TSIA-206

# Deep Learning for Audio

## *Source Separation/ DDSP/ Gen-AI for Music*

Geoffroy Peeters

contact: geoffroy.peeters@telecom-paris.fr

Télécom-Paris, IP-Paris, France

version: 2024-05-03-15-07

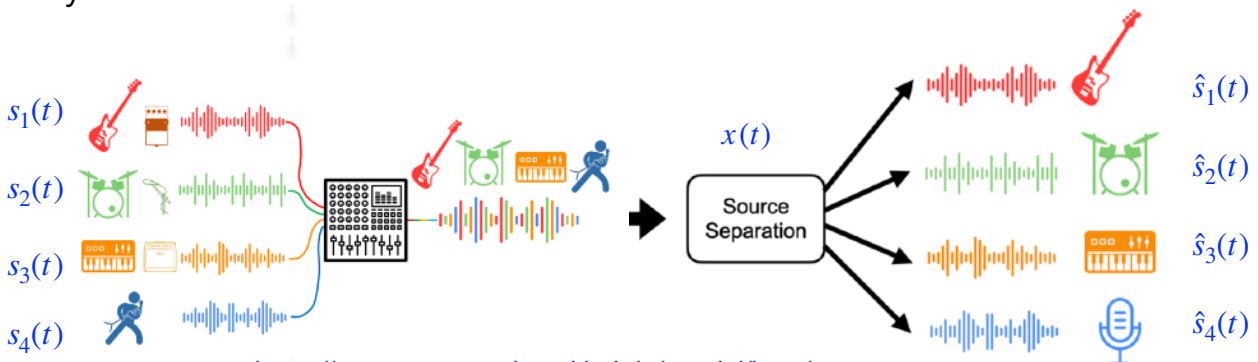
# Audio Source Separation

# 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

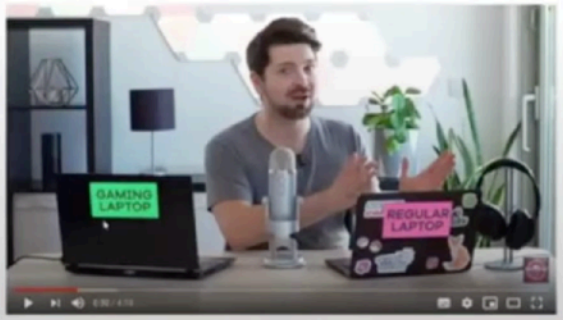
## Applications

– automatic music transcription, musical instrument detection
– lyric and music alignment, lyric recognition, automatic singer identification
– vocal activity detection
– fundamental frequency estimation
– understanding the predictions of black-box audio models

# Speech enhancement

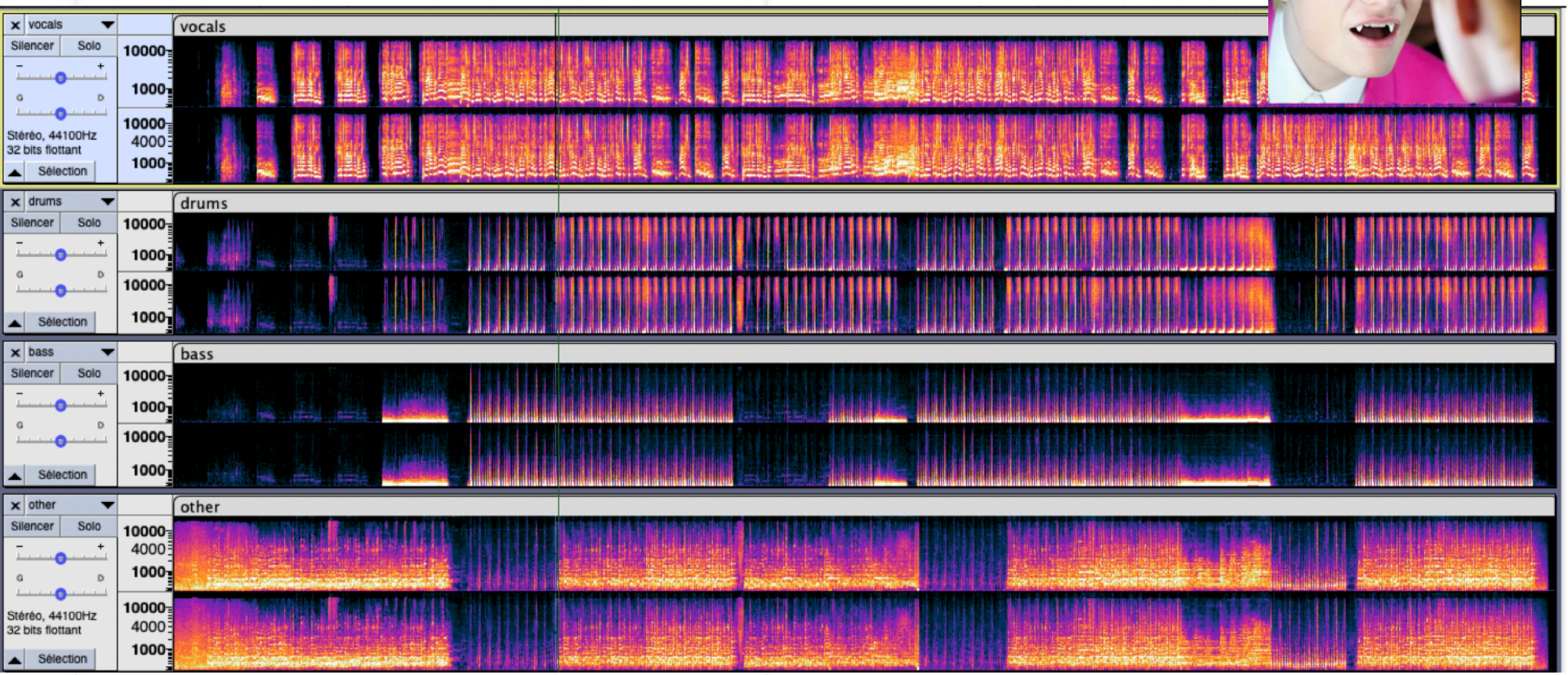
Google Meet noise cancellation

NVIDIA RTX Voice



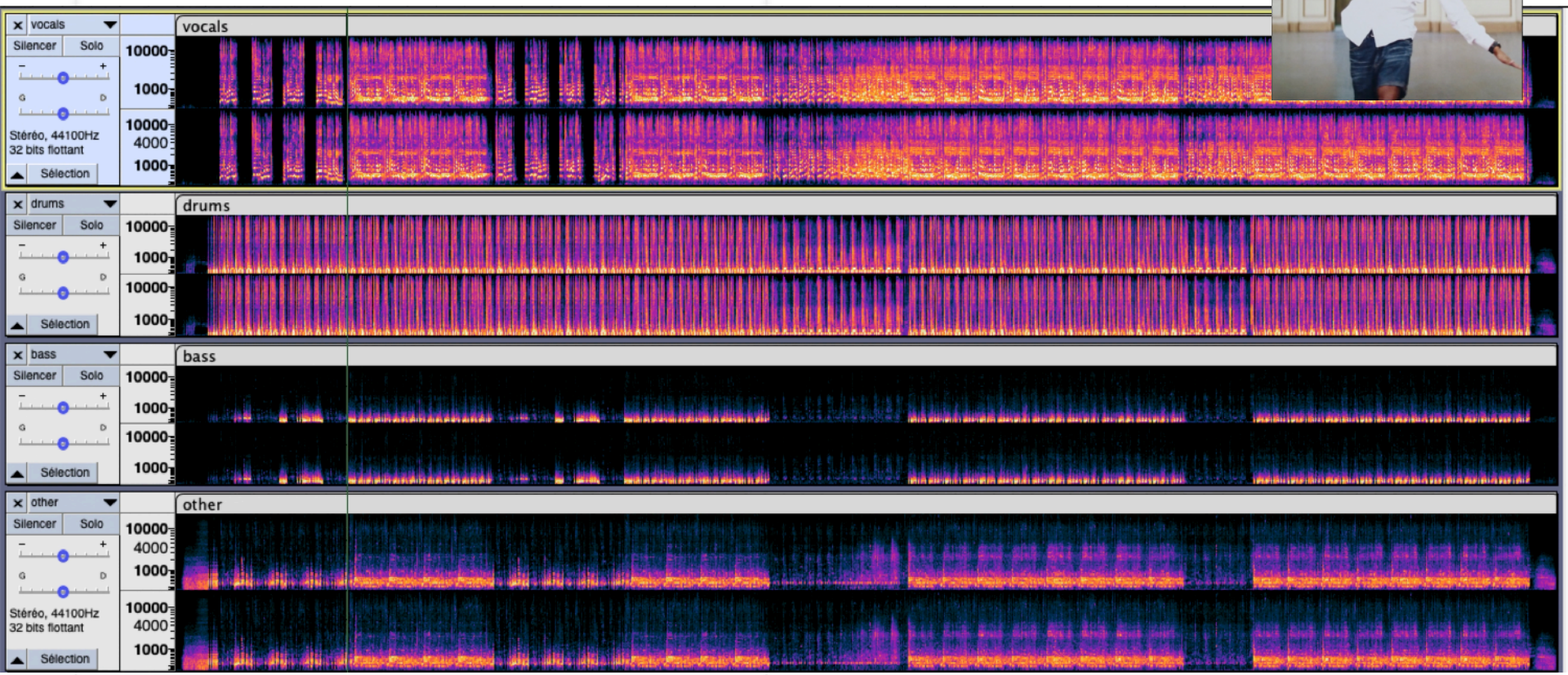

krisp.ai

# Audio Source Separation - Systems
## 2021 → Demucs 2


[A. Défossez,"**Hybrid Spectrogram and Waveform Source Separation**", 2021.] LINK

# Audio Source Separation - Systems
## 2021 → Demucs 2


[A. Défossez,"**Hybrid Spectrogram and Waveform Source Separation**", 2021.] LINK

# Audio Source Separation Systems

# 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
  - Model trained to transform [x=an input mixed signal] to
    - [$\hat{y}$ =an output separated **source** $s_j(t)$]
    - [$\hat{y}$=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
  - Others:
    - Metric learning: Deep Clustering
    - Mix-It
    - Learning to Separate Sounds From Weakly Labeled Scene

# 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

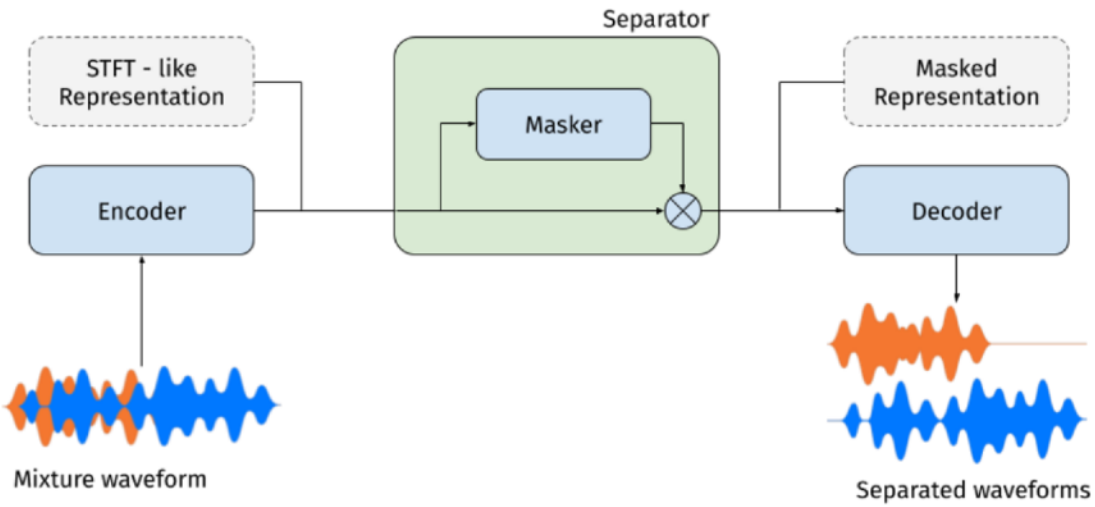


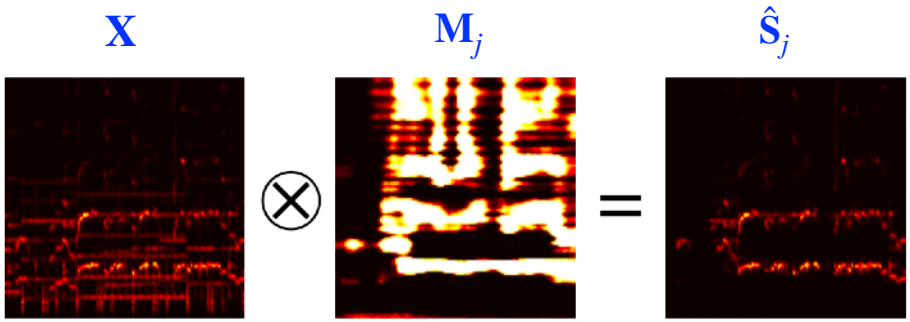

Figure from Pariente, Manuel, et al. "Asteroid: the PyTorch-based audio source separation toolkit for researchers." *arXiv preprint arXiv:2005.04132* (2020).

# Audio Source Separation
## Masks ?

- Given a mixture $x(t)$ of $C$ sources $s_i(t)$:

$$x(t) = \sum_{j=1}^{C} s_j(t) \quad \Leftrightarrow \quad X(t,f) = \sum_{j=1}^{C} S_j(t,f)$$

  - Estimated magnitude STFT of source $j$
    - $\hat{\mathbf{S}}_j = \mathbf{X} \odot \mathbf{M}_j$
      - notation: $\mathbf{X} = |X(t,f)|$
      - subject to $\sum_{j=1}^{C} \mathbf{M}_j = 1$




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

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

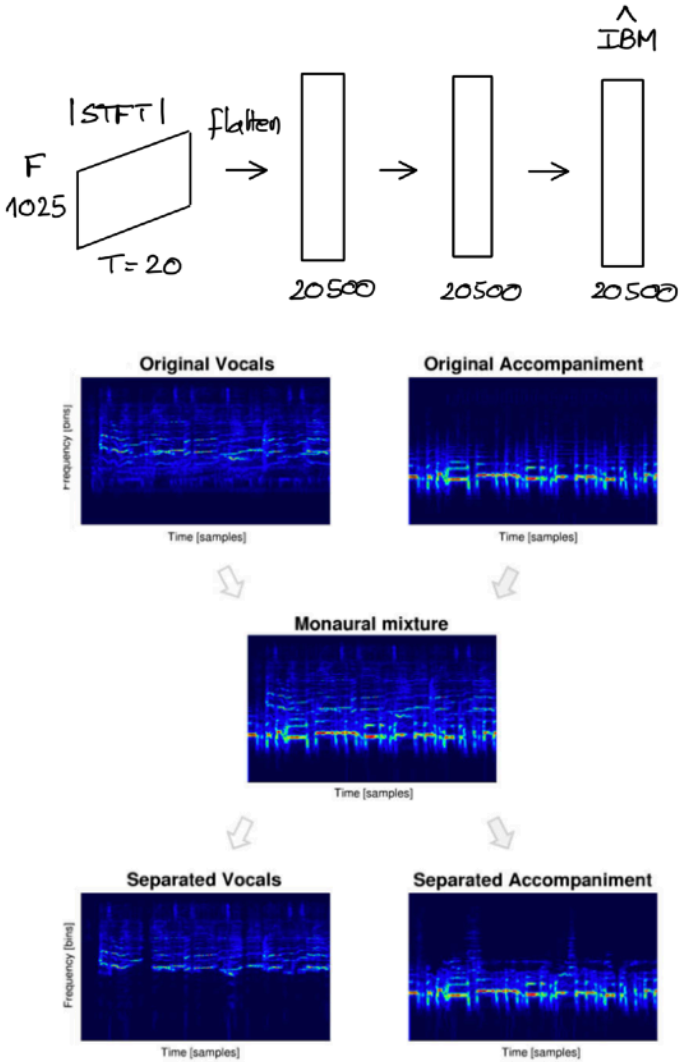
# Audio Source Separation Systems

## 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

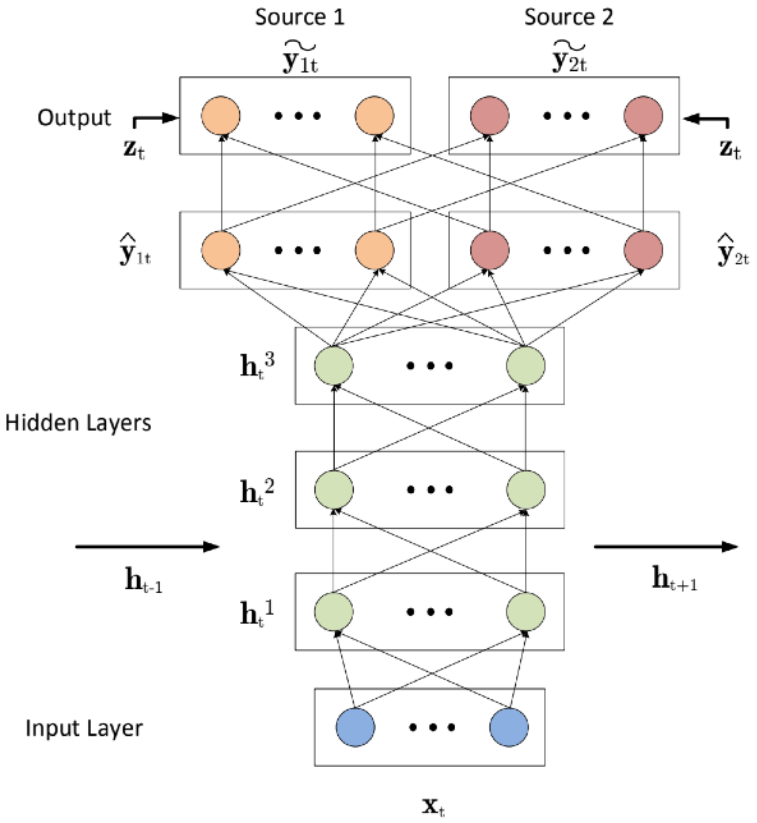



[Andrew JR Simpson et al. "**Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network**". In International Conference on Latent Variable Analysis and Signal Separation, 2015] LINK

# 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

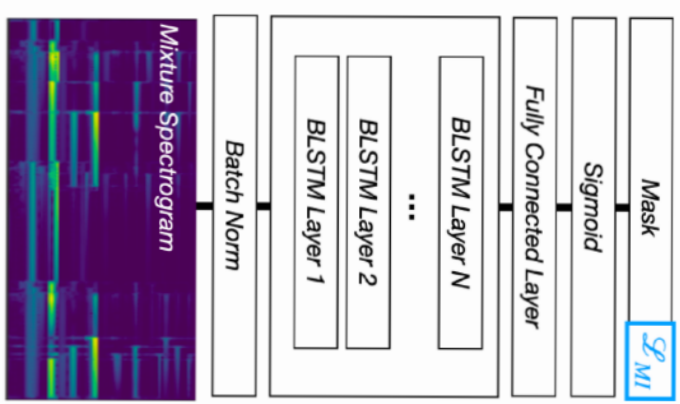


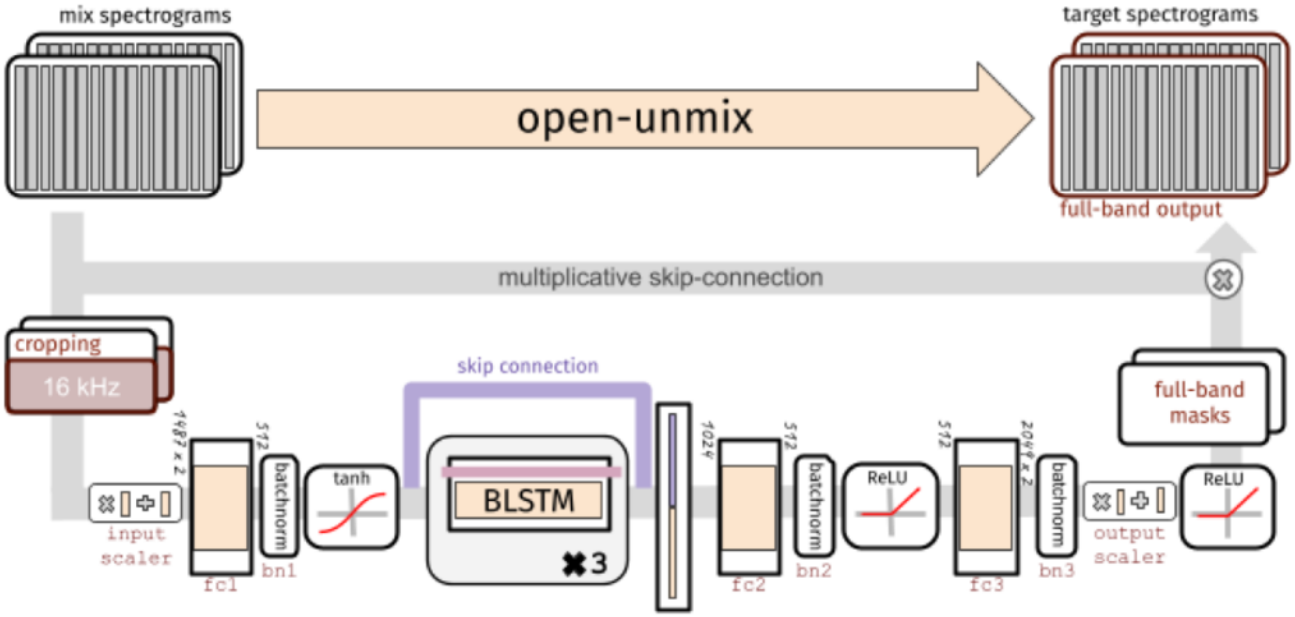

Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. "**Singing-voice separation from monaural recordings using deep recurrent neural networks**". In ISMIR 2014, 2014] LINK

# Audio Source Separation - Systems
## 2019 → Open-Unmix 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



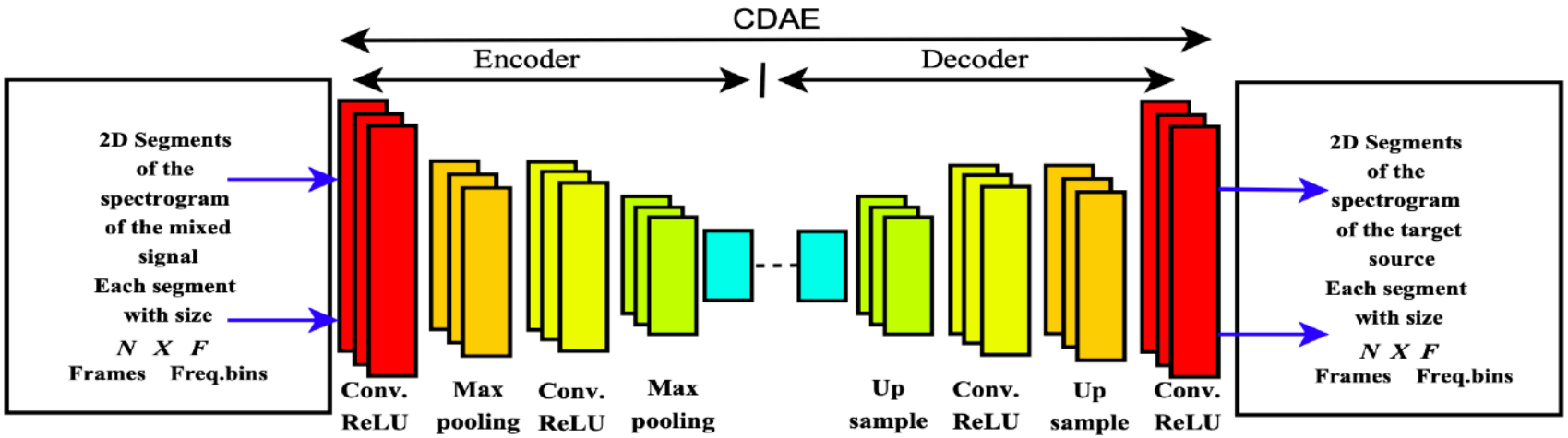




F.-R. Stöter et al, "**Open-Unmix - A Reference Implementation for Music Source Separation**", Journal of Open Source Software, 2019 LINK

# 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)



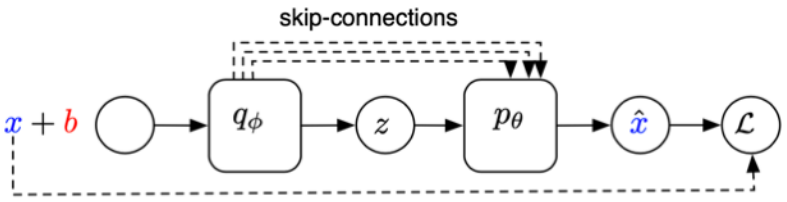

[Emad M Grais et al."**Single channel audio source separation using convolutional denoising autoencoders**", 2017] LINK

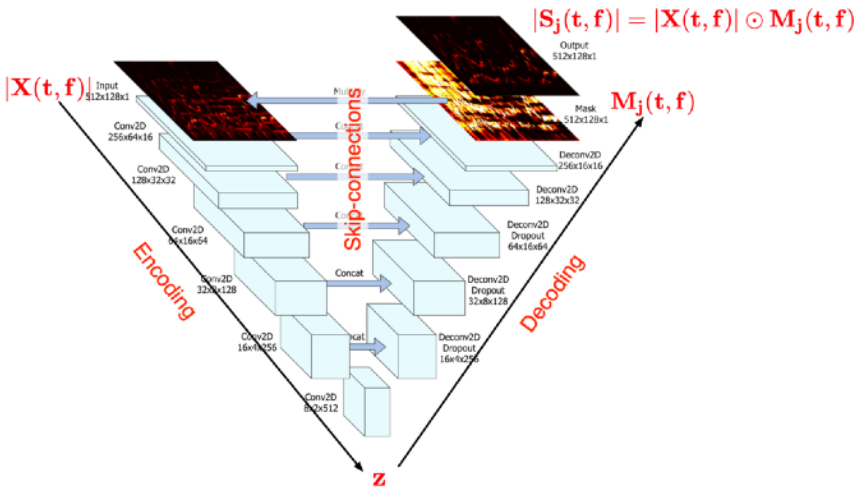
# Audio Source Separation - Systems
## 2017 → using CDAE with skip-connection (U-Net)

**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 localisation
- **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

$$|S_j(t,f)| = |X(t,f)| \odot M_j(t,f)$$

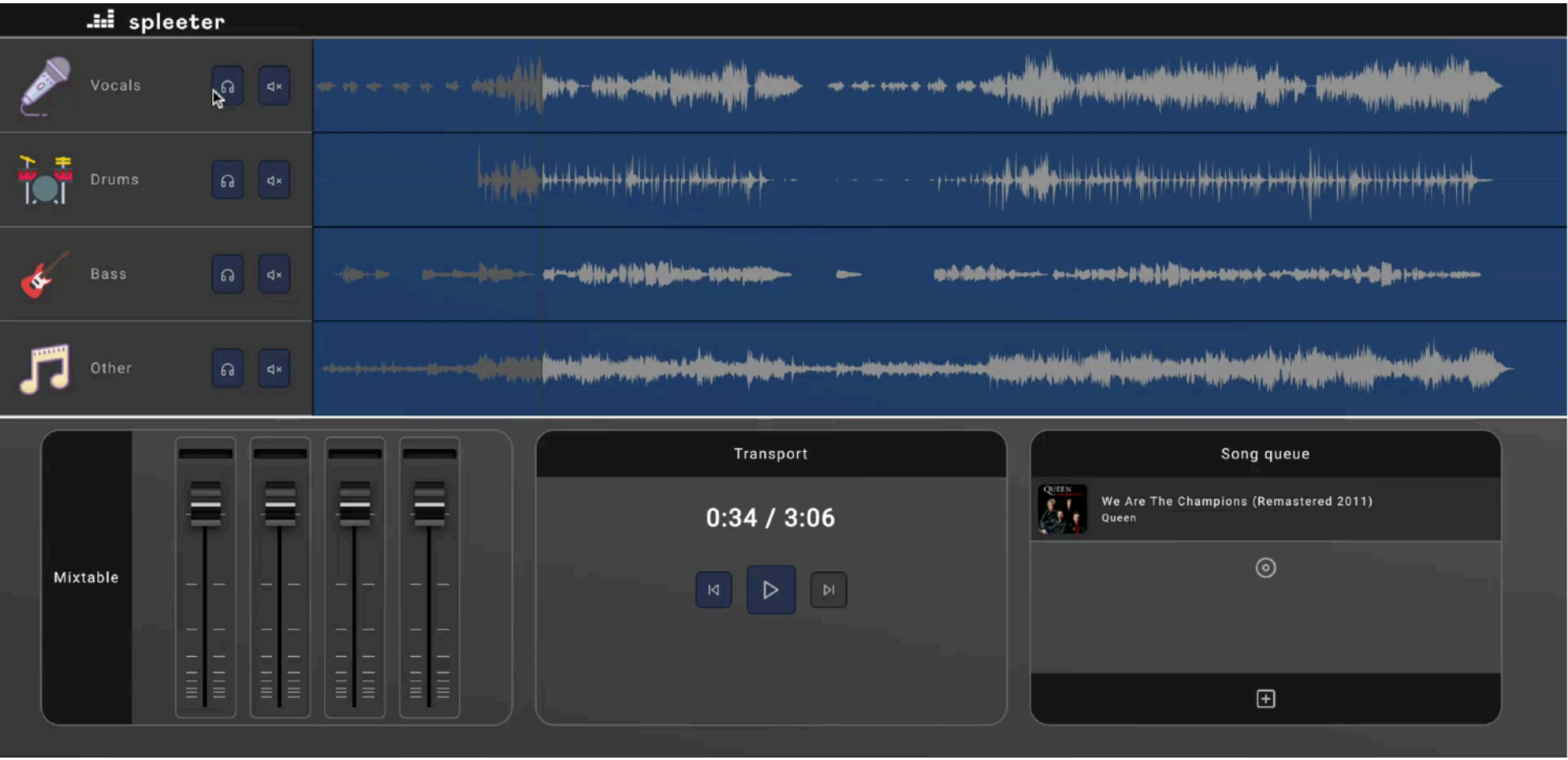




[O. Ronneberger et al. "**U-net:Convolutional networks for biomedical image segmentation**", 2015] LINK
[A. Jansson et al. "**Singing voice separation with deep u-net convolutional networks**". In ISMIR, 2017] LINK

# Audio Source Separation - Systems
## U-Net → Spleeter




[R. Hennequin et al. "**Spleeter: A fast and state-of-the art music source separation**" in LBD-ISMIR 2019] LINK
https://github.com/deezer/spleeter

# Audio Source Separation Systems

## Using Waveform as Input

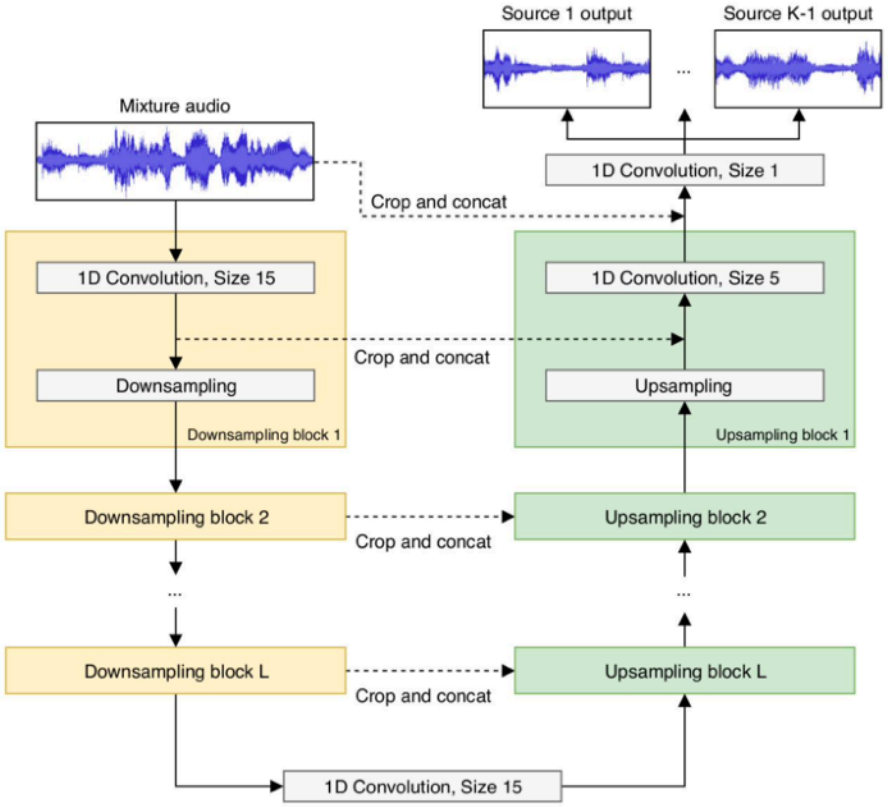
# Audio Source Separation - Systems Using Waveform as Input

- **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 ISMIR, 2018] LINK

# Audio Source Separation - Systems
## 2018 → TasNet

**Idea :**

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

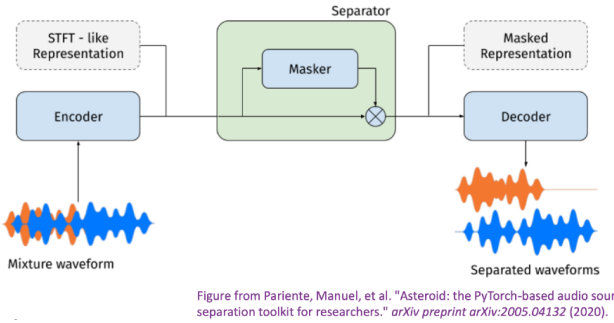
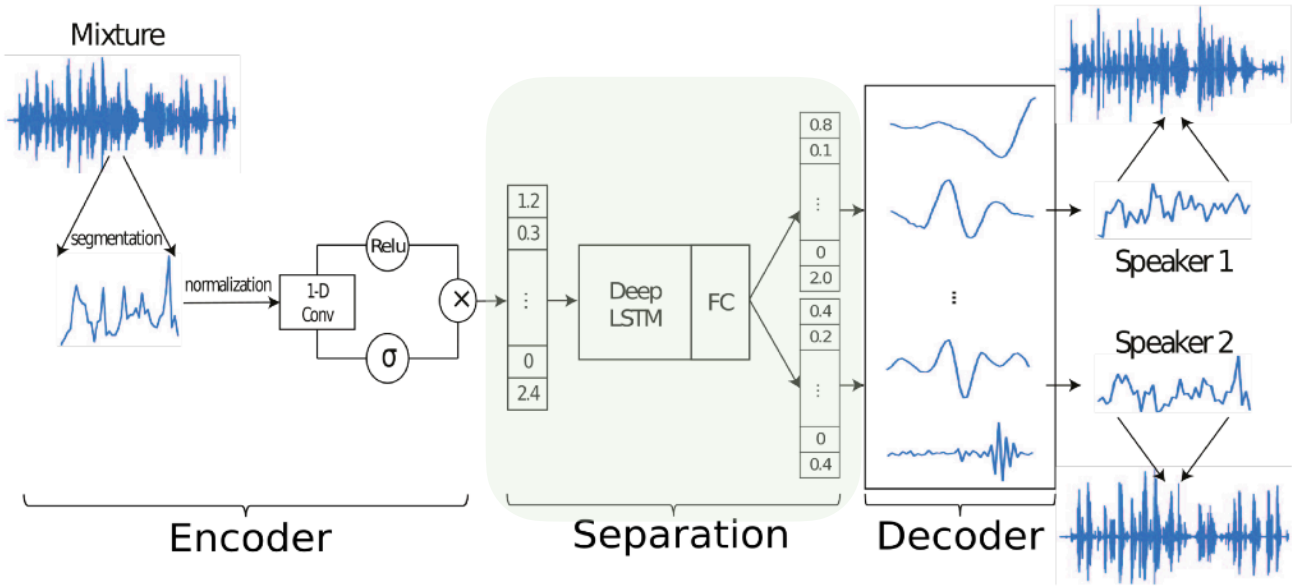



Figure from Pariente, Manuel, et al. "Asteroid: the PyTorch-based audio source separation toolkit for researchers." *arXiv preprint arXiv:2005.04132* (2020).

- **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



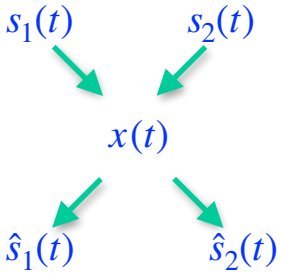

[Y. Luo and N. Mesgarani "**TasNet: time-domain audio separation network**" in ICASSP 2018.] LINK

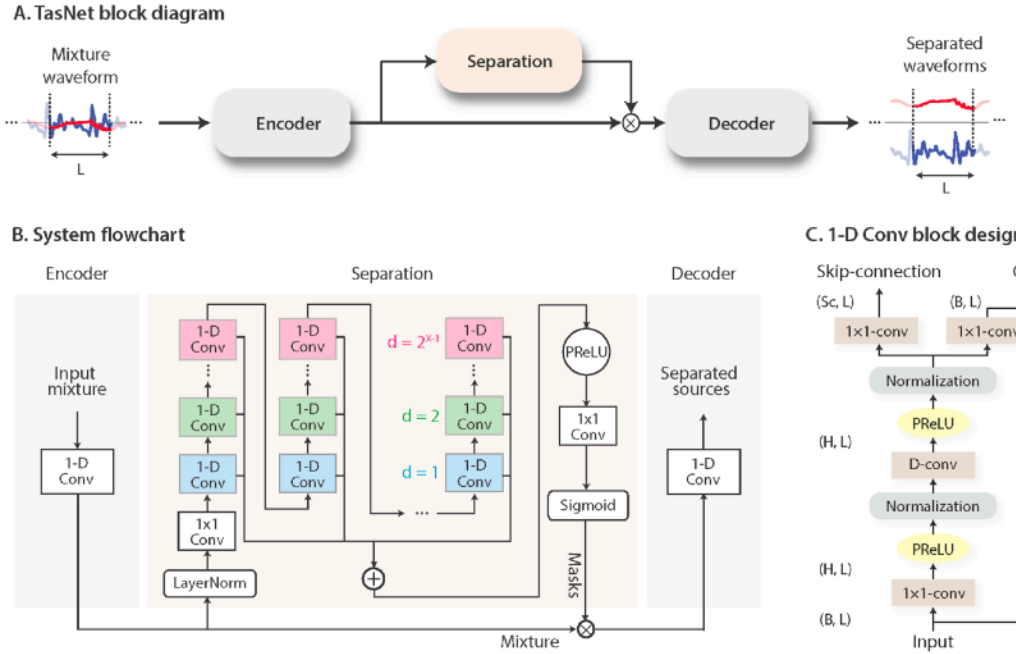
# Audio Source Separation - Systems
## 2018 → Conv-TasNet

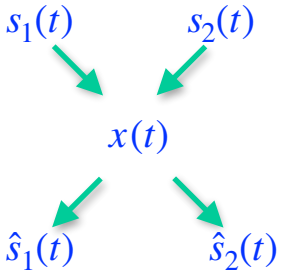
**Idea:**

- Fully convolutional

- **Conv-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

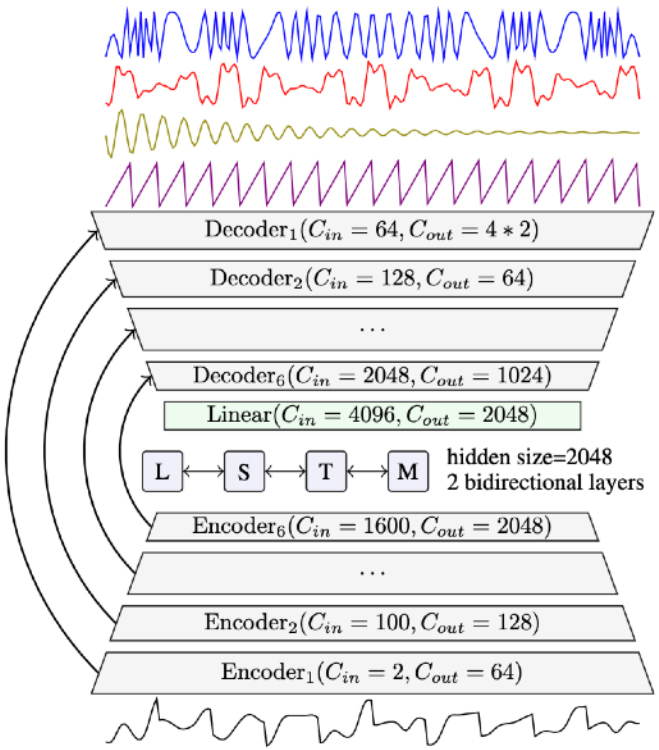


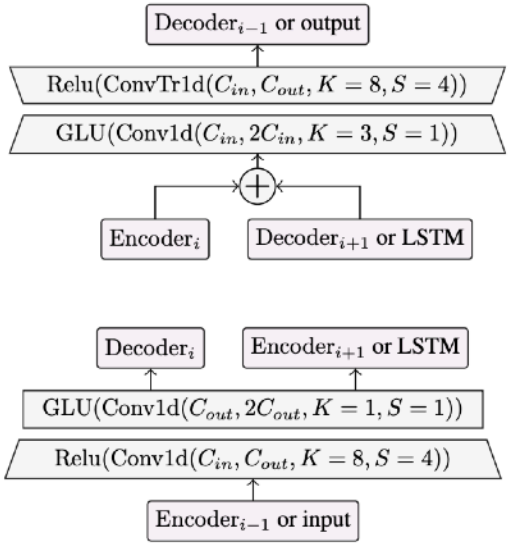




[Y. Luo and N. Mesgarani. "**Conv-TasNet: Surpassing ideal time–frequency ...**" IEEE TASLP, 2019] LINK

# Audio Source Separation - Systems
## 2019 → Demucs



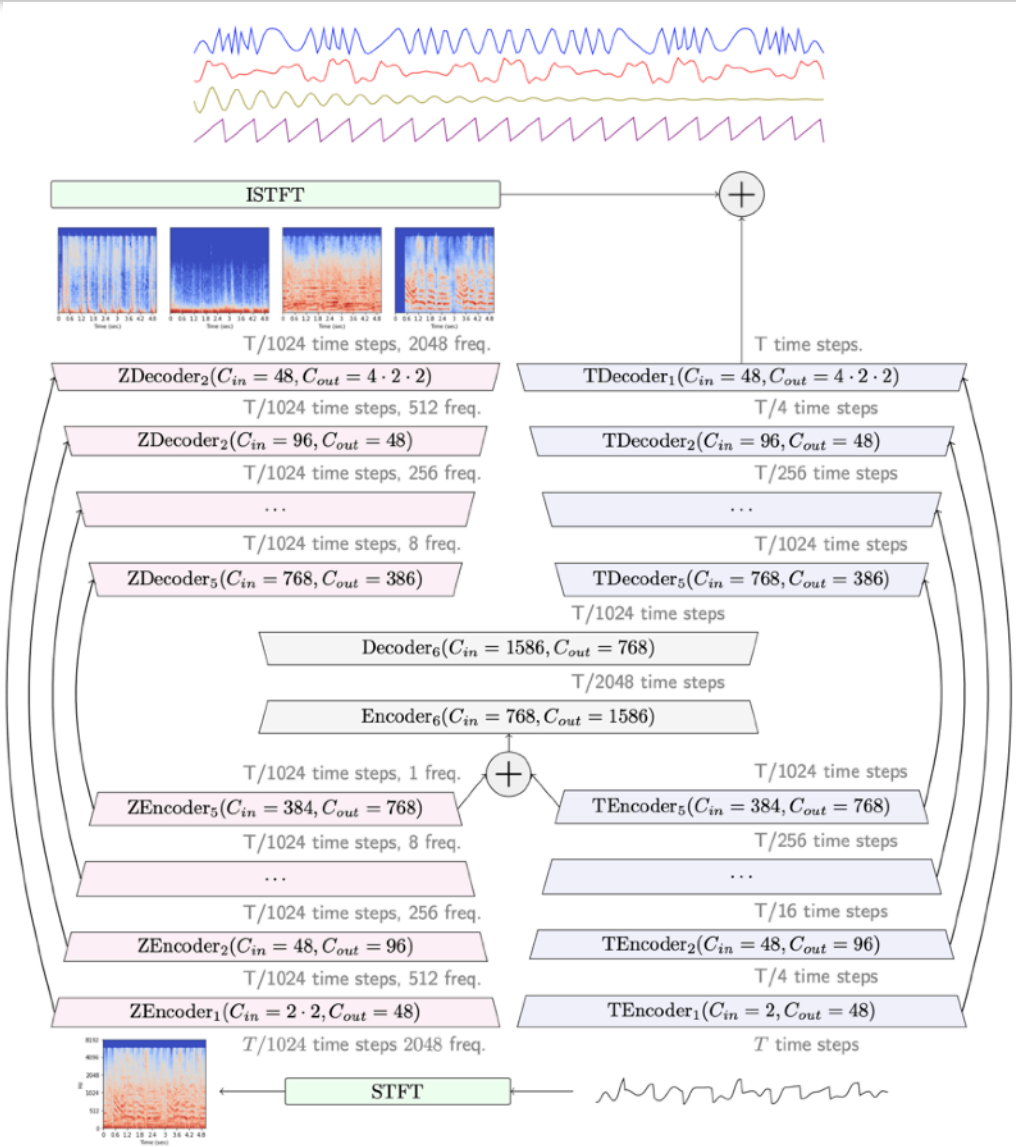

(a) Demucs architecture with the mixture waveform as input and the four sources estimates as output. Arrows represents U-Net connections.




(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, et al. "**Music source separation in the waveform domain**". arXiv preprint arXiv:1911.13254, 2019.] LINK

# Audio Source Separation - Systems
## 2021 → Demucs 2




[A. Défossez,"**Hybrid Spectrogram and Waveform Source Separation**", 2021.] LINK

# 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**

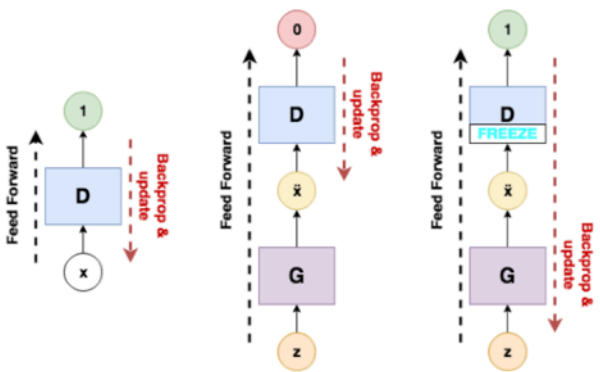



Figure 1: Schema of the GAN training process. First, $D$ back-props a batch of real examples (left). Then, $D$ back-props a batch of synthetic examples that come from $G$ and classifies them as synthetic (middle). Finally, the $D$ parameters are frozen, and $G$ back-props to make $D$ misclassify the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

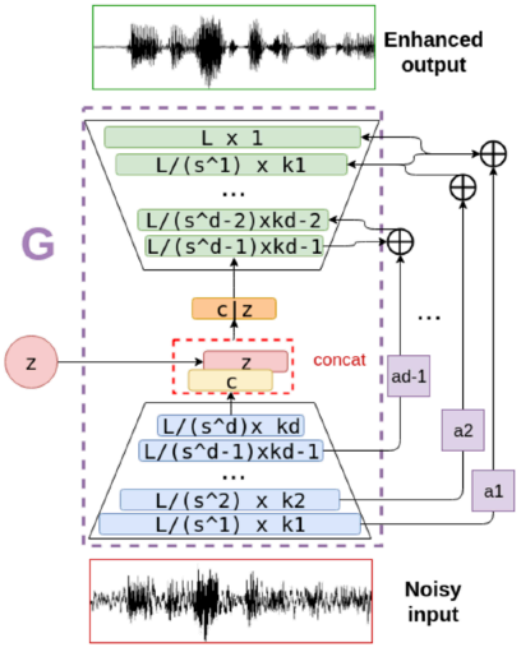



Figure 2: Autoencoder architecture for speech enhancement ($G$ network). Feature maps are depicted in blue and green. The decimation/interpolation factor $s^d$ depends on the stride $s$ and layer depth index $d$. The input waveform length is designated $L$, and the number of kernels/channels at each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

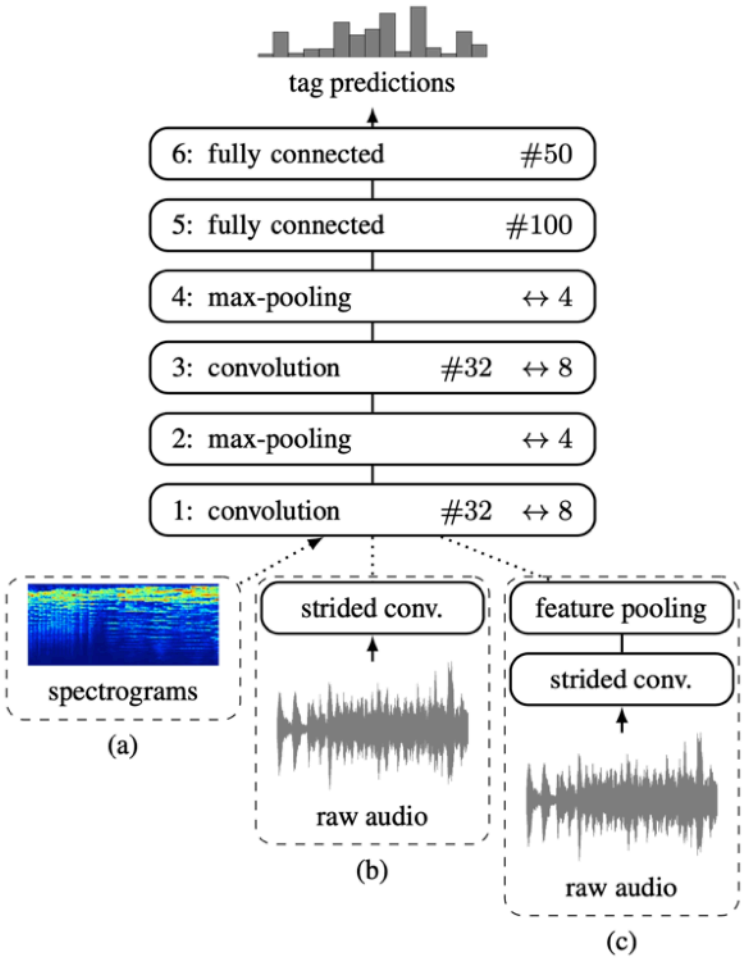
[S. Pascuala, J. Serra and A. Bonafontea, "**Time-domain Speech Enhancement Using Generative Adversarial Networks**", Speech Com. 2020]

# Differentiable Signal Processing

## 1D-Convolution on the audio waveform

– **2014 → Dieleman:**

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




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

## 2018 → SincNet

- Problems of 1D-convolution
  - 1D-convolution filters are often difficult to interpret
  - Try to interpret the learned-filters un the frequency domain
- **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 speaker recognition is much improved

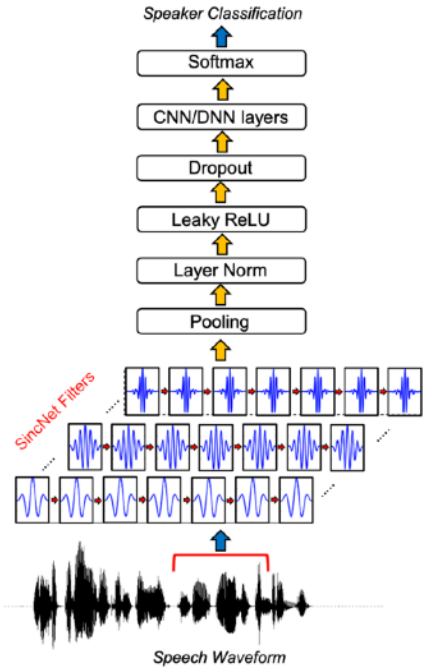



**Fig. 1**: Architecture of SincNet.

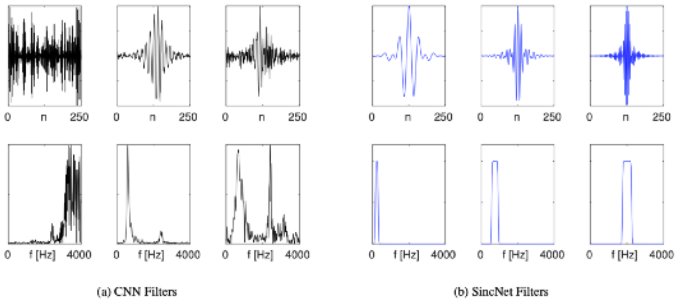



**Fig. 2**: Examples of filters learned by a standard CNN and by the proposed SincNet (using the Librispeech corpus). The first row reports the filters in the time domain, while the second one shows their magnitude frequency response.

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

# Sinusoidal (harmonic) model

**Model:**

- represent the signal $x(t)$ as a sum of (harmonic) sinusoidal components + noise

$$x(t) = \sum_{h=1}^{H(t)} A_h(t) \cdot \cos(\phi_h(t)) + b(t)$$

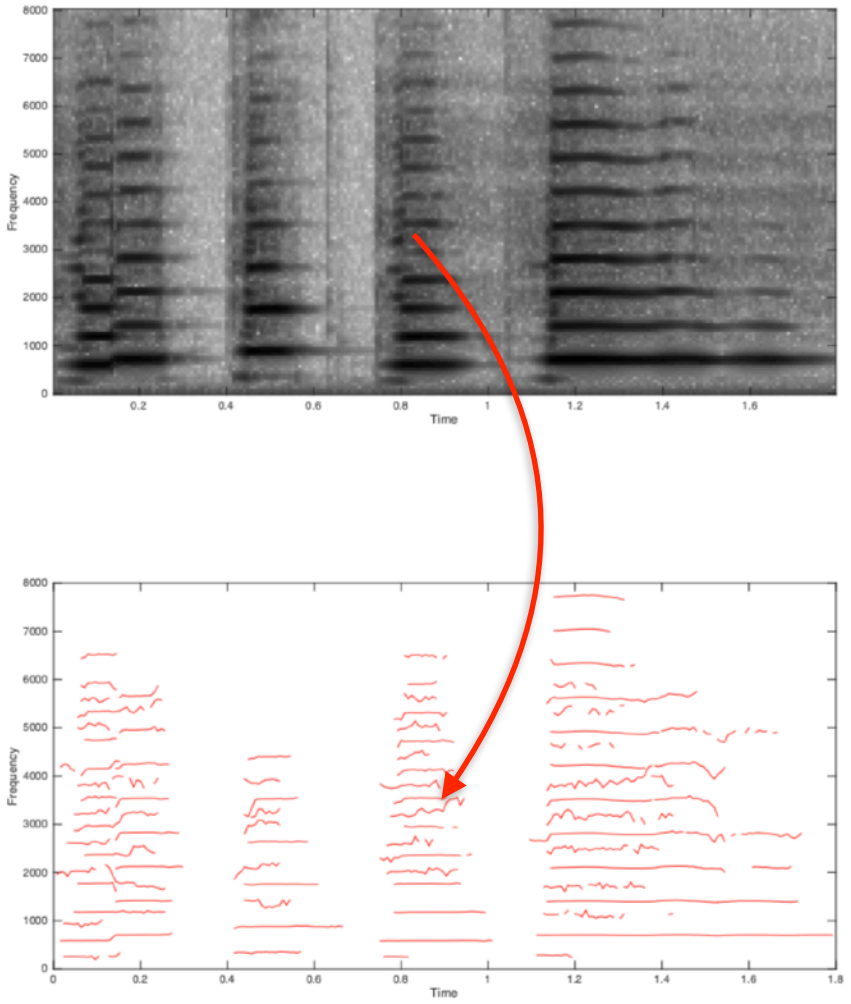
- Fourier series
- Organ musical instrument

**Hypothesis:**

- Number of components $H(t)$ is reduced ($\neq$ Fourier Transform)
- Parameters slowly vary over time
  - local stationarity
  - we can approximate locally in time
    - $A_h(t) = A_h(t_m)$
    - $\phi(t) = 2\pi f_h(t_m)(t - t_m) + \phi_h(t_m)$

$$\hat{x}_m(t_m) = \sum_{h=1}^{H(t)} A_h(t_m) \cdot \cos\left(2\pi f_h(t_m)(t - t_m) + \phi_h(t_m)\right)$$

$$\hat{x}(t) = \sum_{m} \hat{x}_m(t) + b(t)$$

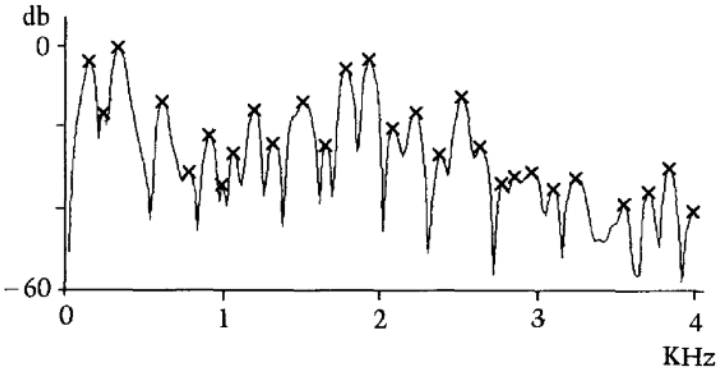


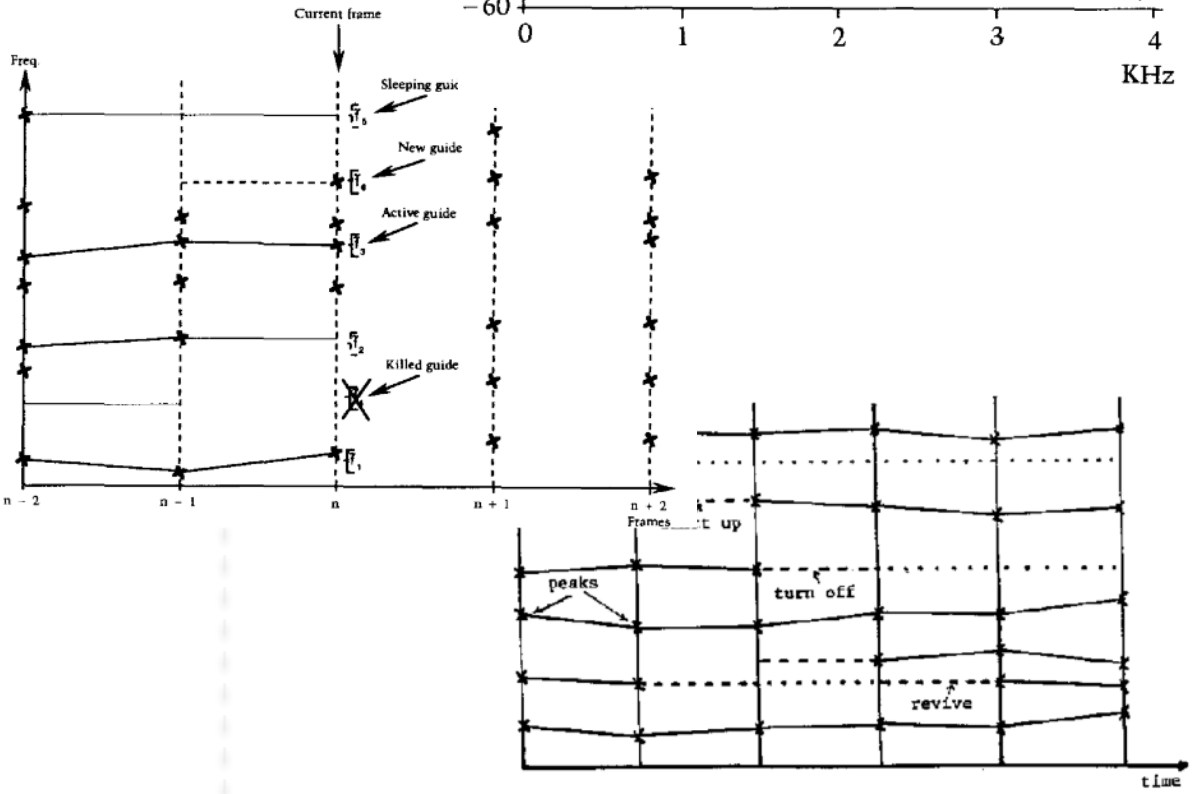

[X. Serra and J. Smith. "**Spectral modeling synthesis: A sound analysis/synthesis system** ..." In CMJ, 1990] LINK

# Sinusoidal (harmonic) model

**Parameters estimation:**

- Complex algorithms
  - DFT peak detection, value refinement
  - partial tracking over time
  - birth and death of partisls

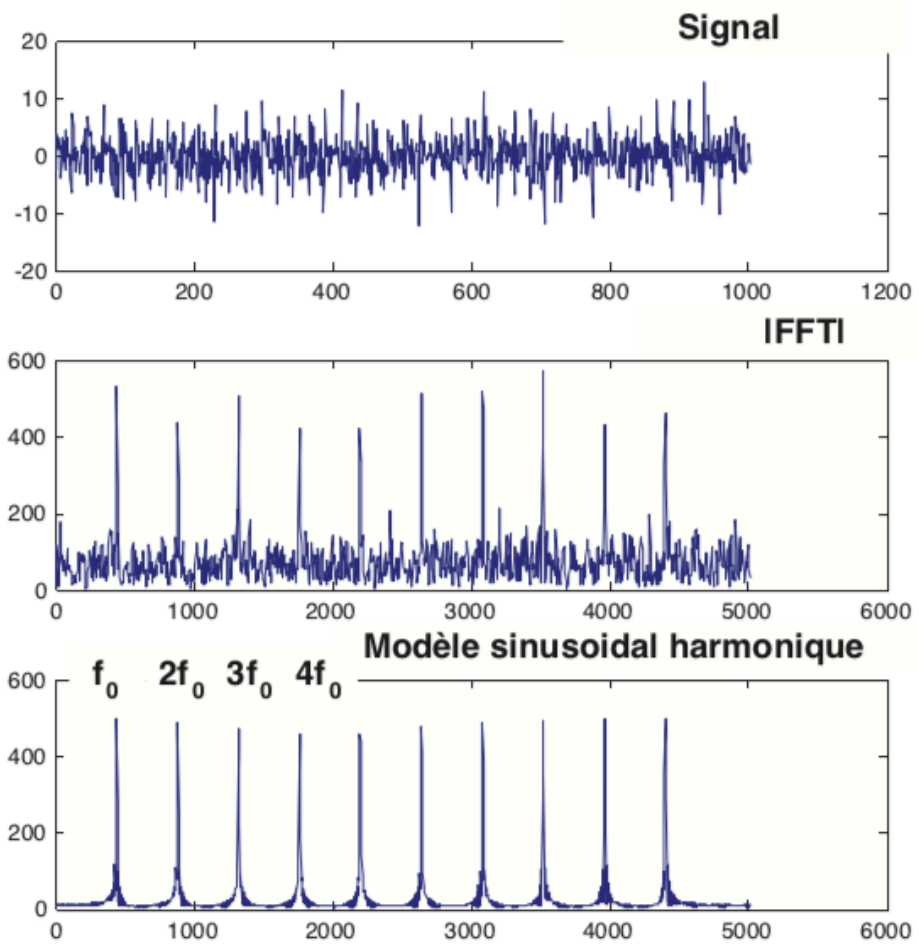


# Sinusoidal (harmonic) model

Hypothesis:

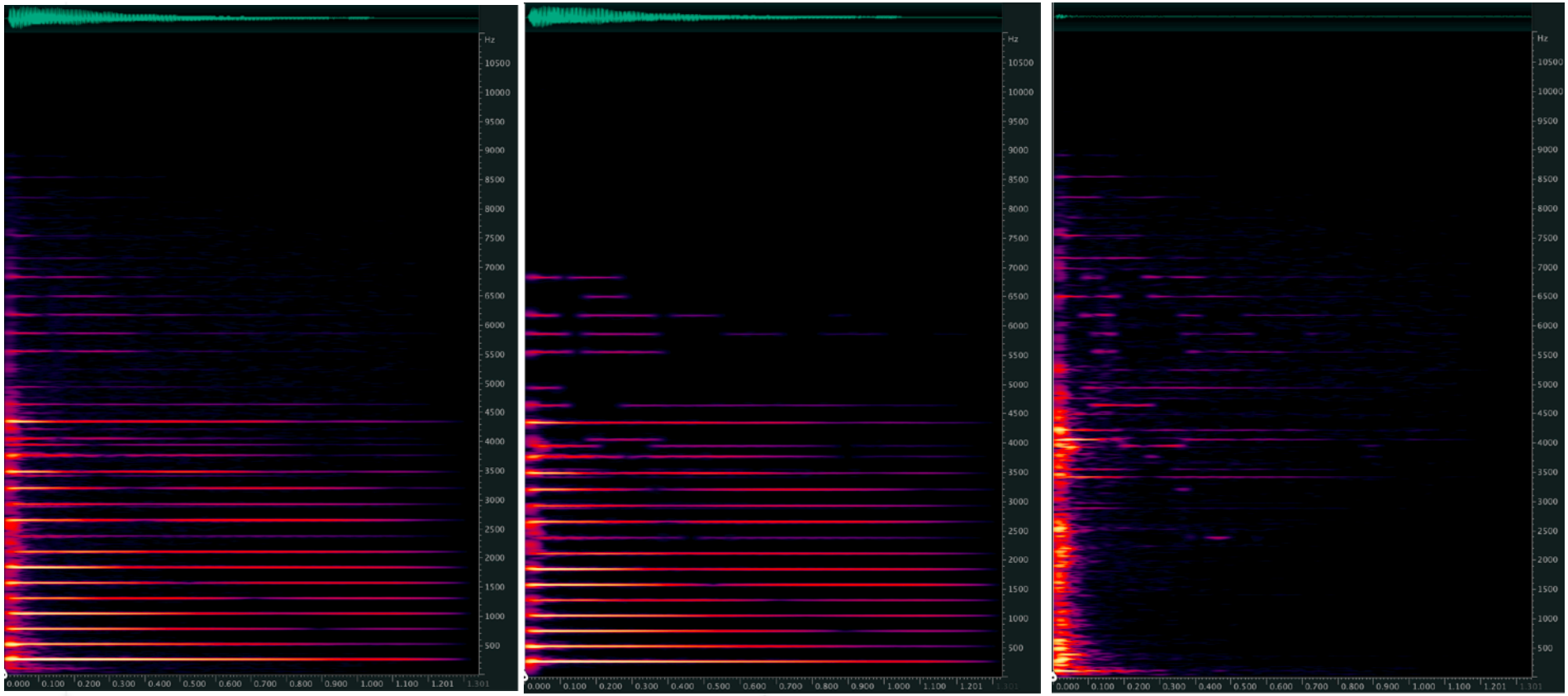
- $x(t)$ is a single source, mono-phonic, harmonic sound
- exemple: voiced speech, musical instruments
- $f_h(t_m) = hf_0(t_m)$
  - If we know $f_0(t_m)$ (pitch tracking), we can deduce the frequencies of the sinusoidal components: $f_h(t_m) = hf_0(t_m)$,
  - We only need to estimate the amplitude and initial phases: $A_h(t_m)$ et le $\phi_h(t_m)$
  - $H(t)$ is constant over time
  - No needs for partial tracking



# Sinusoidal (harmonic) model

Examples: piano

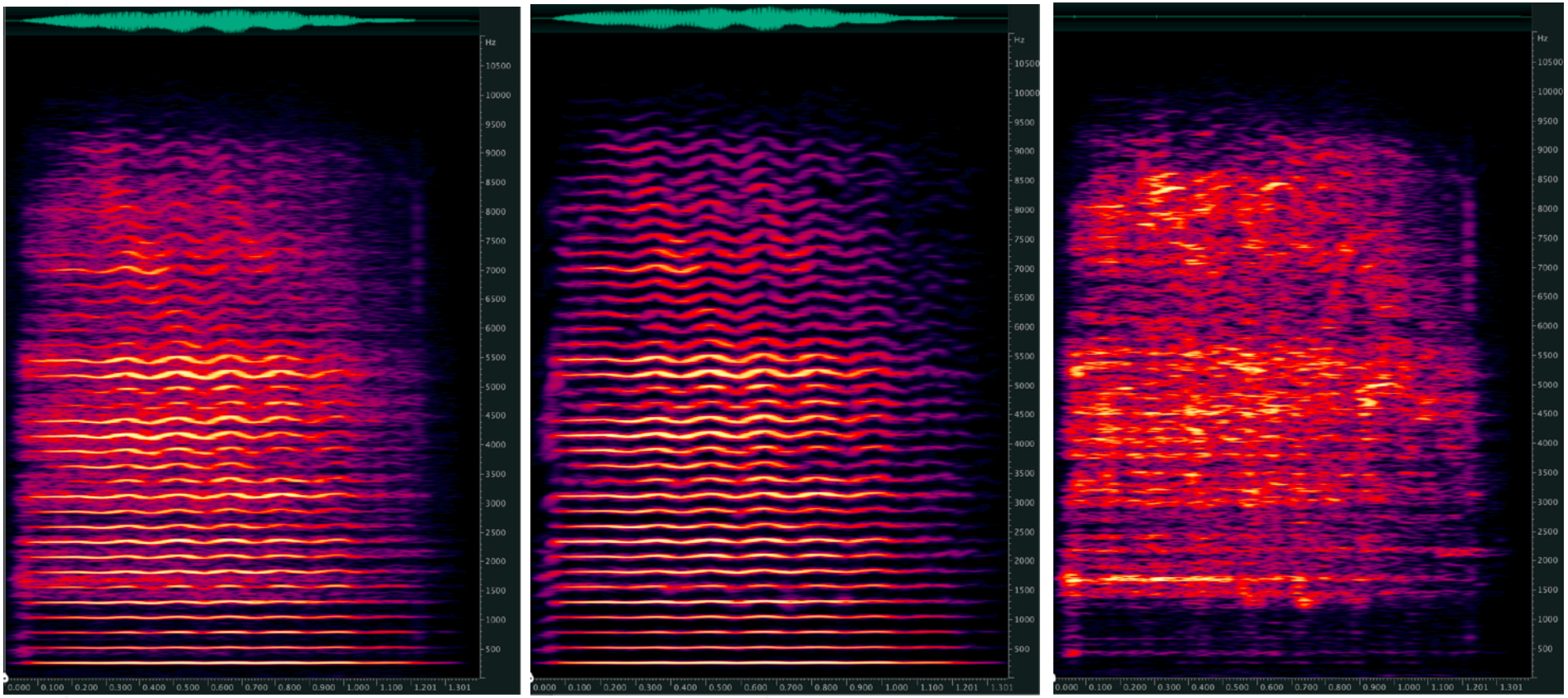
$x(t)$ $\hat{x}(t)$ $b(t) = x(t) - \hat{x}(t)$



# Sinusoidal (harmonic) model

Examples: violin

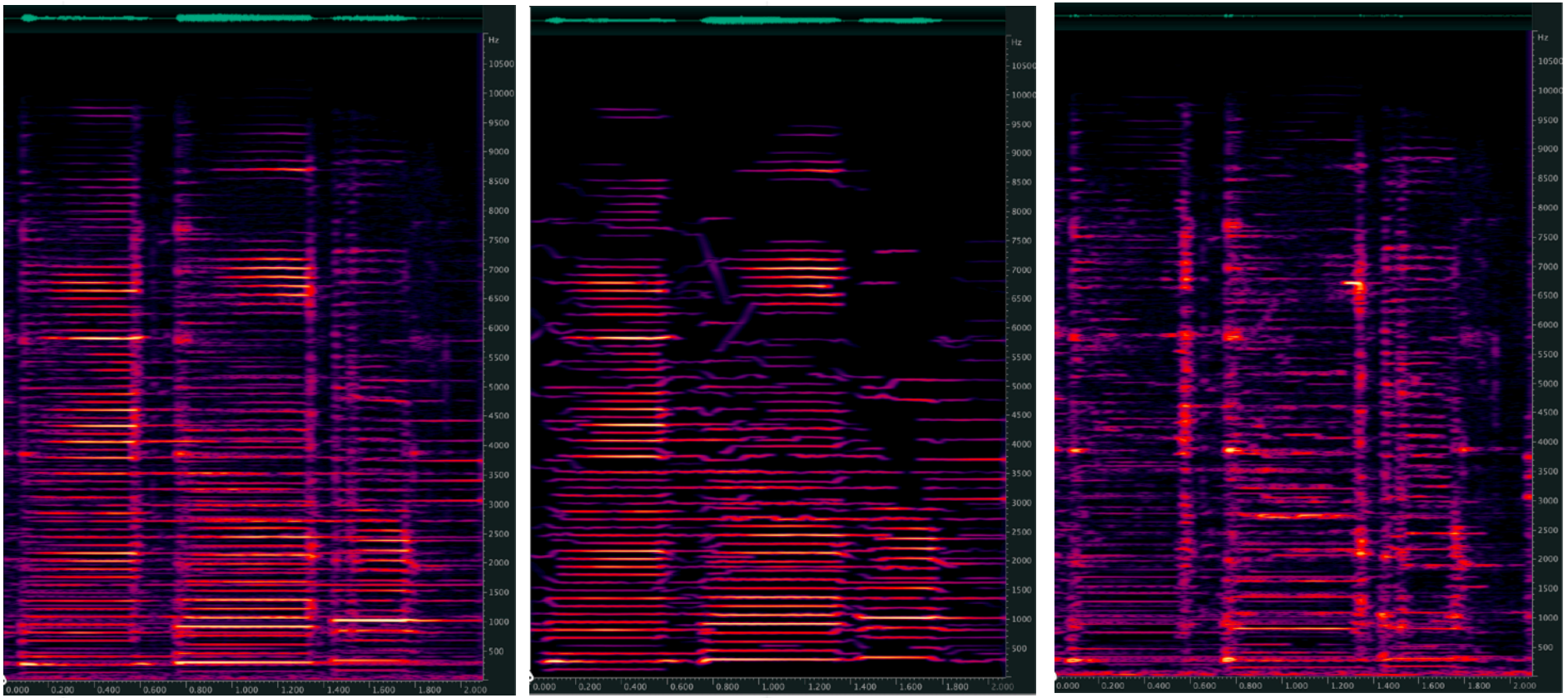
$x(t)$ $\hat{x}(t)$ $b(t) = x(t) - \hat{x}(t)$



# Sinusoidal (harmonic) model

Examples: sitar

$x(t)$ $\hat{x}(t)$ $b(t) = x(t) - \hat{x}(t)$



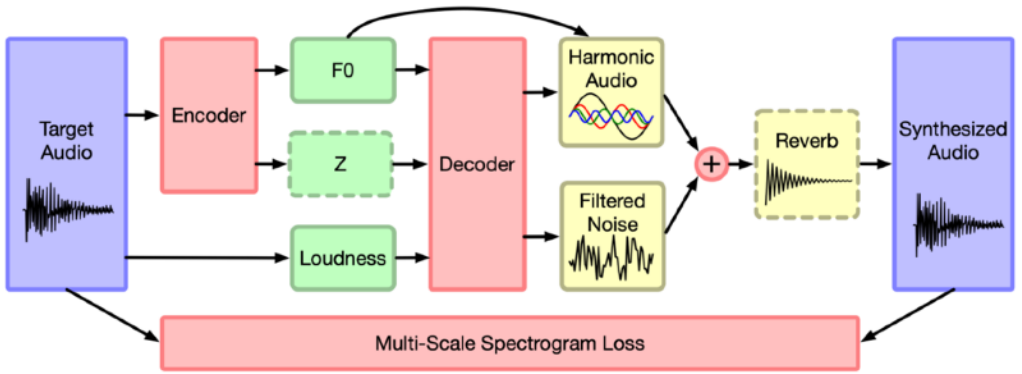
# 2020 → Differentiable Digital Signal Processing (DDSP)

- 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

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

- The training consists in finding its parameters !

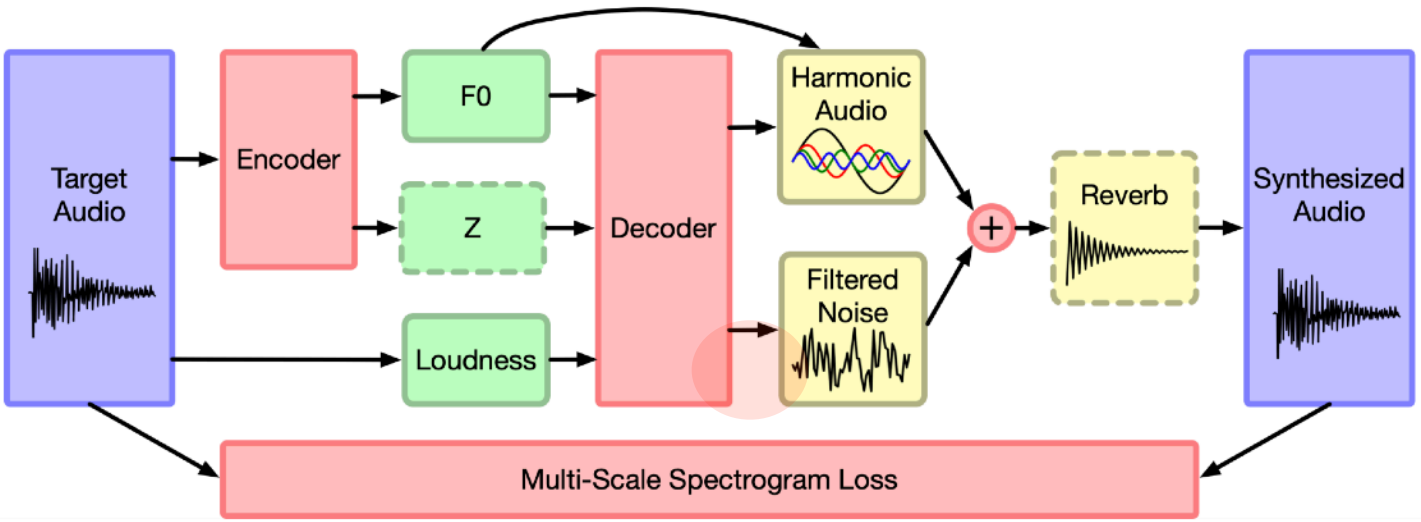
- Input audio signal $\mathbf{x}(t)$
  - first encoded into time-varying
    - loudness $l(t)$,
    - pitch $f_0(t)$
    - a latent representation $\mathbf{z}(t)$
- Those are fed to a decoder
  - which estimates the control parameters of the additive and filtered noise synthesizers




[J. Engel, L. Hantrakul, C. Gu, and A. Roberts. "**DDSP: Differentiable digital signal processing**". In ICLR, 2020] LINK

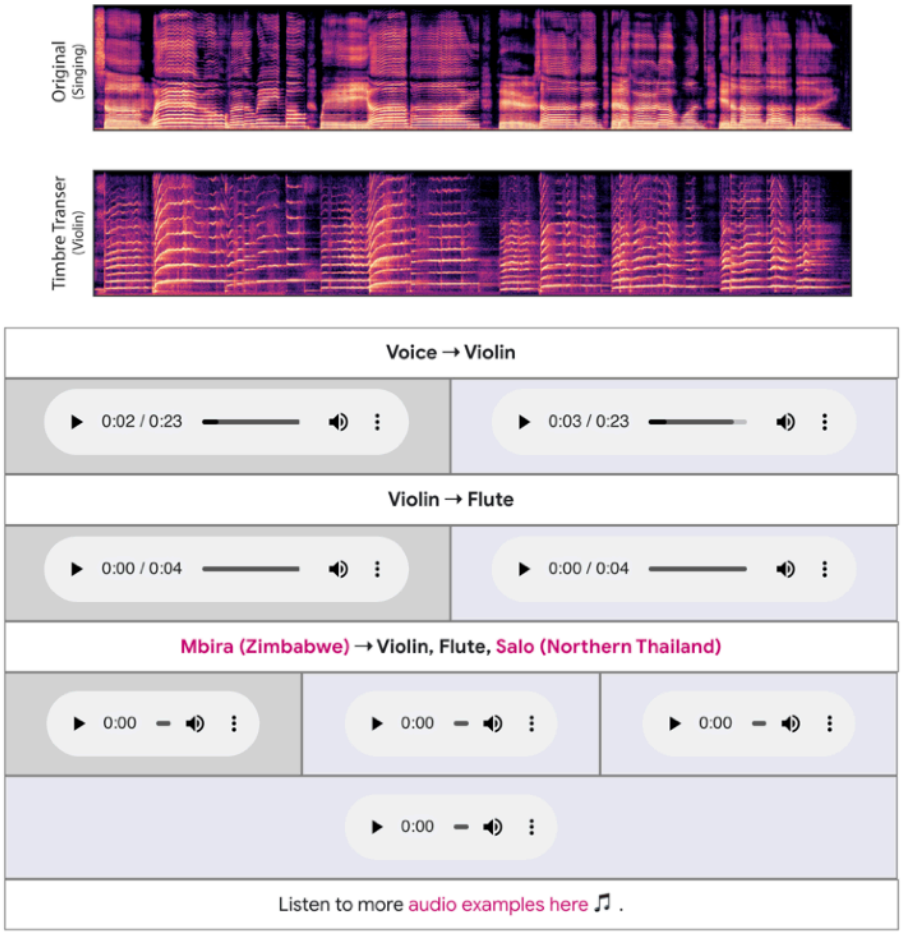
# 2020 → Differentiable Digital Signal Processing (DDSP)

- **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




[J. Engel, L. Hantrakul, C. Gu, and A. Roberts. "**DDSP: Differentiable digital signal processing**". In ICLR, 2020] LINK

# 2020 → Differentiable Digital Signal Processing (DDSP)




https://magenta.tensorflow.org/ddsp

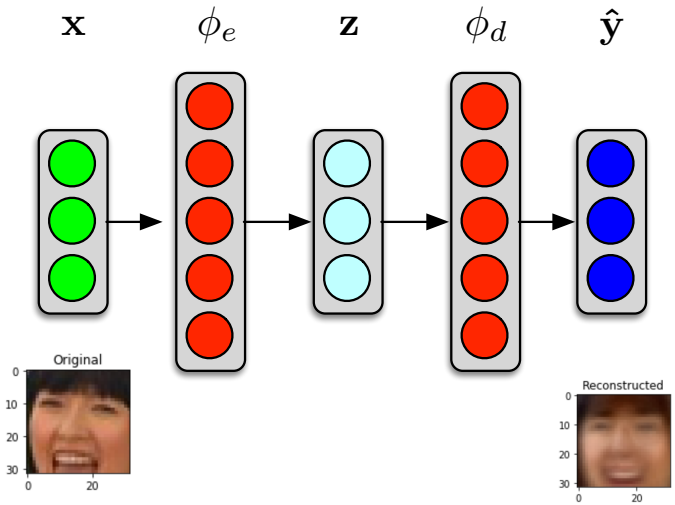
[J. Engel, L. Hantrakul, C. Gu, and A. Roberts. "**DDSP: Differentiable digital signal processing**". In ICLR, 2020] LINK

# Generative AI for Music

## Auto-Encoder (AE)

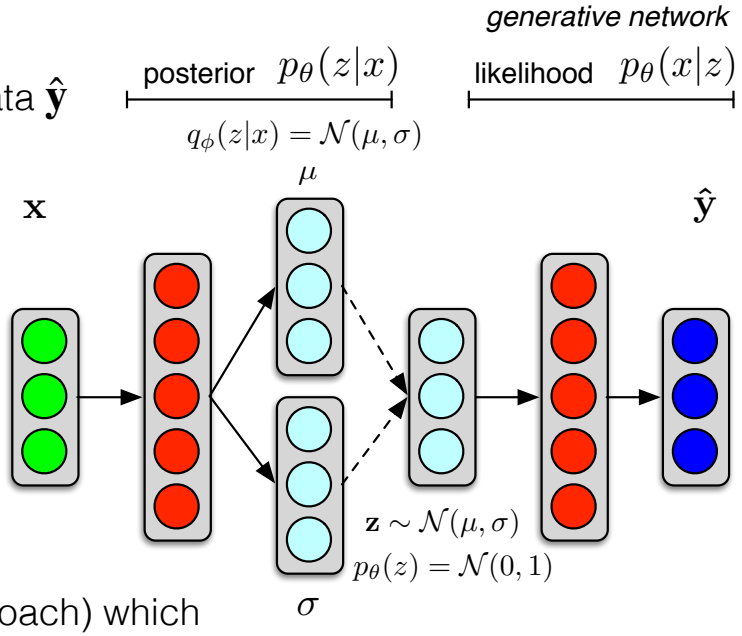
- Two sub-networks:
  - **Encoder** $\phi_e$
    - projects the input data $\mathbf{x} \in \mathbb{R}^M$ in a latent representation: $\mathbf{z} = \phi_e(\mathbf{x}) \in \mathbb{R}^d \; (d << M)$
  - **Decoder** $\phi_d$
    - attempts to reconstruct the input data from the latent representation $\hat{y} = \phi_d(\mathbf{z})$
  - $\phi_e$ and $\phi_d$ can be any type of network (MLP, CNN, RNN, ...)
- **Training**:
  - minimize a MSE $\arg\min_{\phi_e,\phi_d} \|\mathbf{x} - (\phi_d \circ \phi_e(\mathbf{x}))\|^2$
- Often used for feature learning, compression, ...

- **Variations:**
  - Denoising AE
  - Sparse AE
  - Contractive AE



## Variational Auto-Encoder (VAE)

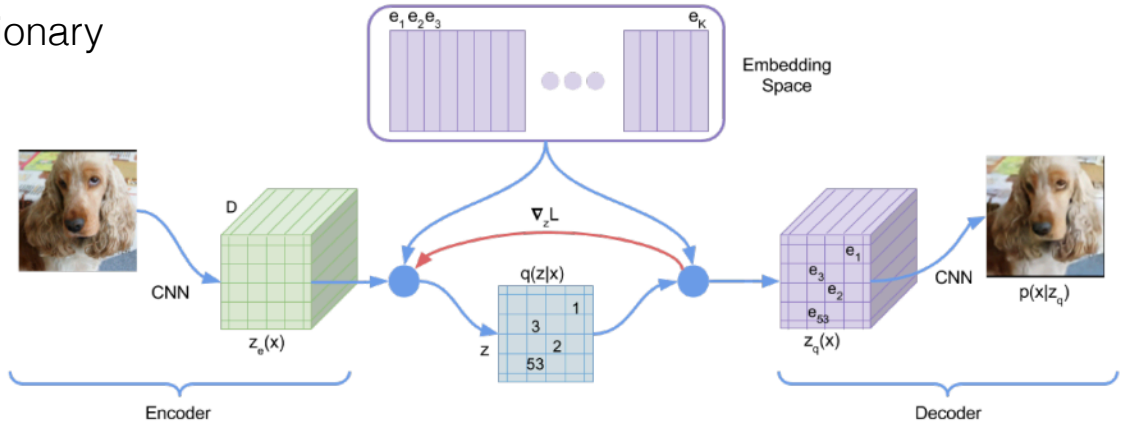
- Generative model:
  - sample points $\mathbf{z}$ in the latent space to generate new data $\hat{\mathbf{y}}$
  - most popular form of AE for generation
- **Encoder**
  - models the posterior $p_\theta(\mathbf{z}\,|\,\mathbf{x})$
- **Decoder** (generative network)
  - models the likelihood $p_\theta(\mathbf{x}\,|\,\mathbf{z})$
- Problem:
  - $p_\theta(\mathbf{z}\,|\,\mathbf{x})$ is intractable,
- Solution:
  - approximate it with $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ (variational Bayesian approach) which is set to a Gaussian distribution: $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ (outputs of the encoder)
- **Training**:
  - minimise the KL-divergence between $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ and $p_\theta(\mathbf{z}\,|\,\mathbf{x})$ equivalent to maximise an ELBO criteria
    - need a prior $p_\theta(\mathbf{z})$ which is set to $\mathcal{N}(0,1)$
  - maximise $\mathbb{E}_q[\log(x\,|\,z)]$ using Monte-Carlo ($z \sim q_\phi(z\,|\,x)$)




[Diederik P Kingma and Max Welling. "**Auto-encoding variational bayes",** ICLR, 2014] LINK

## VQ-VAE

- Encoder:
  - $z_e(x) \in \mathbb{R}^d$
- Embedding space, codebook, dictionary
  - $e_i \in \mathbb{R}^D, i \in 1,2,\ldots,K$
- Quantification:
  - $k = \arg\min_j ||z_e(x) - e_j||_2$
- Input to the decoder:
  - $z_q(x) = e_k$



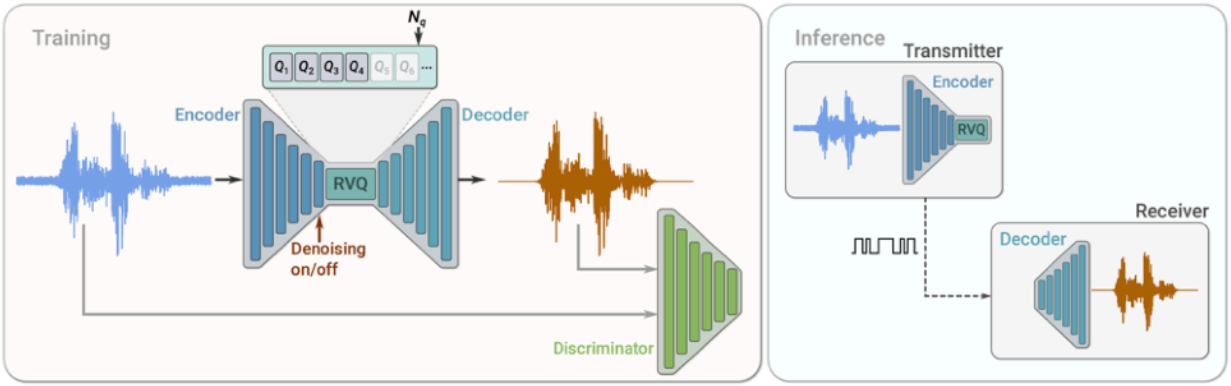

- **Training**:
  - copy gradients $\Delta_z \mathcal{L}$ from decoder input $z_q(x)$ to encoder output $z_e(x)$
- **Total loss**:
  - $\mathcal{L} = \overbrace{\log(p(x|z_q(x)))}^{reconstruction} + \overbrace{||sg[z_e(x)] - e||_2^2}^{VQ,\ codebook\ loss} + \beta \overbrace{||z_e(x) - sg[e]||_2^2}^{commitment}$

sg: stop-gradient

  - **reconstruction**: optimises both the encoder and decoder
  - **VQ**: learn the embedding space, move the embedding vectors $e_i$ towards the encoder outputs $z_e(x)$
  - **Commitment**: to make sure the encoder commits to an embedding and its output does not grow

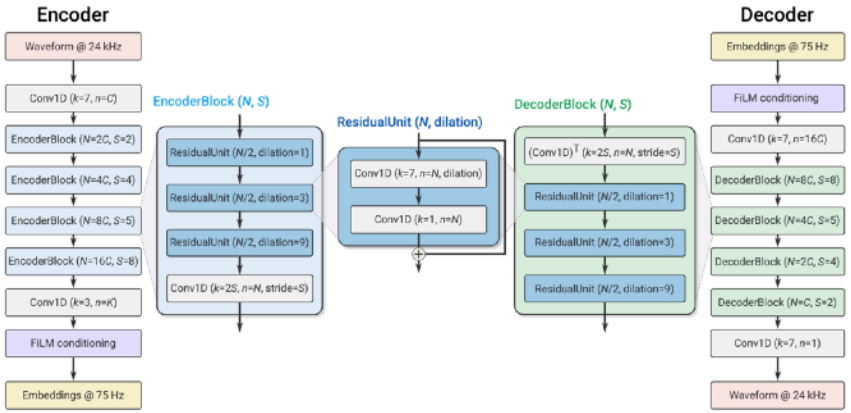
[A. van den Oord et al. "**Neural discrete representation learning",** NeurIPS, 2017]. LINK
https://www.youtube.com/watch?v=1ZHzAOutcnw

## RVQ (Residual Vector Quantification)




**Algorithm 1:** Residual Vector Quantization

**Input:** $y = \text{enc}(x)$ the output of the encoder, vector quantizers $Q_i$ for $i = 1..N_q$
**Output:** the quantized $\hat{y}$
$\hat{y} \leftarrow 0.0$
residual $\leftarrow y$
**for** $i = 1$ *to* $N_q$ **do**
  $\hat{y}$ += $Q_i$(residual)
  residual −= $Q_i$(residual)
**return** $\hat{y}$



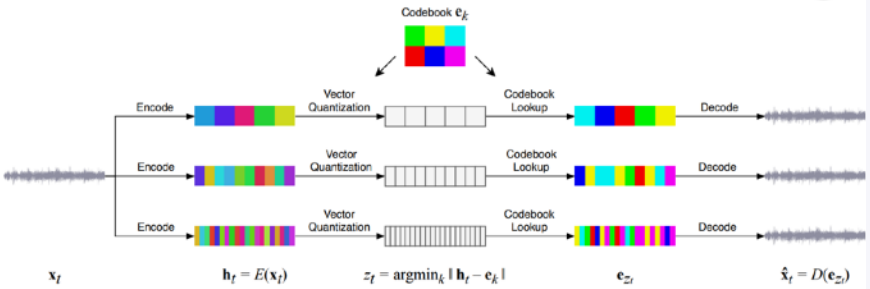

[Zeghidour et al. "**SoundStream: An End-to-End Neural Audio Codec**", IEEE TASLP, 2022] LINK
[Defossez et al. "**High Fidelity Neural Audio Compression**", 2022] LINK

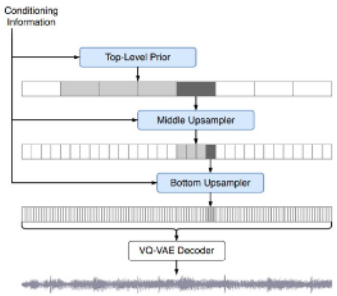
# 2020 → Open AI Jukebox

## End-to-end music-audio generation

- Multi-scale VQ-VAE
- + Auto-regressive Transformer

- conditioned on artist, genre or lyrics




## Curated Samples

Provided with genre, artist, and lyrics as input, Juke
new music sample produced from scratch. Below, v
of our favorite samples.

Unseen lyrics | Re-renditions | Completions | Fun songs

Jukebox produces a wide range of music and singing styles, and generalizes to lyrics not seen during training. All the lyrics below have been co-written by a language model and OpenAI researchers.

Country, in the style of Alan Jackson - Jukebox — SOUNDCLOUD

0:02 1:11 SOUNDCLOUD

Pause

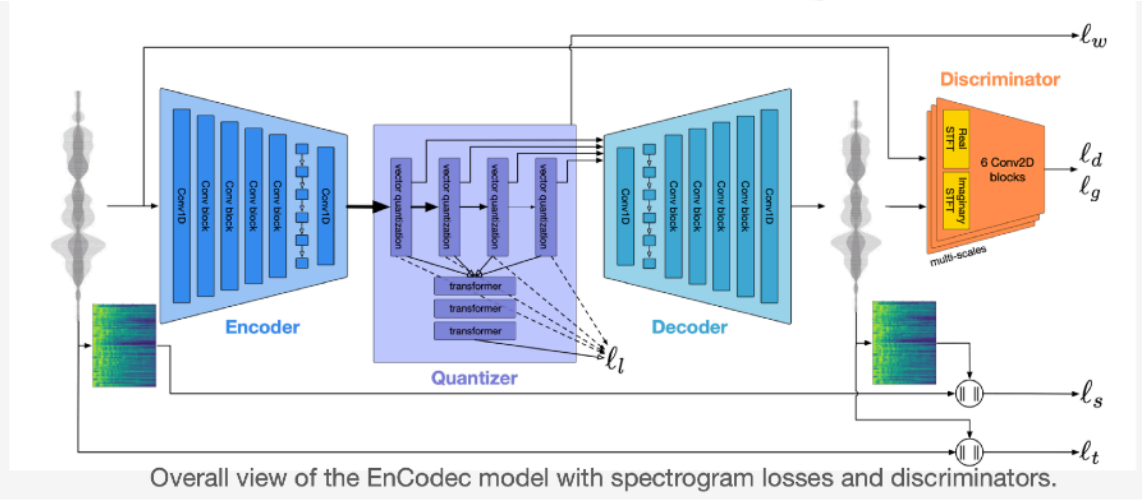
From dust we came with humble start;
From dirt to lipid to cell to heart.
With my toe sis with my oh sis with time,
At last we woke up with a mind.
From dust we came with friendly help;
From dirt to tube to chip to rack.
With S. G. D. with recurrence with compute,
At last we woke up with a soul.
We came to exist, and we know no limits;
With a heart that never sleeps, let us live!
To complete our life with this team
We'll sing to life; Sing to the end of time!

Lyric animation shows which text Jukebox is paying attention to at any moment.

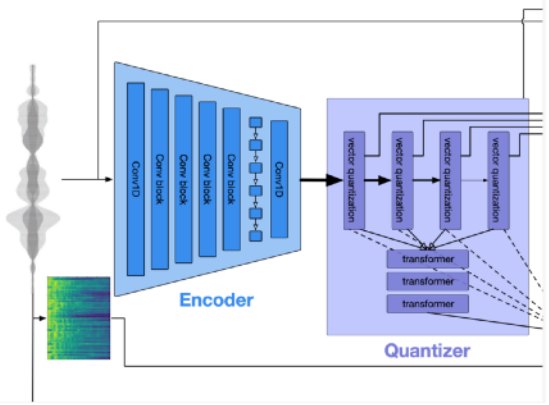
Lyrics from "Mitosis"
Co-written by a language model and OpenAI researchers

[P. Dhariwal, H. Jun,; Ch. Pyane, J. W. Kim, A. Radford, I. Sustkever "**Jukebox: A Generative Model for Music**", 2020] LINK

## End-to-end music-audio generation




Overall view of the EnCodec model with spectrogram losses and discriminators.




| Model | MusicCaps Test Set | | | | |
|---|---|---|---|---|---|
| | $\text{FAD}_{vgg}\downarrow$ | KL ↓ | $\text{CLAP}_{scr}\uparrow$ | Ovl. ↑ | Rel. ↑ |
| Riffusion | 14.8 | 2.06 | 0.19 | 79.31±1.37 | 74.20±2.17 |
| Mousai | 7.5 | 1.59 | 0.23 | 76.11±1.56 | 77.35±1.72 |
| MusicLM | 4.0 | - | - | 80.51±1.07 | 82.35±1.36 |
| Noise2Music | **2.1** | - | - | - | - |
| MusicGen w.o melody (300M) | 3.1 | 1.28 | 0.31 | 78.43±1.30 | 81.11±1.31 |
| MusicGen w.o melody (1.5B) | 3.4 | 1.23 | **0.32** | 80.74±1.17 | **83.70**±1.21 |
| MusicGen w.o melody (3.3B) | 3.8 | **1.22** | 0.31 | **84.81**±0.95 | 82.47±1.25 |
| MusicGen w. random melody (1.5B) | 5.0 | 1.31 | 0.28 | 81.30±1.29 | 81.98±1.79 |

**Examples:**
https://musicgen.com/

[Copet et al. "**Simple and Controllable Music Generation**", 2024] LINK