. "Musical onset detection with convolutional neural networks". In 6th International Workshop on MLM, 2013]
[Jordi Pons, Thomas Lidy, and Xavier Serra. "Experimenting with musically motivated convolutional neural networks". In Proc. of IEEE CBMI, 2016]

Deep Learning For Audio: **learning everything, waveform input**

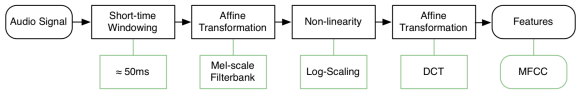
2012 → Humphrey et al. manifesto

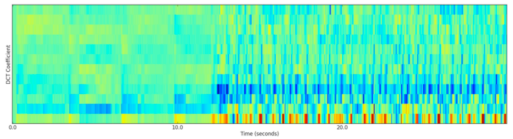
- *a DNN is "a cascade of multiple layers, composed of a few simple operations: affine transforms, point-wise non-linearities and pooling operators. Cascaded non-linearities allow for complex systems composed of simple, linear parts. Music is composed by hierarchies of pitch and loudness forming chords and melodies, phrases and sections, eventually building entirely pieces. Deep structures are well suited to encode these relationships."*
- A large part of the most-commonly used audio signal processing algorithms can be formulated as a cascade of affine transformations, non-linear transforms and pooling operations.
- Example
  - Discrete-Fourier-Transform (DFT)
  - MFCC or Chroma
  - Principal Component Analysis (PCA) or Non Negative Matrix Factorization (NMF)
- The only difference between these transformations lies in their parameterization. DL hence provides a convenient framework to learn such parametrization from the data.

Deep Neural Network

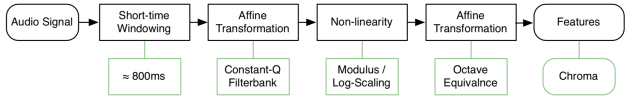


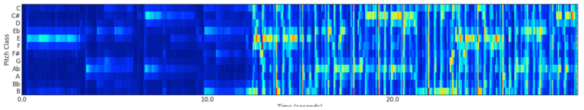
MFCC flowchart





Chroma flowchart





[Humphrey, Bello and LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics". In ISMIR, 2012.]

Deep Learning For Audio: **learning everything, waveform input**

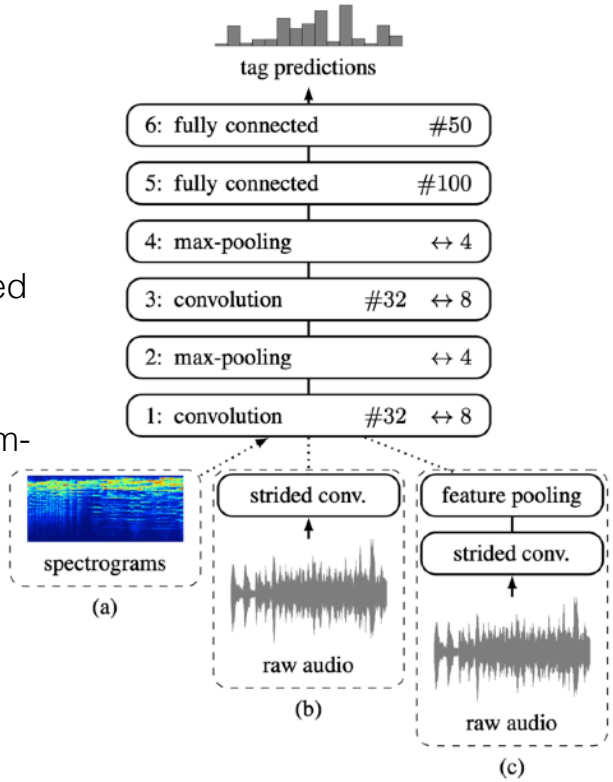
1D-Convolution on the audio waveform

– **2014 → Dieleman:**

- first end-to-end approaches for music auto-tagging,
- use 1D-convolution on the waveform to replace spectrogram input
- reproduce the computation of the spectrogram
  - spectrogram computed using a succession of DFTs computed on a frame of length N and each separated by a hop size S
  - 1D-convolution with 1D-filters of length N and a stride of S
- "end-to-end" approach under-performed the traditional spectrogram-based approach

– **Time Translation Invariance (TTI) property**

- transform insensitive to time translation (or phase shift) of the input.
- amplitude of the DFT (as used in the spectrogram) is TTI
- Mimicking this property with 1D-convolution would require
  - (a) reducing the stride to S=1 (and a huge sampling rate) or
  - (b) having a different 1D- filter for each possible translation.
  - still needs to perform max-pooling over time for (a) over filters for (b).
  - → computationally prohibitive

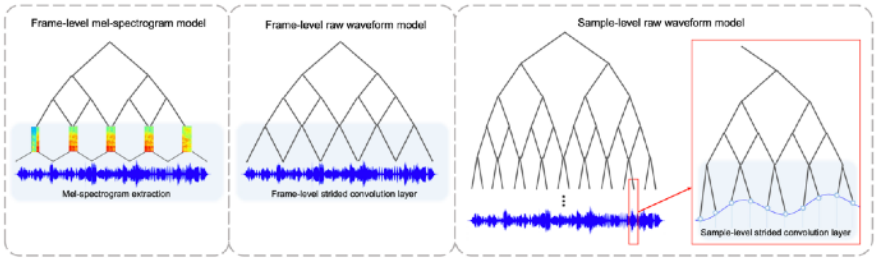


[Sander Dieleman and Benjamin Schrauwen, "End-to-end learning for music audio", In ICASSP, 2014.]

Deep Learning For Audio: **learning everything, waveform input**

2017 → Sample-CNN (1D-convolution on the audio waveform)

- **Idea**:
  - to improve the TTI reduce the size of the 1D-convolution filters (hence also the stride).
  - If the filters are smaller, then the number of time translation to be learned is also reduced
- **2017 → Sample-CNN**
  - a deep stack of 1D-convolution of small (3,1) filters applied to the waveform.
  - equivalent to the VGG-Net for 1D-convolution applied to waveforms.
  - Sample-CNN was shown to slightly outperforms the 2D-CNN on the spectrogram.



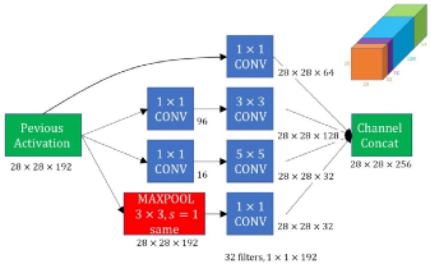
3^9 model, 19683 frames 59049 samples (2678 ms) as input			
layer	stride	output	# of params
conv 3-128	3	19683 × 128	512
conv 3-128 maxpool 3	1 3	19683 × 128 6561 × 128	49280
conv 3-128 maxpool 3	1 3	6561 × 128 2187 × 128	49280
conv 3-256 maxpool 3	1 3	2187 × 256 729 × 256	98560
conv 3-256 maxpool 3	1 3	729 × 256 243 × 256	196864
conv 3-256 maxpool 3	1 3	243 × 256 81 × 256	196864
conv 3-256 maxpool 3	1 3	81 × 256 27 × 256	196864
conv 3-256 maxpool 3	1 3	27 × 256 9 × 256	196864
conv 3-256 maxpool 3	1 3	9 × 256 3 × 256	196864
conv 3-512 maxpool 3	1 3	3 × 512 1 × 512	393728
conv 1-512 dropout 0.5	1 –	1 × 512 1 × 512	262656
sigmoid	–	50	25650
Total params			1.9×10^6

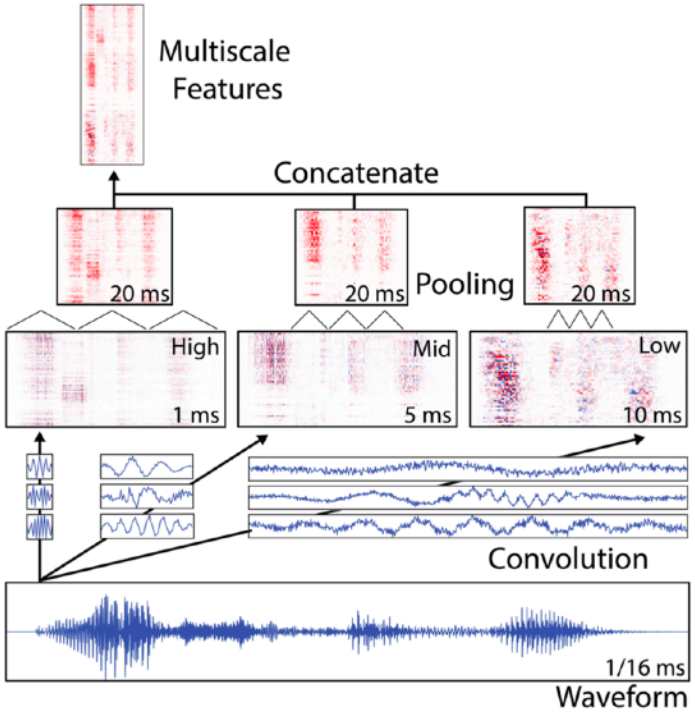
[Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. "Sample- level deep convolutional neural networks for music auto-tagging using raw waveforms". arXiv preprint arXiv:1703.01789, 2017.]
[Taejun Kim, Jongpil Lee, and Juhan Nam. "Sample-level CNN architectures for music auto-tagging using raw waveforms". 2018.]

2016 → Multi-scale (as Inception networks)

- **Idea:**
  - Since we don't know the scale we simultaneously decompose the waveform on filters of difference size

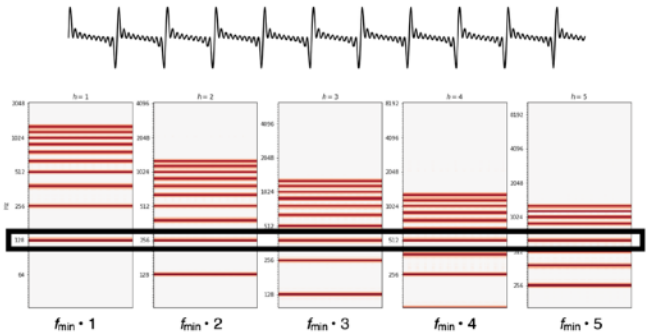
Inception network



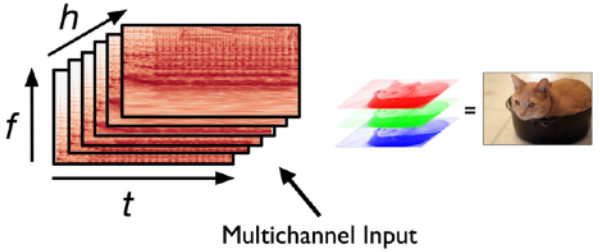


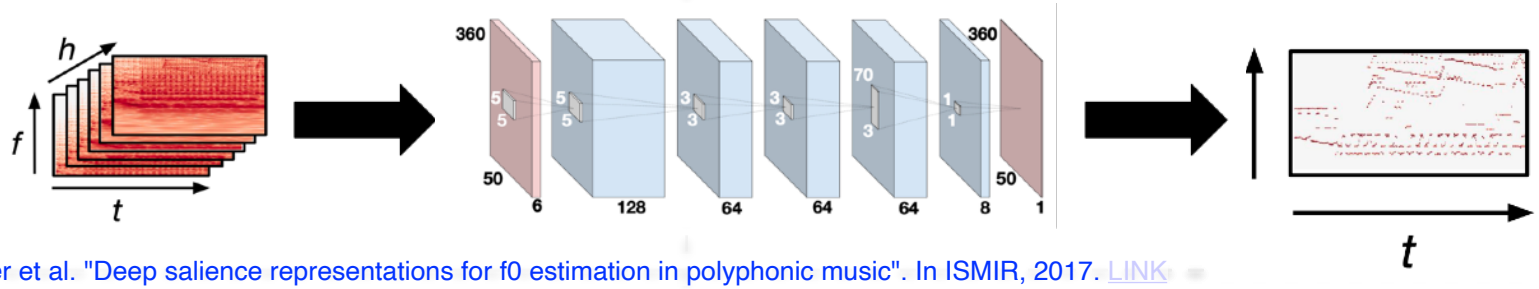
[Z. Zhu, J. H. Engel, and A. Hannun. "Learning multiscale features directly from waveforms" Interspeech, 2016]

2017 → Harmonic-CQT



- Problem of ConvNet on T/F representations
  - **Natural images**: neighboring pixels usually same source
  - **Sound**: harmonics spread over the whole spectrum, can be interleaved with harmonics of other sounds
- **Harmonic-CQT**:
  - bring back the vicinity of the harmonics by projecting each frequency f into a third dimension (the depth of the input) which represents the values of the spectrum at the harmonics hf
- ConvNet-model
  - convolve the Harmonic-CQT with a small time and frequency filter but which extends over the whole depth → allow to model easily the specific harmonic series of pitched sounds hence detecting the dominant melody
- Deep Salience
  - very good for dominant melody and multi-pitch estimation





Rachel Bittner et al. "Deep salience representations for f0 estimation in polyphonic music". In ISMIR, 2017. LINK

2018 → SincNet

– Problems of 1D-convolution
  - 1D-convolution filters are often difficult to interpret

– SincNet:
  - defines the 1D-filters as parametric functions $g(.)$ which theoretical frequency responses are parametrizable band pass filters
  - $g(.)$ is defined in the temporal domain as the difference between two $sinc(.)$ functions which learnable parameters define the low and high cutoff frequencies of the band-pass filters

$$y[n] = x[n] * g[n, \theta]$$

$$G[f, f_1, f_2] = rect\left(\frac{f}{2f_2}\right) - rect\left(\frac{f}{2f_1}\right),$$

$$g[n, f_1, f_2] = 2f_2 sinc(2\pi f_2 n) - 2f_1 sinc(2\pi f_1 n),$$

  - the filters obtained are much more interpretable
  - the performances for a task of speaker recognition is much improved

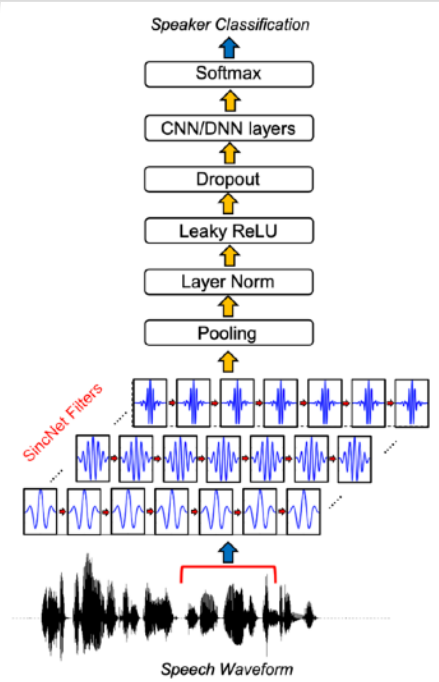


Fig. 1: Architecture of SincNet.

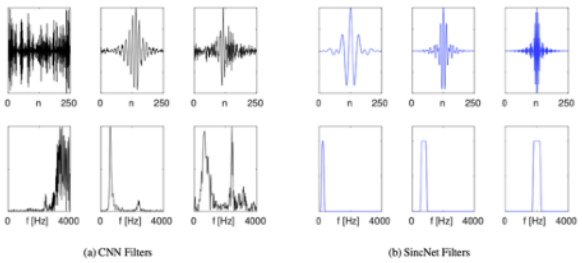


Fig. 2: Examples of filters learned by a standard CNN and by the proposed SincNet (using the Librispeech corpus). The first row reports the filters in the time domain, while the second one shows their magnitude frequency response.

[Mirco Ravanelli and Yoshua Bengio. "Speaker recognition from raw waveform with sincnet". In IEEE SLT, 2018] LINK

2020 → CG-CNN (Complex Gabor)

– Problems of SincNet:

- Time and Frequency trade-off → Gaussian/Gabor window
- SinceNet are real, no shift invariance → Complex kernels

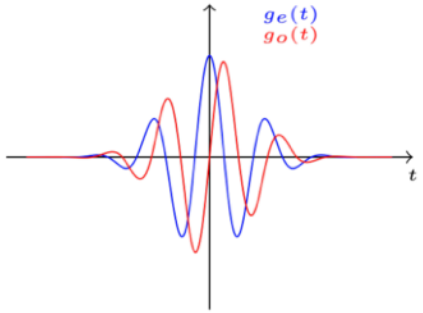


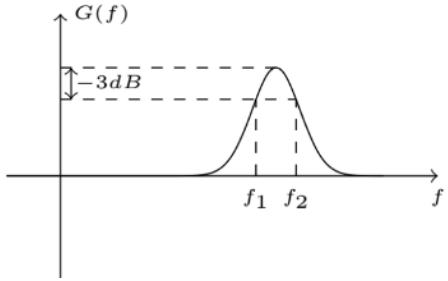
Fig. 1: Illustration of the real (g_e) and imaginary (g_o) parts of the complex Gabor filter impulse response.

– Complex-Gabor:

$$g(t) = w_\sigma(t) e^{i2\pi f_0 t}, \quad = g_e(t) + i g_o(t), \quad w_\sigma(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}},$$

$$G(f) = e^{-2\pi^2\sigma^2(f-f_0)^2}.$$

$$\sigma = \frac{A}{\pi(f_2 - f_1)}, \; f_0 = \frac{f_1 + f_2}{2}$$



Model	Valid.%	Avg. Test%	Best Test%
Gabor-CNN-CTC [18]	-	18.8	18.5
SincNet [2]	-	17.2	-
GaborReal	15.2	17.2	16.9
GaborComplex	15.2	17.1	16.7

- $x(t) \circledast g(t) = x(t) \circledast g_e(t) + i\left(x(t) \circledast g_o(t)\right)$
- output is processed by a CV-CNN (Complex-Valued CNN)
- the performances for a task of TIMIT phoneme recognition task

[Paul-Gauthier Noé et al. "CGCNN: Complex Gabor Convolutional Neural Network on Raw Speech" In ICASSP, 2020] LINK

2020 → Harmonic-CNN

- combine the idea of SincNet with the harmonic model
- 1D-convolution is performed with filters
  - constrained as for SincNet
  - but extended to the harmonic dimensions
    - stacking band-pass filters at harmonic frequencies hf_c

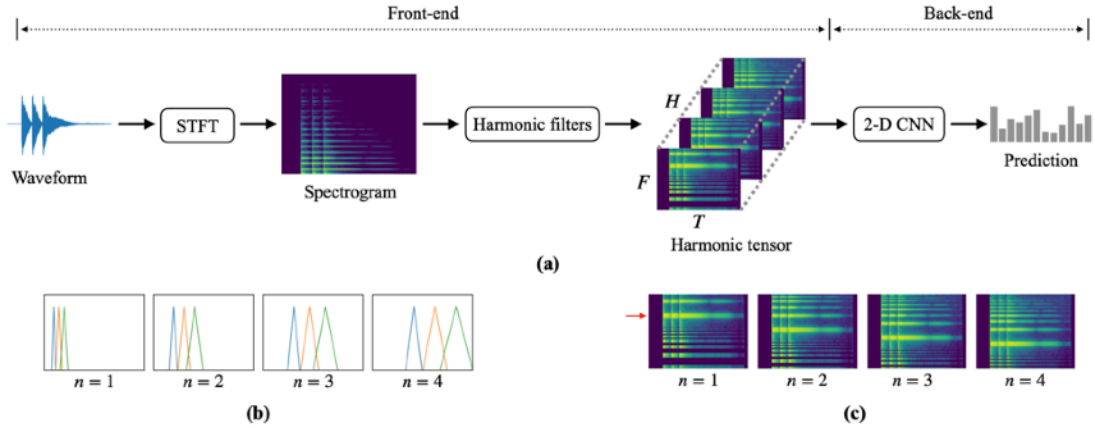
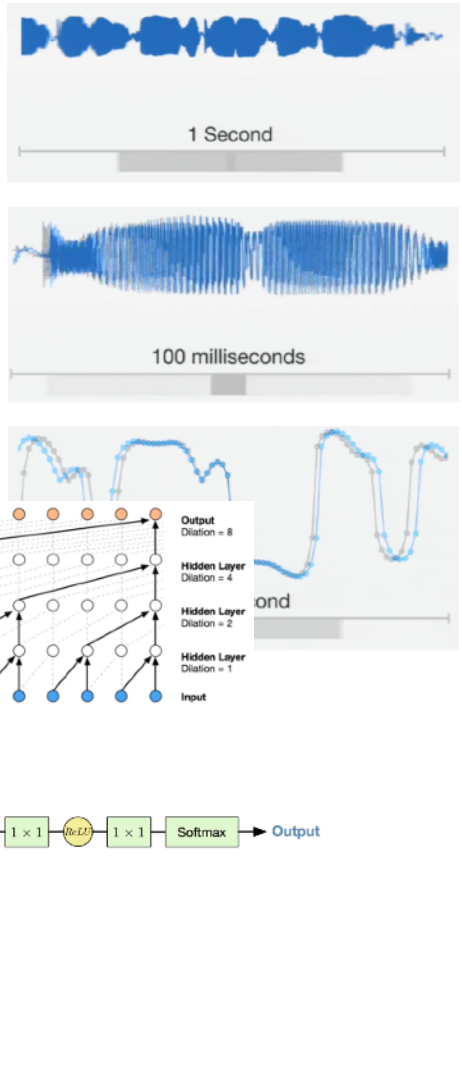


Fig. 1: (a) The proposed architecture using Harmonic filters. The proposed front-end outputs the Harmonic tensor and the back-end processes it depending on the task. The Harmonic filters and the 2-D CNN are data-driven modules that learn parameters during training. (b) Harmonic filters at each harmonic. (c) An unfolded Harmonic tensor. The red arrow indicates the fundamental frequency.

[M. Won et al. "Data-driven harmonic filters for audio representation learning". In ICASSP, 2020] LINK

2016 → Neural-Autoregressive Models: WaveNet

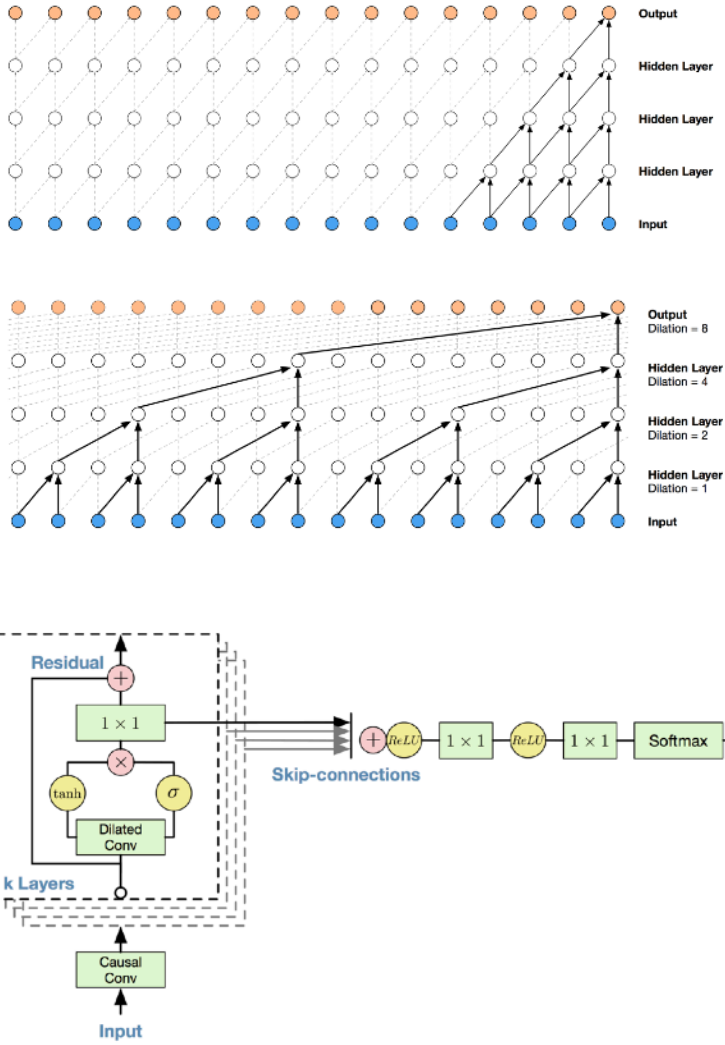
- **Generative model operating directly on audio samples**
  - raw audio = challenging to model because structure at very different scales
  - Based on PixelCNN
  - High resolution signal and long-term dependencies
    - dependencies at **different level**
- **Auto-regressive model**
  - next sample is (almost) reconstructed from linear convolution of past samples
  - $$p(x_1, x_2, \cdots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, \cdots, x_{t-1})$$
  - using RNN ⇒ costly
  - using CNN ⇒ cheap (parallel)



[A. van den Oord et al. "Wavenet: A generative model for raw audio". arXiv preprint arXiv:1609.03499, 2016]
[S. Mehri et al. "Sample-RNN: An unconditional end-to-end neural audio generation model". In ICLR, 2017]

2016 → Neural-Autoregressive Models: Wave

- **Causal Convolution**
  - require many layers or large filters to increase receptive field
- **Dilated Convolution** (convolution à trou)
  - increase the receptive field by orders of magnitude
- **Stacked dilated convolutions**
  - dilation doubled / limit then repeated
    - 1, 2 ... 512, 1, 2 ... 512, 1, 2, ... 512.
- **Input/output signal representation**
  - using μ-law ⇒8 bits ⇒256 categories
- **Softmax layer**
  - trained to maximize log-likelihood
- **Conditional Wavenet:**
  - $$p(\vec{x} \mid \vec{h}) = \prod_{t=1}^{T} p(x_t \mid x_1, \cdots, x_{t-1}, \vec{h})$$
  - conditioning on other input variables

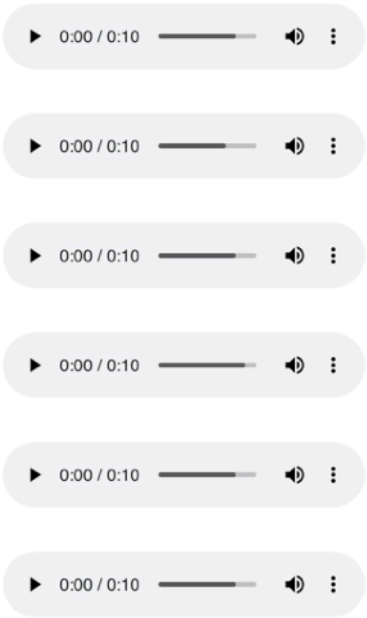


[A. van den Oord et al. "Wavenet: A generative model for raw audio". arXiv preprint arXiv:1609.03499, 2016]

2016 → Neural-Autoregressive Models: WaveNet

Making Music

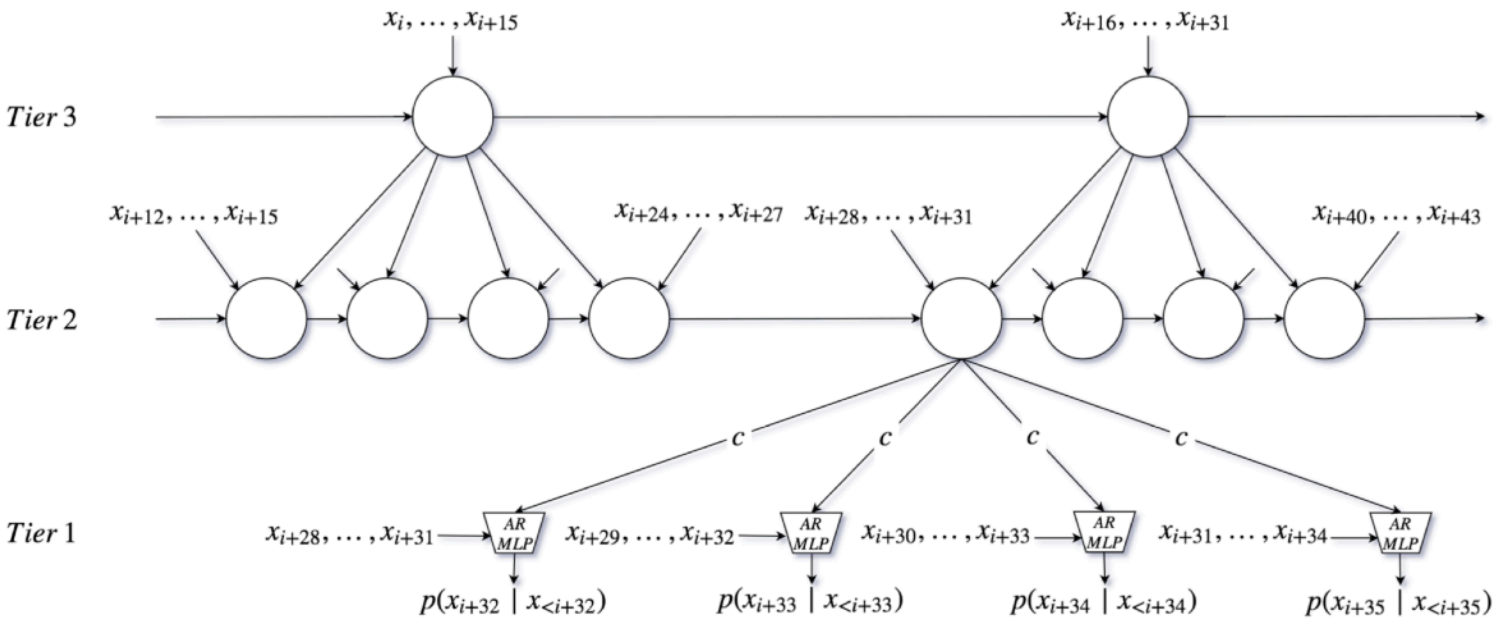
Since WaveNets can be used to model any audio signal, we thought it would also be fun to try to generate music. Unlike the TTS experiments, we didn't condition the networks on an input sequence telling it what to play (such as a musical score); instead, we simply let it generate whatever it wanted to. When we trained it on a dataset of classical piano music, it produced fascinating samples like the ones below:



https://deepmind.com/blog/wavenet-generative-model-raw-audio/

[A. van den Oord et al. "Wavenet: A generative model for raw audio". arXiv preprint arXiv:1609.03499, 2016]

2017 → Neural-Autoregressive Models: SampleRNN



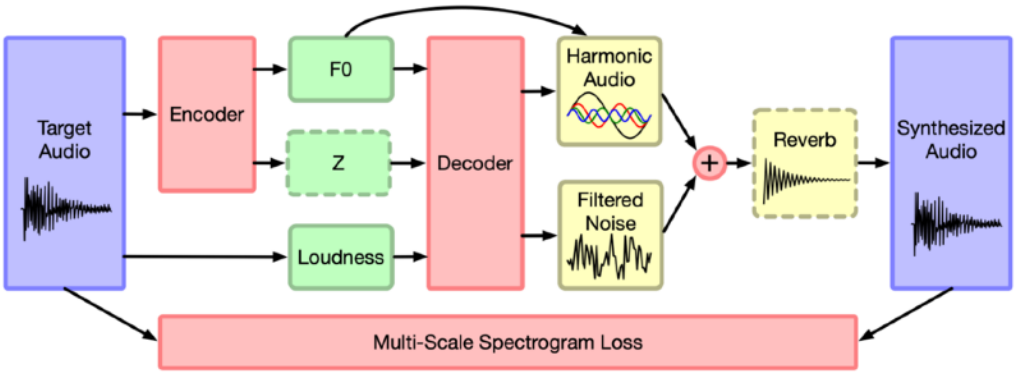
[S. Mehri et al. "Sample-RNN: An unconditional end-to-end neural audio generation model". In ICLR, 2017]

2020 → Differentiable Digital Signal Processing (DDSP)

- DDSP defines the sound production model:
  - the **Spectral Modeling Synthesis (SMS) model [Serra,1990]**
    - combines harmonic additive synthesis (adding together many harmonic sinusoidal components) with subtractive synthesis (filtering white noise);
    - also adds room acoustics to the produced sound through reverberation

$$x(n) = \sum_{k=1}^{K} A_k(n)\sin(\phi_k(n)) \qquad \phi_k(n) = 2\pi \sum_{m=0}^{n} f_k(m) + \phi_{0,k} \qquad f_k(n) = k f_0(n)$$

- The training consists in finding its parameters !

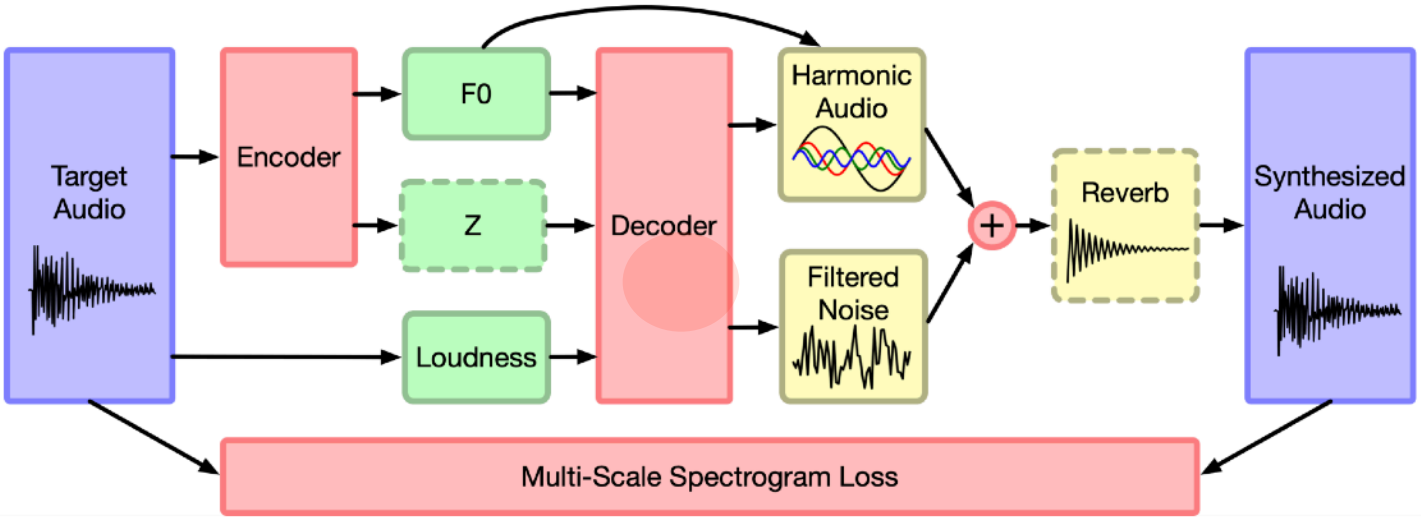


- Input audio signal $\mathbf{x}(t)$
  - first encoded into time-varying
    - loudness $l(t)$,
    - pitch $f_0(t)$
    - a latent representation $\mathbf{z}(t)$
- Those are fed to a decoder
  - which estimates the control parameters of the additive and filtered noise synthesizers

[X. Serra and J. Smith. "Spectral modeling synthesis: A sound analysis/synthesis system ..." In Computer Music Journal, 1990.
[J. Engel, L. Hantrakul, C. Gu, and A. Roberts. "DDSP: Differentiable digital signal processing". In ICLR, 2020]

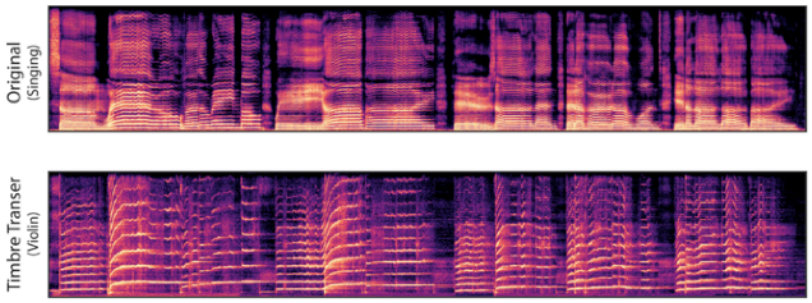
2020 → Differentiable Digital Signal Processing (DDSP)

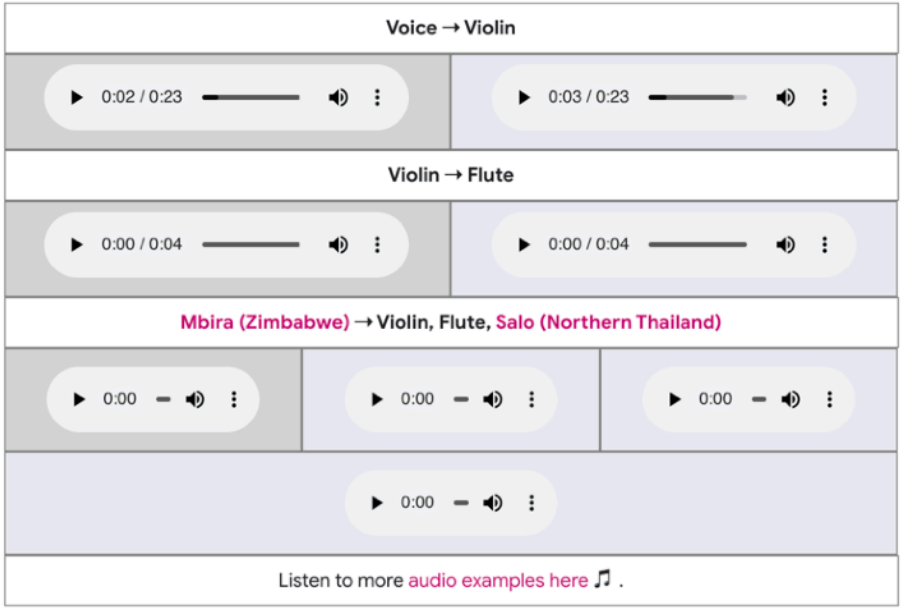
- **Encoder:**
  - Loudness: extracted directly from the audio
  - f_0 pitch: Crepe or ResNet
  - $z(t)$: time-varying latent encoding
- **Decoder:**
  - map $(f(t), l(t), z(t))$to control parameters additive and filtered noise synthesizers



[X. Serra and J. Smith. "Spectral modeling synthesis: A sound analysis/synthesis system ..." In Computer Music Journal, 1990.
[J. Engel, L. Hantrakul, C. Gu, and A. Roberts. "DDSP: Differentiable digital signal processing". In ICLR, 2020]

2020 → Differentiable Digital Signal Processing (DDSP)





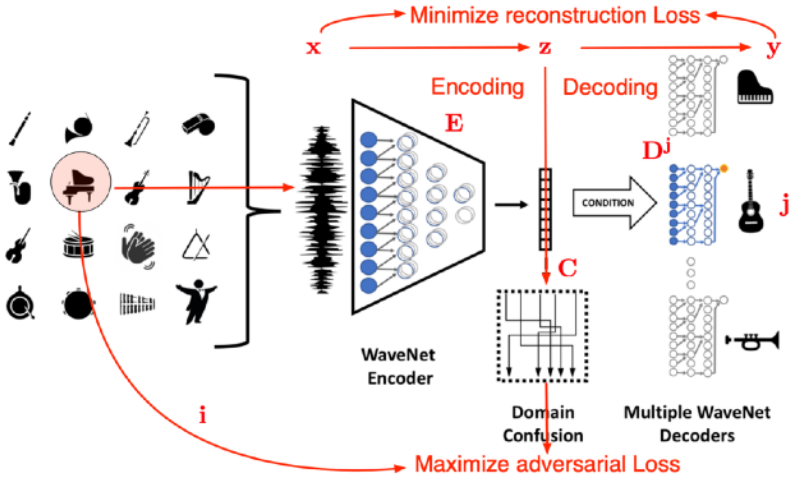
https://magenta.tensorflow.org/ddsp

[X. Serra and J. Smith. "Spectral modeling synthesis: A sound analysis/synthesis system ..." In Computer Music Journal, 1990.
[J. Engel, L. Hantrakul, C. Gu, and A. Roberts. "DDSP: Differentiable digital signal processing". In ICLR, 2020]

Audio applications with DNN

Music translation

- **Task**:
  - translating
    - an input music x with {musical instruments, genres, and style} i to
    - an output y with {musical instruments, genres, and style} j
  - while preserving the musical score
  - without explicitly extracting the musical score and with a single but smart network
- Network has the form of an AE
  - Encoder E (a single WaveNet for all i) is used to project x in the latent space z
  - z is then used to reconstruct an output music track with {musical instruments, genres, and style} j using specific decoders D^j for each j (which are all WaveNet decoders)
- **Training:**
  - Encoder and decoder are trained to minimize a reconstruction loss between x and y.



https://research.fb.com/publications/a-universal-music-translation-network/

Mor, N.,Wolf, L., Polyak, A., Taigman, Y.:A universal music translation network. In: Proc. of ICLR, (2019)

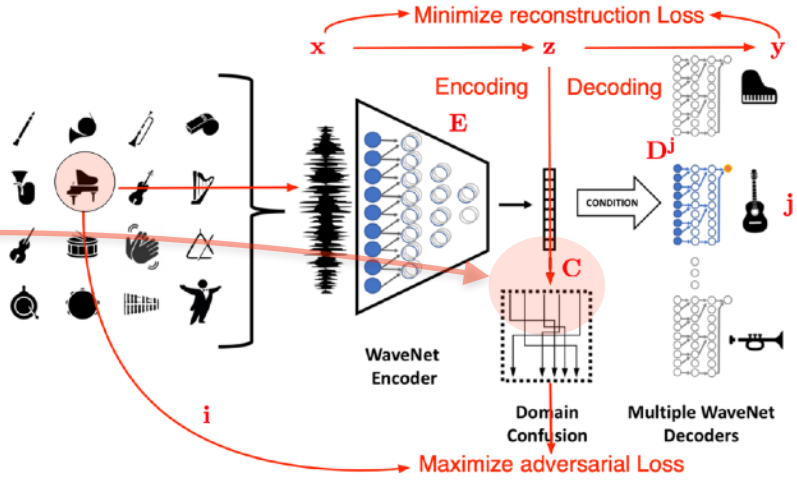
Music translation (cont.)

– **Disentanglement of z ?**

- Important that z does not contain information related to the {musical instruments, genres, and style} i

– How ?

- 1) a classifier C is trained to recognize i from z
- 2) with C fixed, we add an adversarial loss, i.e. we train the encoder E to maximize a classification loss (guarantee that not possible to recognize i from z)



https://research.fb.com/publications/a-universal-music-translation-network/

– **Training:**

- minimize the reconstruction loss and maximize domain classification loss (adversarial loss)
  - prevent z to learn domain characteristic maximize the adversarial loss

$$\sum_j \sum_{s_j} \mathbb{E}_r \mathscr{L}(D^j(E(O(s^j, r))), s^j) - \lambda \mathscr{L}(C(E(O(s^j, r))), j)$$

Mor, N.,Wolf, L., Polyak, A., Taigman, Y.:A universal music translation network. In: Proc. of ICLR, (2019)

N. Mor, L. Wolf, A. Polyak, and Y. Taigman.
"A universal music translation network" in ICLR, 2019

Original (Input): String Quartet, Haydn

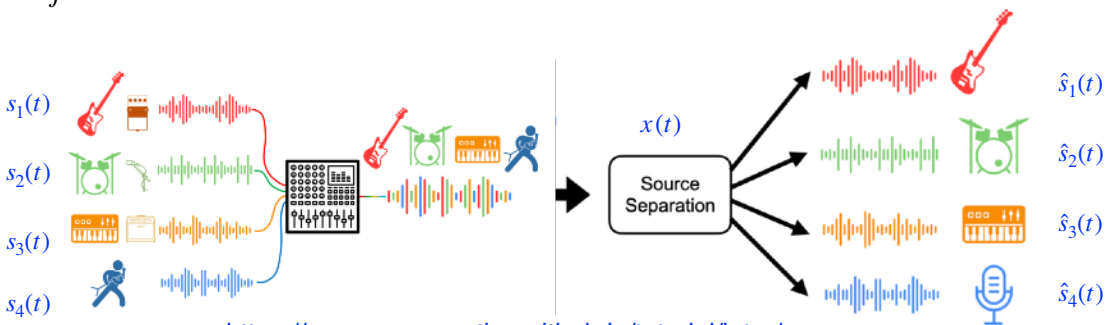
Audio Source Separation

Audio Source Separation Task

Blind Audio Source Separation (BASS)

- Task:
  - recover one or several source signals $s_j(t)$ from a mixture signal $x(t)$ without any additional information

$$x(t) = \sum_{j=1}^{C} s_j(t) \quad \rightarrow \quad \hat{s}_j(t) \quad ?$$



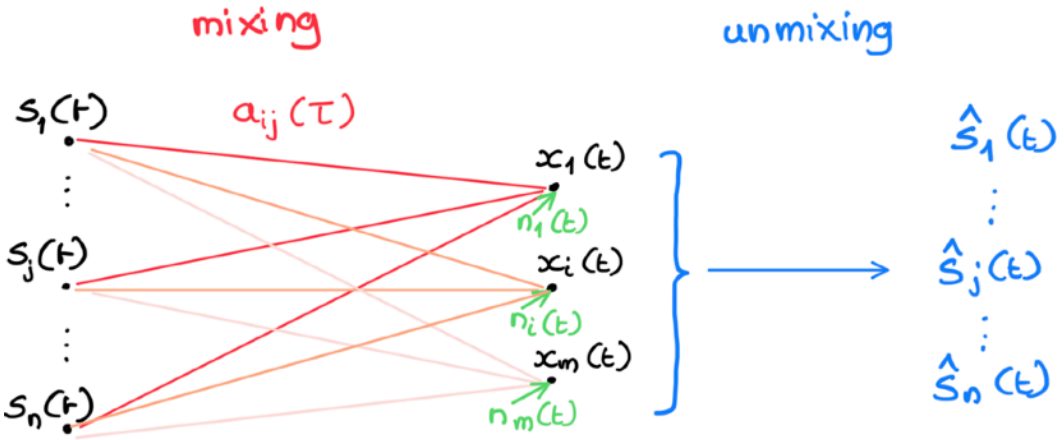
https://source-separation.github.io/tutorial/intro/

- Closely related to speech denoising, speech enhancement
- Applications
  - automatic music transcription
  - lyric and music alignment
  - musical instrument detection
  - lyric recognition
  - automatic singer identification
  - vocal activity detection
  - fundamental frequency estimation
  - understanding the predictions of black-box audio models

Audio Source Separation Evaluation

Model

– sources $s_j(t)$ $\overset{\text{mixing } a_{ij}}{\rightarrow} *$ $\overset{\text{noise } n_i}{\rightarrow}$ microphones $x_i(t)$



$$x_i(t) = \sum_{j=1}^{n} \left[\sum_{\tau=0}^{+\infty} a_{ij}(t) \; s_j(t-\tau) \right] + n_i(t) \quad \forall i \in [1,m]$$

$$\mathbf{x} = \mathbf{A} * \mathbf{s} + \mathbf{n}$$

- $s_{j\in\{1,\ldots,n\}}(t)$ signal emitted by the j^{th} source ()
- $x_{i\in\{1,\ldots,m\}}(t)$ signal recorded by the i^{th} microphone ()
- $a_{ij}(\tau)$ (causal) source-to-microphone filters
- $n_i(t)$ additive sensor noise

Audio Source Separation
Evaluation

bss_eval , mus_eval

- Comparing the estimated source $\hat{s}_j(t)$ to the ground-truth source $s_j(t)$
- Decompose $\hat{s}_j(t)$
  - $\hat{s}_j(t) = s_{target}(t) + e_{interf}(t) + e_{noise}(t) + e_{artif}(t)$
    - $s_{target}(t)$: part of $\hat{s}_j(t)$ coming from the wanted source $s_j(t)$
    - $e_{interf}(t)$: part coming from interferences, from unwanted sources $\left(s_{j'}(t)\right)_{j' \neq j}$
    - $e_{noise}(t)$: perturbing noise (not coming from sources) $\left(n_i\right)_{1 \leq i \leq m}$
    - $e_{artif}(t)$: artifacts from separation algorithm (such as musical noise)

- We decompose $\hat{s}_j$ as the sum of the four terms

$$s_{target} = P_{s_j}\,\hat{s}_j$$
$$e_{interf} = P_s\,\hat{s}_j - P_{s_j}\,\hat{s}_j$$
$$e_{noise} = P_{s,n}\,\hat{s}_j - P_s\,\hat{s}_j$$
$$e_{artif} = \hat{s}_j - P_{s,n}\,\hat{s}_j$$

- If we denote by $\Pi\{y_1, \ldots, y_k\}$ the orthogonal projector onto the subspace spanned by the vectors $y_1, \ldots, y_k$.
- We consider 3 orthogonal projectors

$$P_{s_j} = \Pi\{s_j\}$$
$$P_s = \Pi\{(s_{j'})_{1 \leq j' \leq n}\}$$
$$P_{s,n} = \Pi\{(s_{j'})_{1 \leq j' \leq n}, (n_i)_{1 \leq i \leq m}\}$$

[E. Vincent, R. Gribonval, C. Févotte, "Performance Measurement in Blind Audio Source Separation", 2006, IEEE TASLP]

Audio Source Separation Evaluation

- SDR (Source to Distortion Ratio)

$$SDR = 10\log_{10}\frac{\|s_{target}\|^2}{\|e_{interf} + e_{noise} + e_{artif}\|^2}$$

- SIR (Source to Interference Ratio)

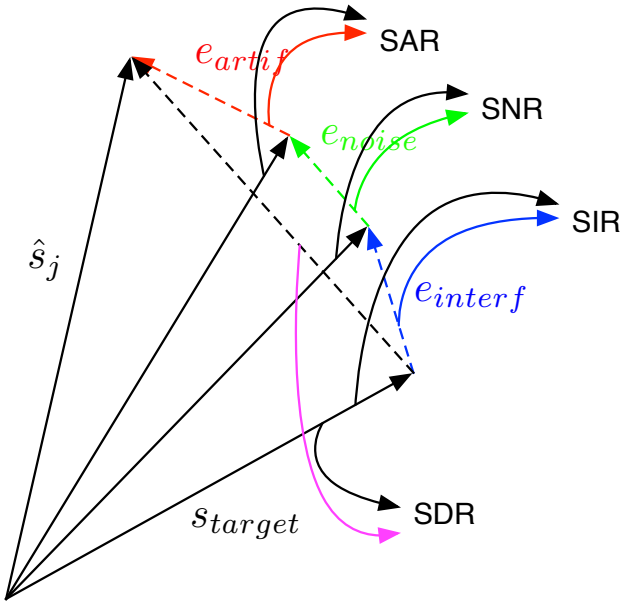
$$SIR = 10\log_{10}\frac{\|s_{target}\|^2}{\|e_{interf}\|^2}$$

- *SNR (Sources to Noise Ratio)*

$$SNR = 10\log_{10}\frac{\|s_{target}+e_{interf}\|^2}{\|e_{noise}\|^2}$$

- SAR (Sources to Artifact Ratio)

$$SAR = 10\log_{10}\frac{\|s_{target}+e_{interf}+e_{noise}\|^2}{\|e_{artif}\|^2}$$



Increasing interference (SAR does not change)

SDR: 56/ SIR: 60/ SAR: 58

SDR: 19/ SIR: 19/ SAR: 57

SDR: 12/ SIR: 12/ SAR: 56

SDR: 07/ SIR: 07/ SAR: 56

Increasing artifact/noise (SIR does not change)

SDR: 19/ SIR: 19/ SAR: 57

SDR: 15/ SIR: 19/ SAR: 17

SDR: 11/ SIR: 19/ SAR: 11

Audio Source Separation
Evaluation

Other measures

– **Scale-Invariant measures**
  - SI-SDR, SI-SIR, SI-SAR

– **Perceptual criteria**
  - PEASS: Perceptual Evaluation of Audio Source Separation
  - PESQ: Perceptual Evaluation of Speech Quality
  - STOI: Short-Time Objective Speech Intelligibility

[Le Roux et al. "SDR – Half-Baked or Well Done?", 2018]
[Emmanuel Vincent, "Improved perceptual metrics for the evaluation of audio source separation", in *LVA/ICA*, 2012]
https://gitlab.inria.fr/bass-db/peass

Audio Source Separation Datasets

Audio Source Separation Datasets

Dataset	Year	Instrument categories	Tracks	Avgerage duration (s)	Full songs	Stereo
MASS	2008	N/A	9	16 ± 7	❌	✅
MIR-1K	2010	N/A	1,000	8 ± 8	❌	❌
QUASI	2011	N/A	5	206 ± 21	✅	✅
ccMixter	2014	N/A	50	231 ± 77	✅	✅
MedleyDB	2014	82	63	206 ± 121	✅	✅
iKala	2015	2	206	30	❌	❌
DSD100	2015	4	100	251 ± 60	✅	✅
MUSDB18	2017	4	150	236 ± 95	✅	✅
Slakh2100	2019	34	2100	249	✅	❌

Audio Source Separation Implementation

- https://github.com/tky823/DNN-based_source_separation

Audio Source Separation Systems

Brief overview of systems evolution

- **Computational Auditory Scene Analysis**
  - REPET-SIM

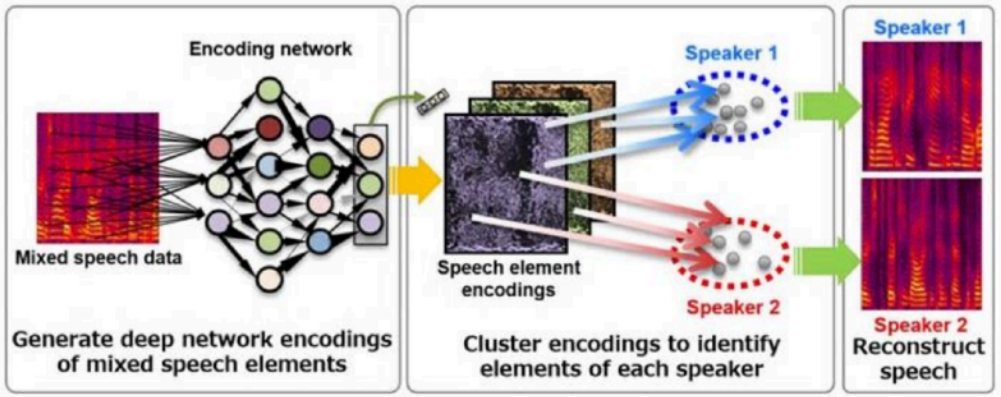
- **Matrix Decomposition methods**
  - Independent Component Analysis (ICA), Subspace (ISA), Non-Negative Matrix Factorization (NMF)

- **Deep Learning approaches**
  - Formulating BASS as a supervised task
  - Metric learning
    - Deep Clustering
  - Model trained to transform [x=an input mixed signal] to
    - [y=an output separated **source** $s_j(t)$]
    - [y=an output separation **mask** $m_j(t)$] to be applied to the input $s_j(t) = x(t) \odot m_j(t)$
  - Mask ? Permutation Invariance Training ? Class-independent ? Universal Training ?
  - Using the **STFT**
    - RNN/ LSTM approaches
    - Convolutional approaches: U-Net [Spleeter], Complex-U-Net
  - Using the **waveform**
    - Wave-U-Net
    - Encoder/Masker/Decoder: Tas-Net, Conv-Tas-Net, Demucs

Audio Source Separation - Systems
2016 → Deep Clustering

Main idea:

- Train a DNN (*Bi-LSTM, Bi-LSTM, FC*) to produce spectrogram embeddings that are discriminative for partition labels given in training data
  - Class independent



Source: https://de.mitsubishielectric.com/en/news-events/releases/global/2017/0524-e/index.html

- **Input**
  - Log-STFT: $X_{n=(t,f)}\ n \in \{1,\ldots N\}$
- **Output**
  - Embedding: $V = f_\theta(x) \in \mathbb{R}^{NxK}$ where K is the size of the projection, we require $\|v_n\|^2 = 1$
- **Ground-Truth**
  - $Y = \{y_{n,c}\}, y_{n,c} = 1$ if n in partition $c \rightarrow$ Ideal pairwise affinity matrix: YY^T
  - Estimated pairwise affinity matrix: VV^T
- **Training**:
  - minimize $C_\theta = |VV^T - YY^T|^2_W$
- **Inference ?**
  - Simply performs clustering (k-means) of the $n = (t, f)$ embedding points $V = f_\theta(x)$

[J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe. Deep clustering, In ICASSP, 2016]

Audio Source Separation - Systems
General Source Separation Architecture

- **Encoder:**
  - Fixed filterbanks (STFT, Mel)
  - Trainable filterbanks
  - Free filter-banks
- **Separator**
  - LSTMs, TCN, U-Net, ...
  - Complex Network
- **Decoder**
  - Inverse encoders (STFT^{-1})
  - Neural vocoder
- **Loss**
  - Fixed or Permutation invariant loss (PIT)
  - Spectrogram loss, l1, l2, SI-SNR

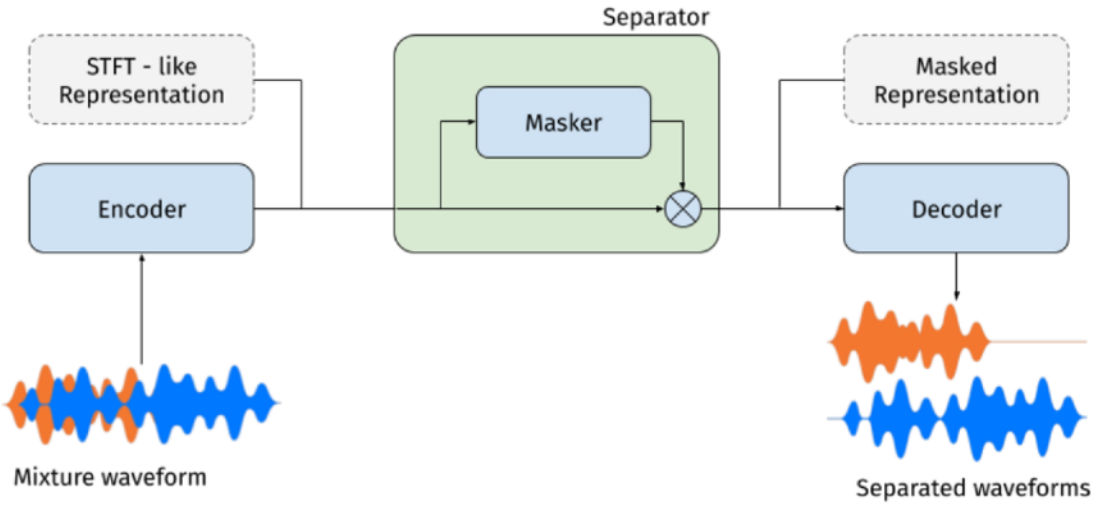
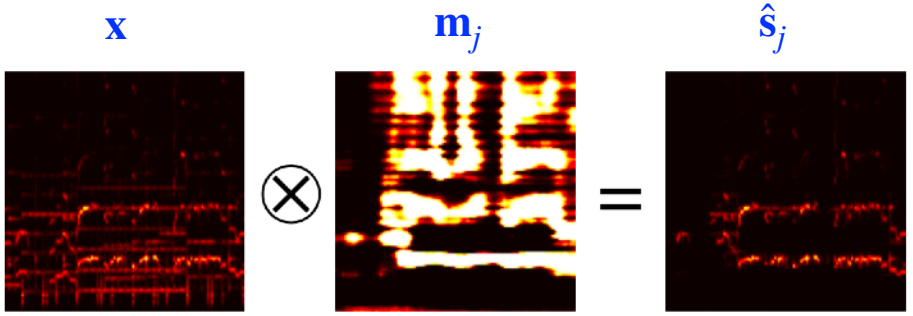


Figure from Pariente, Manuel, et al. "Asteroid: the PyTorch-based audio source separation toolkit for researchers." *arXiv preprint arXiv:2005.04132* (2020).

Audio Source Separation
Masks ?

- Given a mixture $x(t)$ of C sources $s_i(t)$:
  - $x(t) = \sum_{j=1}^{C} s_j(t) \Leftrightarrow X(t,f) = \sum_{j=1}^{C} S_j(t,f)$
  - Estimated magnitude STFT of source j
    - notation: $\mathbf{x} = |X(t,f)|$
    - $\hat{\mathbf{s}}_j = \mathbf{x} \odot \mathbf{m}_j$ subject to $\sum_{j=1}^{C} \mathbf{m}_j = 1$



  - **IBM (Ideal Binary Mask)** for source j is defined as
    - $IBM_{j,tf} = \delta(|\mathbf{s}_{j,tf}| > |\mathbf{s}_{j',tf}|), \forall j' \neq j \quad \in \{0,1\}$
      - i,e. source j it is the most dominant source for this specific STFT bin (t,f)
  - **IRM (Ideal Ratio/Soft Mask)** for source j is defined as
    - $IRM_{j,tf} = \frac{|\mathbf{s}_{j,tf}|}{\sum_{j'=1}^{C} |\mathbf{s}_{j',tf}|} \in [0,1]$
  - **Wiener-filter Like Mask (WFM)** for source j is defined as
    - $WFM_{j,tf} = \frac{|\mathbf{s}_{j,tf}|^2}{\sum_{j'=1}^{C} |\mathbf{s}_{j',tf}|^2}$

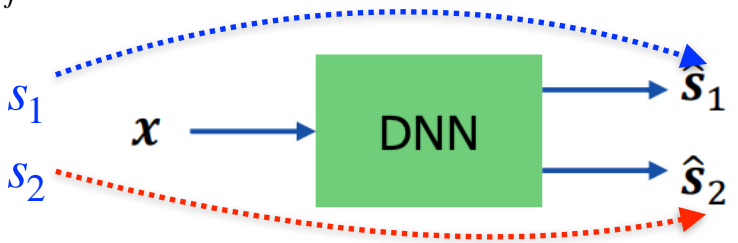
https://arxiv.org/pdf/1707.03634.pdf

Audio Source Separation
Loss ?

- The same architecture can be used for the separation of different source types, but also for same source types

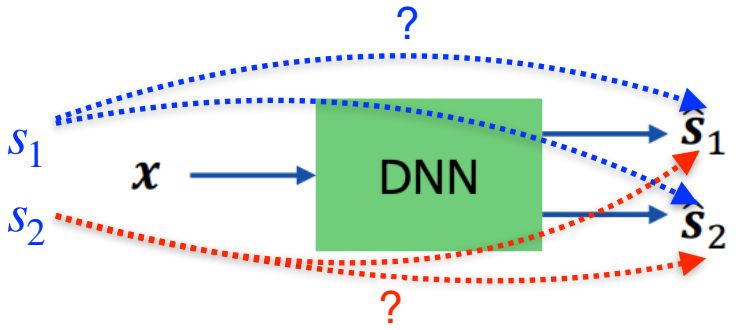
  - **Different source types** (e.g., $\hat{s}_1$=vocals, $\hat{s}_2$=accompaniment)
    - → we know directly the mapping between the $\{s_j\}$ and $\{\hat{s}_j\}$
    - → **Fixed loss**

$$\mathcal{L} = \mathcal{L}(\hat{s}_1, s_1) + \mathcal{L}(\hat{s}_2, s_2)$$



  - **Same source types** (e.g., two unknown speakers or two violins)
    - →several possibilities for mapping the $\{s_j\}$ to the $\{\hat{s}_j\}$
    - →**PIT (Permutation Invariant Training) loss**

$$\mathcal{L} = \min\{\mathcal{L}(\hat{s}_1, s_1) + \mathcal{L}(\hat{s}_2, s_2), \mathcal{L}(\hat{s}_1, s_2) + \mathcal{L}(\hat{s}_2, s_1)\}$$



[Yu, Dong, et al. "Permutation invariant training of deep models for speaker-independent multi-talker speech separation." ICASSP 2017]

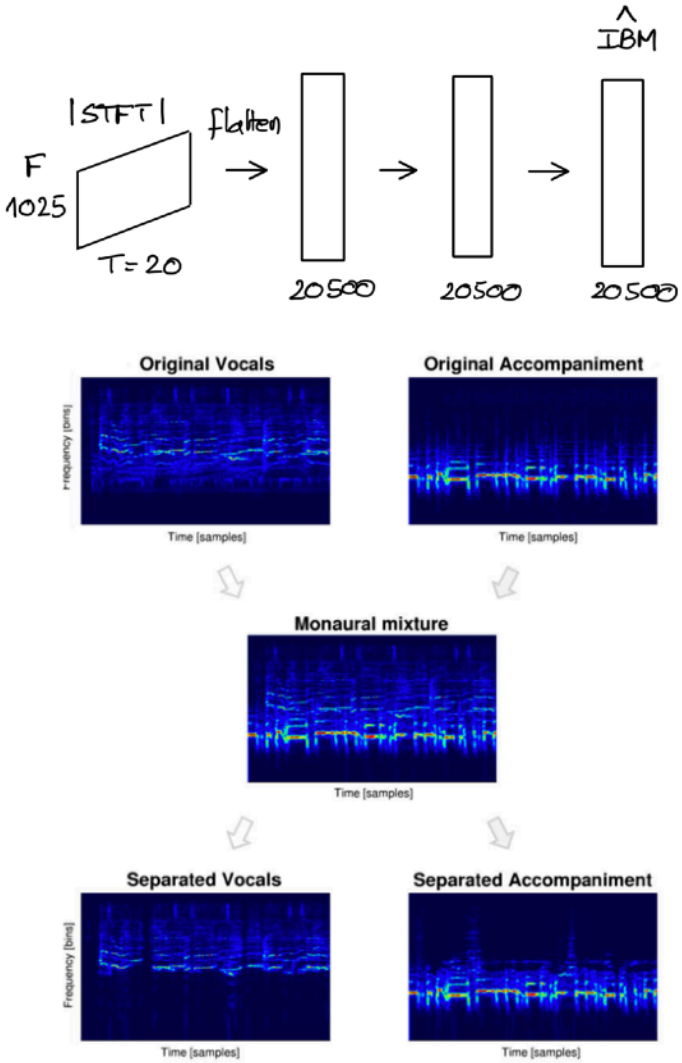
Audio Source Separation Systems

Using STFT as Input

Audio Source Separation - Systems
2015 → using FC

Idea:

- Train a **deep feed-forward** network to learn to estimate an ideal **binary spectrogram mask** that represents the spectrogram bins in which the vocal is more prominent than the accompaniment

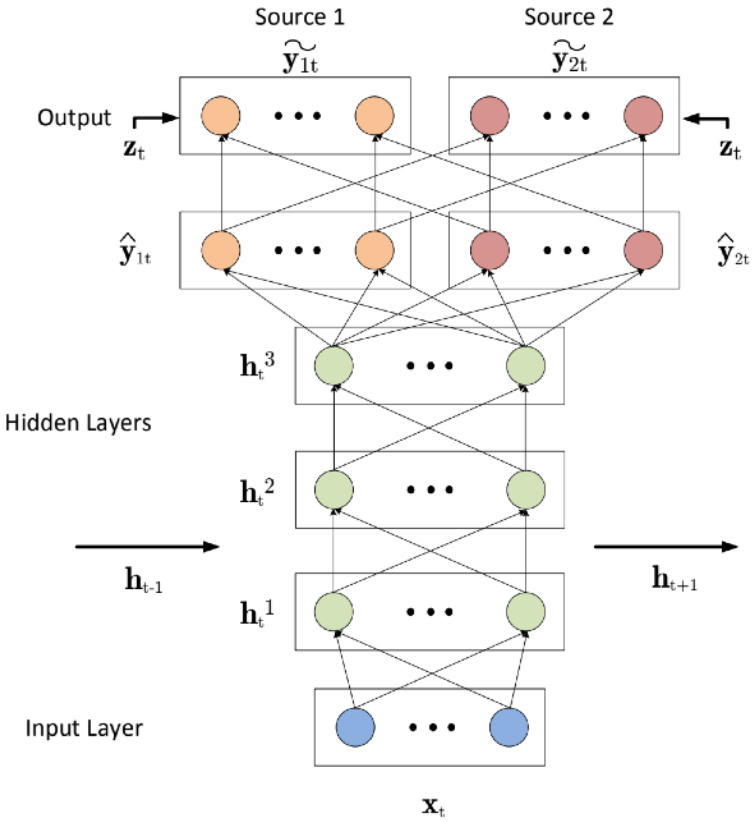


Andrew JR Simpson, Gerard Roma, and Mark D Plumbley. "Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network". In International Conference on Latent Variable Analysis and Signal Separation, 2015.

Audio Source Separation - Systems 2014 → using DeepRNN

Idea:

- Train a deep recurrent architecture to predict **soft masks** that are multiplied with the original signal to obtain the desired isolated source

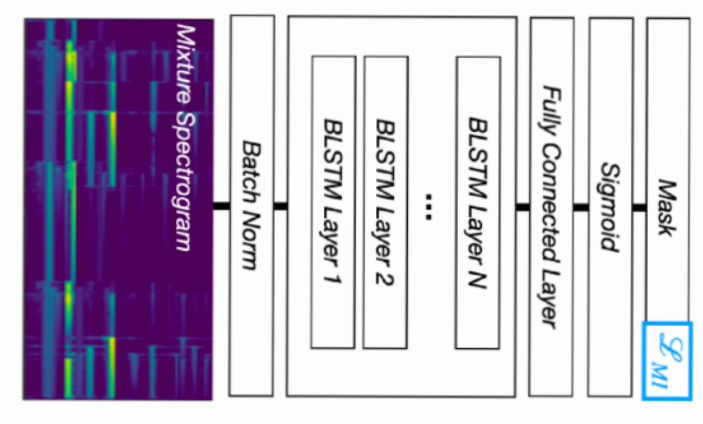


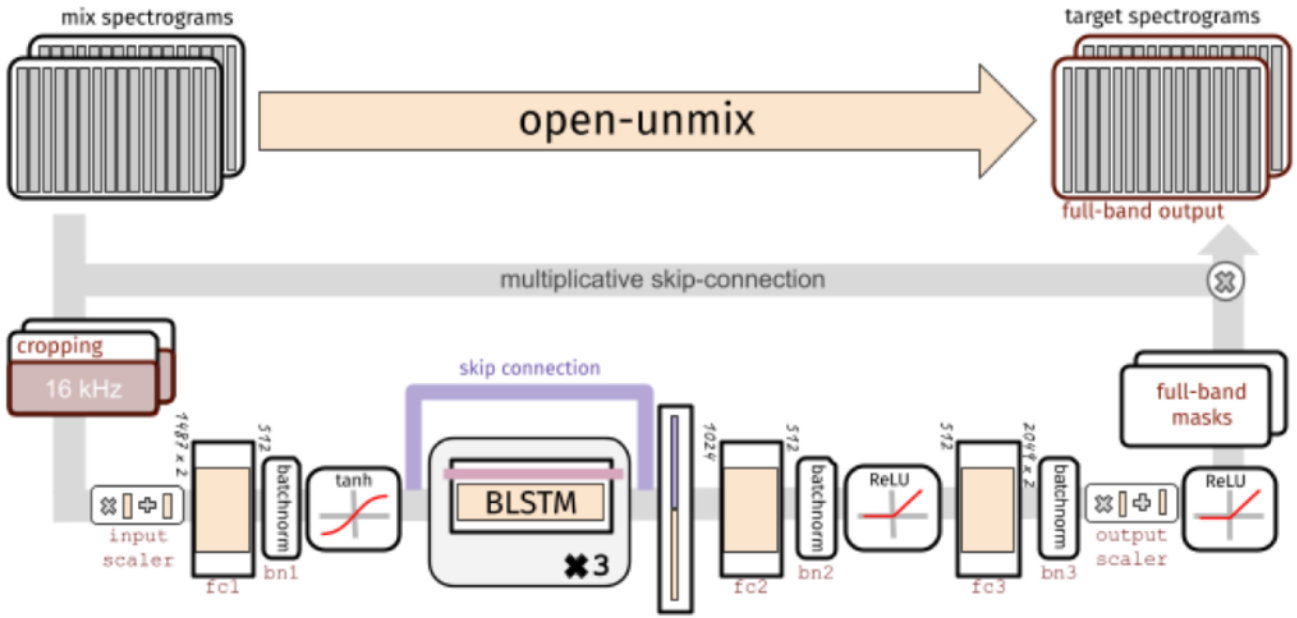
Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. "Singing-voice separation from monaural recordings using deep recurrent neural networks". In ISMIR 2014, 2014.

Audio Source Separation - Systems 2019 → using BLSTM

Idea:

- predict the soft mask using LSTMs
- add skip-connection around LSTMs
- before feeding the LSTMs, first reduce the frequency dimension of the |STFT| using FC



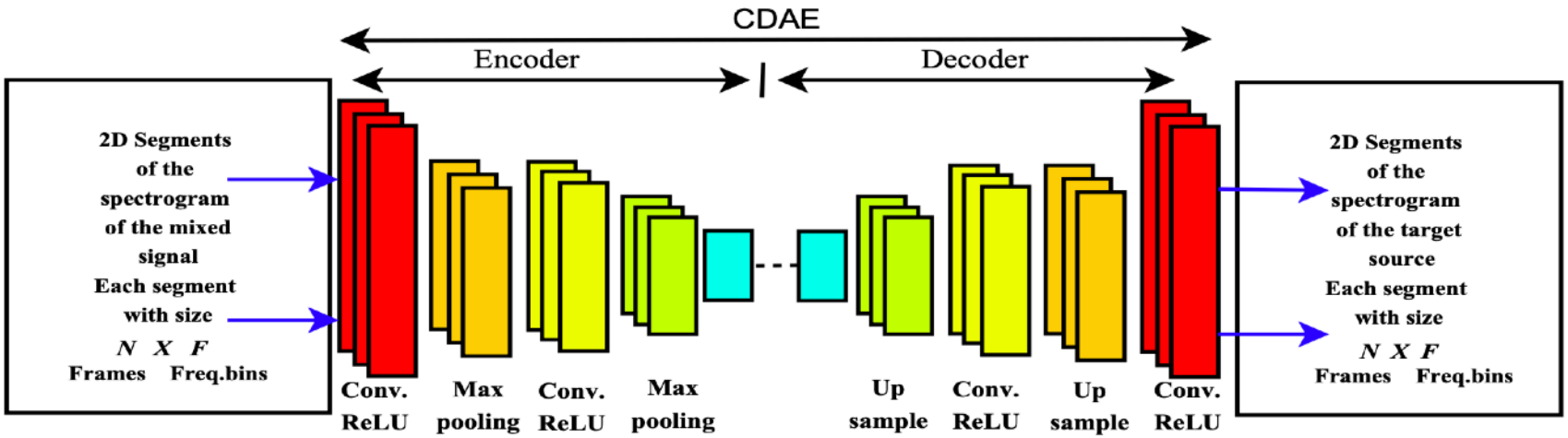


F.-R. Stöter and S. Uhlich and A. Liutkus and Y. Mitsufuji, "Open-Unmix - A Reference Implementation for Music Source Separation", Journal of Open Source Software, 2019

Audio Source Separation - Systems
2017 →using CDAE (Convolutional Denoising Auto-Encoders)

Idea:

- Use a Denoising Auto-Encoder
- Use a Convolutional Network
- Estimate directly the target |STFT| (not the mask)

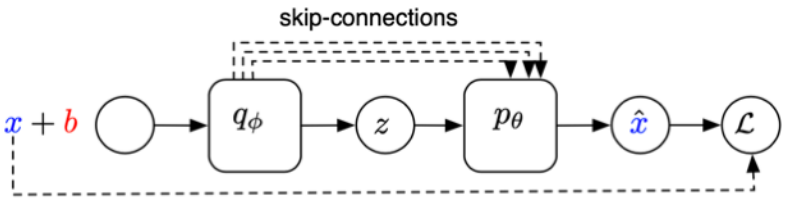


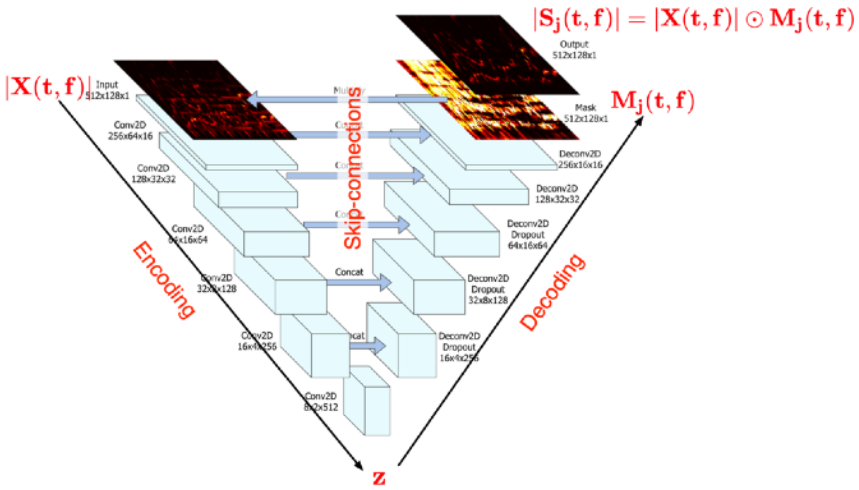
Emad M Grais and Mark D Plumbley. "Single channel audio source separation using convolutional denoising autoencoders". arXiv preprint arXiv:1703.08019, 2017

Audio Source Separation - Systems
2017 → using CDAE with skip-connection (U-Net)

Idea:

- **Limitations of Convolutional Denoising Auto-Encoder (CDAE):**
  - allows to reconstruct a clean but blurred signal from its noisy version
  - but the fine details of the spectrogram (precise harmonic locations) are lost in z
- **U-Net architecture**
  - an AE with skip-connections between E and D
  - contracting path: capture context and a symmetric expanding path that enables precise localization
- **U-Net for source separation**
  - **Goal**: isolate the singing voice from real polyphonic music
  - **Input**: patches of spectrogram representation $|X(t,f)|$ to
  - **Output**: Time/Frequency mask $M_j(t,f)$
  - **Separation**: apply $M_j(t,f)$ to the amplitude STFT of the mixture $|X(t,f)|$ to separate the amplitude STFT of the isolated source
    - $|S_j(t,f)| = |X(t,f)| \odot M_j(t,f)$





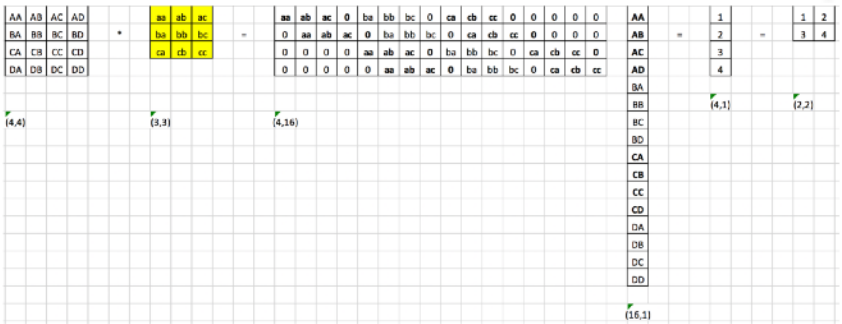
[O. Ronneberger et al. "U-net:Convolutional networks for biomedical image segmentation", 2015]
[A. Jansson et al. "Singing voice separation with deep u-net convolutional networks". In ISMIR, 2017]

Audio applications with DNN using U-Net (up-sampling branch)

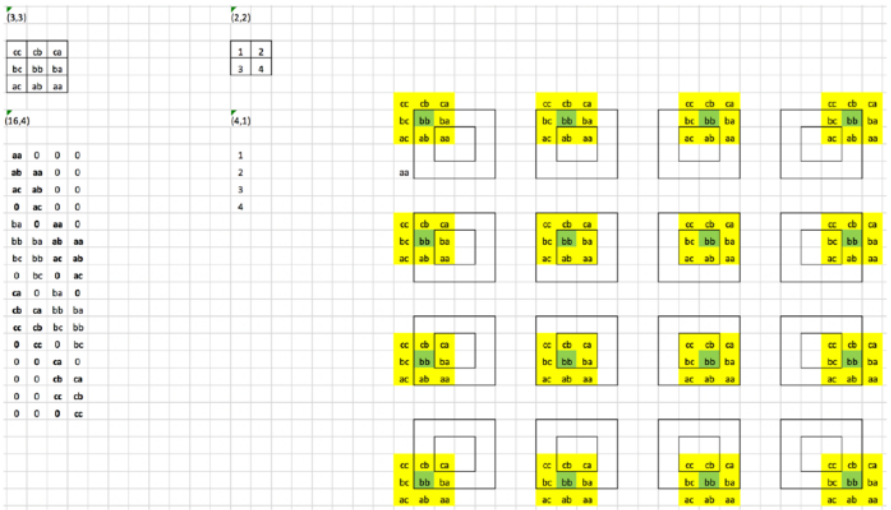
DeConv2D ?

- the term is misleading (there is no de-convolution in the signal processing way)
- It is a convolution with a transposed matrix → **transposed convolution**

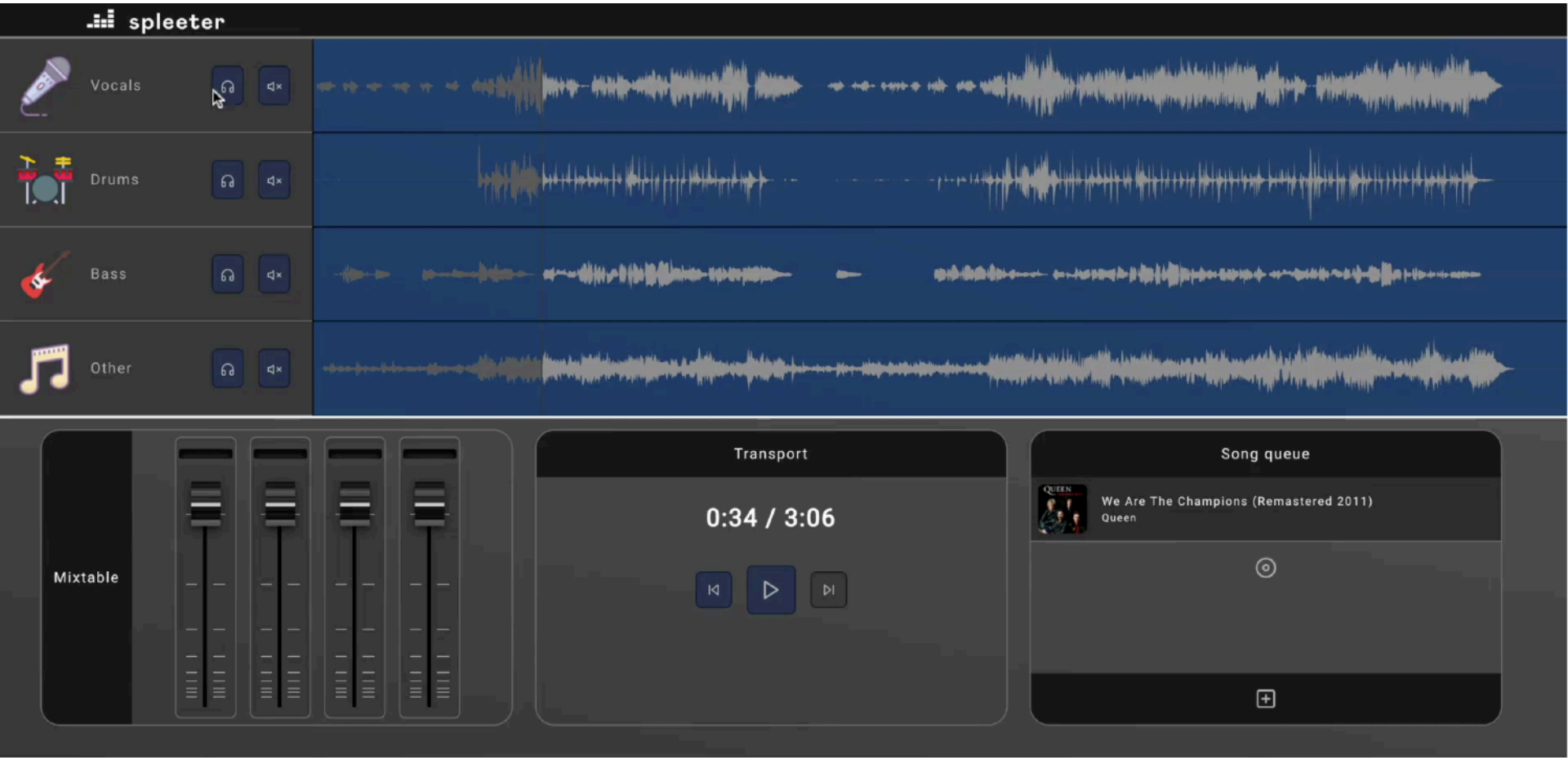
- **Conv2D as a matrix multiplication**
  - $X_{(4,4)} \circledast W_{(3,3)} \Leftrightarrow W^*_{(4,16)} X^*_{(16,1)} = Y^*_{(4,1)} \rightarrow Y_{(2,2)}$



- **Transposed Conv2D**
  - $Y_{(2,2)} \circledast W^T_{(3,3)} \Leftrightarrow \rightarrow W^{*,T}_{(16,4)} Y^*_{(4,1)} = X^*_{(16,1)} \rightarrow X_{(4,4)}$



Audio Source Separation - Systems
U-Net → Spleeter



[R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam. "Spleeter: A fast and state-of-the art music source separation" in LBD-ISMIR 2019]

https://github.com/deezer/spleeter

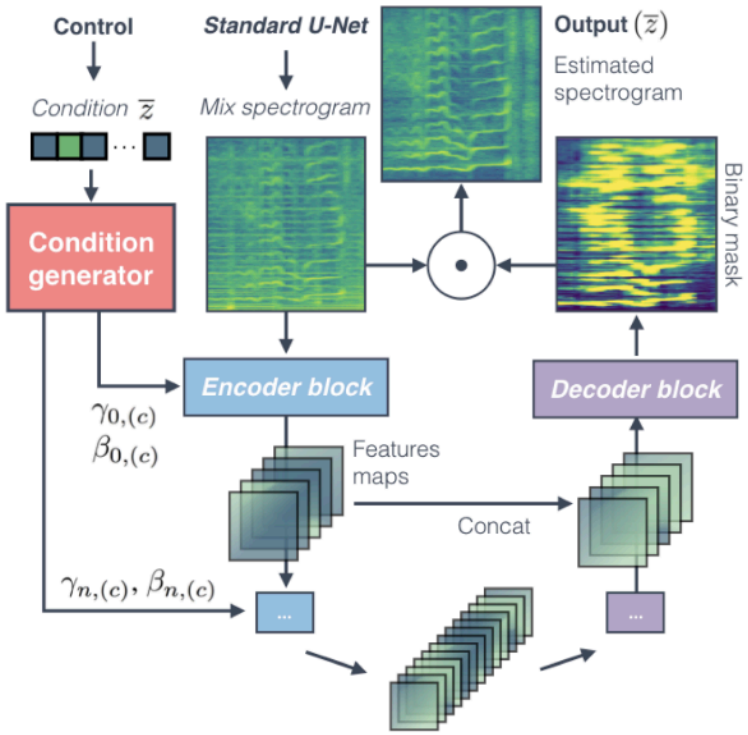
Audio Source Separation - Systems
2019 → do we need to train a new model for each target ?

Idea:

– **Limitations of U-Net**
- we need to train a new model for each source to be separated,

– **Proposal**:
- train a single U-Net which is conditioned on a given target source

– **How to do the conditioning ?**
- Control mechanism
- FiLM (Feature-wise Linear Modulation) layer:
  - apply an independent affine transformation to each layer i and feature map (c)
  - $FiLM_{i,(c)}(x) = \gamma_{i,(c)}(z) \cdot x + \beta_{i,(c)}(z)$
  - $\gamma_{i,(c)}(z)$ and $\beta_{i,(c)}(z)$ are themselves DNN (FC or CNN)
- Condition ?
  - vector $\mathbf{z}$ (a one-hot-encoding) which specify the instrument to be separated, for example $\mathbf{z} = [0,1,0,0]$

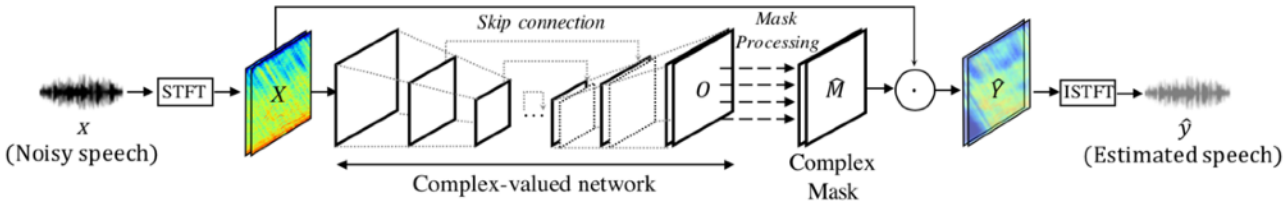


G. Meseguer Brocal and G. Peeters. "Conditioned-u-net: Introducing a control mechanism in the u-net for multiple source separations". In ISMIR, 2019

Audio Source Separation - Systems 2019 → using a Complex-U-Net

Idea:

- **Limitations of U-Net**
  - only trained using magnitude STFT
  - the mask is only applied to the magnitude STFT
- **Proposal**
  - Use a Complex U-Net with complex input/ complex network/ complex output mask



- Complex convolution

$$W = A + iB$$
$$h = x + iy$$
$$W * h = (A * x - B * y) + i(B * x + a * y)$$

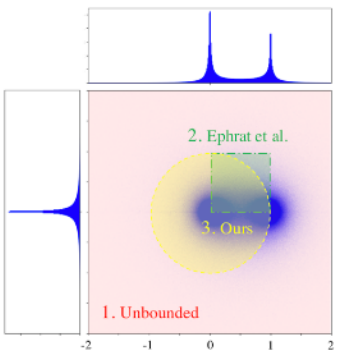
- Complex non-linearity (ℂReLU), pooling, ...
- Complex-value Ideal Ratio Mask (cIRM)

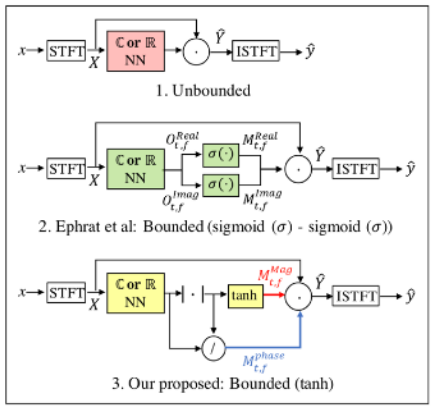
$$\text{cIRM: } M \in \mathbb{C} = \frac{Y}{X} = \frac{|Y|}{|X|} e^{i(\phi_Y - \phi_X)}$$

- Separation:

$$\hat{Y}_{t,f} = \hat{M}_{t,f} \cdot X_{t,f} = |\hat{M}_{t,f}| \cdot |X_{t,f}| \cdot e^{i\left(\phi_{\hat{M}_{t,f}} + \phi_{X_{t,f}}\right)}$$

bounding the cRM





[H.-S. Choi et al. "Phase-aware speech enhancement with deep complex U-net", in ICLR, 2019]

Audio Source Separation Systems

Using Waveform as Input

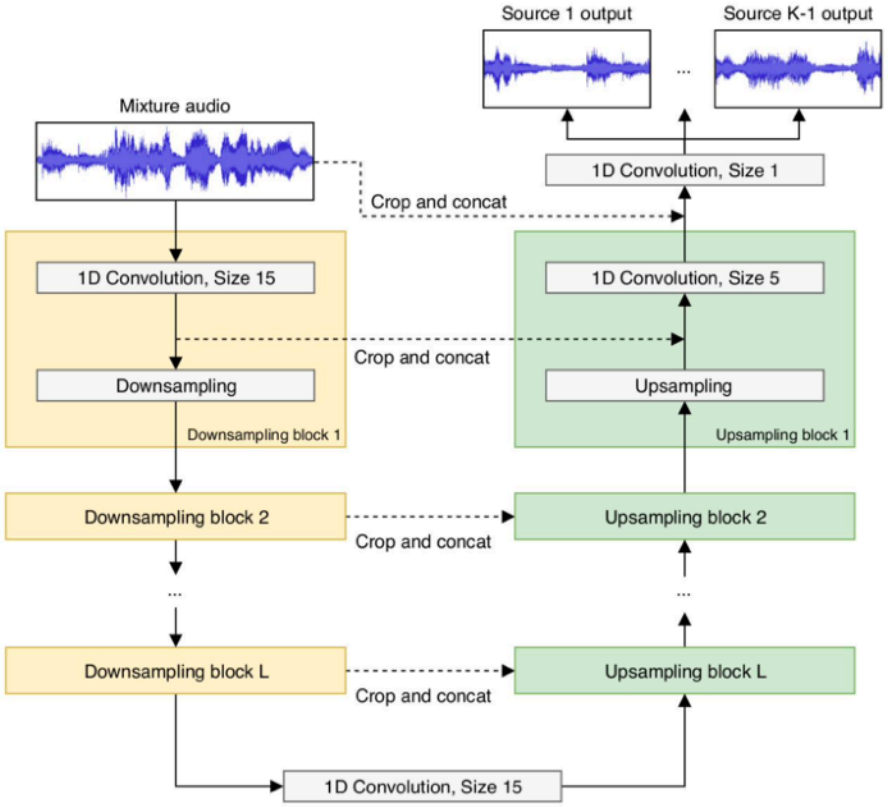
Audio Source Separation - Systems Using Waveform as Input

- **Problems of working in the |STFT| domain**
  - the mask is only applied to |STFT|
  - the audio signal is reconstructed from STFT using original phase
    - not possible to detect/correct potential positive/negative phase interferences
  - using STFT requires choosing analysis parameters (window size, hop size, ...)

- **Working directly in the waveform domain**
  - It is then possible to surpass the Ideal Binary Mask !
  - Two possibilities
    - (a) estimate directly the waveform of separated source
    - (b) estimate mask to be applied to encoded waveform

Audio Source Separation - Systems
2018 → Wave-U-Net

- **Wave-U-Net**
  - estimate directly the waveforms of the separated sources
  - several simultaneous outputs
  - to avoid artifacts:
    - replace *strided-transposed-convolution* (as used in previous works) by upsampling feature maps with *linear interpolation followed by a normal convolution*



[Stoller et al., 2018. "Wave-u-net: A multi-scale neural network for end-to-end audio source separation" in arXiv.]

Audio Source Separation - Systems
2019 → U-Net or Wave-U-Net ?

Idea:

- Is Wave-U-Net better than U-Net ?
- How much data do we need ? Can we solve this with Data Augmentation ?
- Is it better to estimate the STFT or the Mask ?
- Are the skip-connections useful ?

(a) Results for the Wave-U-Net models. Median and median absolute deviation (MAD) of SAR, SIR and SDR.

Model	Dataset used	SAR		SIR		SDR	
		med	MAD	med	MAD	med	MAD
W[8,2]	no-DA	5.52	1.96	10.87	2.02	4.09	2.07
	Stretch[5]	5.60	1.86	**12.11**	2.99	4.20	1.58
	Env.[6]	5.23	1.86	11.22	2.32	3.77	1.61
	Pitch[4]	**6.09**	1.65	10.68	2.40	4.18	1.99
	DA[1]	5.86	1.63	12.02	2.19	**4.67**	1.71
	DA-F[1]	**6.62**	1.80	**13.90**	2.74	**5.42**	1.72
W [9]	No-DA+CCMix[11]					3.96	3.0

(b) Results of U-Net models. Median and MAD of SAR, SIR and SDR.

Model	Dataset used	SAR		SIR		SDR	
		med	MAD	med	MAD	med	MAD
U[7]	no-DA-F[9]	5.76	4.21	11.75	2.05	4.52	2.48
	Stretch-F	5.73	2.28	12.38	2.48	4.85	2.06
	Env.-F	6.06	2.28	11.06	2.66	4.55	2.24
	Pitch-F	6.35	2.21	**12.69**	2.69	**5.20**	2.09
	DA-F	**6.40**	2.20	11.98	2.37	**5.20**	2.22
U [8]	DS-priv[10]	**11.30**		**15.31**			
no-skip	DA-F	5.60	2.39	9.92	2.08	3.44	2.13
no-mask	DA-F	4.87	3.25	**14.71**	3.50	4.18	3.27

[1] DA: all Data Augmentation, [2] 25% of `musdb18` kept for early stopping according to [9], [3] MDB: `MedleyDB`, [23], for the testing phase, [4] Pitch: pitch shifting, [5] Stretch: time stretching, [6] Env.: transformation of the singing voice spectral envelop, [7] U: U-Net model, [8] W: Wave-U-Net model, [9] F: Full `musdb18`, training on full training data - no early stopping, [10]: private dataset see [8], [11] Musdb + CCMixter datasets [9] M3 in table 2.

A. Cohen-Hadria, A. Roebel, and G. Peeters. "Improving singing voice separation using deep u-net and wave-u-net with data augmentation." In Proc. of EUSIPCO, 2019.

Audio Source Separation - Systems
2019 → U-Net or Wave-U-Net ?

Idea:

- Is Wave-U-Net better than U-Net ?
- How much data do we need ? Can we solve this with Data Augmentation ?
- Is it better to estimate the STFT or the Mask ?
- Are the skip-connections useful ?

(a) Results for the Wave-U-Net models. Median and median absolute deviation (MAD) of SAR, SIR and SDR.

Model	Dataset used	SAR		SIR		SDR	
		med	MAD	med	MAD	med	MAD
W[8,2]	no-DA	5.52	1.96	10.87	2.02	4.09	2.07
	Stretch[5]	5.60	1.86	**12.11**	2.99	4.20	1.58
	Env.[6]	5.23	1.86	11.22	2.32	3.77	1.61
	Pitch[4]	**6.09**	1.65	10.68	2.40	4.18	1.99
	DA[1]	5.86	1.63	12.02	2.19	**4.67**	1.71
	DA-F[1]	**6.62**	1.80	**13.90**	2.74	**5.42**	1.72
W [9]	No-DA+CCMix[11]					3.96	3.0

(b) Results of U-Net models. Median and MAD of SAR, SIR and SDR.

Model	Dataset used	SAR		SIR		SDR	
		med	MAD	med	MAD	med	MAD
U[7]	no-DA-F[9]	5.76	4.21	11.75	2.05	4.52	2.48
	Stretch-F	5.73	2.28	12.38	2.48	4.85	2.06
	Env.-F	6.06	2.28	11.06	2.66	4.55	2.24
	Pitch-F	6.35	2.21	**12.69**	2.69	**5.20**	2.09
	DA-F	**6.40**	2.20	11.98	2.37	**5.20**	2.22
U [8]	DS-priv[10]	**11.30**		**15.31**			
no-skip	DA-F	5.60	2.39	9.92	2.08	3.44	2.13
no-mask	DA-F	4.87	3.25	**14.71**	3.50	4.18	3.27

[1] DA: all Data Augmentation, [2] 25% of `musdb18` kept for early stopping according to [9], [3] MDB: `MedleyDB`, [23], for the testing phase, [4] Pitch: pitch shifting, [5] Stretch: time stretching, [6] Env.: transformation of the singing voice spectral envelop, [7] U: U-Net model, [8] W: Wave-U-Net model, [9] F: Full `musdb18`, training on full training data - no early stopping, [10]: private dataset see [8], [11] Musdb + CCMixter datasets [9] M3 in table 2.

A. Cohen-Hadria, A. Roebel, and G. Peeters. "Improving singing voice separation using deep u-net and wave-u-net with data augmentation." In Proc. of EUSIPCO, 2019.

Audio Source Separation - Systems
2019 → U-Net or Wave-U-Net ?

Idea:

- Is Wave-U-Net better than U-Net ?
- How much data do we need ? Can we solve this with Data Augmentation ?
- Is it better to estimate the STFT or the Mask ?
- Are the skip-connections useful ?

(a) Results for the Wave-U-Net models. Median and median absolute deviation (MAD) of SAR, SIR and SDR.

Model	Dataset used	SAR		SIR		SDR	
		med	MAD	med	MAD	med	MAD
W[8,2]	no-DA	5.52	1.96	10.87	2.02	4.09	2.07
	Stretch[5]	5.60	1.86	**12.11**	2.99	4.20	1.58
	Env.[6]	5.23	1.86	11.22	2.32	3.77	1.61
	Pitch[4]	**6.09**	1.65	10.68	2.40	4.18	1.99
	DA[1]	5.86	1.63	12.02	2.19	**4.67**	1.71
	DA-F[1]	**6.62**	1.80	**13.90**	2.74	**5.42**	1.72
W [9]	No-DA+CCMix[11]					3.96	3.0

(b) Results of U-Net models. Median and MAD of SAR, SIR and SDR.

Model	Dataset used	SAR		SIR		SDR	
		med	MAD	med	MAD	med	MAD
U[7]	no-DA-F[9]	5.76	4.21	11.75	2.05	4.52	2.48
	Stretch-F	5.73	2.28	12.38	2.48	4.85	2.06
	Env.-F	6.06	2.28	11.06	2.66	4.55	2.24
	Pitch-F	6.35	2.21	**12.69**	2.69	**5.20**	2.09
	DA-F	**6.40**	2.20	11.98	2.37	**5.20**	2.22
U [8]	DS-priv[10]	**11.30**		**15.31**			
no-skip	DA-F	5.60	2.39	9.92	2.08	3.44	2.13
no-mask	DA-F	4.87	3.25	**14.71**	3.50	4.18	3.27

[1] DA: all Data Augmentation, [2] 25% of `musdb18` kept for early stopping according to [9], [3] MDB: `MedleyDB`, [23], for the testing phase, [4] Pitch: pitch shifting, [5] Stretch: time stretching, [6] Env.: transformation of the singing voice spectral envelop, [7] U: U-Net model, [8] W: Wave-U-Net model, [9] F: Full `musdb18`, training on full training data - no early stopping, [10]: private dataset see [8], [11] Musdb + CCMixter datasets [9] M3 in table 2.

A. Cohen-Hadria, A. Roebel, and G. Peeters. "Improving singing voice separation using deep u-net and wave-u-net with data augmentation." In Proc. of EUSIPCO, 2019.

Audio Source Separation - Systems
2018 → Tas-Net

Idea :

- reformulate the system as an encoder/masker/decoder
- use LSTM for masker

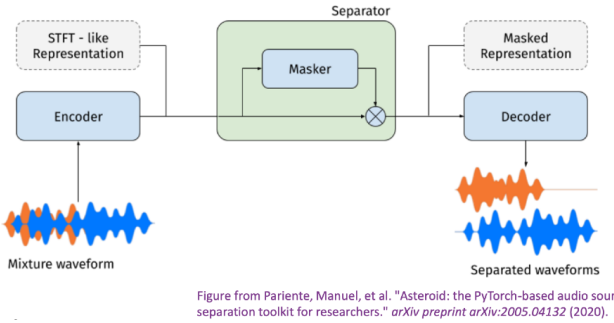
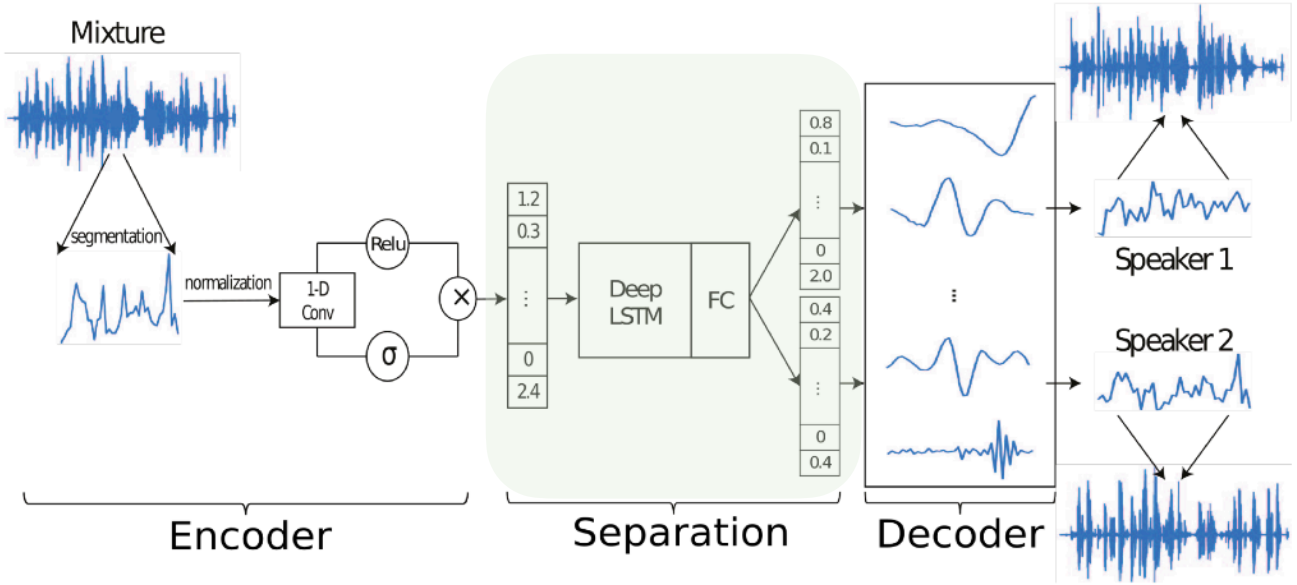


Figure from Pariente, Manuel, et al. "Asteroid: the PyTorch-based audio source separation toolkit for researchers." *arXiv preprint arXiv:2005.04132* (2020).

- **TasNet**: **T**ime-domain **A**udio **S**eparation network
  - **Encoder**: Learn a projection of the audio waveform
  - **Separation**: Learn the mask to be applied to the projection
  - **Decoder**: regenerate the separated audio from the masked projection



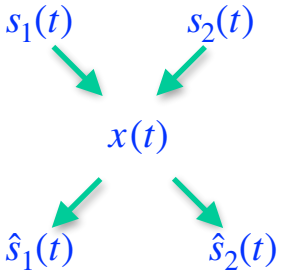
[Y. Luo and N. Mesgarani "TasNet: time-domain audio separation network" in ICASSP 2018.]

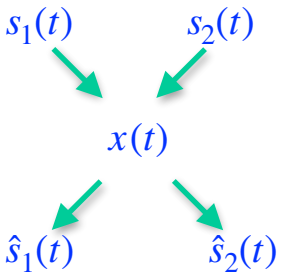
Audio Source Separation - Systems
2018 → Conv-Tas-Net

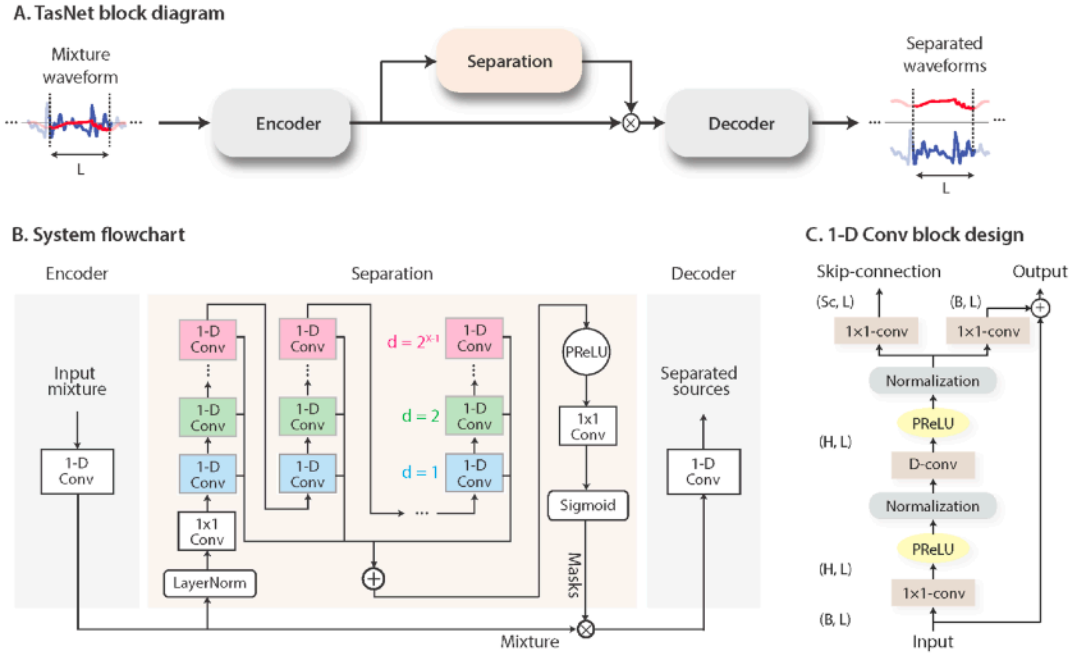
Idea:

- Fully convolutional

- **ConvTasNet**: **T**ime-domain **A**udio **S**eparation network
  - **Encoder**: Learn a projection of the audio waveform
  - **Separation**: Learn the mask to be applied to the projection
  - **Decoder**: regenerate the separated audio from the masked projection

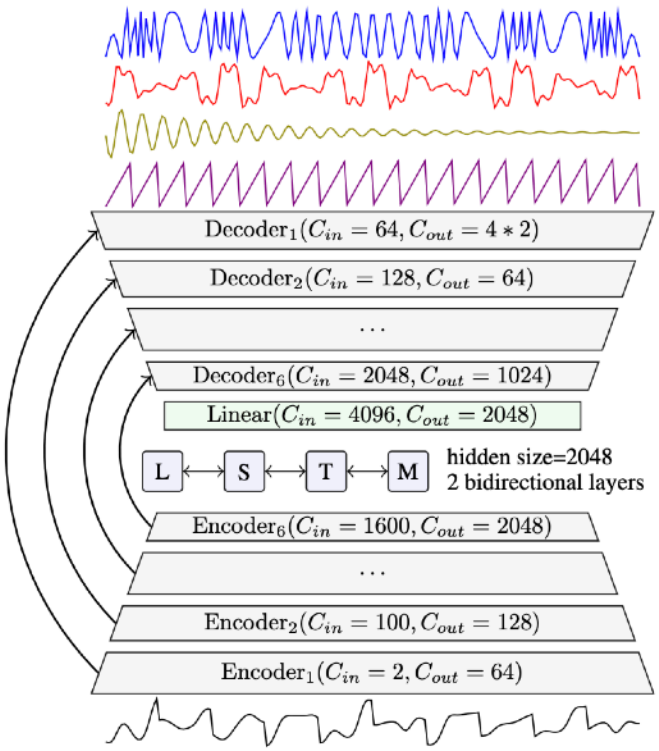




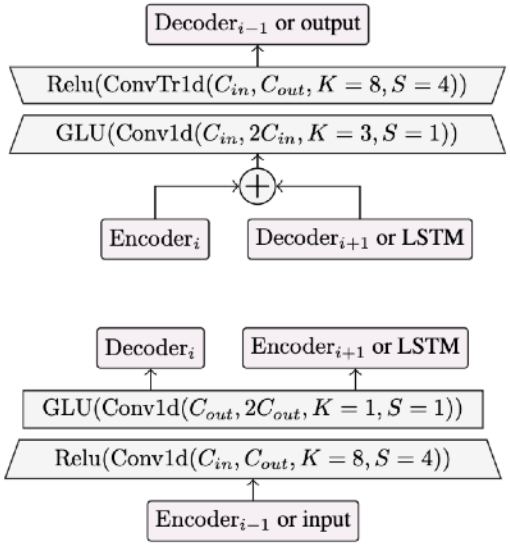


[Y. Luo and N. Mesgarani. "Conv-TasNet: Surpassing ideal time–frequency ..." IEEE TASLP, 2019]

Audio Source Separation - Systems
2019 → Demucs



(a) Demucs architecture with the mixture waveform as input and the four sources estimates as output. Arrows represents U-Net connections.



(b) Detailed view of the layers Decoder$_i$ on the top and Encoder$_i$ on the bottom. Arrows represent connections to other parts of the model. For convolutions, C_in (resp C_out) is the number of input channels (resp output), K the kernel size and S the stride.

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

Audio Source Separation - Systems
2020 → SEGAN

Idea:

- Formulate the separation/denoising problem as a generative process
- Use GAN
- **SEGAN: Speech Enhancement Using Generative Adversarial Networks**

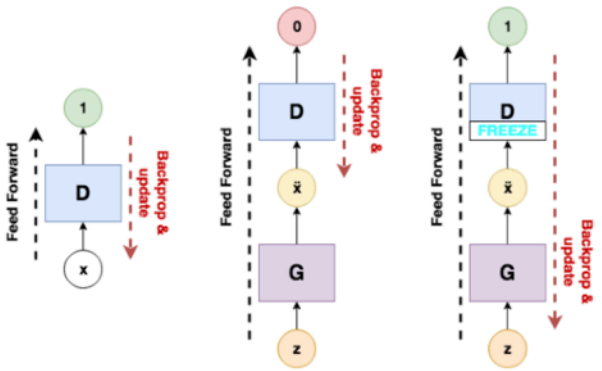


Figure 1: Schema of the GAN training process. First, D back-props a batch of real examples (left). Then, D back-props a batch of synthetic examples that come from G and classifies them as synthetic (middle). Finally, the D parameters are frozen, and G back-props to make D misclassify the examples (right).

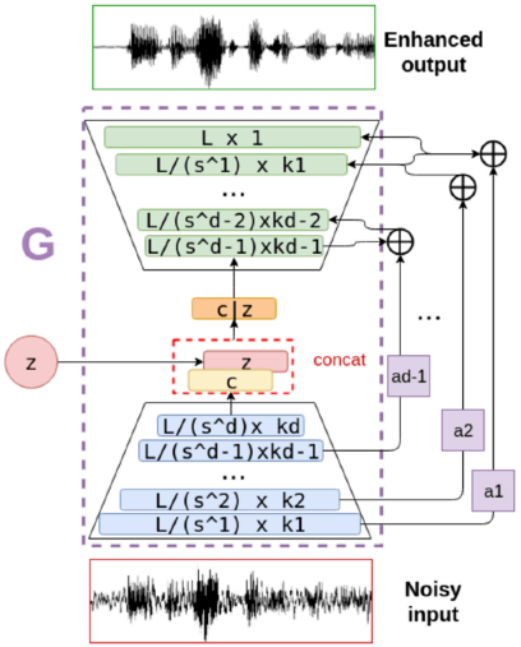


Figure 2: Autoencoder architecture for speech enhancement (G network). Feature maps are depicted in blue and green. The decimation/interpolation factor s^d depends on the stride s and layer depth index d. The input waveform length is designated L, and the number of kernels/channels at each layer is k_d. The right-side arrows denote skip connections, which have a multiplicative scalar factor a_d.

[S. Pascuala, J. Serra and A. Bonafontea, "Time-domain Speech Enhancement Using Generative Adversarial Networks", Speech Com. 2020]