2023 September 25

Deep Learning for Music Analysis and Generation

Timbre

music classification
(audio → class)

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

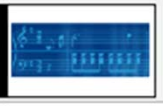
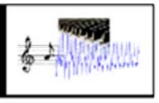
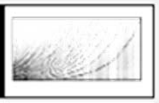
Objectives

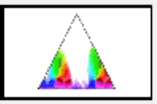
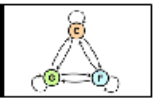
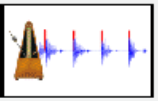
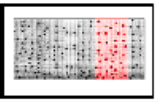
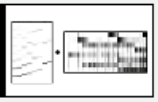
- To get to know the idea of “timbre”
  - Timbre features, timbre classifier, timbre encoder, timbre decoder

- Music classification as a type of music **analysis** task
  - 1D vs 2D convolution kernels
  - Contrastive learning

Reference 1: FMP Notebook

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S3_Timbre.html

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

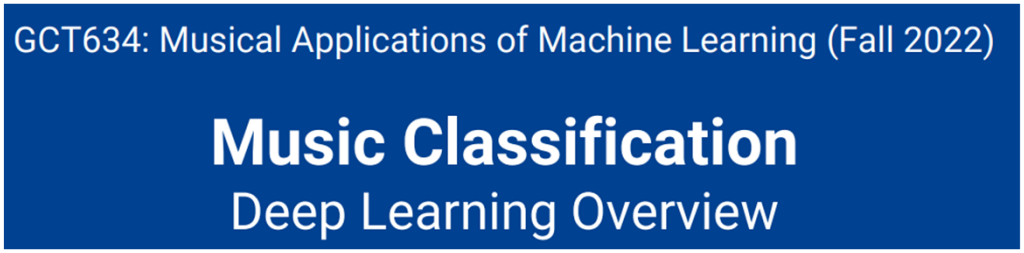
Part	Title	Notions, Techniques & Algorithms	HTML	IPYNB
4 	Music Structure Analysis	Similarity matrix, repetition, thumbnail, homogeneity, novelty, evaluation, precision, recall, F-measure, visualization, scape plot	[html]	[ipynb]
5 	Chord Recognition	Harmony, music theory, chords, scales, templates,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

Not much covered by FMP

Reference 2: KAIST Course & ISMIR 2021 Tutorial

For fundamentals of deep learning and music classification

- https://mac.kaist.ac.kr/~juhan/gct634/Slides/[week4-1]%20music%20classification%20-%20deep%20learning%20overview.pdf



- https://music-classification.github.io/tutorial/landing-page.html

Outline

- **General idea**
- Representing timbre: signal processing perspective
- Representing timbre: deep learning perspective
- Other related topics on music classification
- HW1

Timbre: Different Instruments

https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S3_Timbre.html

- **Timbre** (tone color; tone quality)
  - Allows a listener to distinguish the musical tone of a violin, an oboe, or a trumpet even if the tone is played at the same **pitch** and with the same **loudness**
  - Hard to grasp & *subjective*
  - May be described with words such as *bright*, *dark*, *warm*, and *harsh*

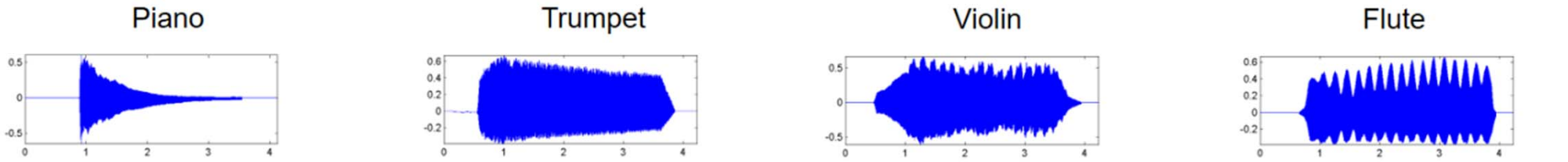


Figure 1.23 from [Müller, FMP, Springer 2015]

0:04 / 0:04 0:00 / 0:04 0:00 / 0:04 0:00 / 0:04

Timbre: Different Tones / Audio Effects

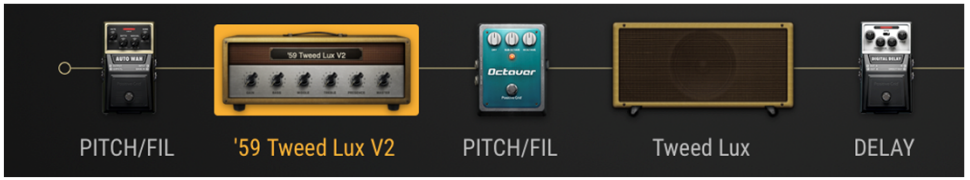
(Examples provided by Positive Grid®)

Dry (single coil EG)

In the Clouds

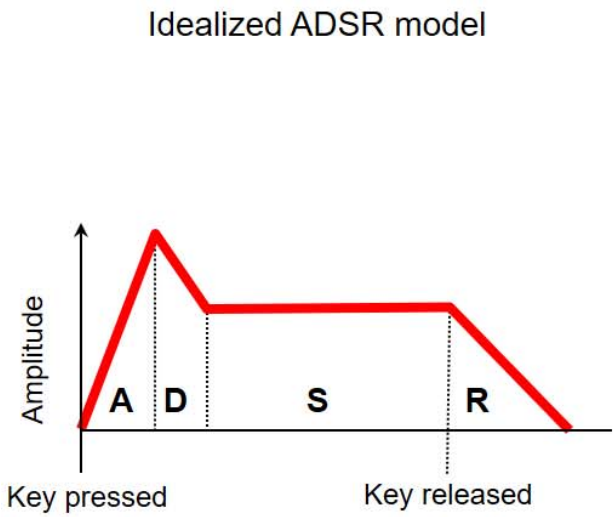
Overdriven Verb Icon

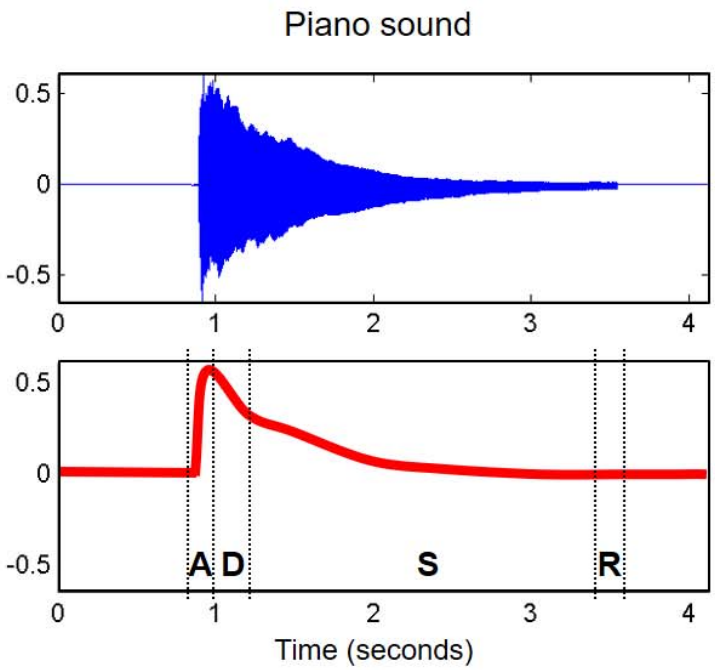
Lazy Down



Temporal Characteristics

- **ADSR**: attack (A), decay (D), sustain (S), and release (R)





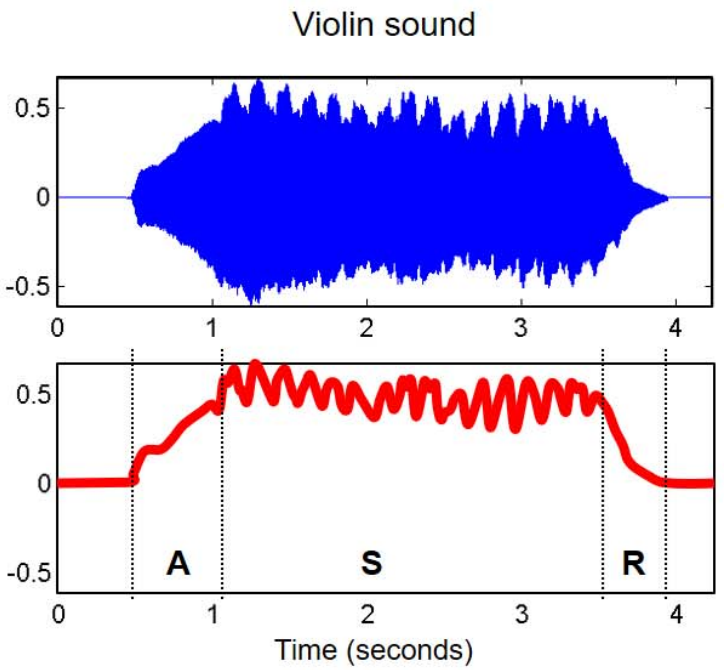


Figure 1.22b and Figure 1.23 from [Müller, FMP, Springer 2015]

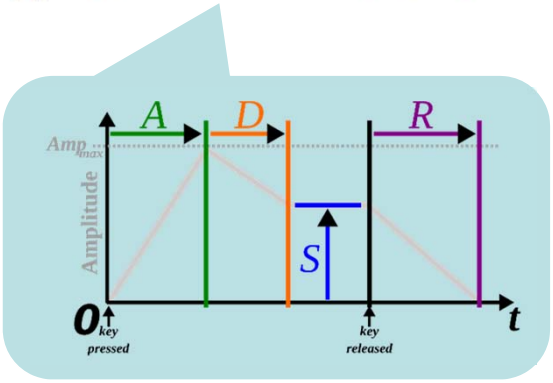
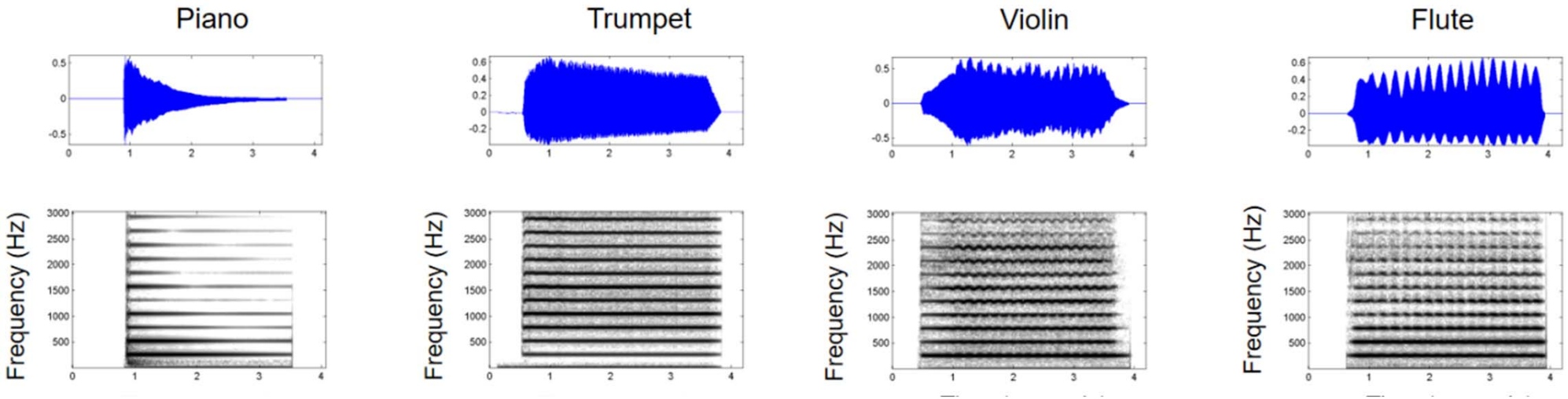


Figure from:
https://en.wikipedia.org/wiki/Envelope_(music)

Spectral Characteristics

- Fundamental frequency (**F0**)
- **Partials** (harmonics): usually at frequencies which are integer multiples of F0
  - The partials in different instruments may exhibit **relative strengths**
  - The frequency may deviate from the ideal harmonics (this is called “inharmonicity”)



Demo: Tone Transfer

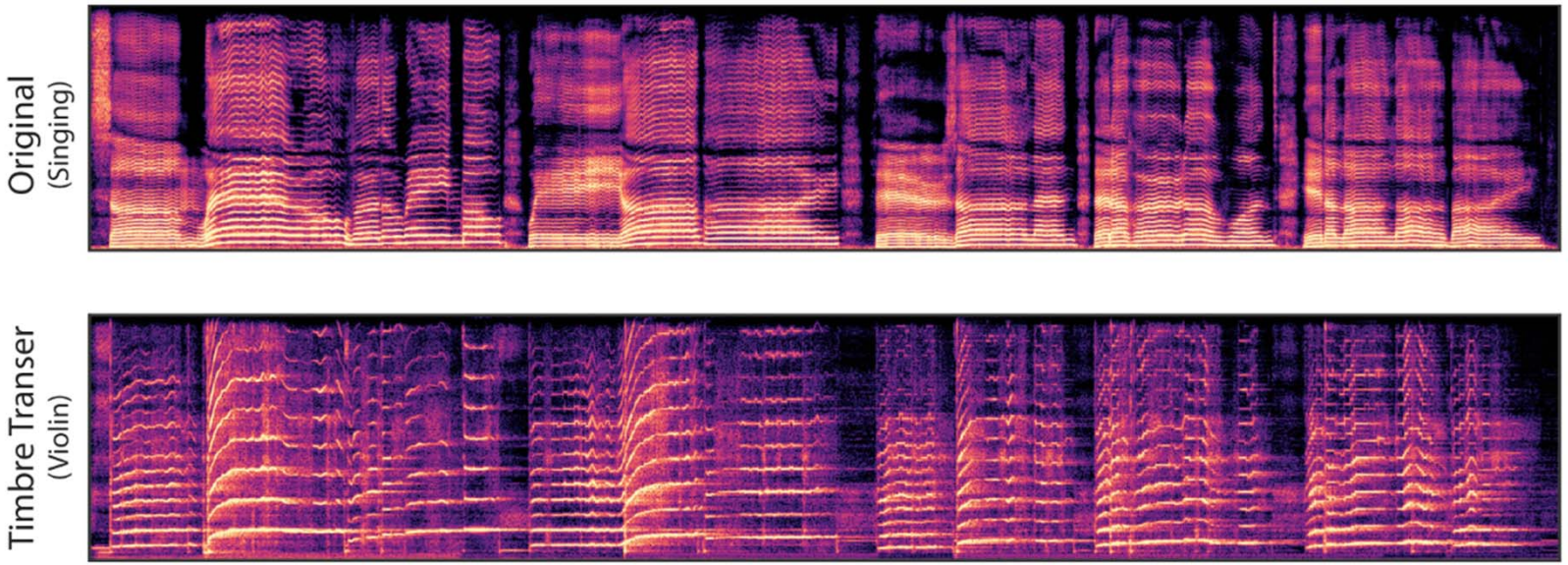
https://sites.research.google/tonetransfer

Tone Transfer — Behind the Scenes

https://magenta.tensorflow.org/tone-transfer

https://magenta.tensorflow.org/ddsp

- Keeping the **frequency** and **loudness** *the same* and changing the sound character



Ref: Engel et al., “DDSP: Differentiable digital signal processing,” ICLR 2020

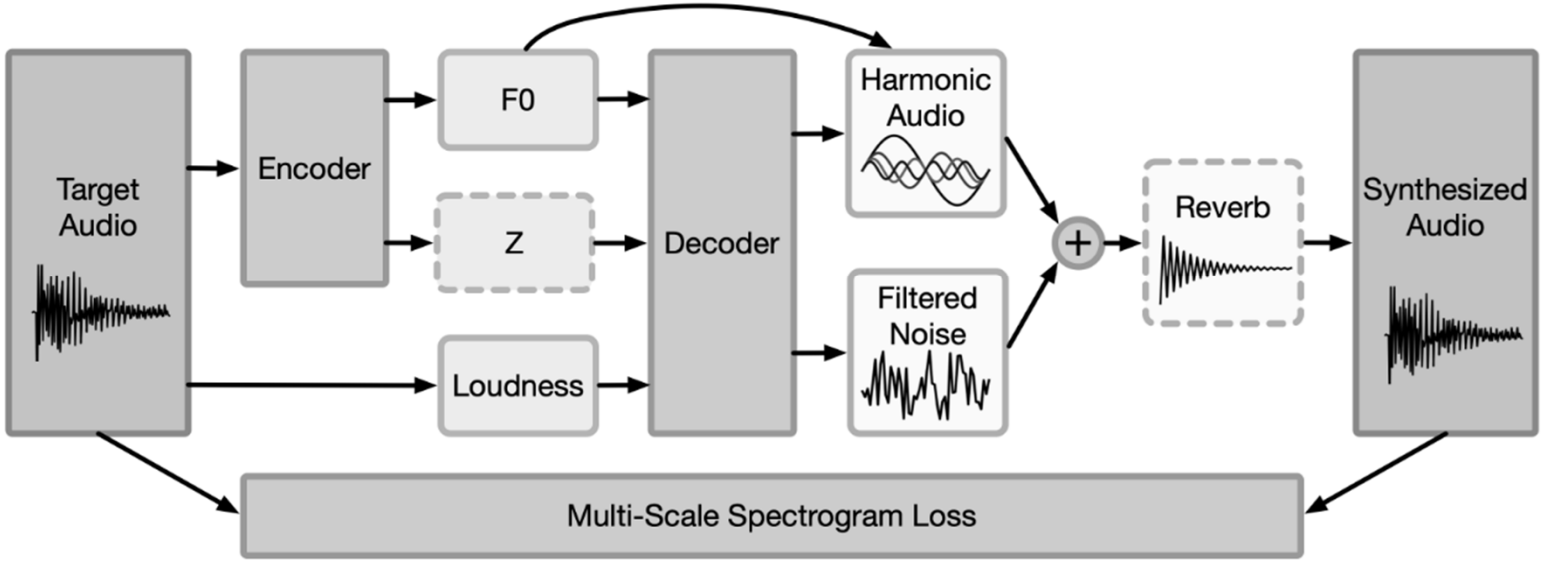
Decomposition of Elements of Music

(for monophonic signals)

https://magenta.tensorflow.org/ddsp

- **Music = F0 (pitch + duration) + loudness + "z" (timbre)**

content *style*



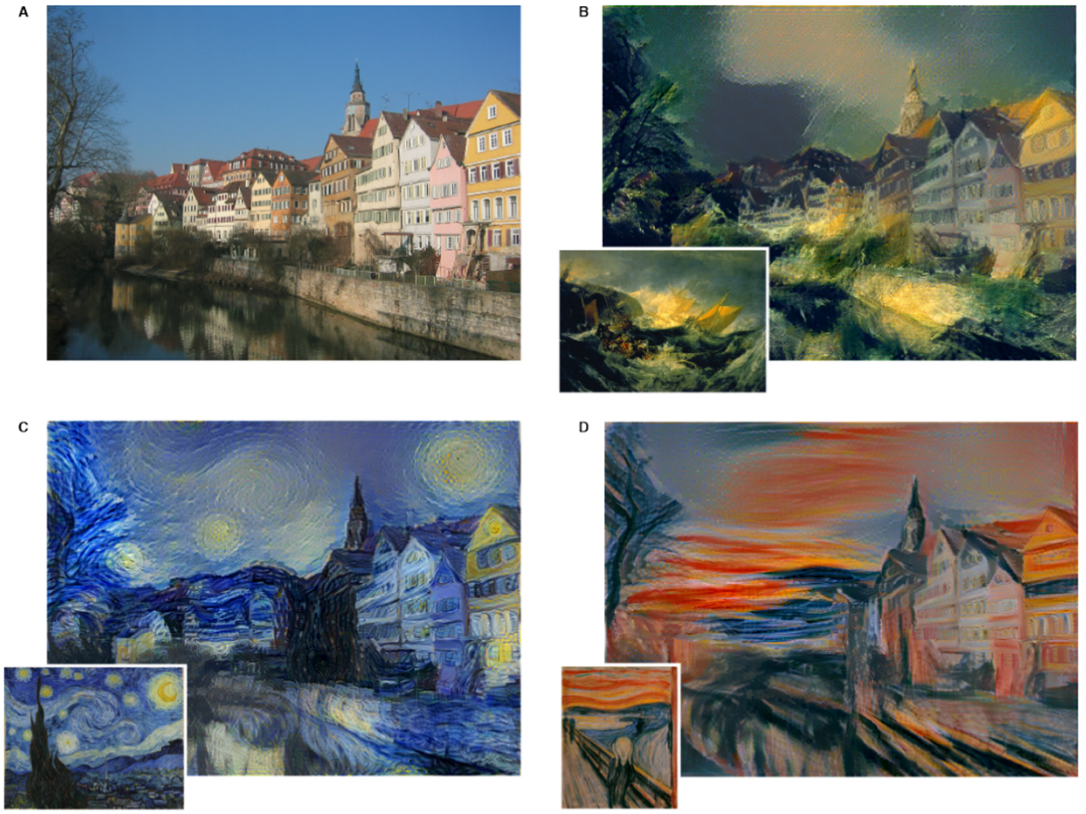
Ref: Engel et al., "DDSP: Differentiable digital signal processing," ICLR 2020

Will Talk About DDSP Later

- W1. Introduction to the course
- W2. Fundamentals & Music representation
- W3. **Analysis** I (timbre): Automatic music classification and representation learning
- (**HW1**: Singer classification)
- W4. **Generation** I: Source separation
- W5. **Generation** II: GAN & Vocoders
- W6. **Generation** III: Synthesis of notes and loops
- (**HW2**: GAN-based Mel-vocoder)
- W7. **Analysis** II (pitch): Music transcription, Melody extraction, and Chord Recognition
- W8. **Generation** IV: Symbolic MIDI generation
- W9. **Generation** V: Symbolic MIDI generation: Advanced Topics
- (**HW3**: Pop music Transformer)
- W10. **Generation** VI: Singing voice generation
- W11. **Generation** VII: Text-to-music generation
- **W12**. Proposal of ideas of final projects
- W13. **Generation** VIII: Differentiable DSP models and automatic mixing
- W14. Miscellaneous Topics
- (W15. Break)
- **W16**. Oral presentation of final projects

Decomposition of Elements of Images

- Image = **content** + **style**



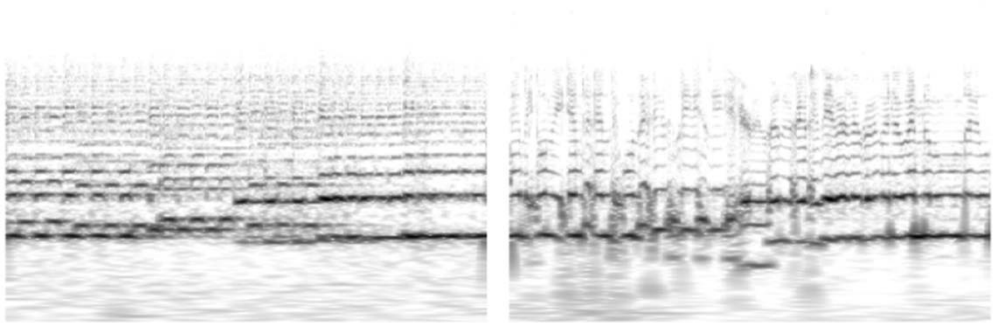
Ref: Gatys et al., “A neural algorithm of artistic style,” arXiv **2015**

Decomposition of Elements of Music

(for more general classes of audio signals)

https://anonymous84654.github.io/RAVE_anonymous/

Strings to speech transfer



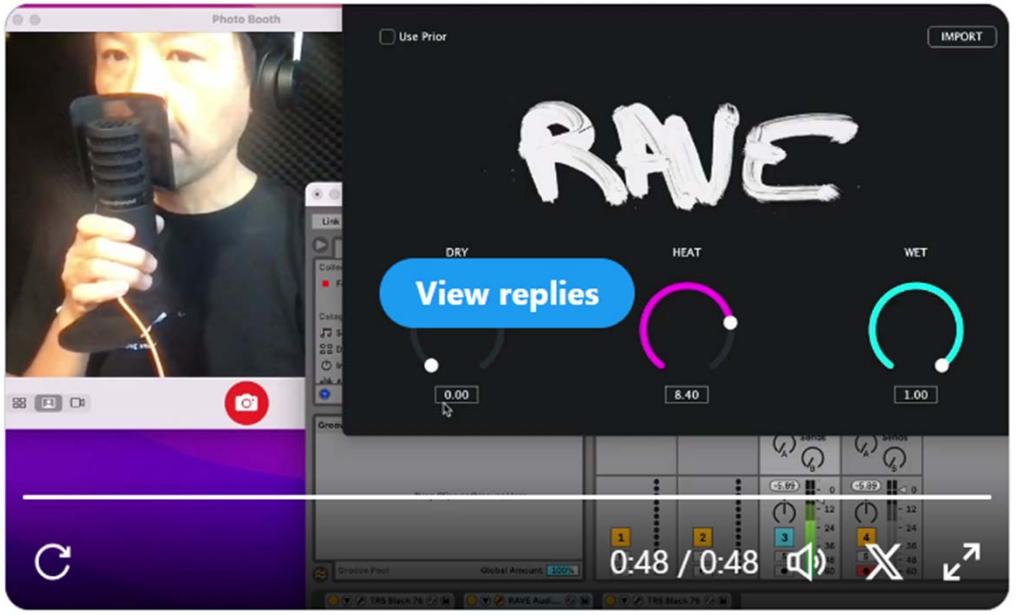
original

0:00 / 0:04

reconstructed

0:00 / 0:04

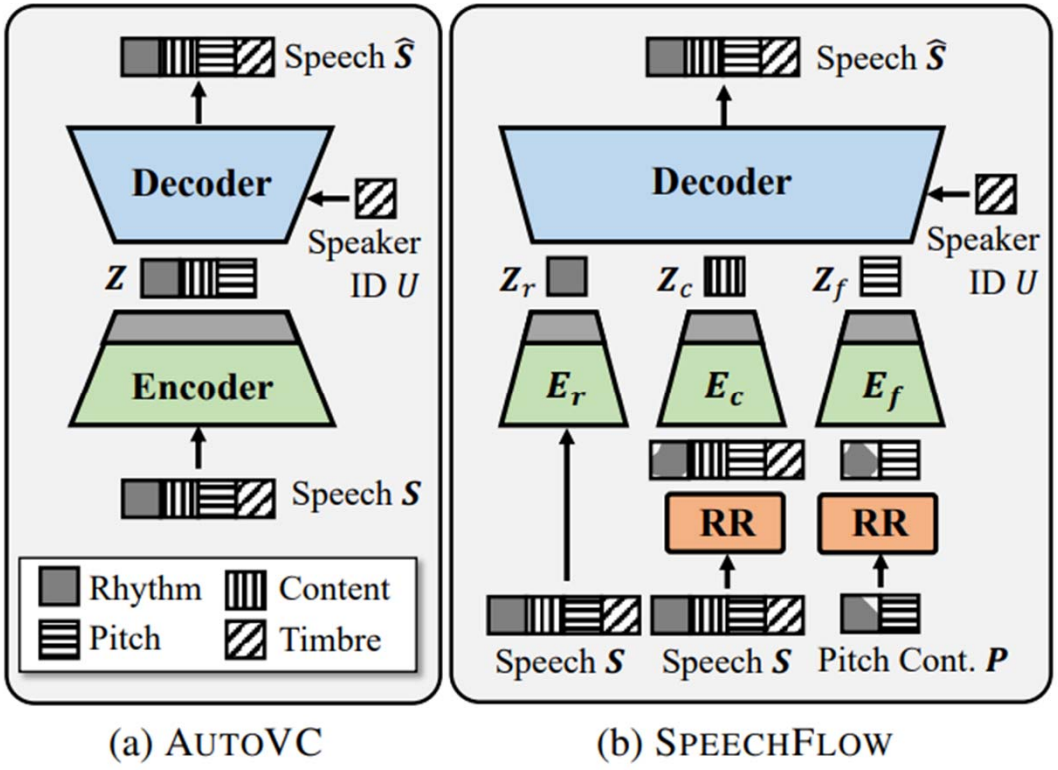
Realtime Voice to Darbuka transfer



Ref: Caillon and Esling, “RAVE: A variational autoencoder for fast and high-quality neural audio synthesis,” arXiv 2021

Decomposition of Elements of Speech

- Speech = pitch + rhythm + content (phoneme) + timbre



(a) AUTOVC (b) SPEECHFLOW

Ref1: Qian et al., "AUTOVC: Zero-shot voice style transfer with only autoencoder loss," ICML 2019

Ref2: Qian et al., "Unsupervised speech decomposition via triple information bottleneck," ICML 2020

Decomposition of Elements of Singing

https://github.com/svc-develop-team/so-vits-svc

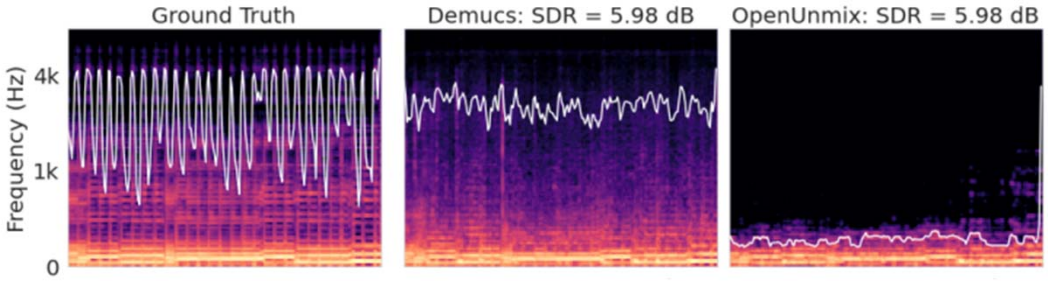
Representation of Timbre

- Four different ways
  - Timbre **features** (audio → features)
  - Timbre **classifier** (audio → class)
  - Timbre **encoder** (audio → embeddings)
  - Timbre **decoder** (embeddings → audio)

Representation of Timbre: Timbre Features

(audio → features)

- Hand-crafted features that can be computed with equations or an algorithm
  - Clear physical meaning
  - Interpretable
- Used as input to music classifiers in the early days
- Can be used as objective metrics for deep learning models



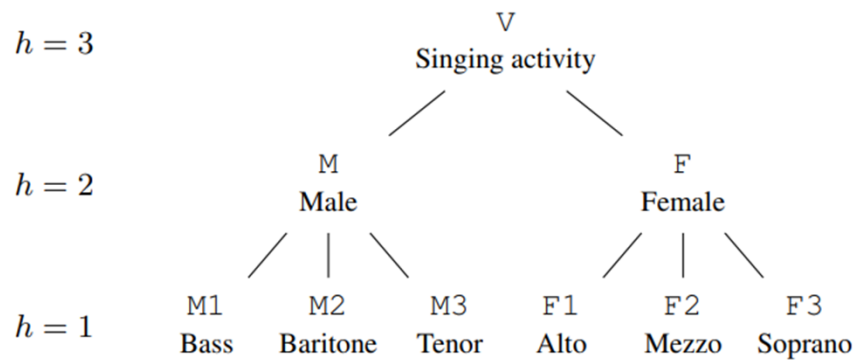
"Our analysis of the errors produced by source separators shows that waveform models [*Demucs*; middle subfigure] tend to **introduce more high-frequency noise**, while spectrogram models [*Open-Unmix*; right subfigure] tend to **lose transients and high frequency content**. We introduce objective measures [*spectral rolloff*] to quantify both kinds of errors"

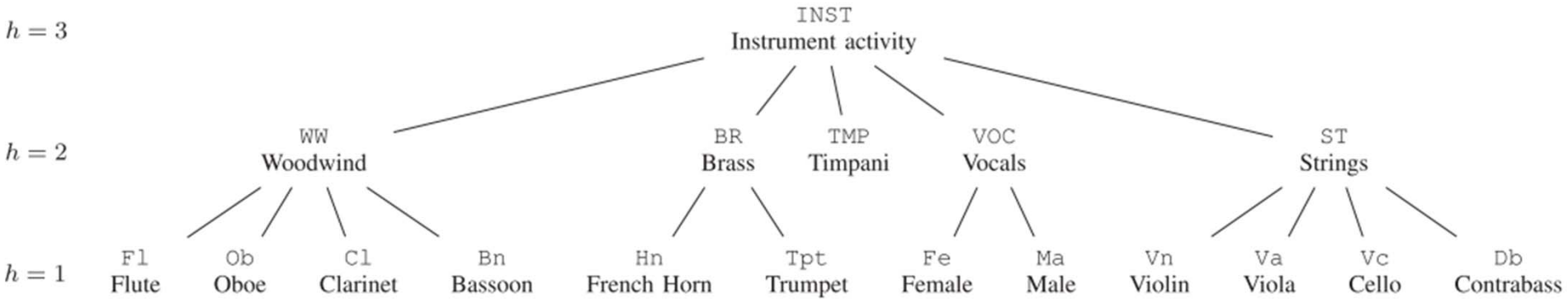
Ref: Schaffer et al., "Music separation enhancement with generative modeling," ISMIR 2022

Representation of Timbre: Timbre Classifier

(audio → class)

- Per song
- Per segment/second
  - Singing activity detection
  - Instrument activity detection





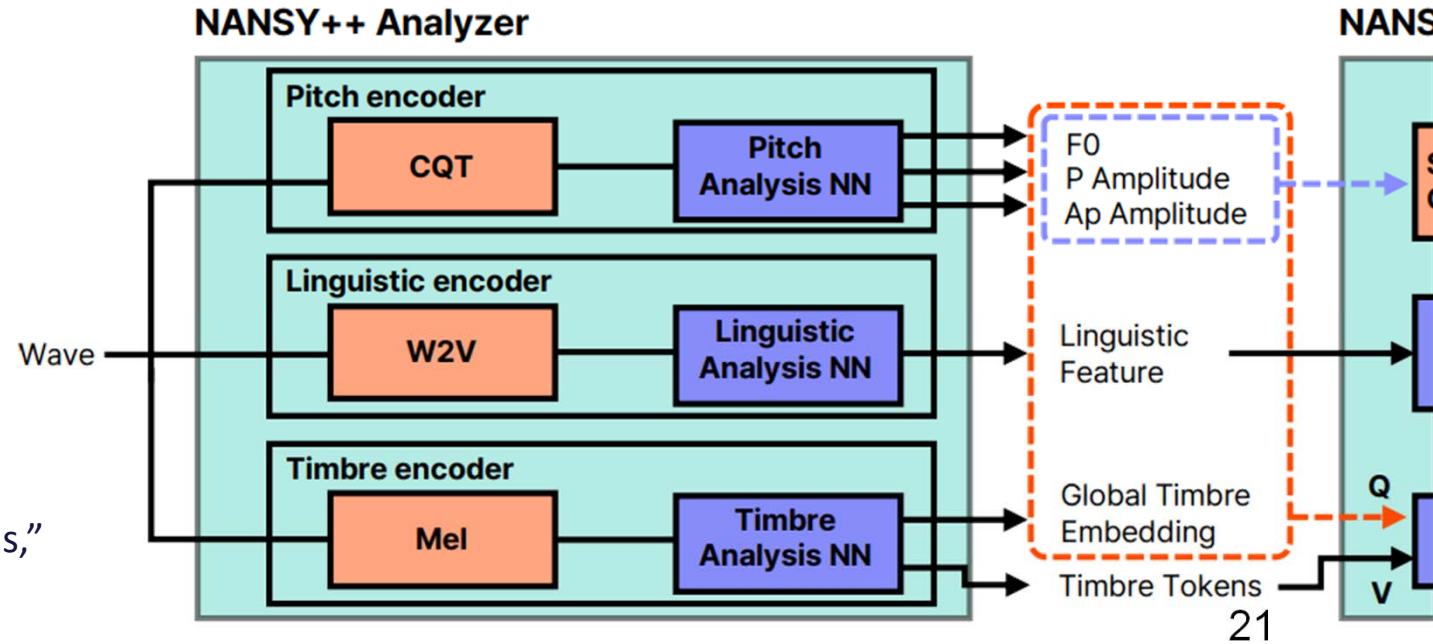
Ref1: Krause et al., “Hierarchical classification of singing activity, gender, and type in complex music recordings,” ICASSP 2022

Ref2: Krause et al., “Hierarchical classification for instrument activity detection in orchestral music recordings,” TASLP 2023

Representation of Timbre: Timbre Encoder

(audio → embeddings)

- Turn audio into **timbre embeddings**
- *Only* encode timbre information, while ignoring the others
  - Some kind of "disentanglement" is needed
  - Harder and *not yet well studied* in the literature



Ref: Choi et al., "NANSY++: Unified voice synthesis with neural analysis and synthesis," ICLR 2023

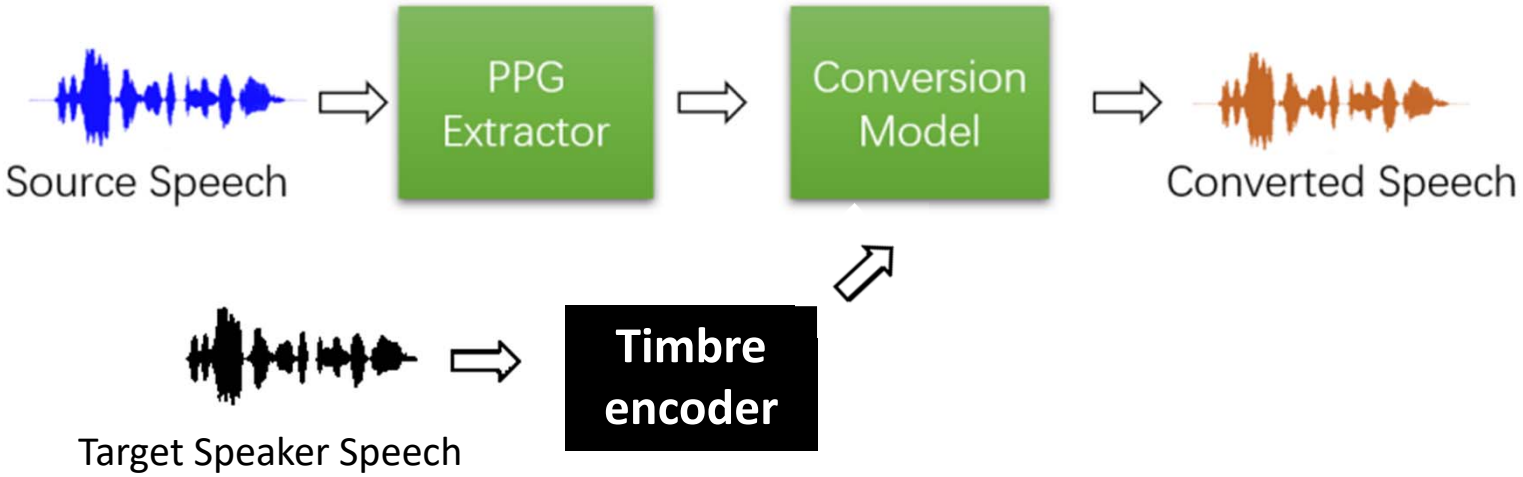
Representation of Timbre: Timbre Decoder
(embeddings → audio)

- Generate audio given some **“timbre-invariant” embeddings**
- Example: **“many-to-one” voice conversion**
  - Use output of automatic speech recognition (ASR) models, a.k.a., phonetic posteriorgrams (PPG), as the model input
    - Note: ASR should focus on the linguistic content, rather than the speaker timbre
  - The decoder **“injects” timbre information** while creating the output

Ref: Liu et al., “Any-to-many voice conversion with location-relative sequence-to-sequence modeling,” TASLP 2021

Timbre Decoder vs. Timbre Encoder

- **Many-to-one** voice conversion
  - **Content** encoder
  - **Timbre** decoder
- **Many-to-many voice** conversion
  - **Content** encoder
  - **Timbre** encoder
  - **Timbre** decoder



Representation of Timbre

- Four different ways
  - Timbre **features** (audio → features) — signal processing
  - Timbre **classifier** (audio → class) — discriminative model
  - Timbre **encoder** (audio → embeddings) — discriminative/generative model
  - Timbre **decoder** (embeddings → audio) — generative model

Outline

- General idea
- **Representing timbre: signal processing perspective**
  - **Timbre features**
- Representing timbre: deep learning perspective
  - Timbre classifier & encoder
- Other related topics on music classification
- HW1

LibROSA

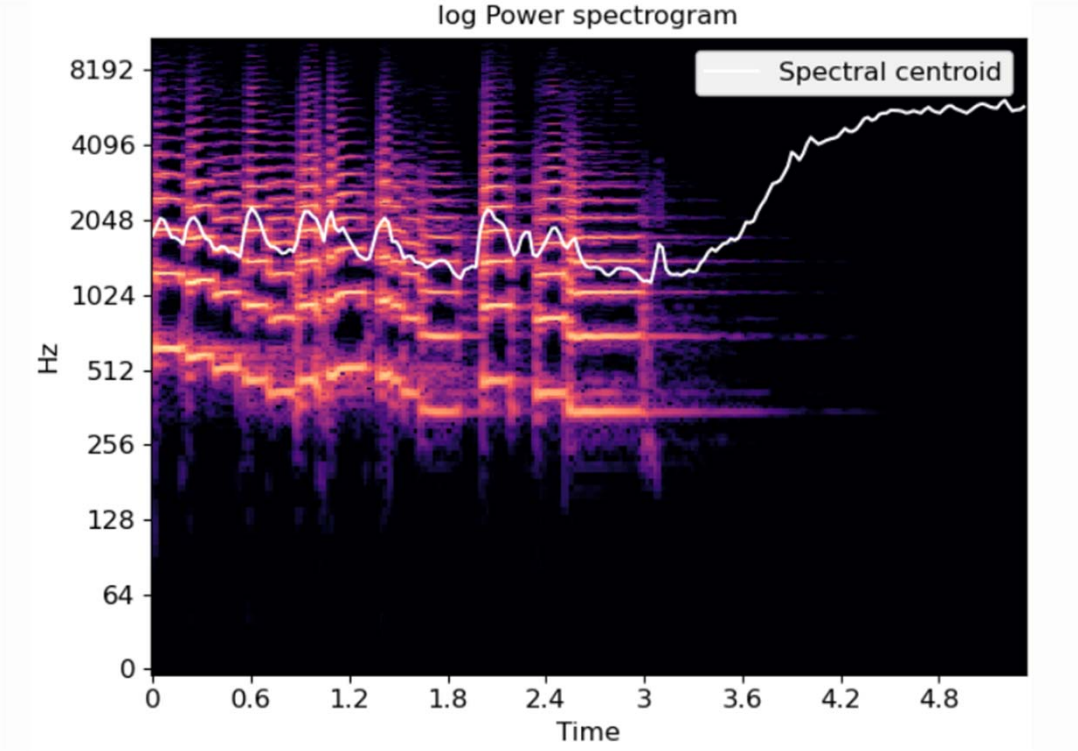
Spectral features

`melspectrogram` (*[, y, sr, S, n_fft, ...])	Compute a mel-scaled spectrogram.
`mfcc` (*[, y, sr, S, n_mfcc, dct_type, norm, ...])	Mel-frequency cepstral coefficients (MFCCs)
`rms` (*[, y, S, frame_length, hop_length, ...])	Compute root-mean-square (RMS) value for each frame, either from t or from a spectrogram `S`.
`spectral_centroid` (*[, y, sr, S, n_fft, ...])	Compute the spectral centroid.
`spectral_bandwidth` (*[, y, sr, S, n_fft, ...])	Compute p'th-order spectral bandwidth.
`spectral_contrast` (*[, y, sr, S, n_fft, ...])	Compute spectral contrast
`spectral_flatness` (*[, y, S, n_fft, ...])	Compute spectral flatness
`spectral_rolloff` (*[, y, sr, S, n_fft, ...])	Compute roll-off frequency.

Ref: McFee et al., “librosa: Audio and music signal analysis in python,” 2015

Spectral Centroid

- Each frame of a magnitude spectrogram is normalized and treated as a distribution over frequency bins, from which the mean (centroid) is extracted per frame
- A measure of spectral shape
- Higher centroid values correspond to "**brighter**" textures with more high frequencies
- Other statistics can also be used
  - bandwidth, skewness, kurtosis



Ref: Tzanetakis and Cook, "Musical genre classification of audio signals," TASLP 2002

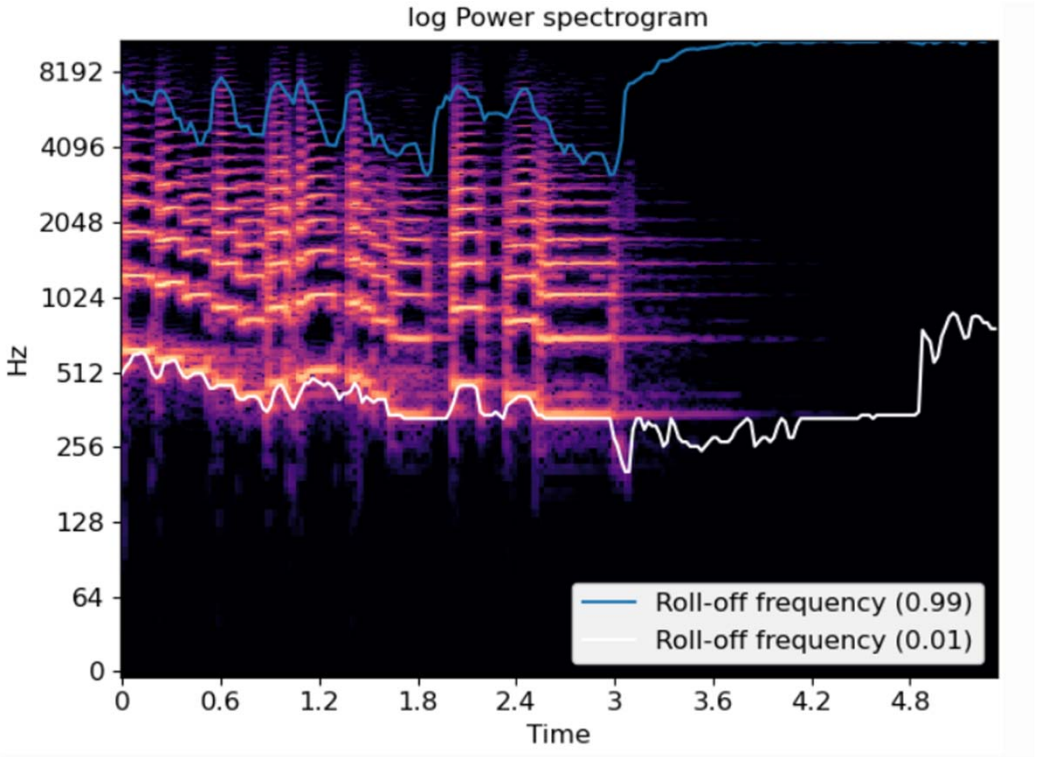
Spectral Contrast

- Each frame of a spectrogram is divided into sub-bands. For each sub-band, the energy contrast is estimated by comparing the mean energy in the top quantile (peak energy) to that of the bottom quantile (valley energy). High contrast values generally correspond to **clear, narrow-band signals**, while low contrast values correspond to **broad-band noise**

Ref: McFee et al., “librosa: Audio and music signal analysis in python,” 2015

Spectral Rolloff

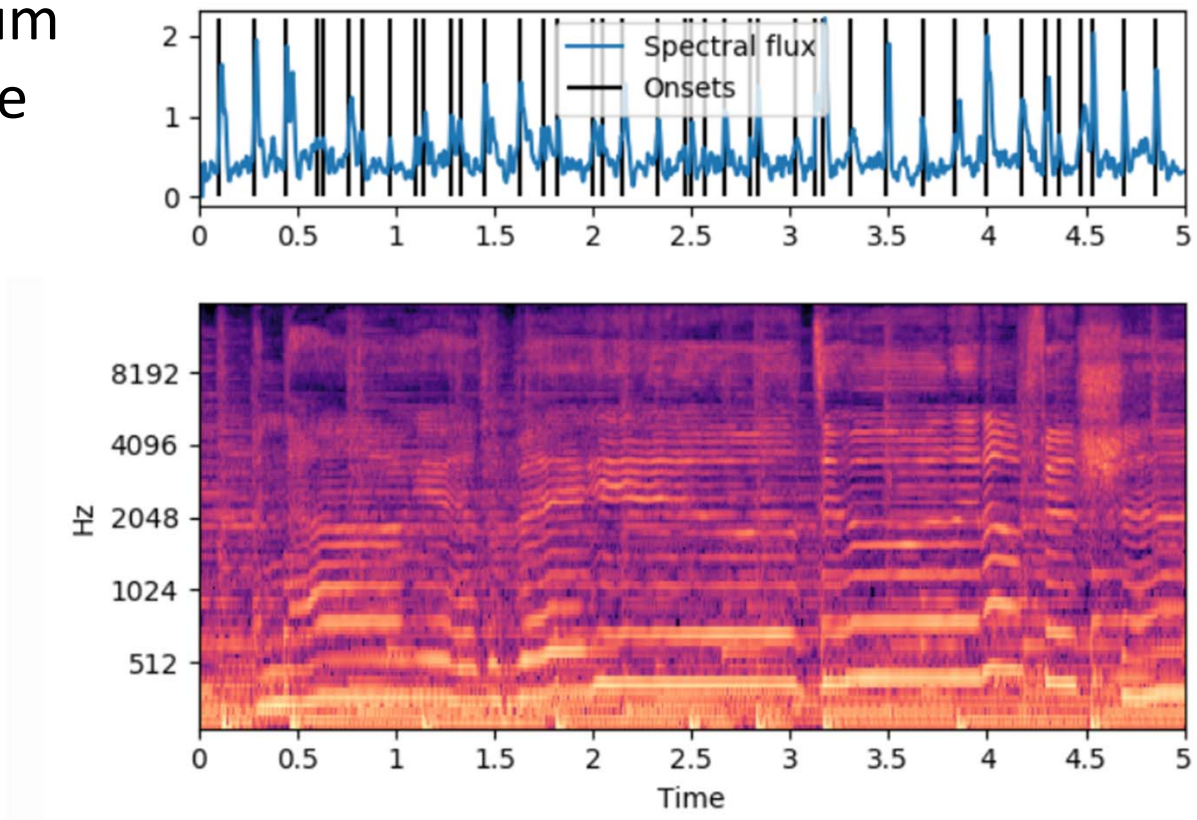
- The frequency for a spectrogram bin such that at least *roll_percent* (0.85 by default) of the **energy** of the spectrum in a frame is contained **in this bin and the bins below**
- Can be used to approximate the **maximum** (or **minimum**) frequency by setting roll_percent to a value close to 1 (or 0)
- Another measure of spectral shape



Ref: McFee et al., "librosa: Audio and music signal analysis in python," 2015

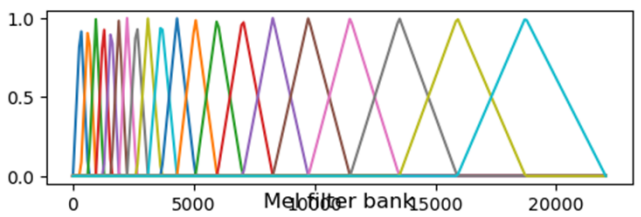
Spectral Flux

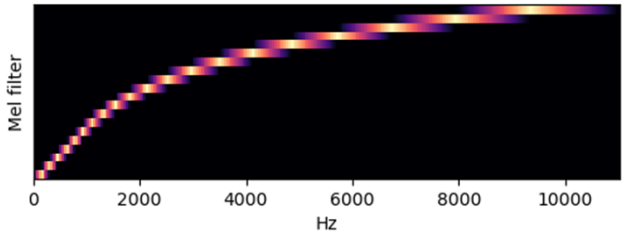
- How quickly the power spectrum of a signal is changing over time
- Usually calculated as the L2-norm between two adjacent normalised spectra
- Usually used for musical onset detection



Ref1: https://en.wikipedia.org/wiki/Spectral_flux
Ref2: https://librosa.org/librosa_gallery/auto_examples/plot_superflux.html

Mel-frequency cepstral coefficients (MFCC)





- Procedure
  1. Log-amplitude of the magnitude spectrum
  2. Grouping the FFT bins according to the *perceptually motivated* Mel-filter bank
     - **Linear** till 1,000 Hz and **logarithmic** above it
     - Usually with triangular overlapping windows
  3. Taking logs and DCT for uncorrelating the resulting features

- **Compact representation** of the spectrum (1,024-dim → 128 or 13 coefficients)
  - Somehow capture the relative strength of partials (without knowing the F0)
  - Less interpretable

Ref: McFee et al., "librosa: Audio and music signal analysis in python," 2015

Demonstrations 1

https://colab.research.google.com/github/stevetjoa/musicinformationretrieval.com/blob/gh-pages/spectral_features.ipynb

Signal Analysis and Feature Extraction

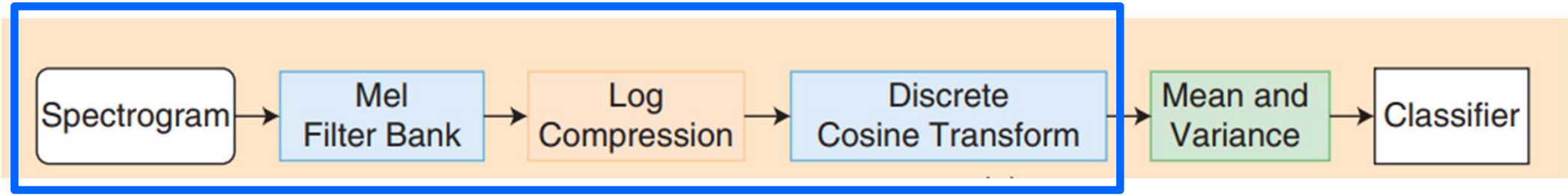
1. Basic Feature Extraction
2. Segmentation
3. Energy and RMSE
4. Zero Crossing Rate
5. Fourier Transform
6. Short-time Fourier Transform and Spectrogram
7. Constant-Q Transform and Chroma
8. Video: Chroma Features
9. Magnitude Scaling
10. Spectral Features
11. Autocorrelation
12. Pitch Transcription Exercise

Demonstrations 2

http://www.ifs.tuwien.ac.at/~schindler/lectures/MIR_Feature_Extraction.html

```
def spectral_centroid(wavedata, window_size, sample_rate):

    magnitude_spectrum = stft(wavedata, window_size)

    timebins, freqbins = np.shape(magnitude_spectrum)

    # when do these blocks begin (time in seconds)?
    timestamps = (np.arange(0,timebins - 1) * (timebins / float(sample_rate)))

    sc = []

    for t in range(timebins-1):

        power_spectrum = np.abs(magnitude_spectrum[t])**2

        sc_t = np.sum(power_spectrum * np.arange(1,freqbins+1)) / np.sum(power_spectrum)

        sc.append(sc_t)


    sc = np.asarray(sc)
    sc = np.nan_to_num(sc)

    return sc, np.asarray(timestamps)
```

Spectral Features Can be Used to Build a Classifier



- Procedure
  1. Compute the features per STFT frame (e.g., 13-D frame-level features)
  2. **Temporal pooling** over time for each audio clip (e.g., by taking the **mean and variance**; leading to 26-D clip-level features)
  3. Use that as input to a **classifier** (e.g., random forest, or support vector machine)
- Limits
  - *Clear* physical meaning but not sophisticated enough and *limited* **semantic** meaning
  - Only the classifier is trainable

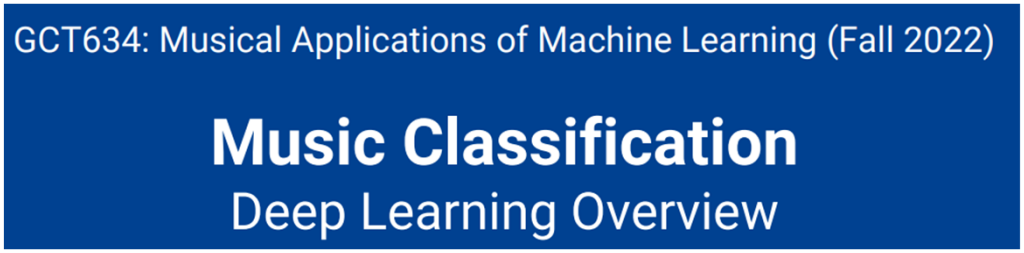
Outline

- General idea
- Representing timbre: signal processing perspective
  - Timbre features
- **Representing timbre: deep learning perspective**
  - **Timbre classifier & encoder**
- Other related topics on music classification
- HW1

KAIST Course & ISMIR 2021 Tutorial

For fundamentals of deep learning and music classification

- https://mac.kaist.ac.kr/~juhan/gct634/Slides/[week4-1]%20music%20classification%20-%20deep%20learning%20overview.pdf



- https://music-classification.github.io/tutorial/landing-page.html

Feature Learning

(the blocks inside the black lines are learned)

(a) Feature engineering

(b) Low-level feature learning

(c) Convolution neural networks

(d) End-to-end learning

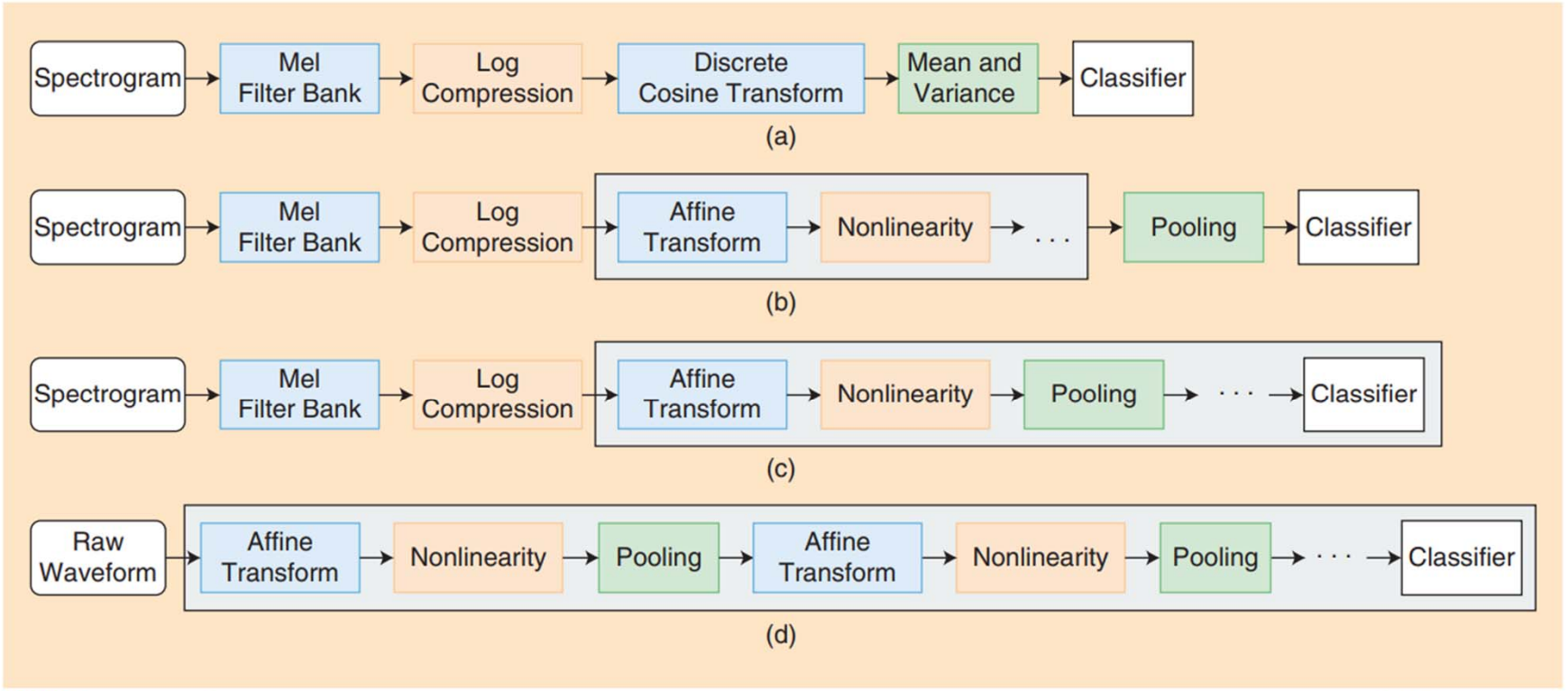


FIGURE 1. The transition of feature representation for music classification: (a) feature engineering [mel-frequency cepstral coefficients (MFCCs)], (b) low-level feature learning, (c) convolutional neural networks, and (d) end-to-end learning. The blocks inside the black lines indicate that they are learned by the algorithms.

Ref: Nam et al., "Deep learning for audio-based music classification and tagging," IEEE Signal Processing Magazine, 2019

Different Convolution Approaches

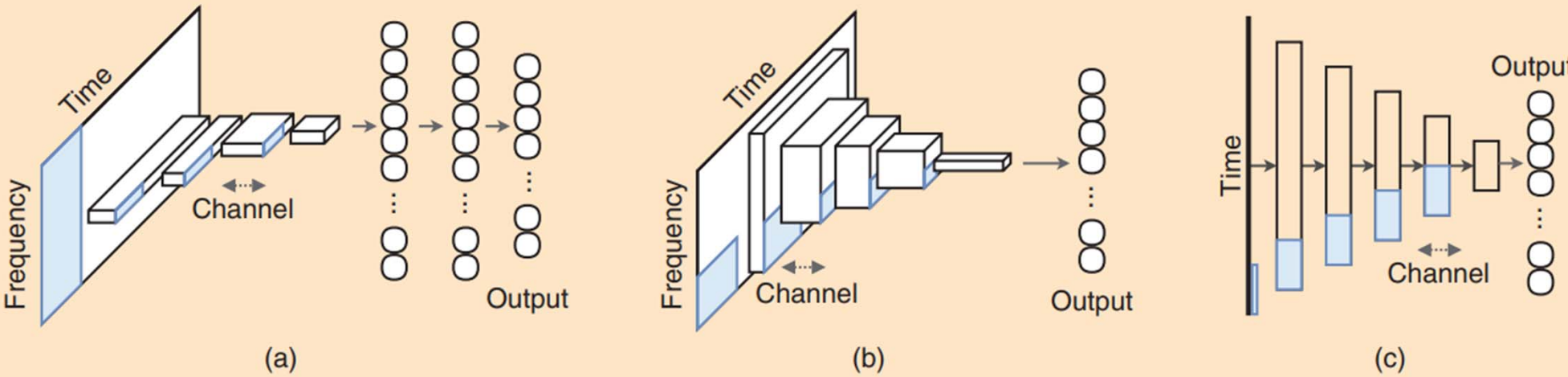


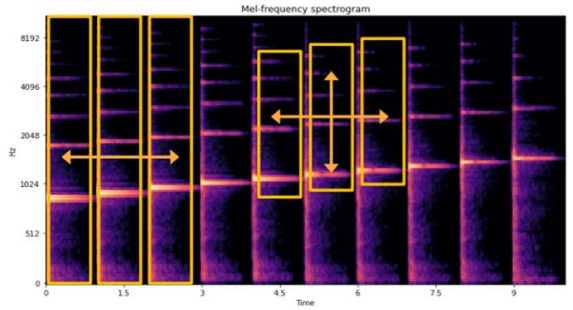
FIGURE 2. Block diagrams of (a) 1-D, (b) 2-D, and (c) sample-level CNNs. (a) and (b) are based on 2-D time–frequency representation inputs (e.g., mel-spectrograms or short-time Fourier transforms), and (c) is based on a time-series input.

- 1D CNNs
- 2D CNNs
- Sample-level CNNs

Convolution: Locality and Translation Invariance

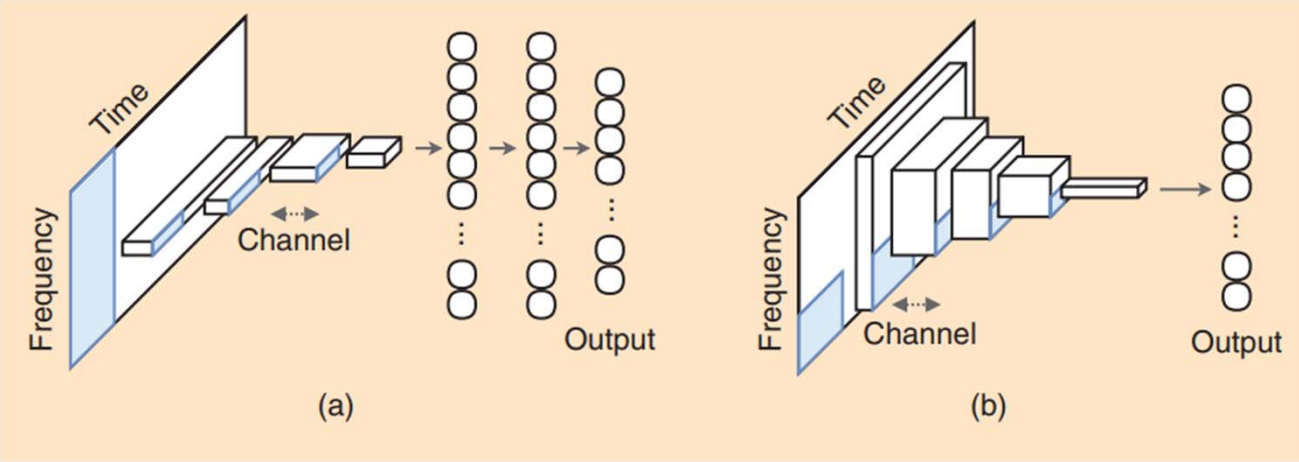
- Handle high-dimensional input based on locality and translation invariance of objects
  - Locality: the objects of interest tend to have a local spatial support
    Important parts of the object structures are locally correlated
  - Translation invariance: object appearance is independent of location

- The locality and translation invariance works in the audio domain
  - Both 1D and 2D translation are possible
  - 2D translation is only on the log-frequency scale which makes harmonic patterns approximately shifted-invariant



- "Convolution + pooling" may lead to translation invariance
- See Dr. Juhan Nam's slides (GCT634)

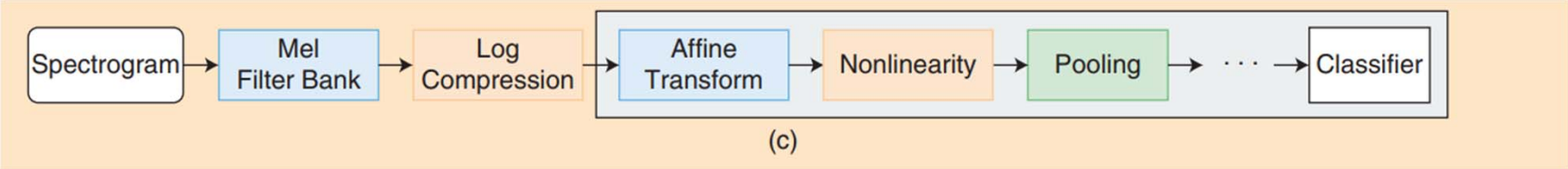
1D CNNs vs. 2D CNNs



- **1D CNNs**
  - The filter size of the first conv layer covers the *entire* frequency range (can cover multiple *frames*)
  - Fast to train
  - **Time**-invariant but not **pitch**-invariant
- **2D CNNs**
  - Significantly increases the number of parameters and thus need more computational resources
  - More flexible and powerful

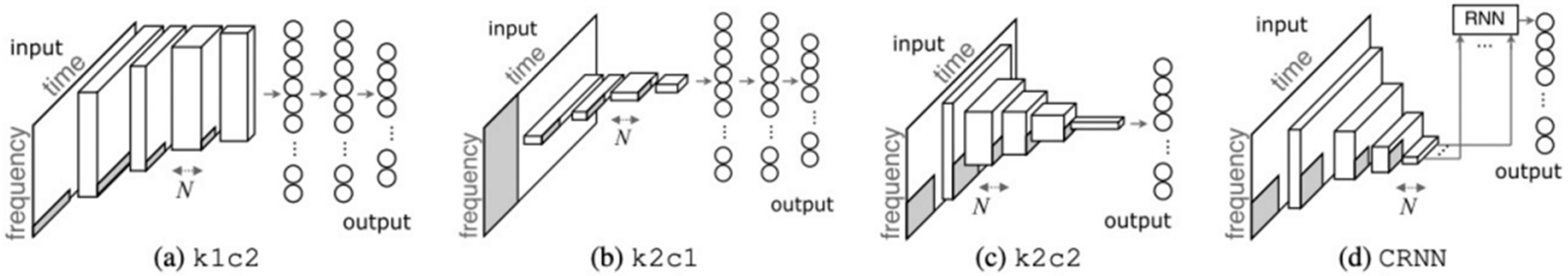
Input Audio Representation

https://music-classification.github.io/tutorial/part2_basics/input-representations.html



- **Log-magnitude STFT**
  - The most "raw" kind of spectrograms
  - Discard phase: the human auditory system is insensitive to phase information
  - Log compresssion: human perception of loudness is closer to a logarithmic scale
- **Melspectrograms**
  - Based on a Mel-scale, which is nonlinear and approximates human perception
  - Reduces the number of frequency band greatly (1,024 → 128)

Convolutional Recurrent Neural Networks

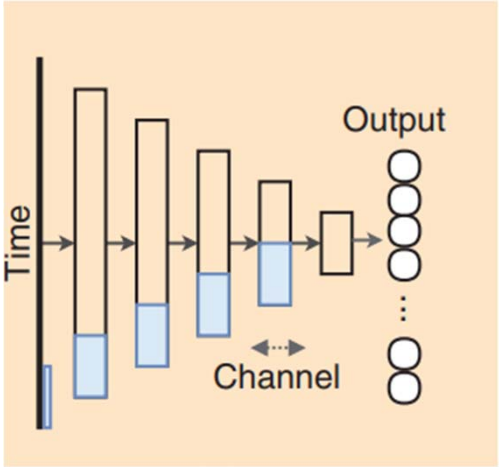


- See Dr. Juhan Nam's slides (GCT634)
- Use RNN for temporal summary
- The CRNN model slightly outperforms the CNN models but it is slower

Ref: Choi et al., "Convolutional recurrent neural networks for music classification," ICASSP 2017

Sample-level CNN

- Work on audio samples (e.g., two or three samples) rather than a typical window size (e.g., 512 samples)
- Longer training time; need larger compute



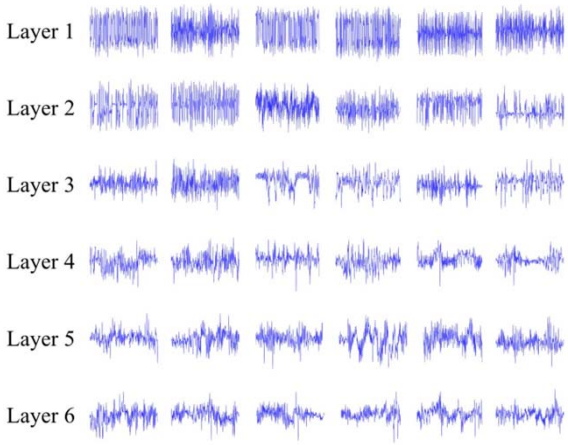
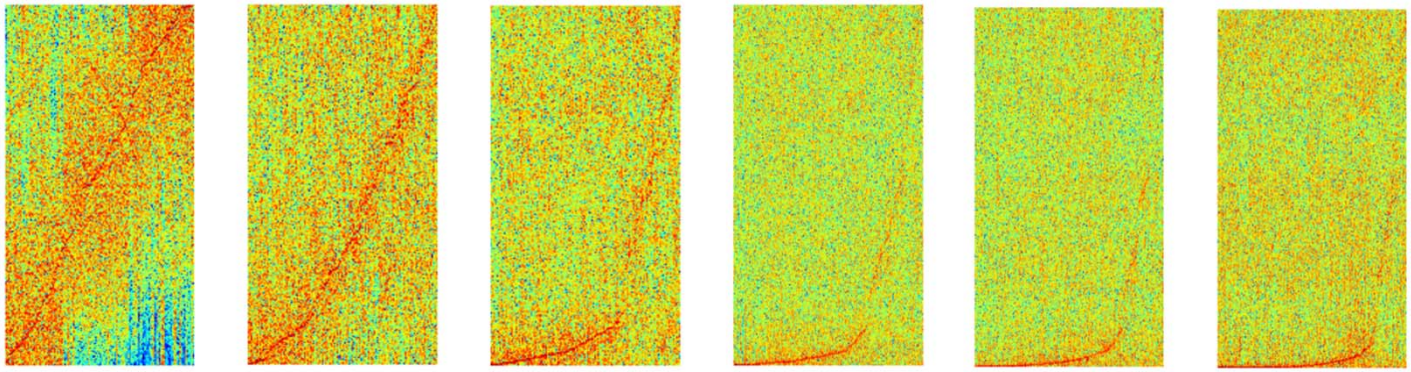


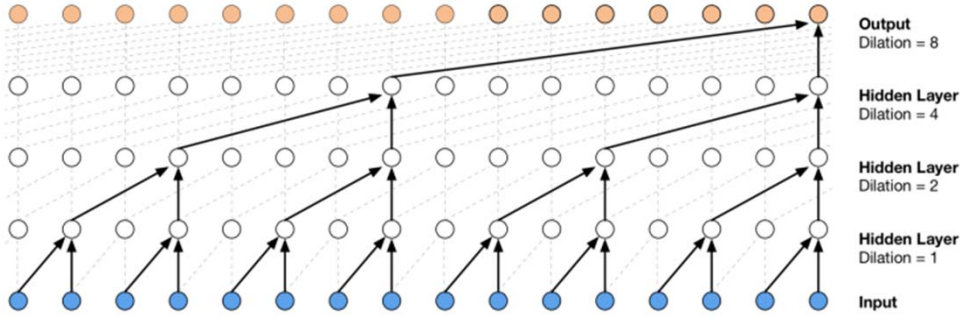
Figure 3. Examples of learned filters at each layer.



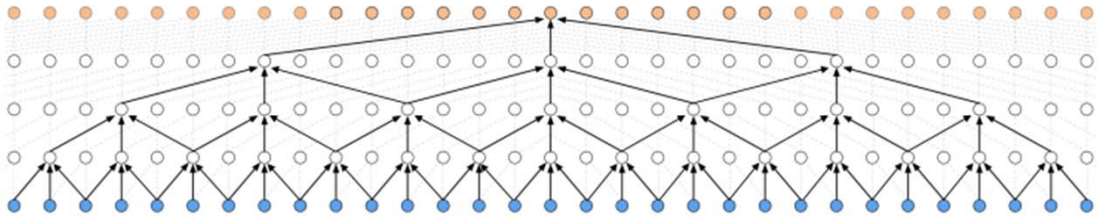
Ref: Lee et al., “Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms,” SMC 2017

Temporal Convolution Network (TCN)

- *WaveNet*-like sample-level **dilated** convolution
  - Convolutions across sub-sampled input representations
- Will talk more about this in later lectures



a stack of dilated causal convolutional layers



a stack of dilated non-causal convolutional layers

Ref 1: van den Oord et al, “Wavenet: A generative model for raw audio,” ISCA 2016
Ref 2: Lemaire & Holzapfel, “Temporal convolutional networks for speech and music detection in radio broadcast,” ISMIR 2019

Exemplar Models: Short-Chunk CNN

https://github.com/minzwon/sota-music-tagging-models

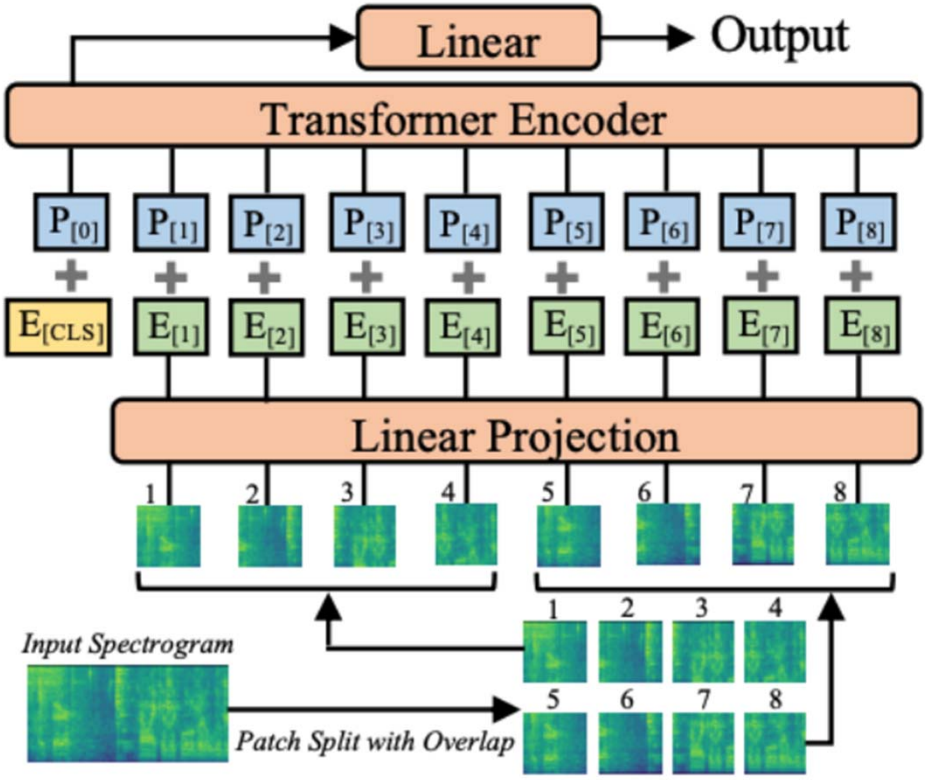
Available Models

- **FCN** : Automatic Tagging using Deep Convolutional Neural Networks, Choi et al., 2016 [arxiv]
- **Musicnn** : End-to-end Learning for Music Audio Tagging at Scale, Pons et al., 2018 [arxiv]
- **Sample-level CNN** : Sample-level Deep Convolutional Neural Networks for Music Auto-tagging Using Raw Waveforms, Lee et al., 2017 [arxiv]
- **Sample-level CNN + Squeeze-and-excitation** : Sample-level CNN Architectures for Music Auto-tagging Using Raw Waveforms, Kim et al., 2018 [arxiv]
- **CRNN** : Convolutional Recurrent Neural Networks for Music Classification, Choi et al., 2016 [arxiv]
- **Self-attention** : Toward Interpretable Music Tagging with Self-Attention, Won et al., 2019 [arxiv]
- **Harmonic CNN** : Data-Driven Harmonic Filters for Audio Representation Learning, Won et al., 2020 [pdf]
- **Short-chunk CNN** : Prevalent 3x3 CNN. So-called *vgg*-ish model with a small receptieve field.
- **Short-chunk CNN + Residual** : Short-chunk CNN with residual connections.

Exemplar Models: Audio Spectrogram Transformer

https://github.com/YuanGongND/ast

- Use Vison **Transformer** (ViT) based architecture
  - The first convolution-free, purely attention-based model for audio classification
- May need larger amount of training data and compute



Ref: Gong et al., "AST: Audio Spectrogram Transformer," INTERSPEECH 2021

Exemplar Models: PANNs

https://github.com/qiuqiangkong/audioset_tagging_cnn

- Purely convolutional
- Can be used as a pre-trained model
  - Produce audio embeddings that have been used by **CLAP** (https://github.com/LAION-AI/CLAP) in learning audio-text joint embedding space for **text-to-audio** generation

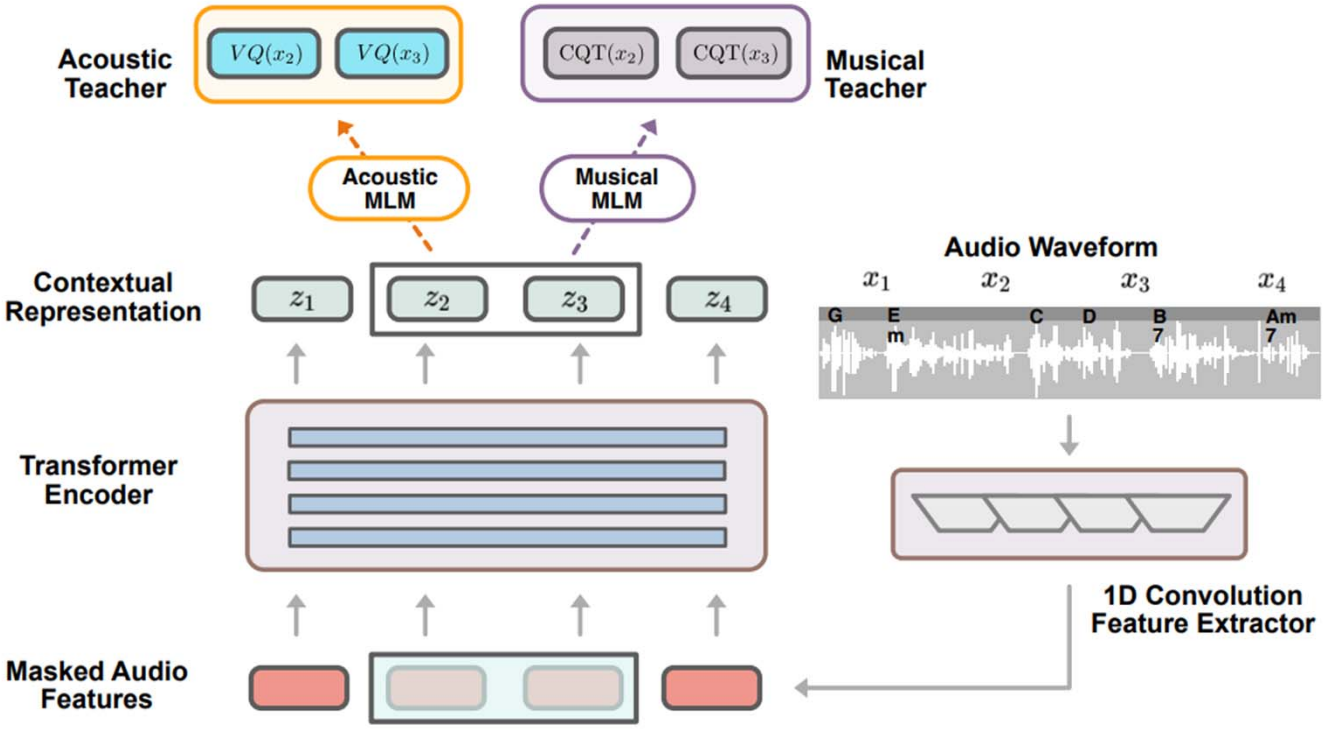
VGGish [1]	CNN6	CNN10	CNN14
Log-mel spectrogram 96 frames × 64 mel bins	Log-mel spectrogram 1000 frames × 64 mel bins		
3 × 3 @ 64 ReLU	5 × 5 @ 64 BN, ReLU	(3 × 3 @ 64 BN, ReLU) × 2	(3 × 3 @ 64 BN, ReLU) × 2
MP 2 × 2	Pooling 2 × 2		
3 × 3 @ 128 ReLU	5 × 5 @ 128 BN, ReLU	(3 × 3 @ 128 BN, ReLU) × 2	(3 × 3 @ 128 BN, ReLU) × 2
MP 2 × 2	Pooling 2 × 2		
(3 × 3 @ 256 ReLU) × 2	5 × 5 @ 256 BN, ReLU	(3 × 3 @ 256 BN, ReLU) × 2	(3 × 3 @ 256 BN, ReLU) × 2
MP 2 × 2	Pooling 2 × 2		
(3 × 3 @ 512 ReLU) × 2	5 × 5 @ 512 BN, ReLU	(3 × 3 @ 512 BN, ReLU) × 2	(3 × 3 @ 512 BN, ReLU) × 2
MP 2 × 2 Flatten	Global pooling		Pooling 2 × 2
FC 4096 ReLU × 2	FC 512, ReLU		(3 × 3 @ 1024 BN, ReLU) × 2
FC 527, Sigmoid	FC 527, Sigmoid		Pooling 2 × 2
			(3 × 3 @ 2048 BN, ReLU) × 2
			Global pooling
			FC 2048, ReLU
			FC 527, Sigmoid

Ref: Kong et al., "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," arXiv 2020

Pre-trained Models: MERT

https://github.com/yizhilll/MERT

- Use **sample-level** CNNs
- Pre-trained on millions of songs using self-supervised learning objectives
- Promising result in downstream tasks
  - https://marble-bm.shef.ac.uk/leaderboard



Ref: Li et al., "MERT: Acoustic music understanding model with large-scale self-supervised training," arXiv 2023

Question: Do They Produce “Timbre Embeddings”?

- These are music classifiers
  - May learn to exploit information other than timbre for higher accuracy

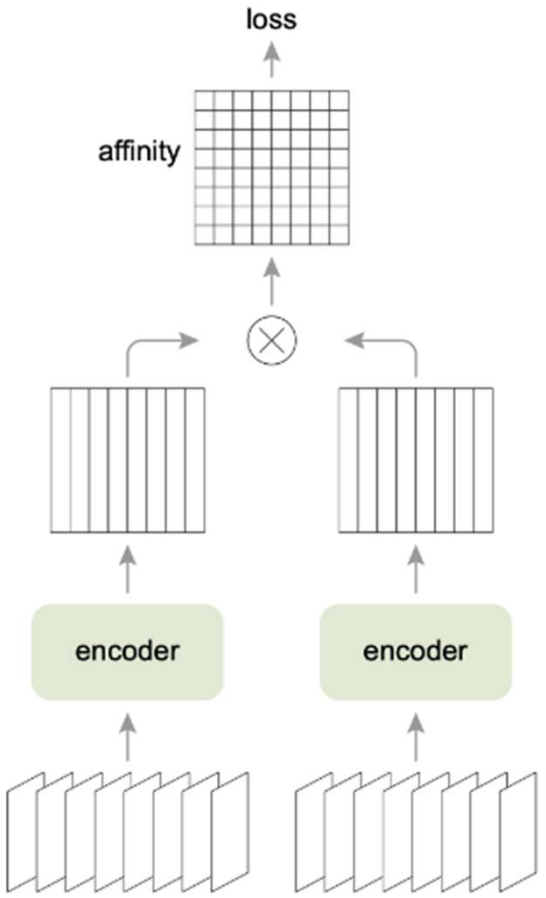
- May learn timbre embeddings when the classification task is **about timbre**
  - Examples: **singer recognition**, speaker recognition, instrument recognition
  - But there may be other **confounding factors**

- There are techniques to further reduce non-timbre information
  - Data augmentation
  - **Contrastive learning**
  - Information bottleneck

Contrastive Learning

https://music-classification.github.io/tutorial/part5_beyond/methods.html

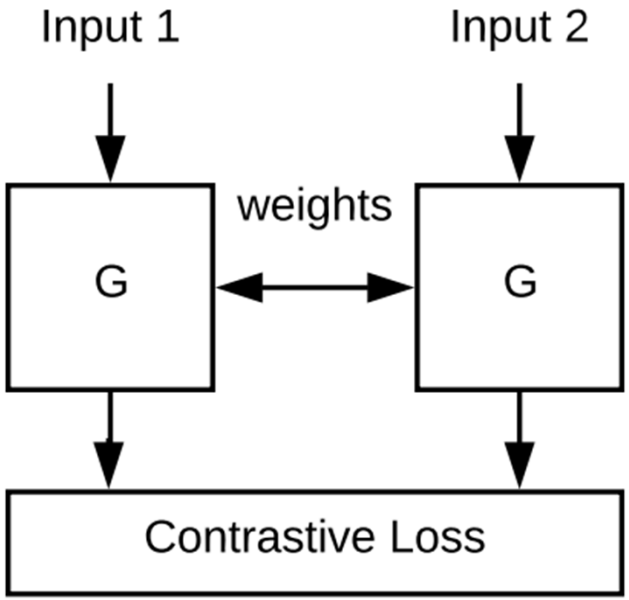
- Learning representations by modeling **similarity** from natural variations of data
- *Self-supervised learning* (SSL): Need no labels
- Can be used to establish "foundation models"
  - Pre-train on a big, unlabeled data using SSL
  - Freeze the network
  - Add a few dense layers which are trained on downstream tasks



Ref: Lee and Nam, "Learning a joint embedding space of monophonic and mixed music signals for singing voice," ISMIR 2019

Contrastive Learning for Learning Timbre Representation

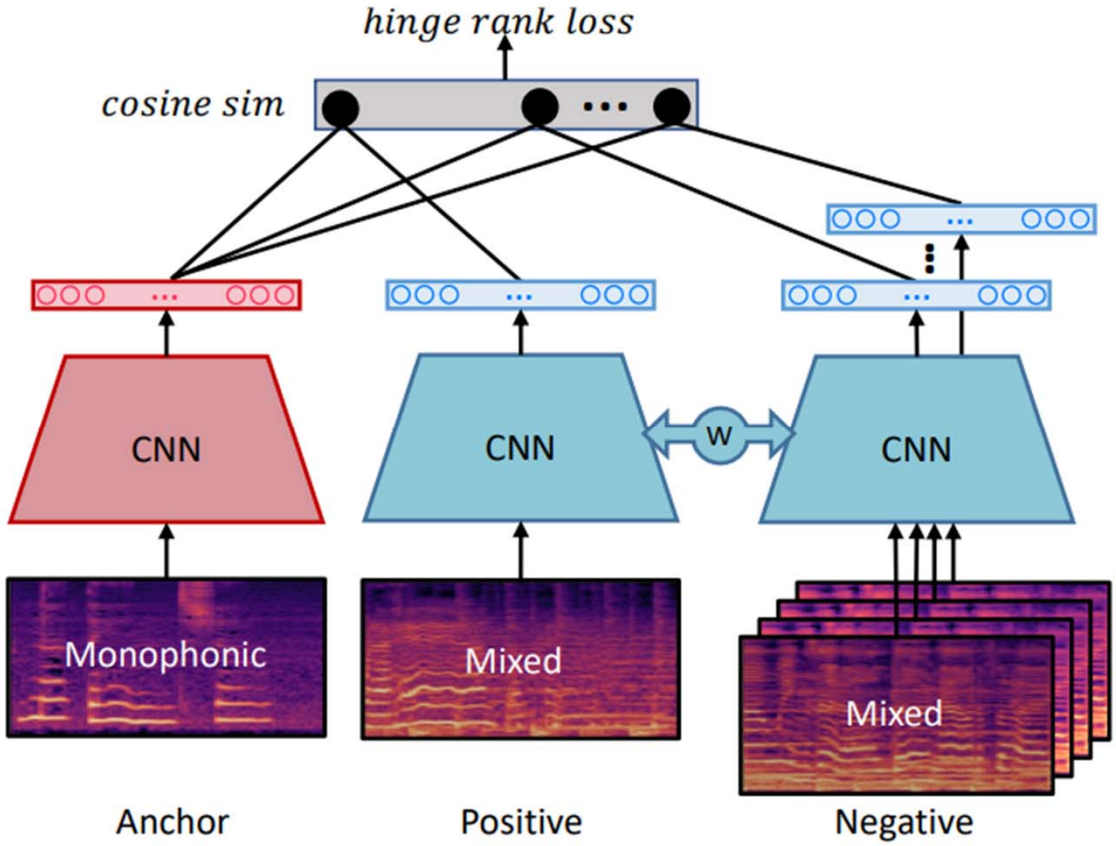
- [wang18icassp]
  - Learn an embedding space that recordings sang by the **same singers are close** to each other, while the those sang by **different singers are far away** from each other
  - *Siamese* architecture: Tie two identical neural networks, with shared weights



Ref: Wang and Tzanetakis, "Singing style investigation by residual siamese convolutional neural networks," ICASSP 2018

Contrastive Learning for Learning Timbre Representation

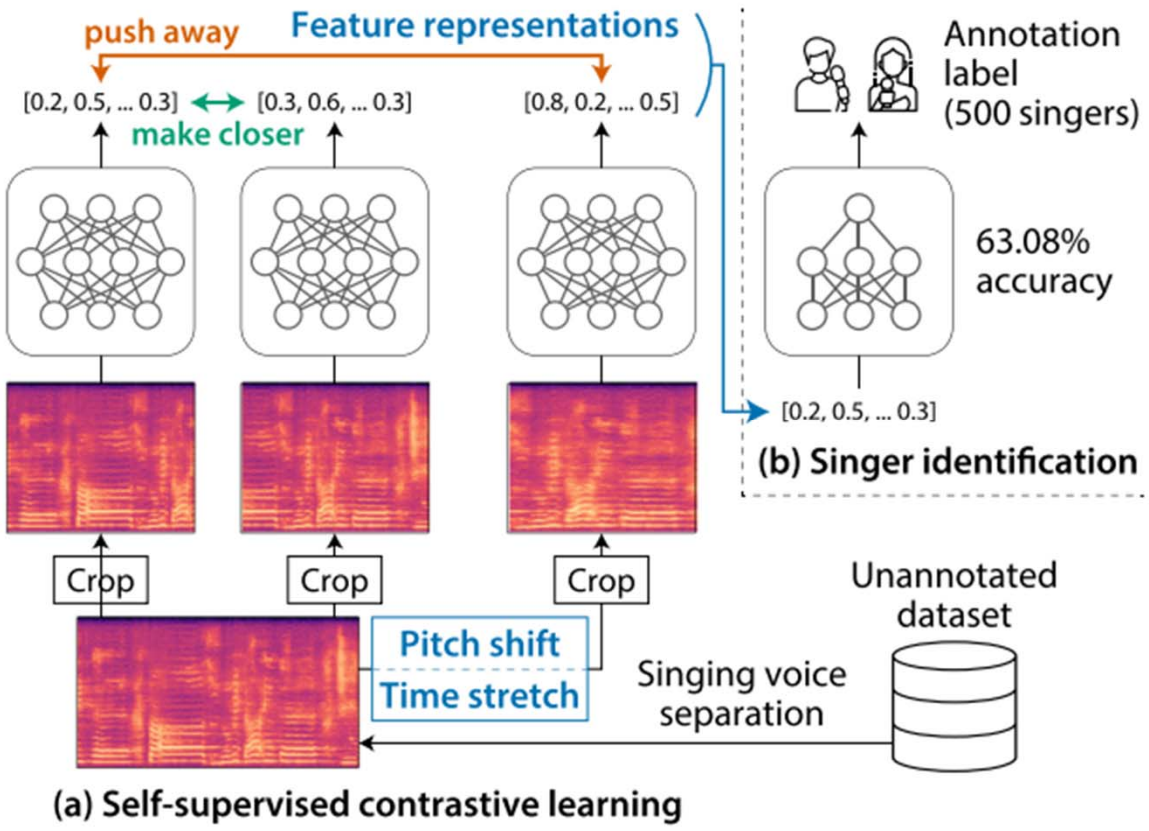
- [lee19ismir]
  - Positive pairs: **monophonic** and **mixed** tracks of the same singer
  - May close the gap between monophonic and mixed tracks without source separation
    - Application: given a monophonic track of a singer as a query, retrieving mixed tracks sung by the same singer from the database



Ref: Lee and Nam, "Learning a joint embedding space of monophonic and mixed music signals for singing voice," ISMIR 2019

Contrastive Learning for Learning Timbre Representation

- [yakura23taslp]
  - Positive pairs: **different cuts** from the **same song**
  - Negative pairs: **transformed versions** of the **same song**
  - Pitch shifting
    - Change both pitch and formants → lead to change in *timbre*
  - Time stretching
    - Change vibrato rate and F0 contour in note transitions → lead to change in *expression*

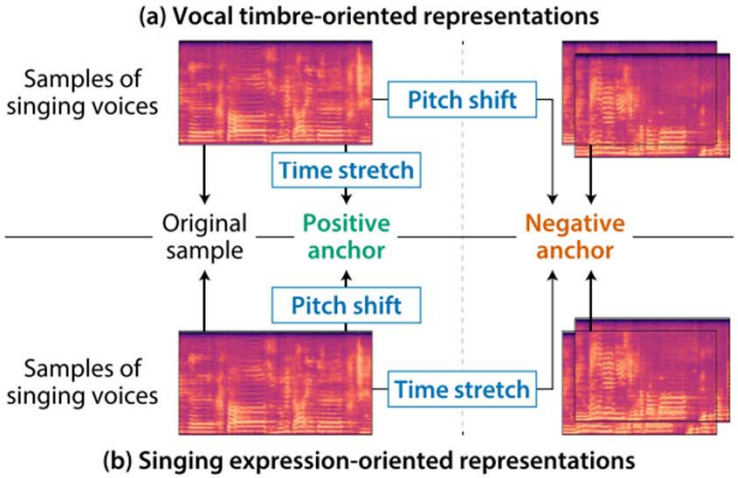


Ref: Yakura et al., "Self-supervised contrastive learning for singing voices," TASLP 2023

Self-Supervised Contrastive Learning for Singing Voices

ACCURACY OF SINGER IDENTIFICATION (500 SINGERS)

Feature extraction	Top-1 acc.	Top-5 acc.
Conventional features (MFCC, etc.)	0.20%	1.00%
CLMR [41]	47.96%	73.64%
Proposed	53.96%	76.60%
Proposed + time stretching	61.00%	81.16%
Proposed + pitch shifting	61.28%	81.72%
Proposed + both	**63.08%**	**82.16%**



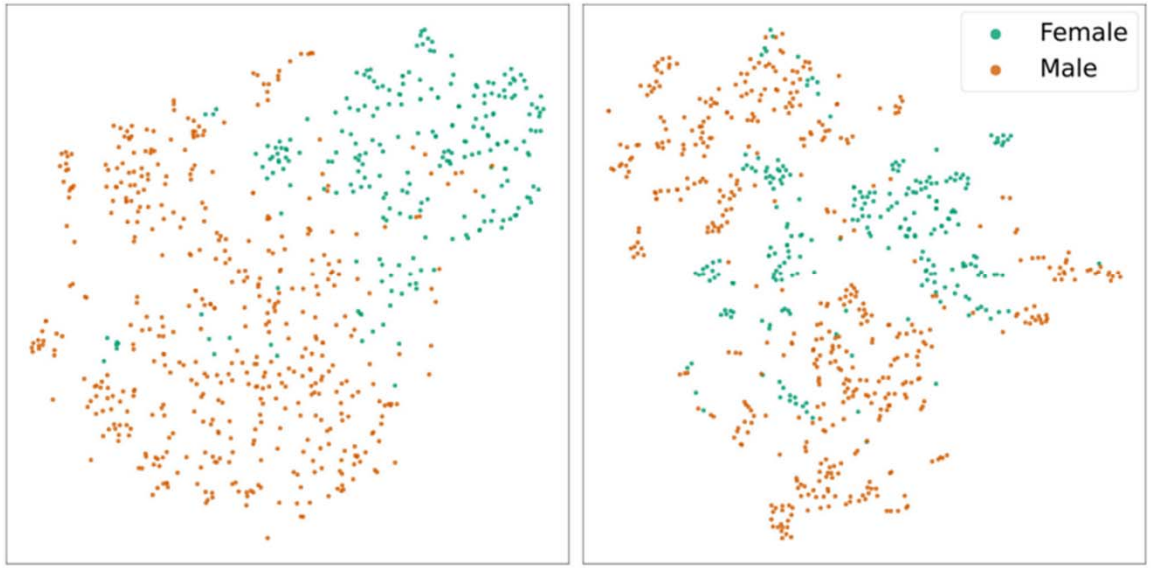


Fig. 4. Feature representations visualized by t-SNE. The plot of (a) vocal timbre-oriented representations provided a clearer distinction between male and female singers than (b) singing expression-oriented representations.

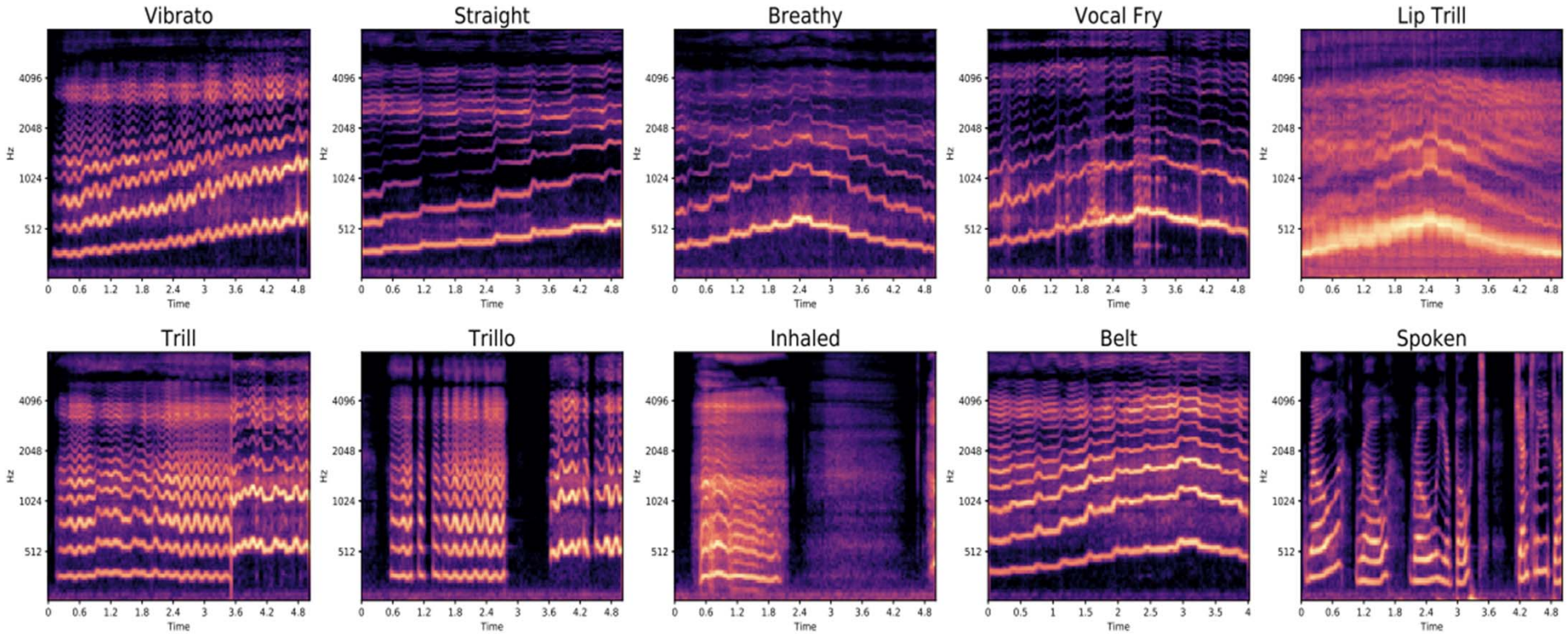
Ref: Yakura et al., "Self-supervised contrastive learning for singing voices," TASLP 2023

Outline

- General idea
- Representing timbre: signal processing perspective
- Representing timbre: deep learning perspective
- **Other related topics on music classification**
- HW1

Other Classification Problems: Playing Technique

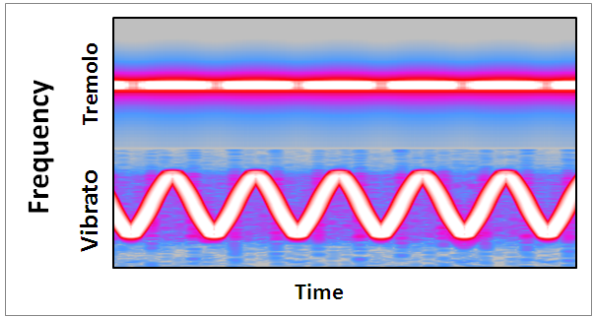
VocalSet (https://zenodo.org/record/1193957)



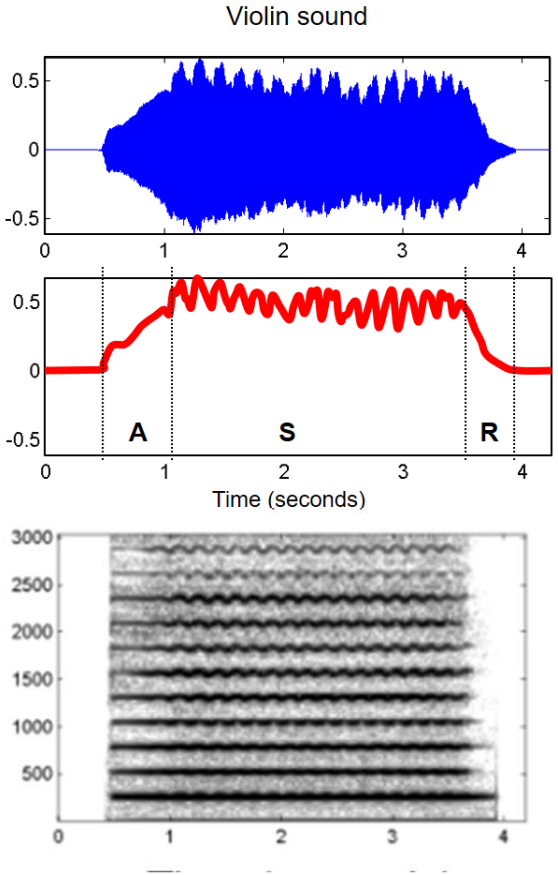
Ref: Wilkins et al., "VocalSet: A singing voice dataset," ISMIR 2018

Vibrato and Tremolo

- **Tremolo**: periodic variations in amplitude (amplitude modulations)
- **Vibrato**: periodic variations in frequency (frequency modulations)
  - Wind and bowed instruments generally use vibratos with an extent of less than half a *semitone* either side
  - Tremolo and vibrato do not necessarily evoke a perceived change in loudness or pitch of the tone



https://en.wikipedia.org/wiki/Vibrato



Ref: Sundberg, "Acoustic and psychoacoustic aspects of vocal vibrato," 1994

Other Classification Problems: Playing Technique

E-Guitar playing technique detection (https://zenodo.org/record/1414806)

- **Bend**: stretching the string with left hand to increase the pitch of the bended note either gradually or instantly
- **Vibrato**: minute and rapid variations in pitch
- **Hammer-on**: when a note is sounded, a left hand finger is used to quickly press down a fret that is on the same string while the first note is still ringing
- **Pull-off**: when you have strummed one note and literally pull off of the string to a lower note
- **Slide**: the action of slide left hand finger across one or more frets to reach another note

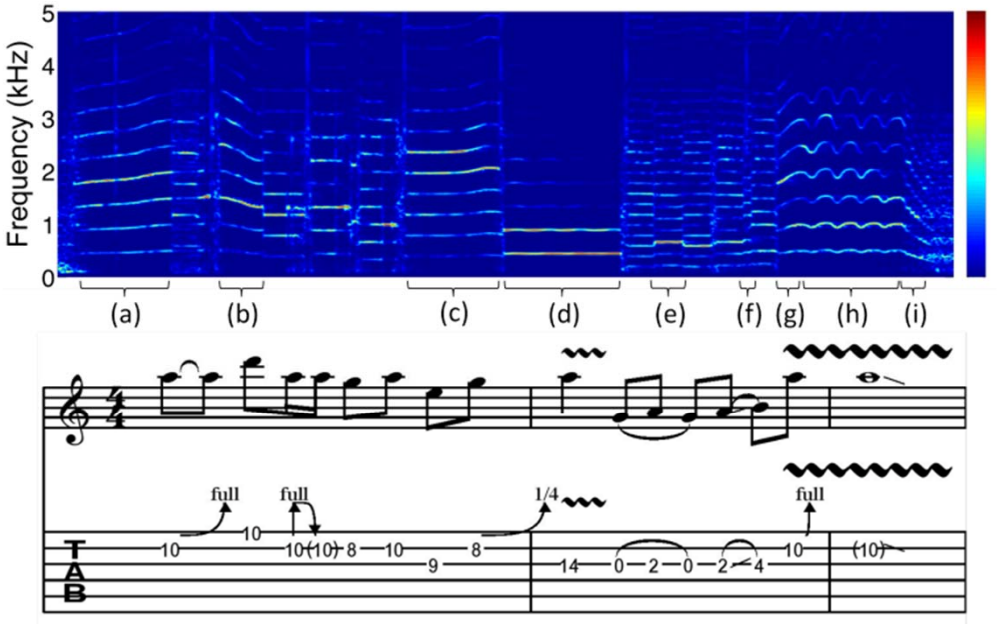


Figure 1. The spectrogram and tablature of a guitar phrase that contains the following techniques: bend (a, b, c, g), vibrato (d, h), hammer-on & pull-off (e) and slide (f, i).

Ref: Chen et al., "Electric guitar playing technique detection in real-world recording based on F0 sequence pattern recognition," ISMIR 2015

Other Classification Problems: Emotion

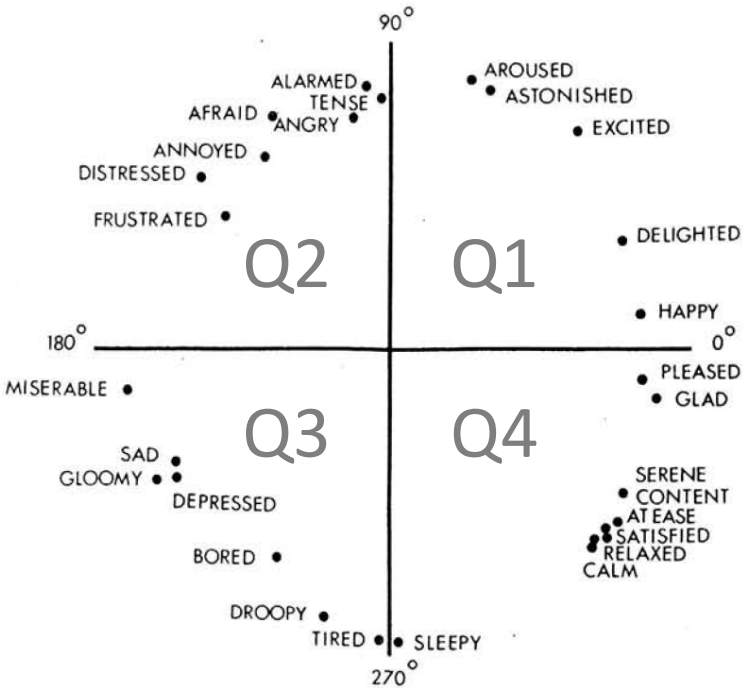
EMOPIA (https://annahung31.github.io/EMOPIA/)

Model	4Q	A	V
Logistic regression	.523	**.919**	.558
Short-chunk ResNet [50]	**.677**	.887	**.704**

Table 5. Audio-domain classification performance.

Arousal

energy or neurophysiological stimulation level



Valence

pleasantness or positive/negative affective states

Ref: Hung et al., "EMOPIA: A multi-modal pop piano dataset for emotion recognition and emotion-based music generation," ISMIR 2021

Other Classification Problem: Auto-Tagging

MagnaTagATune (https://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset)

- Top 50 by categories (source)
  - **genre**: classical, techno, electronic, rock, indian, opera, pop, classic, new age, dance, country, metal
  - **instrument**: guitar, strings, drums, piano, violin, vocal, synth, female, male, singing, vocals, no vocals, harpsichord, flute, no vocal, sitar, man, choir, voice, male voice, female vocal, harp, cello, femal voice, choral
  - **mood**: slow, fast, ambient, loud, quiet, soft, weird
  - **etc**: beat, solo, beats

Ref: Law et al., "Evaluation of algorithms using games: the case of music annotation," ISMIR 2009

MARBLE

Music Audio Representation Benchmark for universaL Evaluation

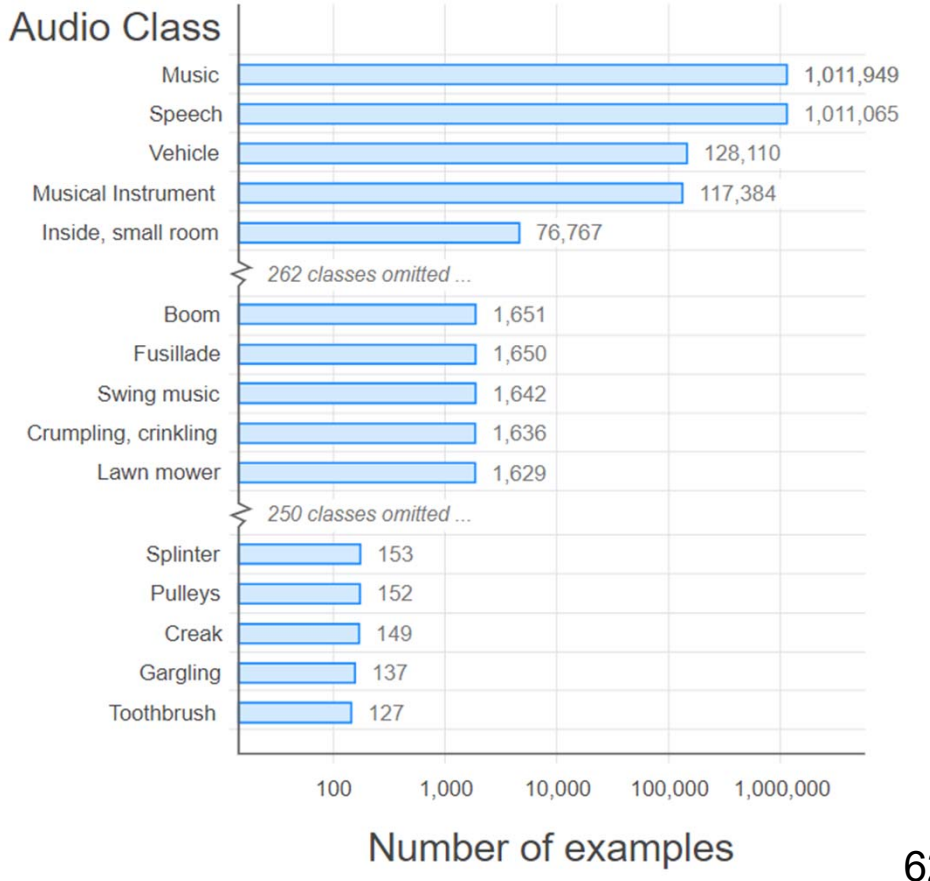
https://marble-bm.shef.ac.uk/

Dataset	MTT		GS	GTZAN	GTZAN	EMO		Nsynth	NSynth	VocalSet	VocalSet
Task	Tagging		Key	Genre	Rhythm	Emotion		Instrument	Pitch	Tech	Singer
Metric	ROC	AP	$Acc^{Refined}$	Acc	$F1^{beat}$	$R2^{V}$	$R2^{A}$	Acc	Acc	Acc	Acc
MAP-MERT-v0-95M	90.7	38.2	64.1	74.8	**88.3**	52.9	69.9	70.4	92.3	73.6	77.0
MAP-MERT-v0-95M-public	90.7	38.4	**67.3**	72.8	88.1	59.1	72.8	70.4	92.3	75.6	78.0
MAP-MERT-v1-95M	91.0	39.3	63.5	74.8	**88.3**	55.5	**76.3**	70.7	92.6	74.2	83.7
MAP-MERT-v1-330M	91.1	39.5	61.7	77.6	87.9	59.0	75.8	72.6	**94.4**	**76.9**	87.1
MAP-Music2Vec	90.0	36.2	50.6	74.1	68.2	52.1	71.0	69.3	93.1	71.1	81.4
MusiCNN	90.3	37.8	14.4	73.5	-	44.0	68.8	72.6	64.1	70.3	57.0
CLMR	89.5	36.0	14.8	65.2	-	44.4	70.3	67.9	47.0	58.1	49.9
Jukebox-5B	**91.4**	**40.6**	63.8	**77.9**	-	57.0	73.0	70.4	91.6	76.7	82.6
MULE	91.2	40.1	64.9	75.5	-	**60.7**	73.1	**74.6**	88.5	75.5	**87.5**

Other Classification Problem: General Audio

Audio Set (http://research.google.com/audioset/)

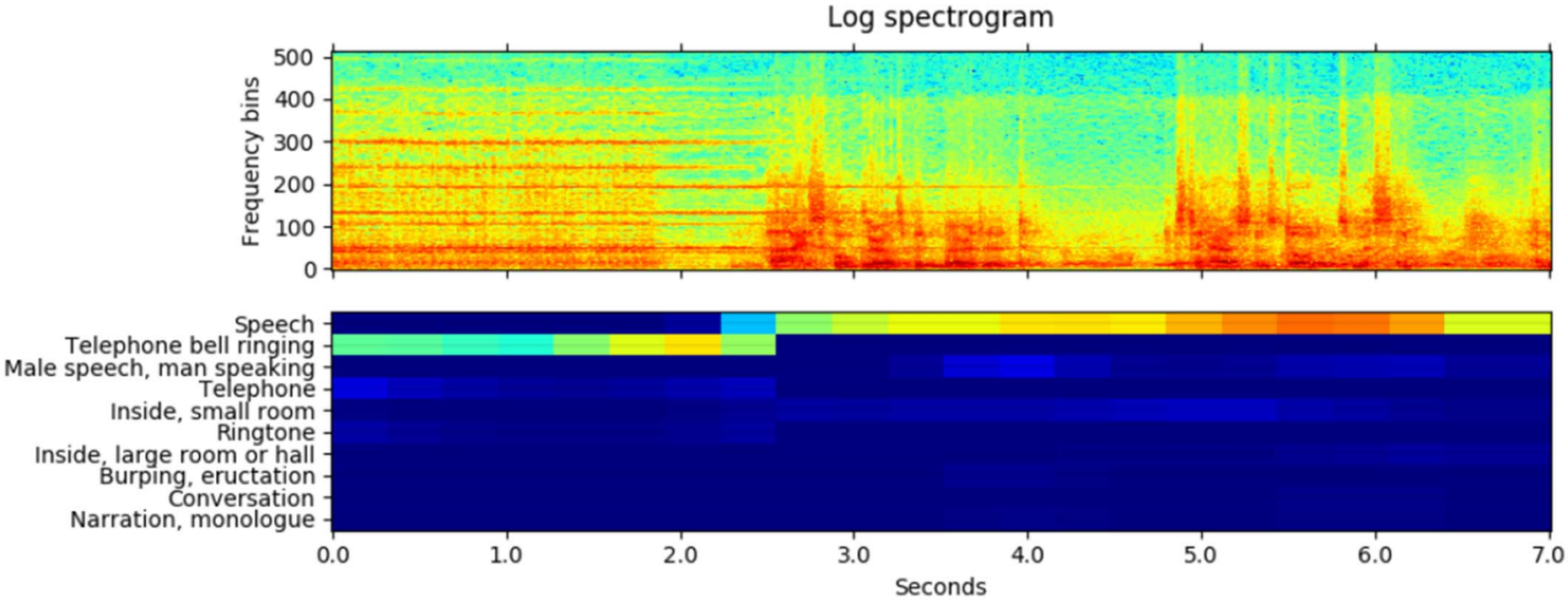
- 635 audio classes
  - Over 2M audio clips
  - Each clip is a 10 second segment of YouTube videos
- Widely used benchmark for **audio classification** and **audio captioning**
- Can be useful for **sound design**
  - Example: *Speak-like-a-dog* (link)



Ref: Gemmeke et al., "Audio Set: An ontology and human-labeled dataset for audio events," ICASSP 2017

Other Classification Problem: Audio Event Detection

https://github.com/qiuqiangkong/audioset_tagging_cnn



- Useful for
  - Music/singing/speech detection; instrument activity detection

Copyright-free Music Audio Datasets

- Free Music Archive (FMA)
  - https://freemusicarchive.org/
  - https://github.com/mdeff/fma
- Jamendo dataset
  - https://www.jamendo.com/
  - https://github.com/MTG/mtg-jamendo-dataset
- FreeSound
  - https://freesound.org/
  - https://labs.freesound.org/datasets/

Data Augmentation

https://music-classification.github.io/tutorial/part3_supervised/data-augmentation.html

Code Libraries

Name	Author	Framework	Language	License	Link
Muda	B. McFee et al. (2015)	General Purpose	Python	ISC License	source code
Audio Degradation Toolbox	M. Mauch et al. (2013)	General Purpose	MATLAB	GNU General Public License 2.0	source code
rubberband	-	General Purpose	C++	GNU General Public License (non-commercial)	website, pyrubberband

Audio Degradation Toolbox

https://github.com/sevagh/audio-degradation-toolbox

- Exemplar use case: simulate the case of smartphone recording

```
{ "name": "noise", ["snr": 20, "color": "pink"] }
{ "name": "mp3", ["bitrate": 320] }
{ "name": "gain", ["volume": 10.0] }
{ "name": "normalize" }
{ "name": "low_pass", ["cutoff": 1000.0] }
{ "name": "high_pass", ["cutoff": 1000.0] }
{ "name": "trim_millis", ["amount": 100, "offset": 0] }
{ "name": "mix", "path": STRING, ["snr": 20.0] }
{ "name": "speedup", "speed": FLOAT }
{ "name": "resample", "rate": INT }
{ "name": "pitch_shift", "octaves": FLOAT }
{ "name": "dynamic_range_compression", ["threshold": -20.0, "ratio": 4.0, "attack": 5.0, "release": 50.0]
{ "name": "impulse_response", "path": STRING }
{ "name": "equalizer", "frequency": FLOAT, ["bandwidth": 1.0, "gain": -3.0] }
{ "name": "time_stretch", "factor": FLOAT }
{ "name": "delay", "n_samples": INT }
{ "name": "clipping", ["n_samples": 0, "percent_samples": 0.0] }
{ "name": "wow_flutter", ["intensity": 1.5, "frequency": 0.5, "upsampling_factor": 5.0 ] }
{ "name": "aliasing", ["dest_frequency": 8000.0] }
```

Outline

- General idea
- Representing timbre: signal processing perspective
- Representing timbre: deep learning perspective
- Other related topics on music classification
- **HW1**