Course assessment survey

Take a few minutes and do it now

NYU Center for Data Science

Week N: What do I do?

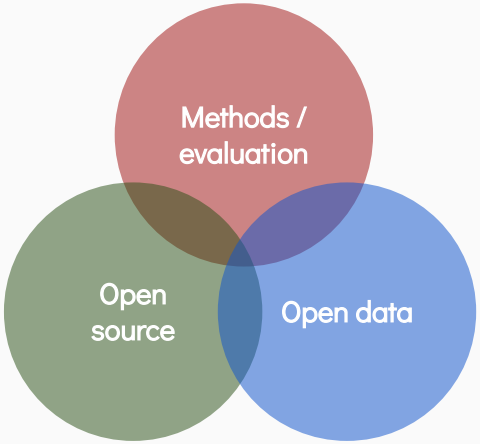
DS-GA 1004: Big Data

Todays’ plan

1. Quick overview of MIR and audio processing

2. Highlight reel of some projects…
    a. Automatic chord recognition
    b. Structure analysis
    c. Dataset construction

My research…



- Improve the **accessibility of audio data**
- Develop methods to **exploit structure** in audio and music
- **Infrastructure** to support music information retrieval (MIR) research

What is Music Information Retrieval?

- Technology to help machines and humans understand collections of music (and audio)

- Examples:
  - **Audio→score transcription**
  - **Recommender systems**
  - **Cover and plagiarism detection**
  - **Instrument / source recognition**
  - **Synthesis and generative models**

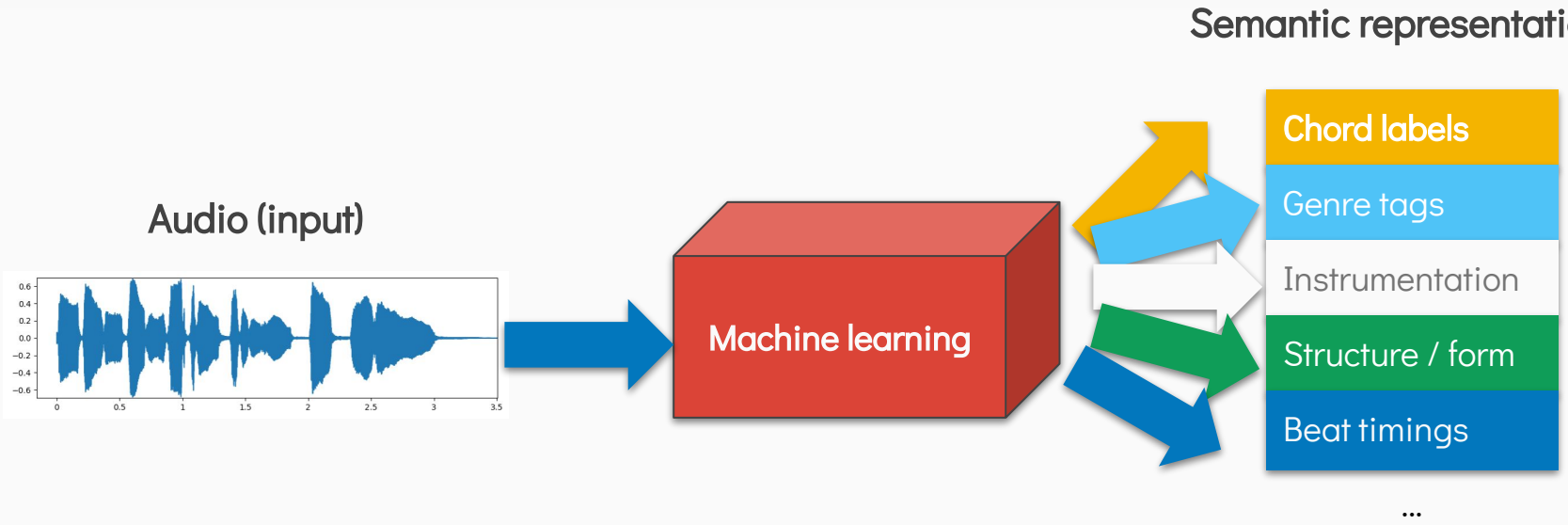
- Many connections to…
  - **Search engines** and **information retrieval**
  - **Signal processing** and **machine learning**
  - **Environmental sound** and **bioacoustics**

ISMIR

International Society for Music Information Retrieval (ISMIR)

https://ismir.net/

Audio-based MIR



ML strategy:

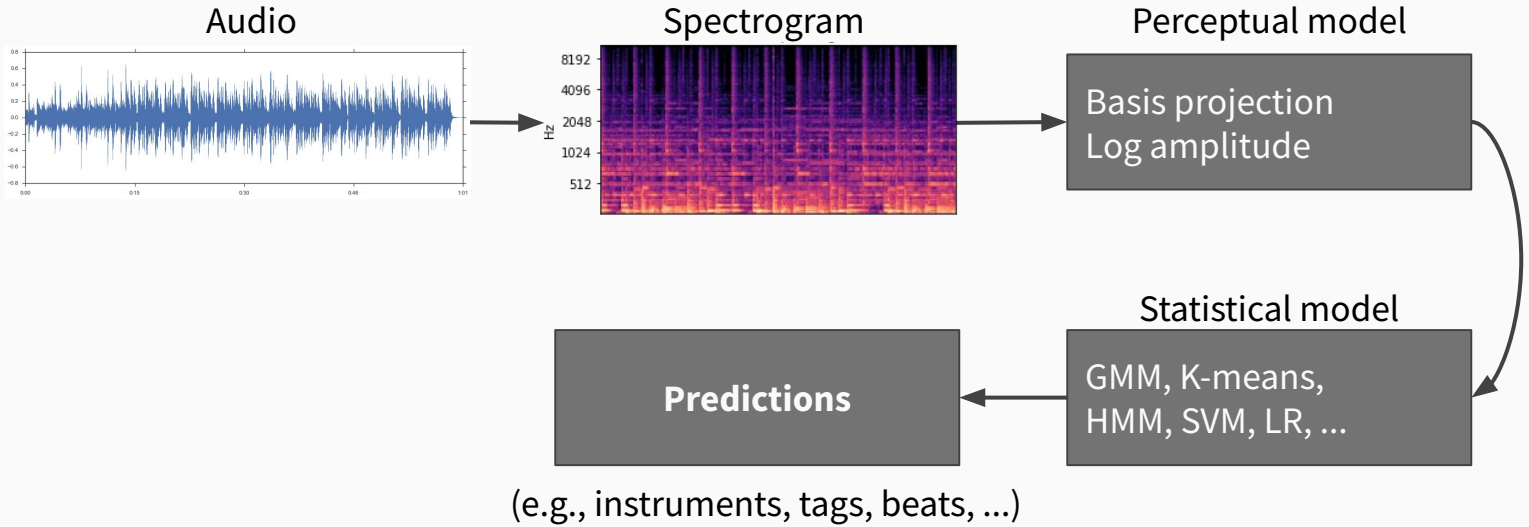
- Collect paired examples of **input** + **output**
- Teach the **machine** to map **input** → **output**

Quantity and **quality** of data is the **limiting factor**!

https://librosa.org/

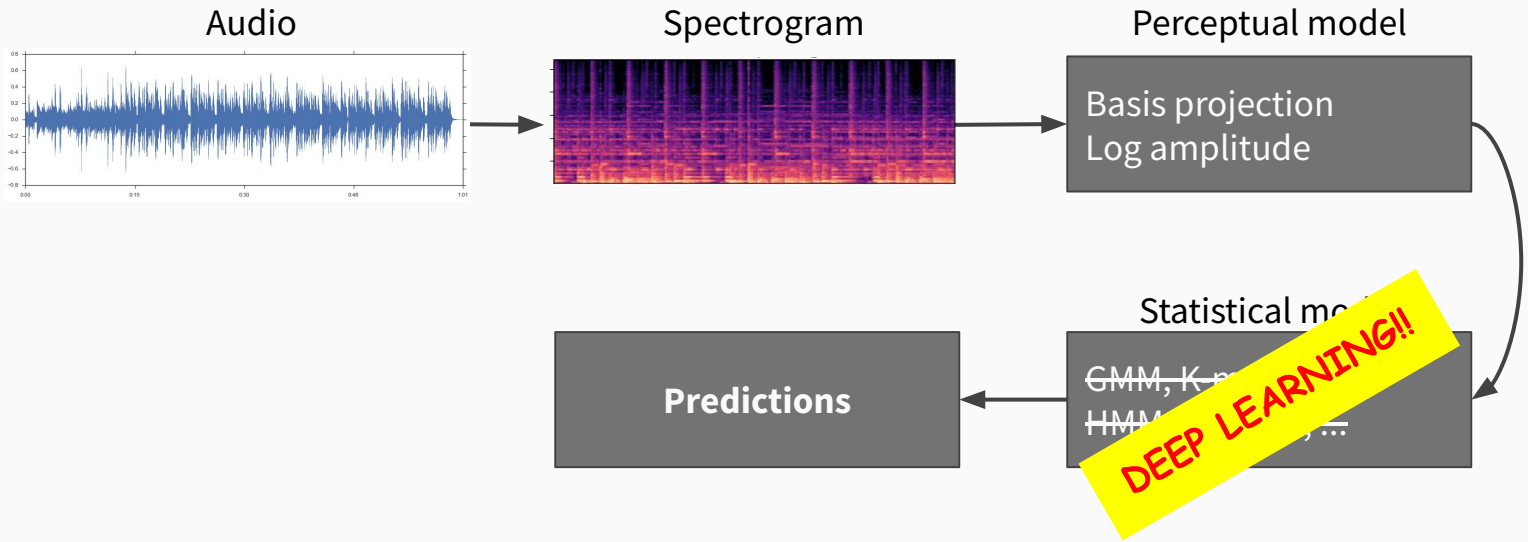
- Standardized DSP tools in Python for music and audio analysis
- In development since 2012
  - Latest version 0.9.1 (2022/02)
  - 80+ contributing authors
- 1.0 next year?

A standard pipeline, ca. 2002-2011



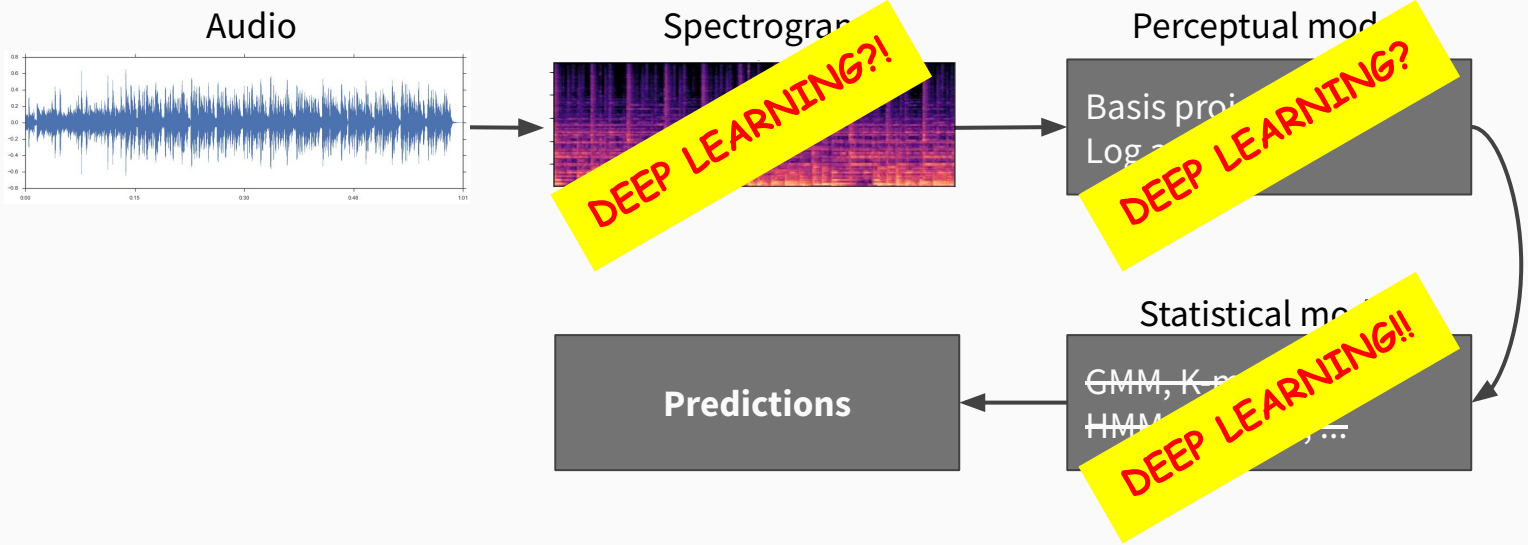
Going deep

- Stick a deep network on top of the existing feature stack
- Add time-convolution or recurrence to exploit temporal dependencies

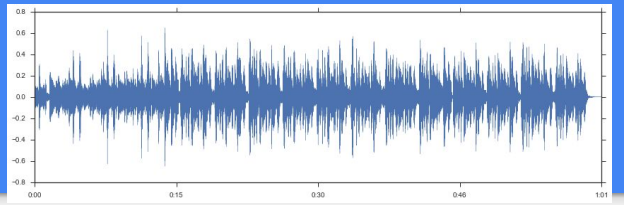


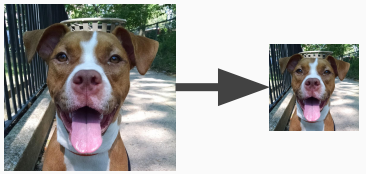
Going deeper...

- Stick a deep network directly on spectrogram
  - Or on the *waveform* itself
- Usually needs **much more** training data

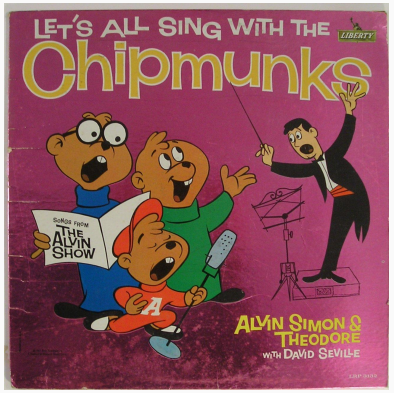


Audio vs. Images

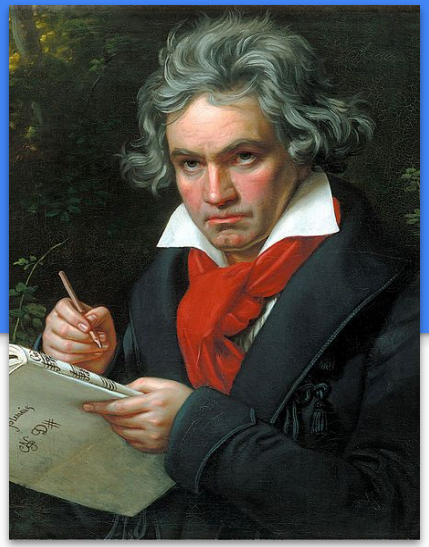




- 1D Input: (mono) amplitude over time
- Interactions can be both **local** and **long-range**
- ***Not scale-invariant!***
  - Unlike in images, downsampling does not preserve structures
  - Models should support variable-length input



Music vs. Speech



- Higher variability of observations
- Sources overlap in time and frequency
- Ambiguity and subjectivity are huge factors!
- ***Ground truth*** usually doesn't exist, so act accordingly

How does structure help?

We often know things about the **input**:

- Most music has **sustained tones** and/or **rhythmic structure**
- **Hierarchical organization**
  - Recording → Sections → Bars → Beats → ...
- We can leverage work in auditory perception and cognition
- Physics can help too!

... and the output space:

- Regularity in event timing (beats, onsets)
- Structure of chords (from music theory)
- Melody is piece-wise continuous
- Some instrument combinations are more likely than others

Fundamentals

No, not fundamental frequency.

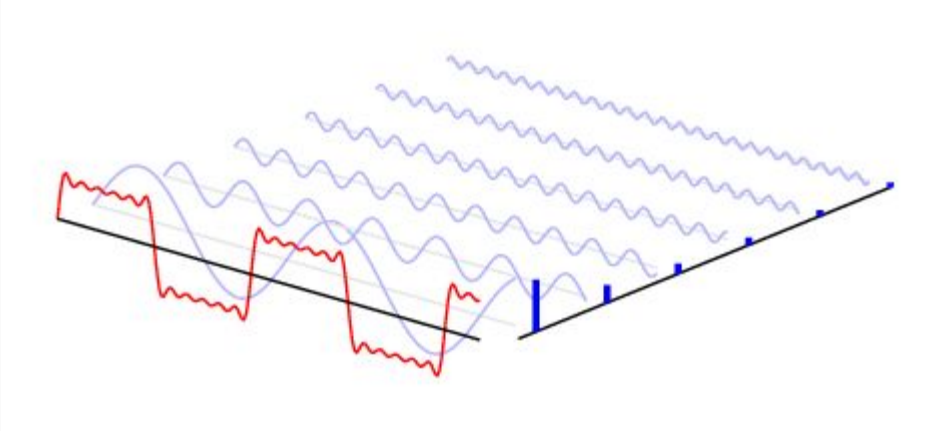
Signal processing and machine learning!

Okay, maybe fundamental frequencies too.

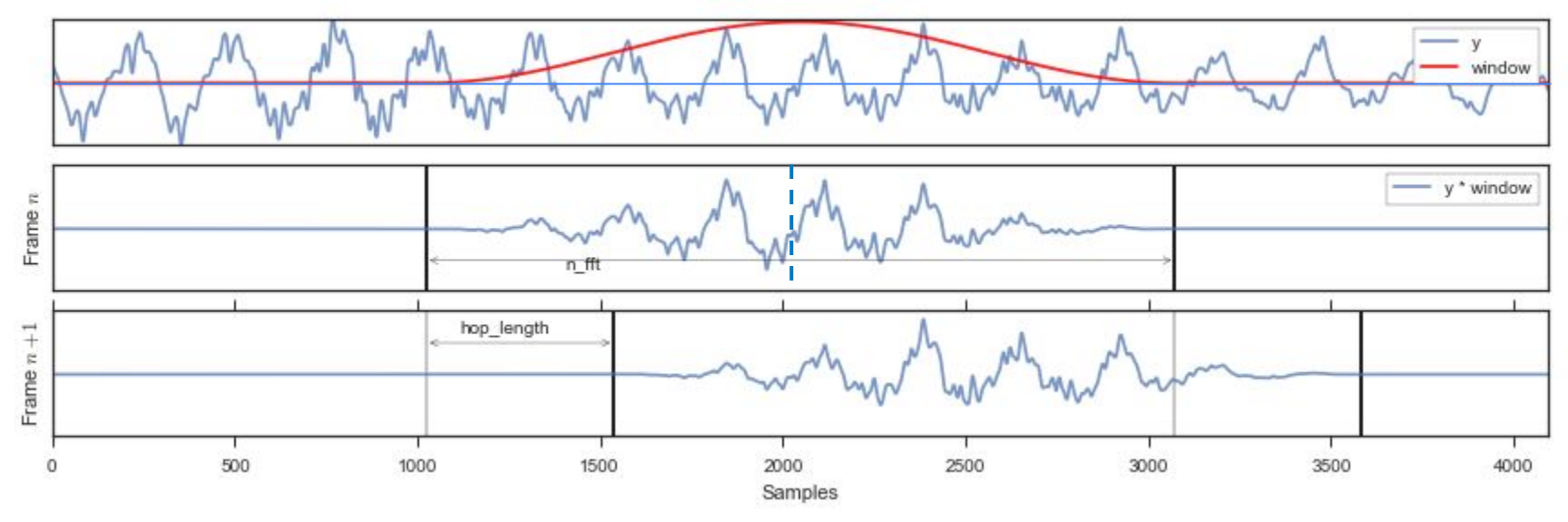
https://commons.wikimedia.org/wiki/File:Music_Class_at_St_Elizabeths_Orphanage_New_Orleans_1940.jpg

Spectrograms

- Break sound into different frequencies
  - **Fourier analysis**
- Measure how energy changes
  - **Short-time Fourier transform**
- Break audio into small, overlapping **frames**

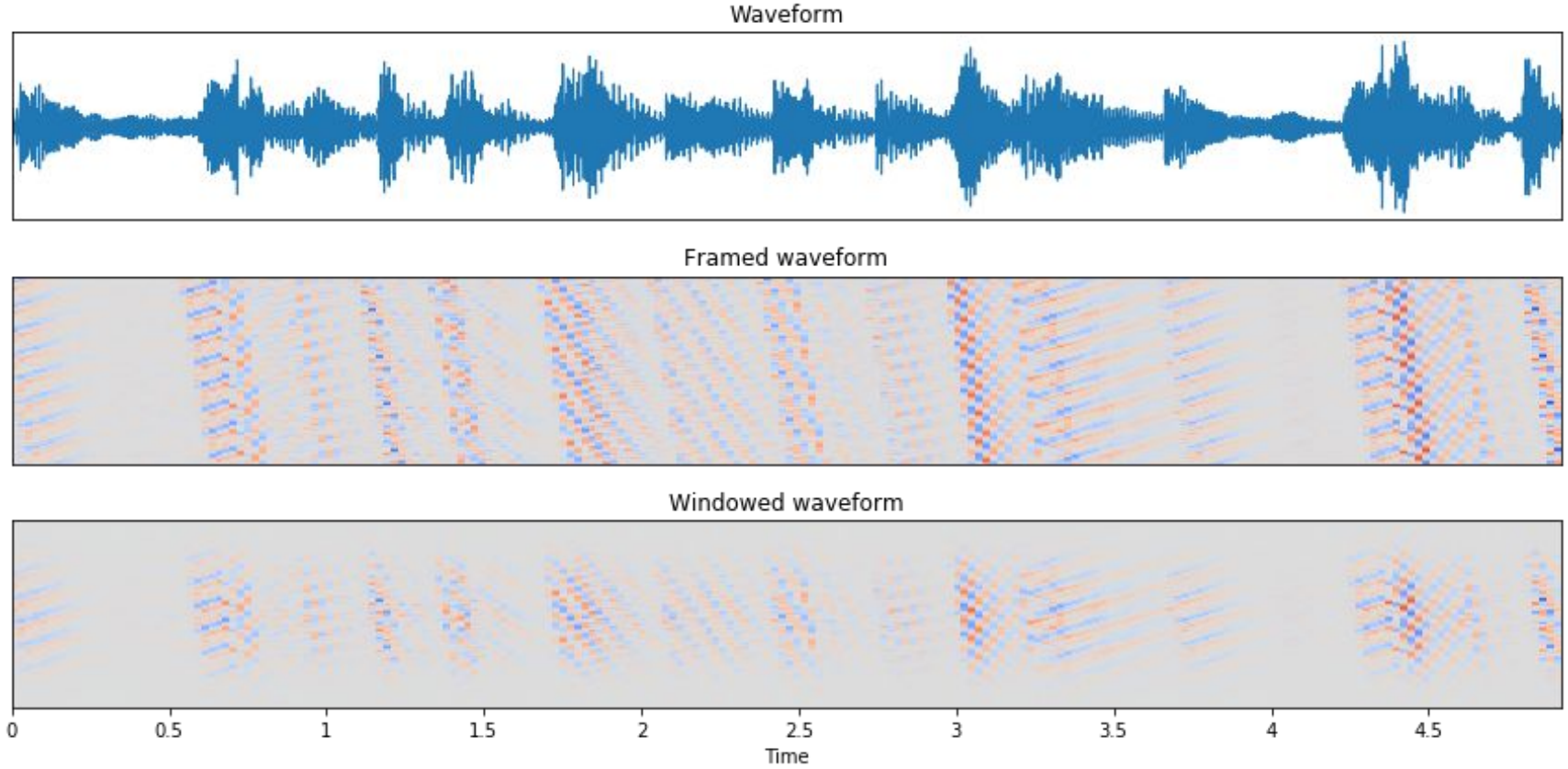


http://pgfplots.net/tikz/examples/fourier-transform/

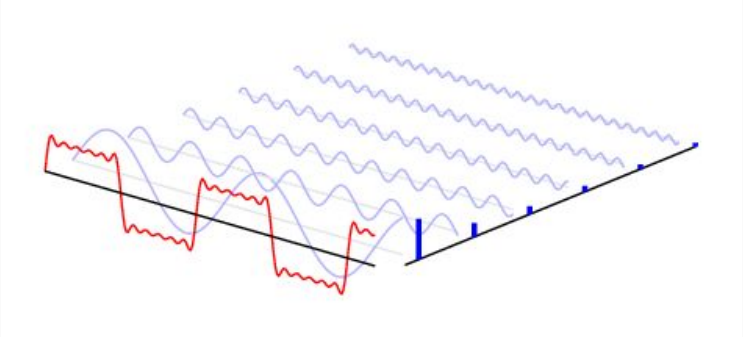


frame[n] → frame[n] * window

Neighboring frames differ by shifts when the signal is stationary



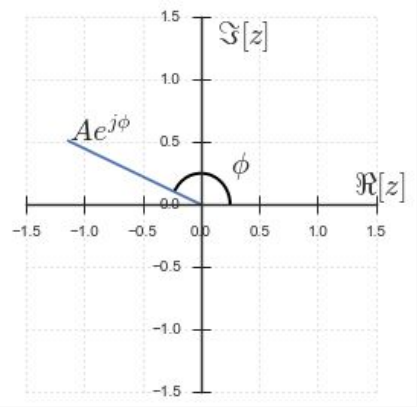
(Discrete) Fourier transform



- **Complex**-valued coefficient for each frequency

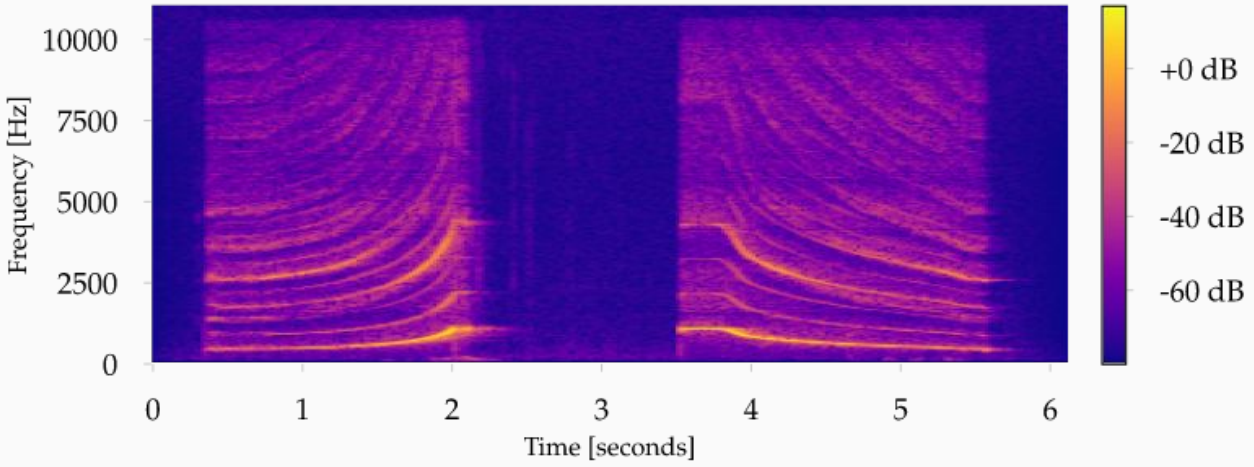
  $X(f) = Ae^{j\varphi}$

  - A = Magnitude (energy)
  - φ = Phase (shift)



- Usually we discard phase and analyze only the magnitude
- Interpretation: how much energy is there at frequency f?

Reading a spectrogram



Horizontal = Time Vertical = Frequency Color = amplitude

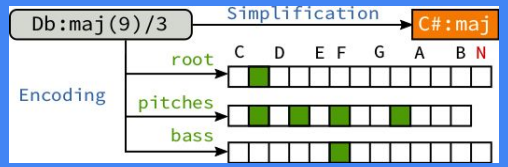
Each vertical slice is the DFT (magnitude) of one frame

Neighboring frames overlap

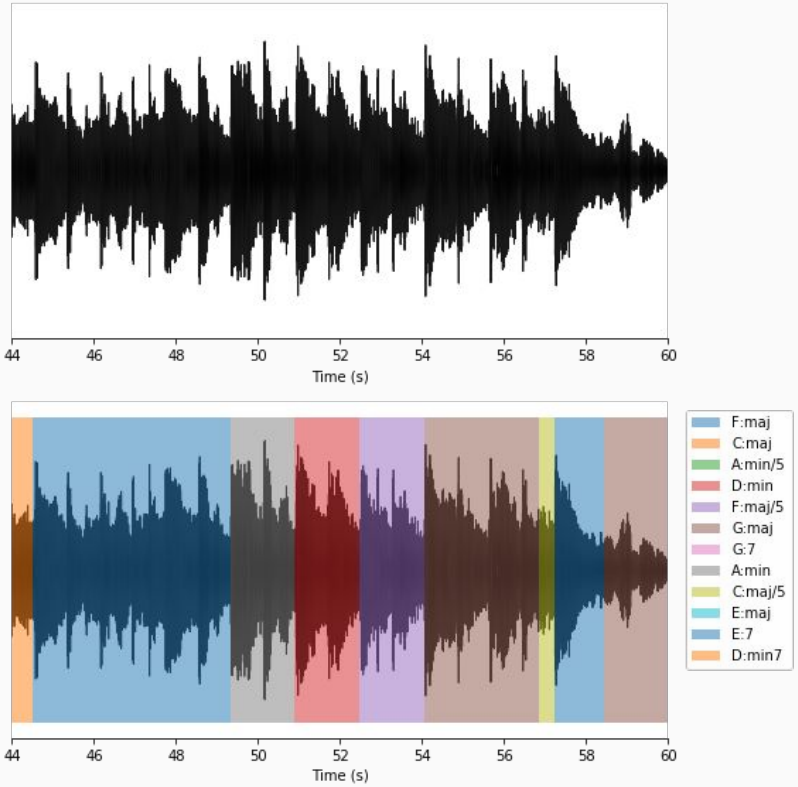
Data scale

- We typically acquire audio at
  f_s = 44100 [samples / second]
- Using float32, this is 172 [kB / sec] of audio
- Say a typical song is 3 minutes ⇒ 30 [MB / song]
- Short-time Fourier transform is [2× - 4×] redundant due to frame overlap
  ⇒ 100 [MB / song]
- A collection of ~1000 songs ⇒ **100 GB**
  - … and 1000 songs is tiny by machine learning standards
  - Complex tasks require much larger training sets than that!

Automatic chord recognition



Automatic chord estimation



- Transcribe audio as symbolic chord labels
  (*A:min*, *C:7*, etc)
- Why automate this?
  - Music education applications
  - Computational musicology
  - Creative applications
  - Facilitate downstream analysis

Small chord vocabularies

N	
C:maj	C:min
C#:maj	C#:min
D:maj	D:min
...	...
B:maj	B:min

- Standard approach: ***1-of-K*** classification
  - 25 classes: N + (12 × min) + (12 × maj)
  - Frames → chord labels → labeled segments
  - Hidden Markov Models, convolutional networks, etc.
- Classes are approximately balanced
- Implicit assumption:

All mistakes are equally bad

Large chord vocabularies

Chord quality	Frequency
maj	53%
min	14%
7	10%
...	
hdim7	0.17%
dim7	0.07%
minmaj7	0.04%

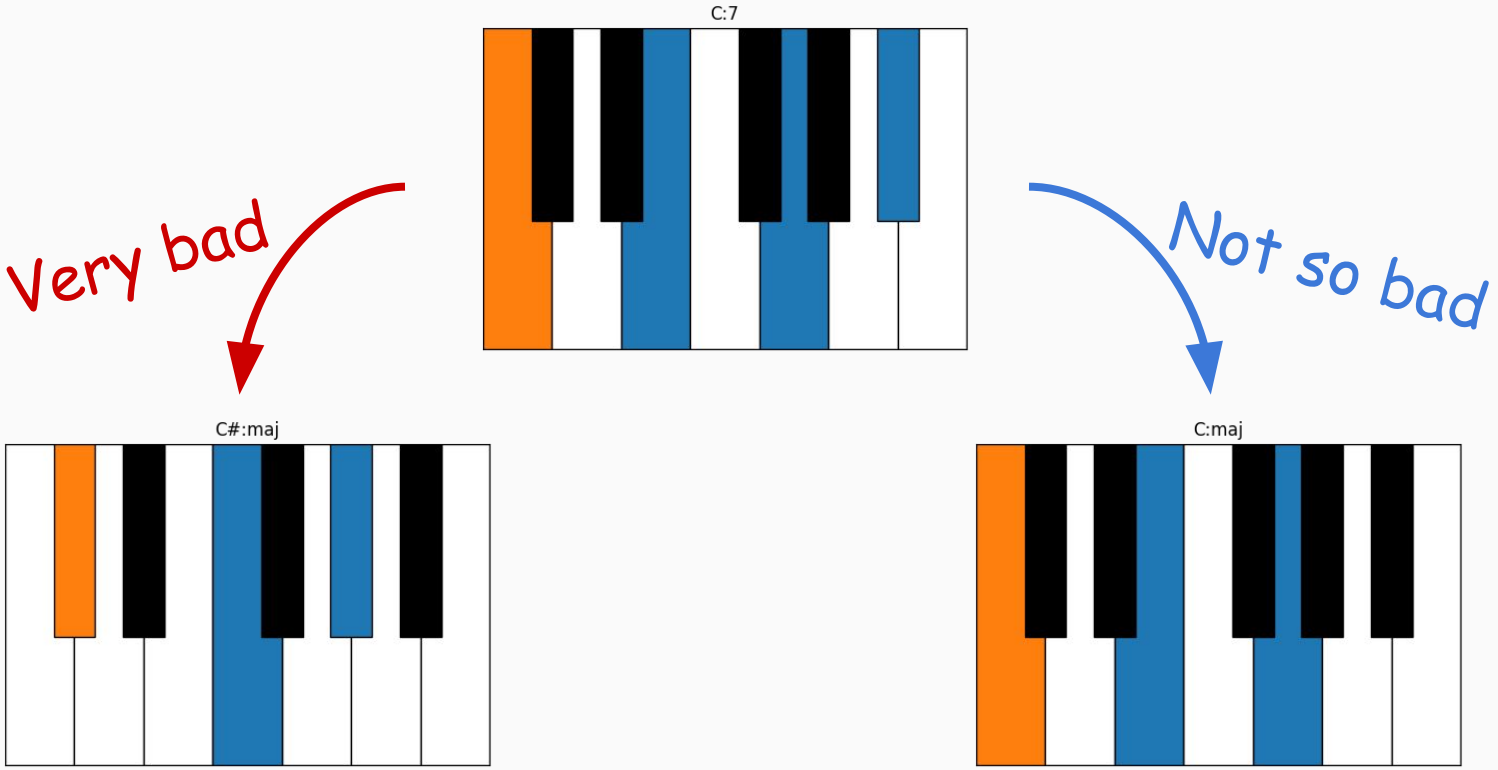
Distribution of the 1217 dataset

- Class distribution is **highly skewed**
- Classes are **not well-separated**
  - Overlapping pitch classes: *C:7* = *C:maj* + *m7*
  - Identical pitch classes: *C:sus4* vs. *F:sus2*
- Improving **rare classes** hurts **common classes**
- Annotations require expertise ⇒ expense!
- Not easy to get many observations for rare classes

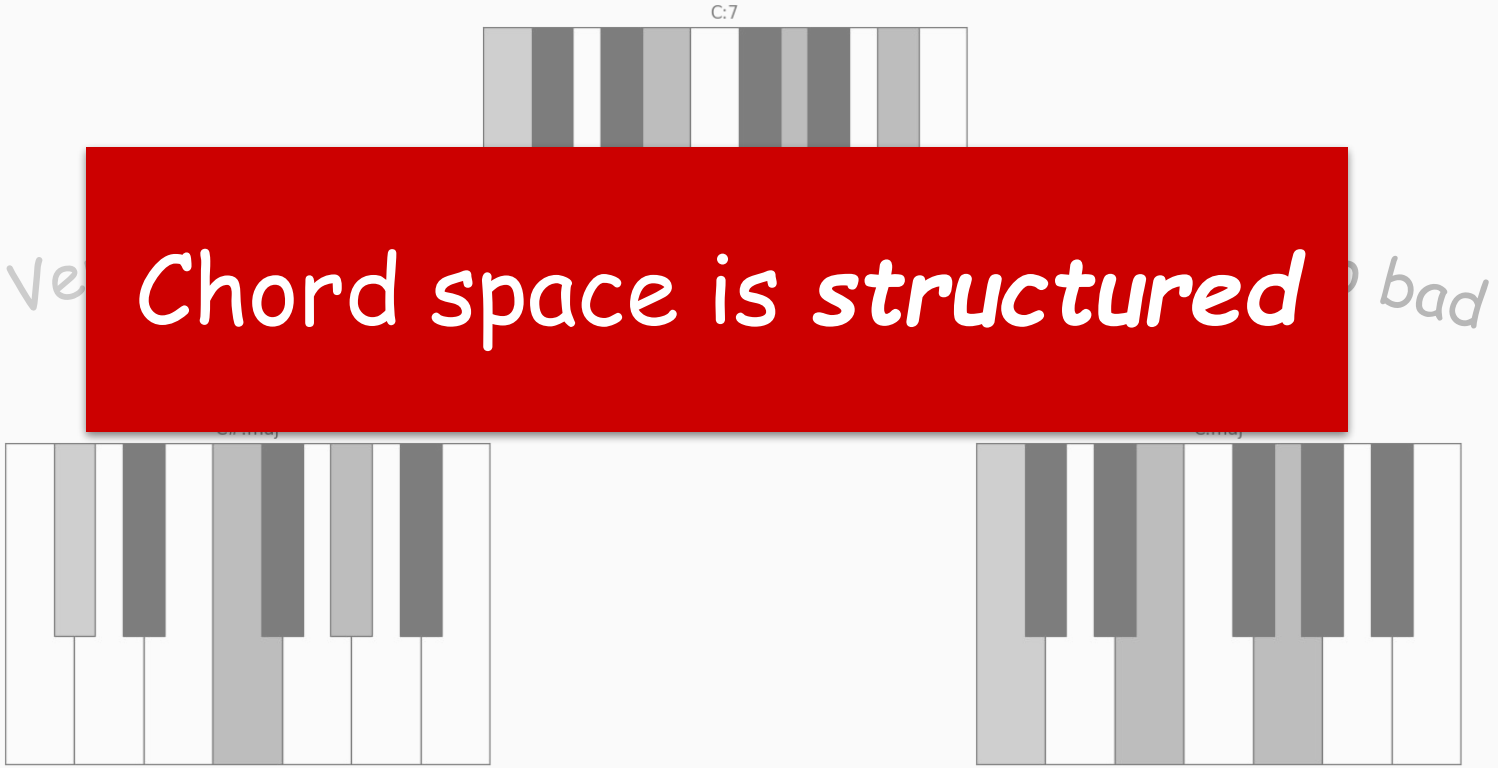
Standard classification setup

- Discard inversions **/3 → NUL**
- Discard added/suppressed notes: **(9) → NUL**
- Simplify above-octave extensions: **G:9 → G:7**
- Resolve enharmonic equivalence: **D ♭ → C#**
- Learn to predict the **simplified chord label**

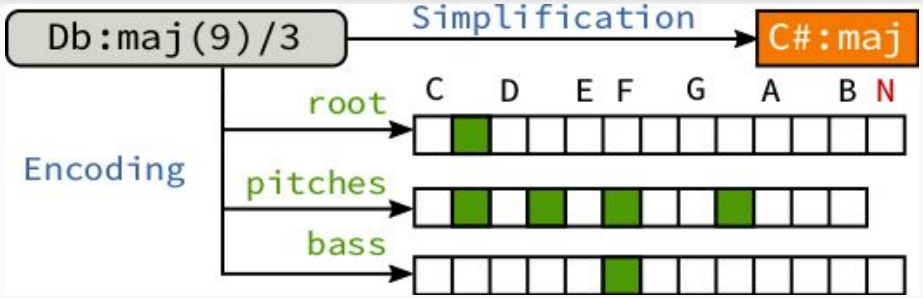
All mistakes are not equal



All mistakes are not equal



Structured training



- Represent **chord labels** as **binary encodings**

- Representation is **structured**:
  - **Similar chords** with **different labels** will have **similar encodings**
  - **Dissimilar chords** will have **dissimilar encodings**

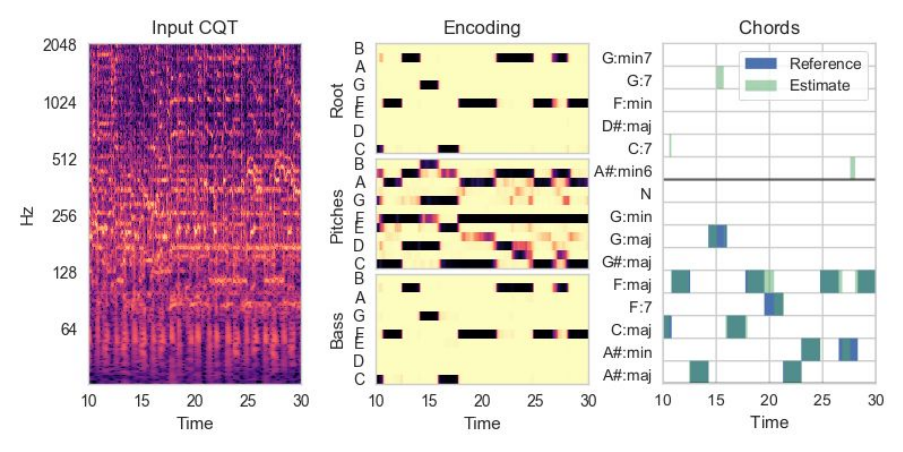
- Predict the **binary encoding**
- Learn to decode into **chord labels**
- Jointly optimize all four outputs: **root** + **pitches** + **bass** + **label**

12 × 14 + 2 = 170 classes

	min	maj	dim	aug	min6	maj6	min7	minmaj7	maj7	7	dim7	hdim7	sus2	sus4
C														
C#														
...														
B														

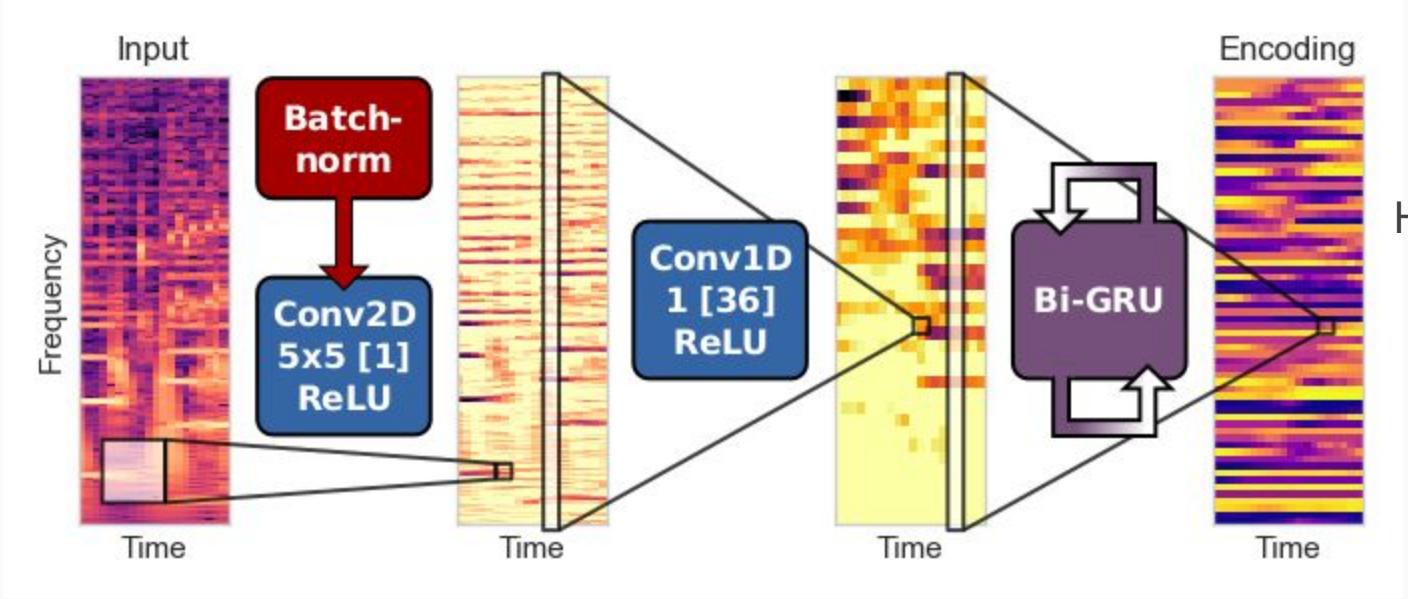
N	No chord (e.g., silence)
X	Out of gamut (e.g., power chords)

Model architectures



- **Input**: constant-Q spectral patches
  - 6 octaves, 3 bins per semitone
  - ~10 Hz frame rate
- **Outputs**: (per-frame)
  - **Root** **[multiclass, 13]**
  - **Pitches** **[multilabel, 12]**
  - **Bass** **[multiclass, 13]**
  - **Chords** **[multiclass, 170]**

Convolutional-recurrent network



Hidden state at frame t:

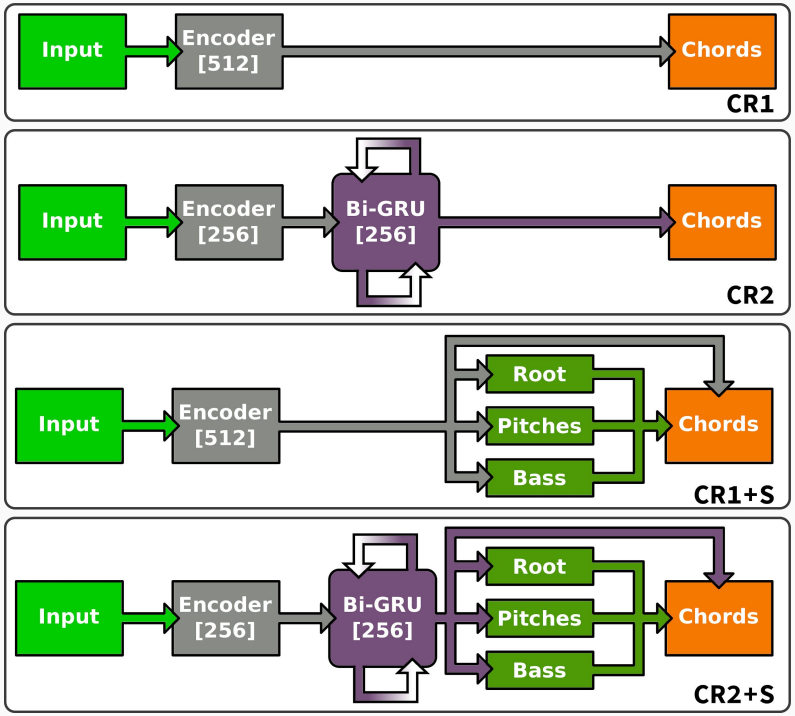
$h(t) \in [-1, +1]^D$

What about root bias?

- Quality and root should be independent
- But the data is **inherently biased**
- Solution: **data augmentation**!
  - **muda** [**M**., Humphrey, Bello, ISMIR 2015]
  - Jointly deform *audio* with *annotations*
- Each training track → ± 6 semitone shifts
  - All qualities are observed in all root positions
  - All roots, pitches, and bass values are observed

http://photos.jdhancock.com/photo/2012-09-28-001422-big-data.html

Decoder architectures



Chords = Logistic regression from GRU state

Chords = GRU + LR

Chords = LR from GRU state + root/pitch/bass

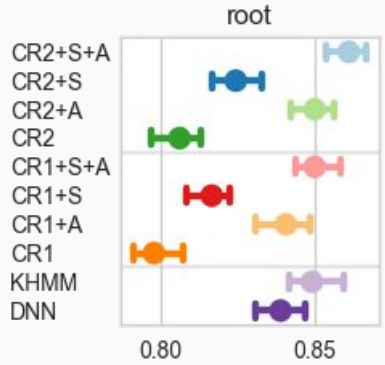
All of the above

Evaluation

- 8 configurations
  - ± data augmentation
  - ± structured training
  - 1 vs. 2 recurrent layers
- 1217 full-length recordings
  - 5-fold cross-validation
- Baseline models:
  - DNN [Humphrey & Bello, 2015]
  - KHMM [Cho, 2014]

Results

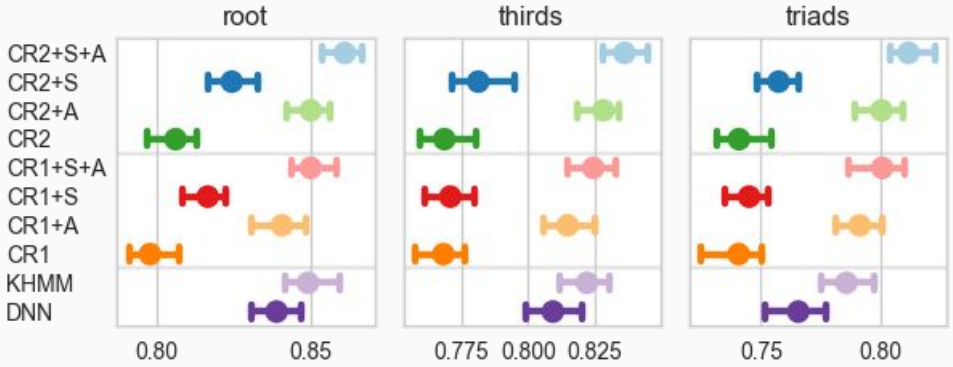
CR1: 1 recurrent layer
CR2: 2 recurrent layers
+A: data augmentation
+S: structure encoding



Data augmentation (+A) is necessary to match baselines.

Results

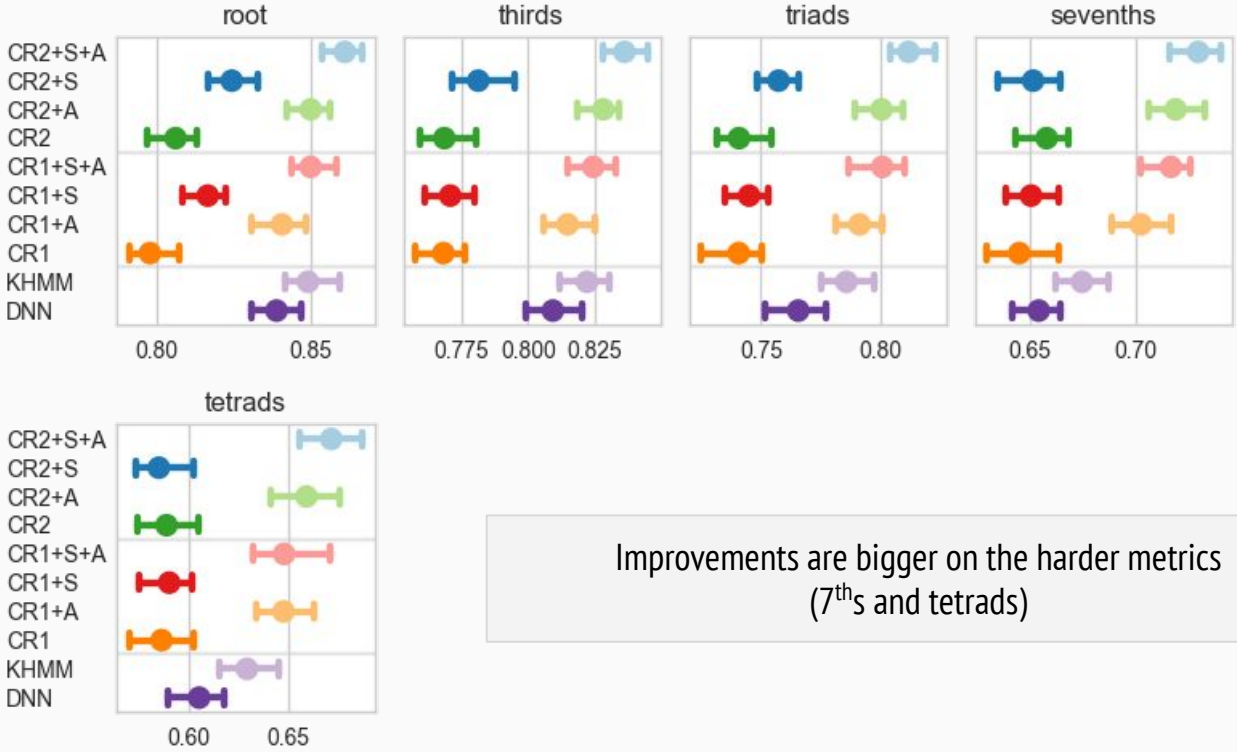
CR1: 1 recurrent layer
CR2: 2 recurrent layers
+A: data augmentation
+S: structure encoding



Structured training (+S) and deeper models improve over baselines.

Results

CR1: 1 recurrent layer
CR2: 2 recurrent layers
+A: data augmentation
+S: structure encoding



Improvements are bigger on the harder metrics
(7^{th}s and tetrads)

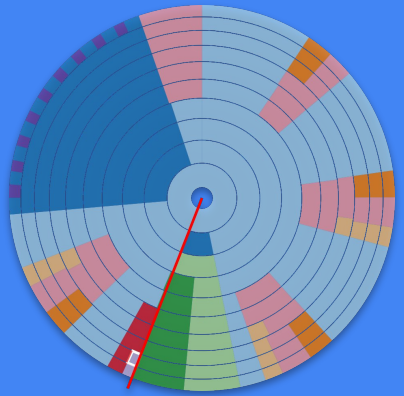
Making it work

- Pre-processing training data
  - ~5GB of compressed audio (mp3), 50GB raw
  - Data augmentation (12× increase) 60GB / 600GB
  - Audio → CQT → HDF5 storage
  - Parallelized by **joblib** (**Dask** would also work here)
- Training data does not fit entirely in memory
  - Out-of-core sampling/streaming via **pescador** (**pytorch** Dataloader now does some of this)
  - Work on short patches (~10sec) at a time
  - Train on **GPU**
- Fully replicable workflow
  - Numbered scripts:
    01-augment.py
    02-train.py
    03-evaluate.py
    ...

Take-aways

- Sometimes more data isn’t the answer
- Some domain knowledge can reduce the amount of (labeled) data you need
- Data augmentation can complement domain knowledge

Song structure analysis

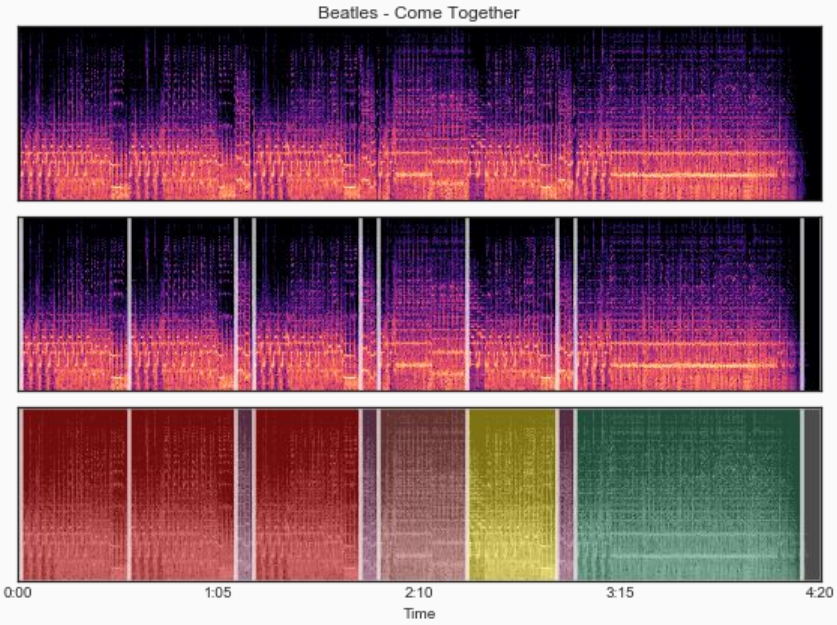


The standard approach: classification

1. Compute time-series features

2. Break a song into pieces
   (detect **segment boundaries**)

3. Re-connect the pieces
   (detect **similar/dissimilar segments**)



The standard approach: classification

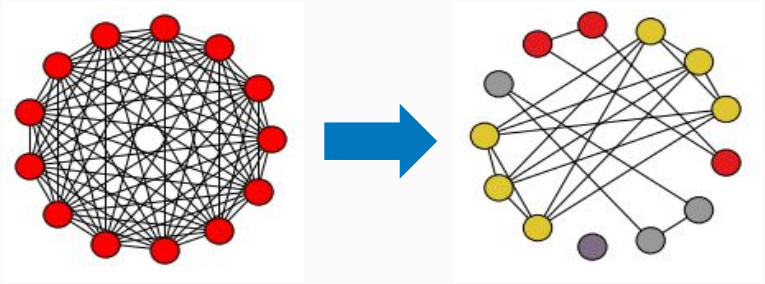
1. Compute time-series features
2. Break a song into pieces
   (detect **segment boundaries**)
3. Re-connect the pieces
   (detect **similar/dissimilar segments**)

Problems with this approach:

1. Error propagation
2. Repetition is a strong cue for segmentation
3. Does not generalize to multi-level analysis

⇒ **Structure analysis is not a classification problem**

Musical structure encodes graph ***topology*** over ***time***

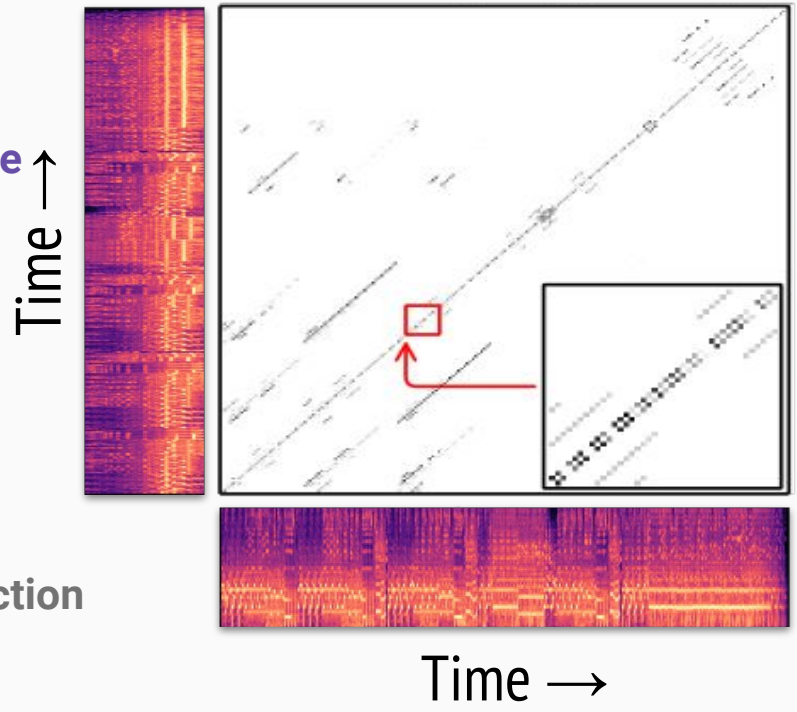


- **Design a graph** that exposes structures of interest
- **Partition the graph** to recover structure
- Represent structure at **multiple levels** by iterative partitioning

The infinite jukebox graph

[M. & Ellis, ISMIR 2014]

1. Link each instant to **neighbors in time**
2. Link each instant to **neighbors in feature space**
3. Weight edges by feature similarity
4. Balance time- and feature-links



Intuition:

A random walk should not easily escape a section

Kind of similar to PageRank, but we're interested in **subgraphs**, not individual **vertices**. ⇒ Spectral clustering!

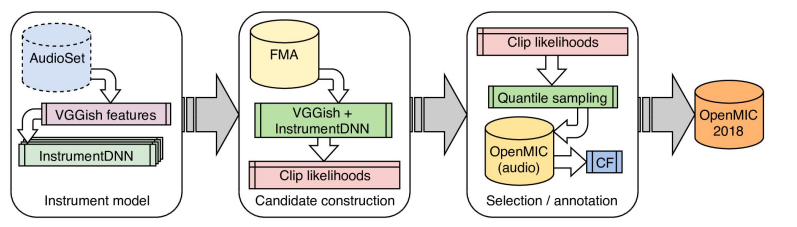
Demo

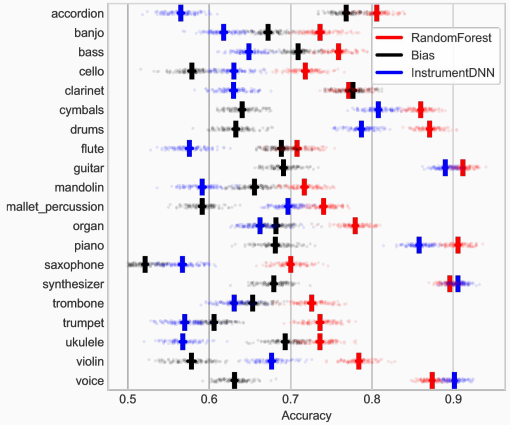
Take-aways…

- Core operation is graph construction / decomposition
  - Accelerated by fast nearest neighbor
- A good visualization can make it easier to navigate large collections
- Some algorithmic know-how will go a long way

Instrument dataset construction

OpenMIC 2018



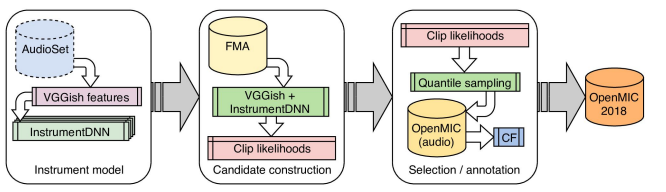


- 20k polyphonic, 10s **audio** clips from the Free Music Archive
- 20 instrument classes
  - > 500 positive examples per class
  - > 1500 total examples per class
- Crowd-sourced annotations
- Over 40k consensus labels

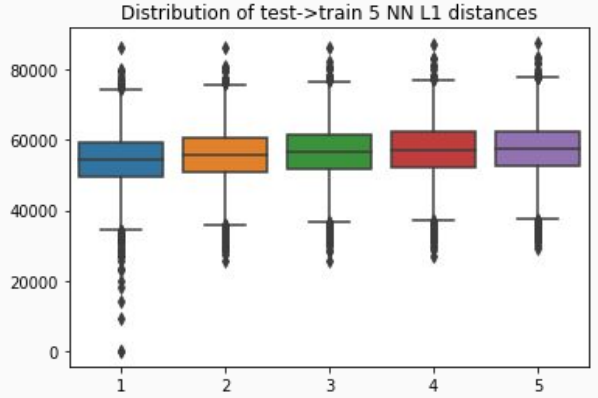
http://bit.ly/openmic-2018

[Humphrey, Durand, **M.**, ISMIR 2018]

What went into OpenMIC-2018?



- Training a preliminary instrument detector on **AudioSet**
  - https://research.google.com/audioset/
  - ~206,000 excerpts from youtube (just music instruments)
  - Huge parallel grid search over model architectures / hyper-parameters
- Estimations on the **Free Music Archive** (FMA) dataset
  - ~108,000 songs, 7 million excerpts
  - Per-class likelihood quantile sampling (10th -, 90th +)
- **Crowd-sourced** annotations
  - ~230,000 ratings, 2500 human annotators
- **Train-test split** construction, **de-duplication**
  - Approximate multi-label stratification
  - Approximate nearest neighbors strikes again!



Closing remarks…

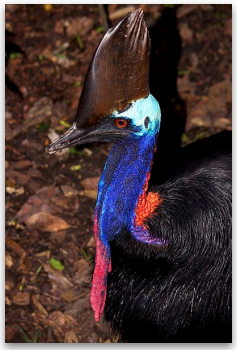
What should you know from this class?

1. You have a lot of tools at your disposal.
   Know their strengths and weaknesses.

2. Communication is often the bottleneck.
   Make your data as small as you can as quickly as you can.

3. Tools change rapidly. Documentation is often bad.
   Be flexible.

4. Always think critically about where your data comes from, and
   how your systems impact people.



Thanks!

- You've been a great class!
- This has been a hard year.

  And I'm impressed with how well you've all handled it.
- Enjoy the summer, hang in there, and I hope to see you in the Fall!

Thanks to the course staff:
Artie, Bo, Jack, Safwan, Sanae, Saumyaa, Xintong

And the NYU-HPC crew:
Shenglong, Wensheng, Eric, John