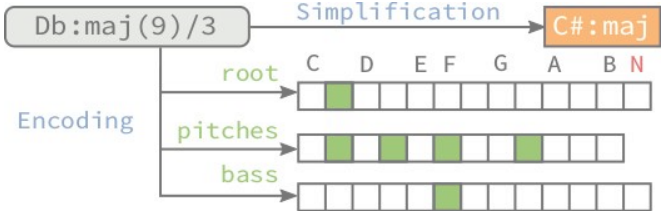
Structured training for large-vocabulary chord recognition

Brian McFee* & Juan Pablo Bello

NYU | Center for Data Science

MARL



Small chord vocabularies

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

- Typically a supervised learning problem
  - Frames ⟶ chord labels
- ***1-of-K*** classification models are common
  - 25 classes: N + (12 × min) + (12 × maj)
  - Hidden Markov Models, Deep convolutional networks, etc.
  - Optimize accuracy, log-likelihood, etc.

Small chord vocabularies

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

- Typically a supervised learning problem
  - Frames ⟶ chord labels
- ***1-of-K*** classification models are common
  - 25 classes: N + (12 × min) + (12 × maj)
  - Hidden Markov Models, Deep convolutional networks, etc.
  - Optimize accuracy, log-likelihood, etc.
- Implicit training assumption:

  All mistakes are equally bad

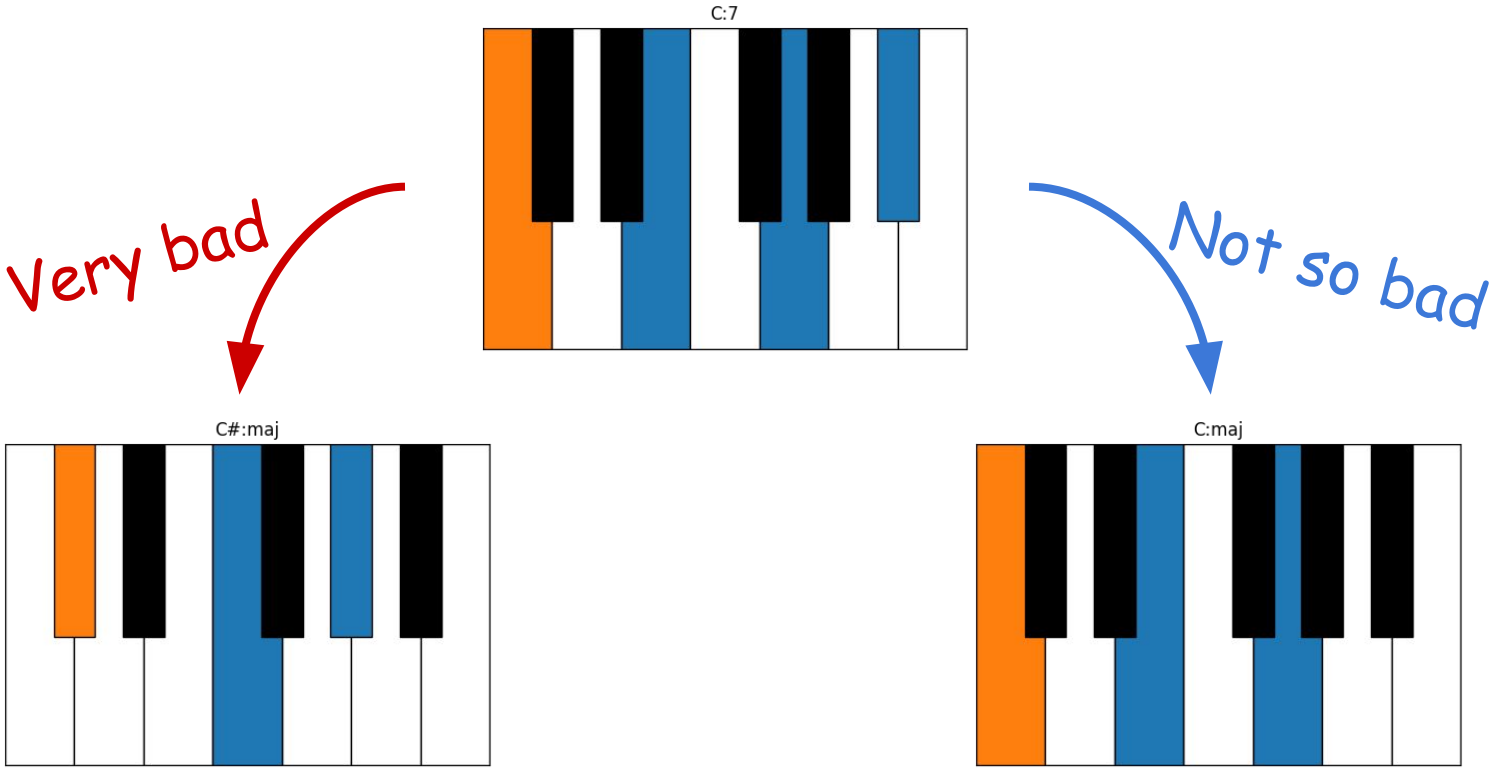
Large chord vocabularies

Chord quality	Frequency
maj	52.53%
min	13.63%
7	10.05%
...	
hdim7	0.17%
dim7	0.07%
minmaj7	0.04%

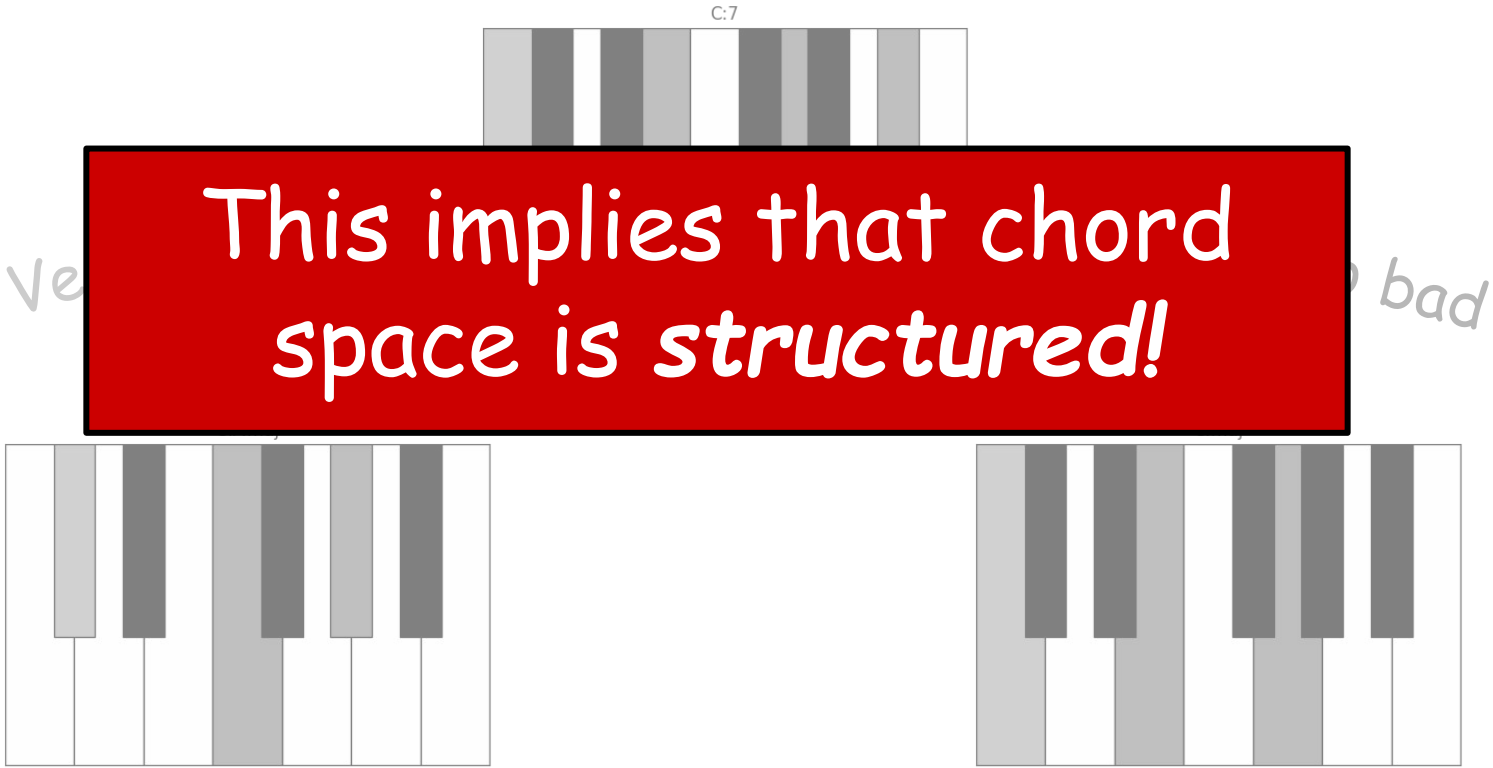
Distribution of the 1217 dataset

- Classes are **not well-separated**
  - C:7 = C:maj + m7
  - C:sus4 vs. F:sus2
- Class distribution is **non-uniform**
- **Rare classes are hard to model**

Some mistakes are better than others



Some mistakes are better than others



Our contributions

- Deep learning architecture to **exploit structure** of chord symbols
- Improve accuracy in rare classes
  Preserve accuracy in common classes
- Bonus: package is online for you to use!

Chord simplification

- All classification models need a finite, canonical label set

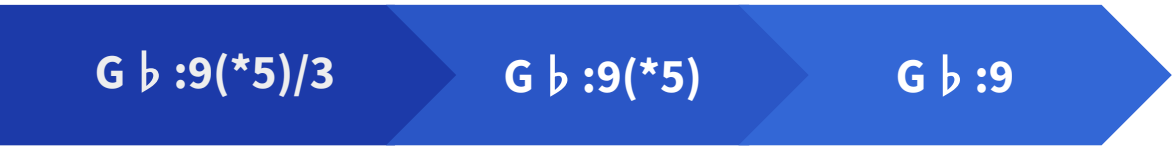
Chord simplification

- All classification models need a finite, canonical label set
- Vocabulary simplification process:
  a. Ignore inversions

Chord simplification

- All classification models need a finite, canonical label set

- Vocabulary simplification process:
  a. Ignore inversions
  b. Ignore added and suppressed notes



Chord simplification

- All classification models need a finite, canonical label set

- Vocabulary simplification process:
  a. Ignore inversions
  b. Ignore added and suppressed notes
  c. Template-match to nearest quality

Chord simplification

- All classification models need a finite, canonical label set

- Vocabulary simplification process:
  a. Ignore inversions
  b. Ignore added and suppressed notes
  c. Template-match to nearest quality
  d. Resolve enharmonic equivalences

Chord simplification

- All classification models need a finite, canonical label set

- Vocabulary simplification process:
  a. Ignore inversions
  b. Ignore added and suppressed notes
  c. Template-match to nearest quality
  d. Resolve enharmonic equivalences

Simplification is lossy!
(but all chord models do it)

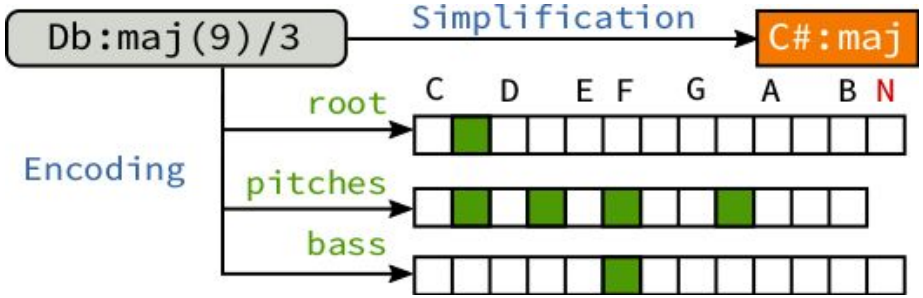
$14 \times 12 + 2 = 170$ classes

14 qualities

	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)

Structural encoding

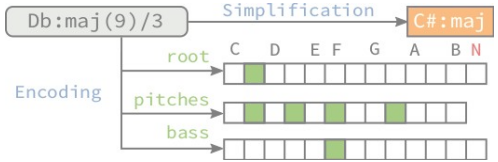


- Represent **chord labels** as **binary encodings**
- Encoding is **lossless*** and **structured**:
  - **Similar chords** with **different labels** will have **similar encodings**
  - **Dissimilar chords** will have **dissimilar encodings**
- Learning problem:
  - Predict the **encoding** from audio
  - Learn to decode into **chord labels**

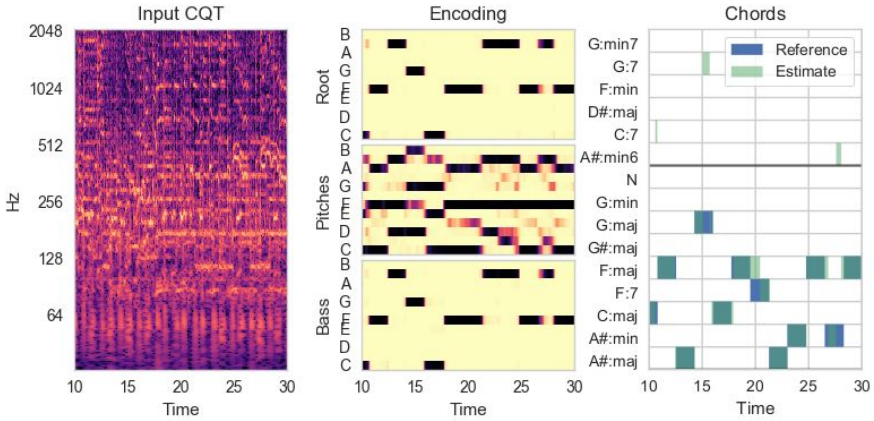
* up to octave-folding

The big idea

- Jointly estimate **structured encoding** AND **chord labels**
- Full objective = **root loss** + **pitch loss** + **bass loss** + **decoder loss**

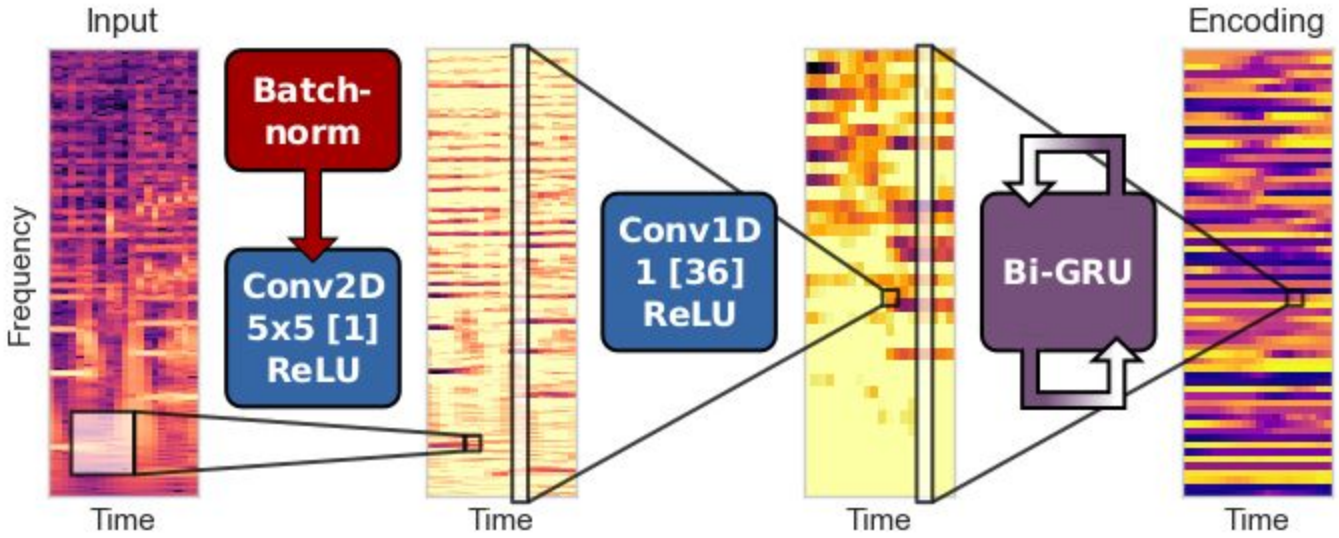


Model architectures



- Input: constant-Q spectral patches
- Per-frame outputs:
  - **Root** **[multiclass, 13]**
  - **Pitches** **[multilabel, 12]**
  - **Bass** **[multiclass, 13]**
  - **Chords** **[multiclass, 170]**
- Convolutional-recurrent architecture (encoder-decoder)
- End-to-end training

Encoder architecture



Hidden state at frame *t*:

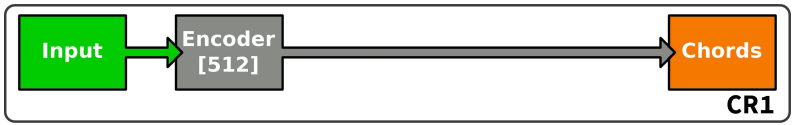
$h(t) \in [-1, +1]^D$

Suppress transients

Encode frequencies

Contextual smoothing

Decoder architectures

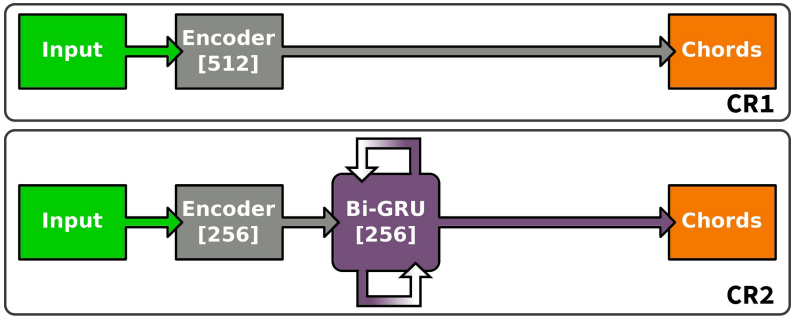


Chords = Logistic regression from encoder state

Frames are independently decoded:

$$\boldsymbol{y(t)} = \text{softmax}(W\, h(t) + \beta)$$

Decoder architectures



Chords = Logistic regression from encoder state

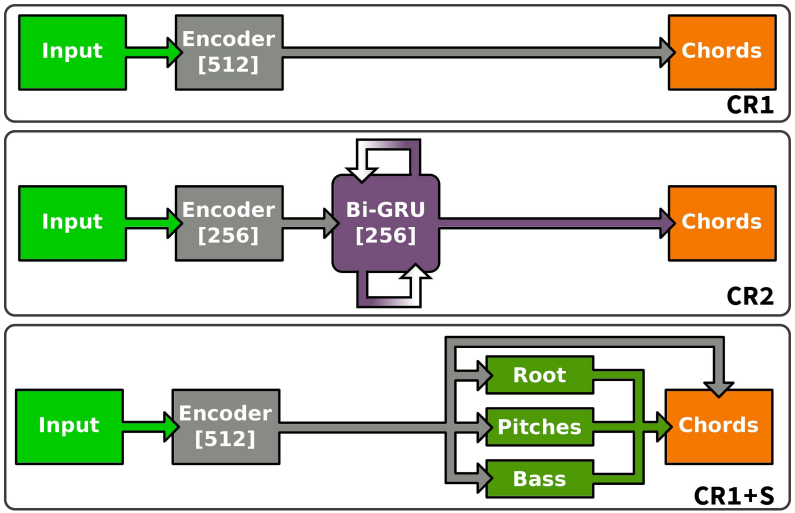
Decoding = GRU + LR

Frames are recurrently decoded:

$$h_2(t) = \text{Bi-GRU}[h](t)$$

$$y(t) = \text{softmax}(W\, h_2(t) + \beta)$$

Decoder architectures



Chords = Logistic regression from encoder state

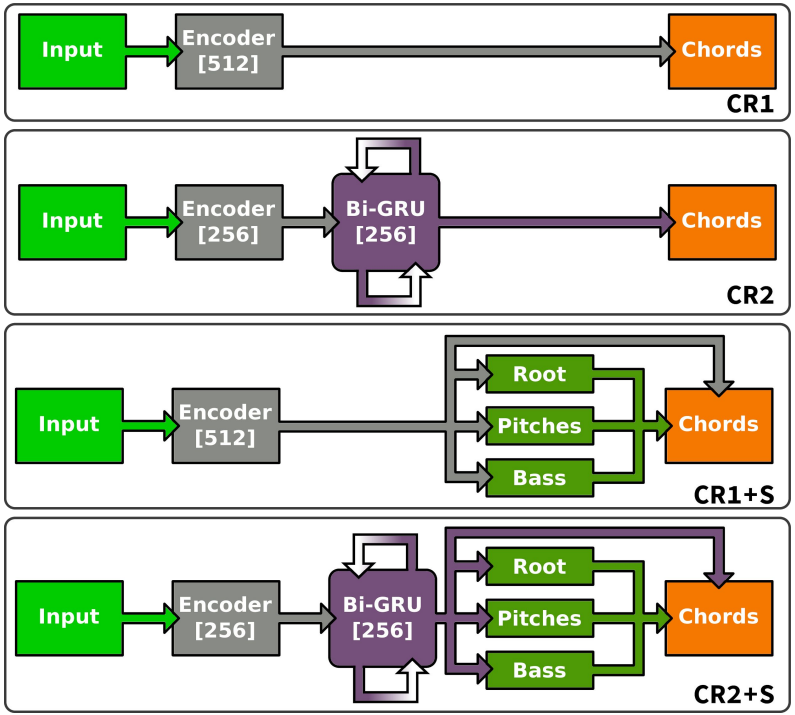
Decoding = GRU + LR

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

Frames are independently decoded with structure:

$$\mathbf{y}(t) = \text{softmax}(W_r\,\mathbf{r}(t) + W_p\,\mathbf{p}(t) + W_b\,\mathbf{b}(t) + W_h\,h(t) + \beta)$$

Decoder architectures



Chords = Logistic regression from encoder state

Decoding = GRU + LR

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

All of the above

What about root bias?

- Quality and root should be independent
- But the data is **inherently biased**
- Solution: **data augmentation**!
  - **muda** [McFee, Humphrey, Bello 2015]
  - Pitch-shift the audio and annotations simultaneously
- Each training track ⟶ ± 6 semitone shifts
  - All qualities are observed in all root positions
  - All roots, pitches, and bass values are observed

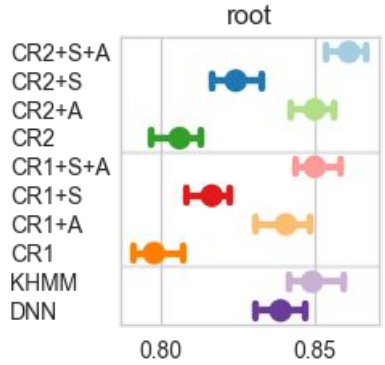
http://photos.jdhancock.com/photo/2012-09-28-001422-big-data.html

Evaluation

- 8 configurations
  - ± data augmentation
  - ± structured training
  - 1 vs. 2 recurrent layers
- 1217 recordings
  (Billboard + Isophonics + MARL corpus)
  - 5-fold cross-validation
- Baseline models:
  - DNN [Humphrey & Bello, 2015]
  - KHMM [Cho, 2014]

Results

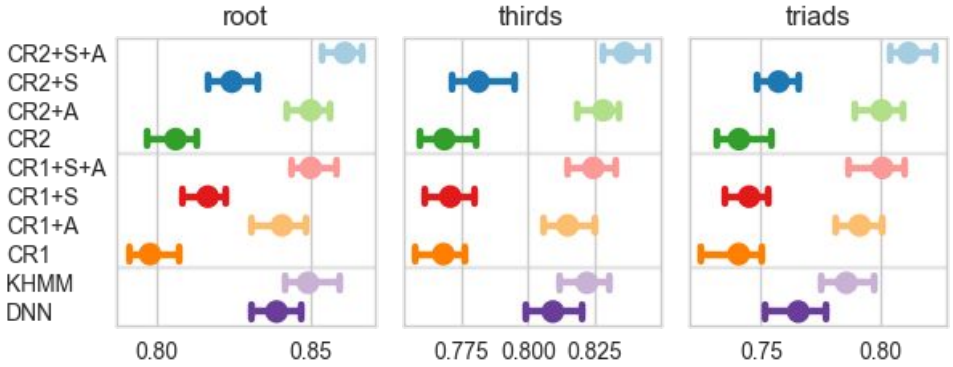
CR1: 1 recurrent layer
CR2: 2 recurrent layers
+A: data augmentation
+S: structure encoding



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

Results

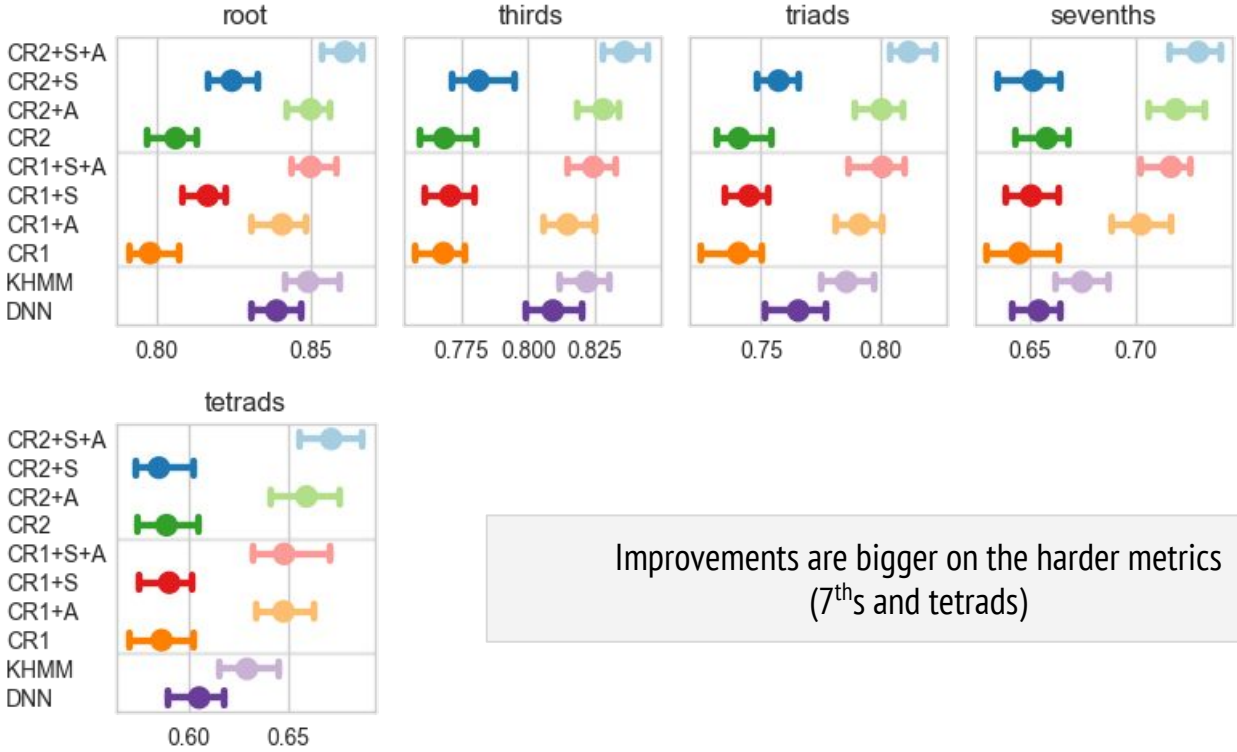
CR1: 1 recurrent layer
CR2: 2 recurrent layers
+A: data augmentation
+S: structure encoding



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

Results

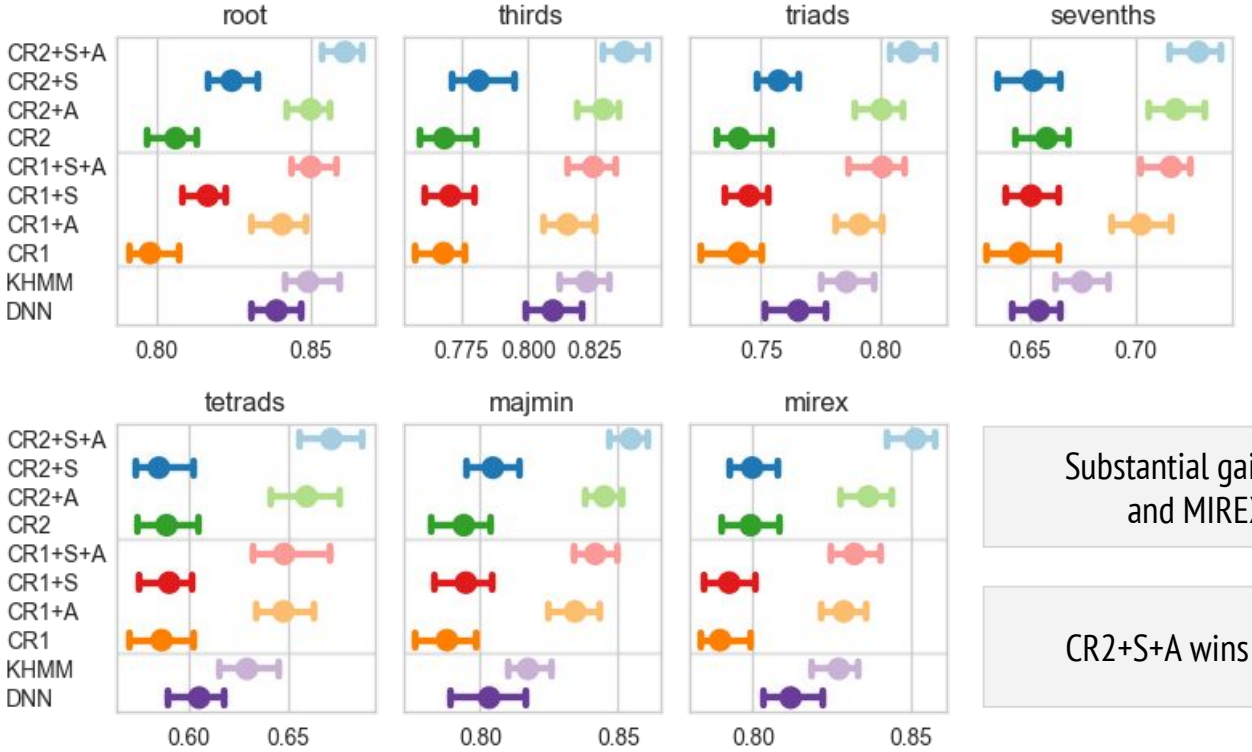
CR1: 1 recurrent layer
CR2: 2 recurrent layers
+A: data augmentation
+S: structure encoding



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

Results

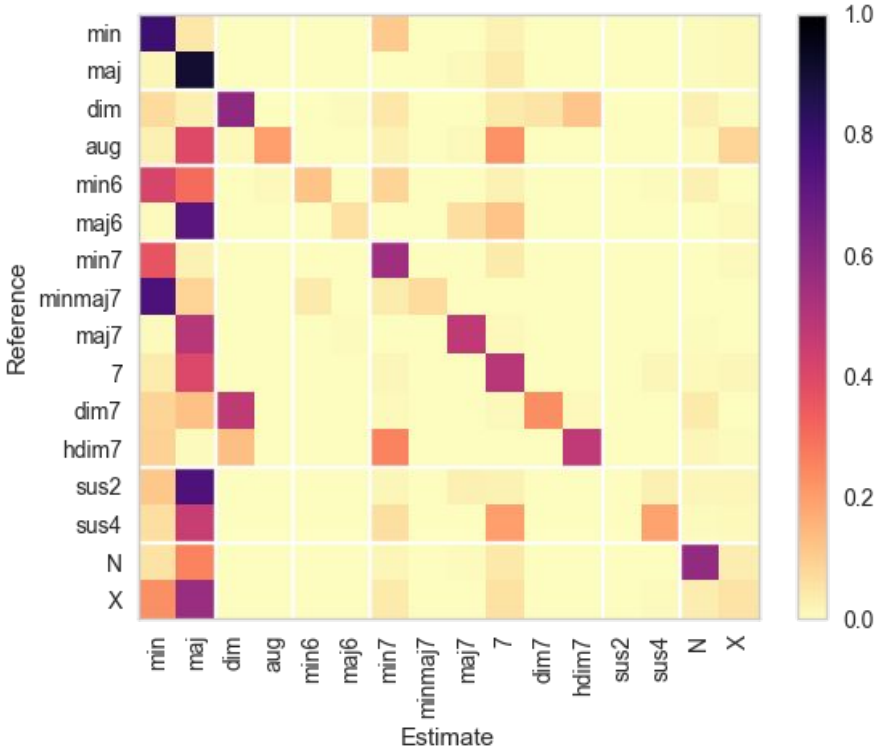
CR1: 1 recurrent layer
CR2: 2 recurrent layers
+A: data augmentation
+S: structure encoding



Substantial gains in maj/min and MIREX metrics

CR2+S+A wins on all metrics

Error analysis: quality confusions



Errors tend toward simplification

Reflects maj/min bias in training data

Simplified vocab. accuracy: 63.6%

Summary

- Structured training helps
- Deeper is better
- Data augmentation is critical
  - **pip install muda**
- Rare classes are still hard
  - We probably need new data

Thanks!

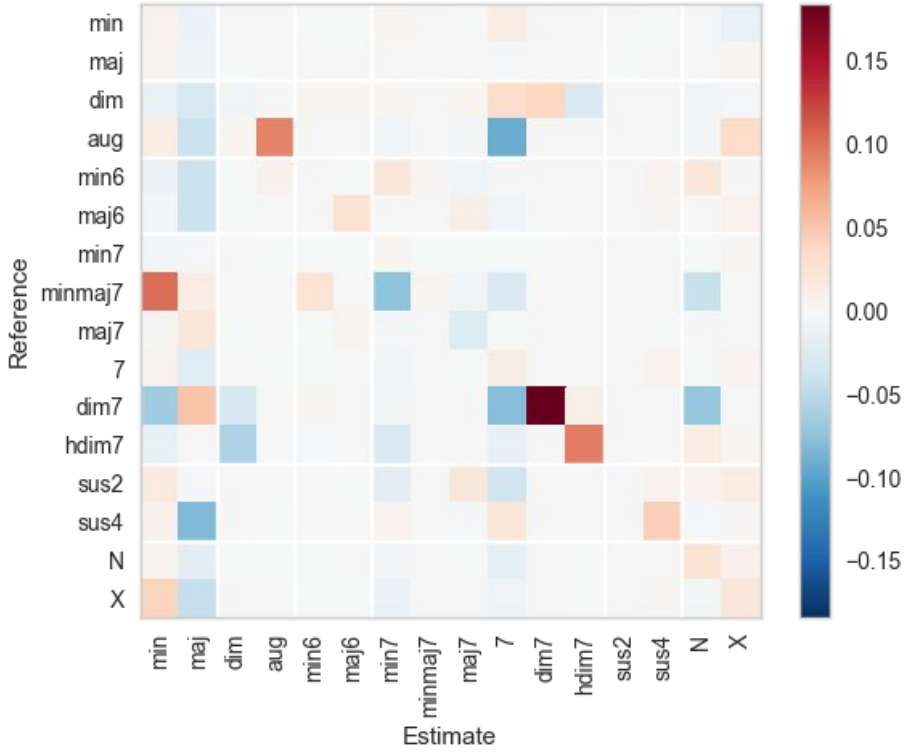
- Questions?
- Implementation is online
  - https://github.com/bmcfee/ismir2017_chords
  - **pip install crema**

brian.mcfee@nyu.edu

https://bmcfee.github.io/

Extra goodies

Error analysis: CR2+S+A vs CR2+A



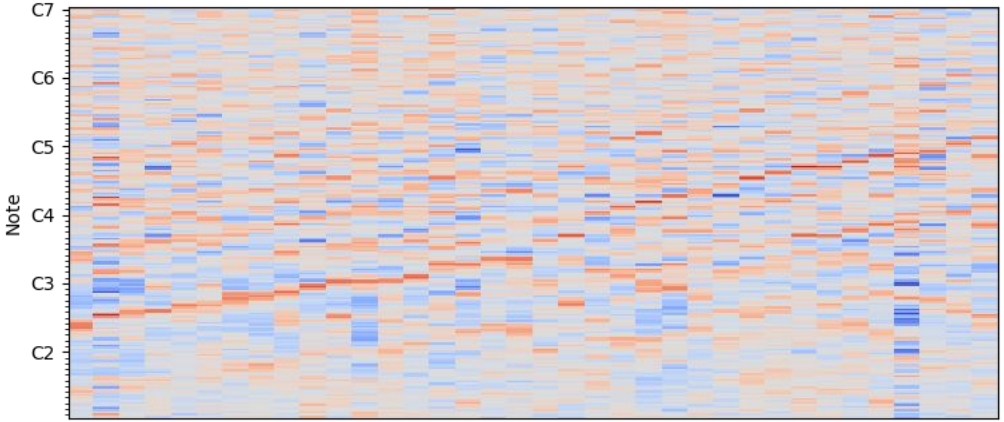
Reduction of confusions to major

Improvements in rare classes:
aug, maj6, dim7, hdim7, sus4

Learned model weights

- Layer 1: Harmonic saliency

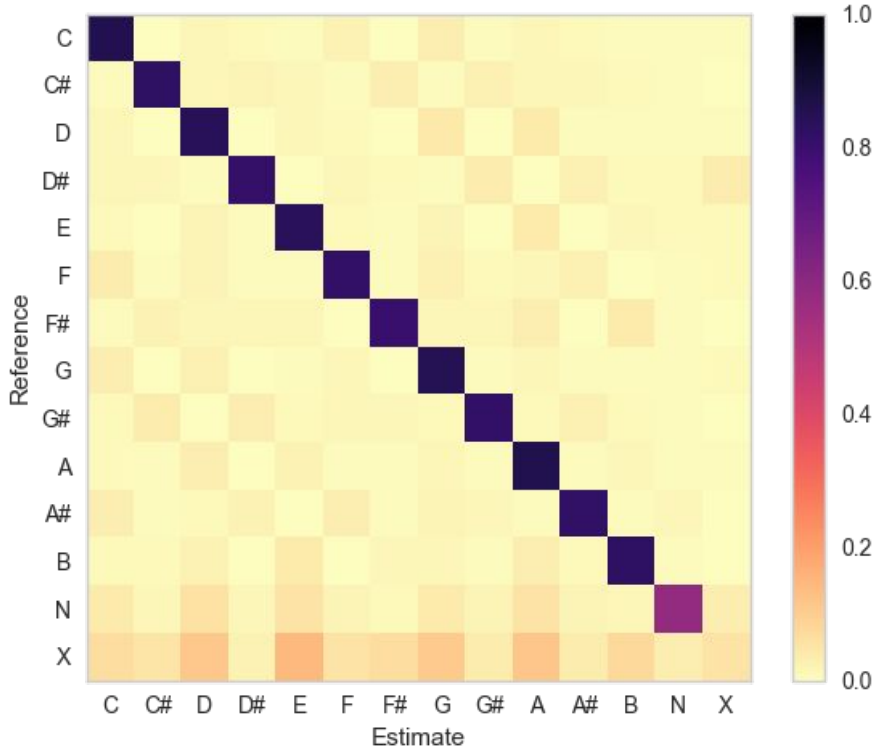
- Layer 2: Pitch filters (sorted by dominant frequency)



Training details

- Keras / TensorFlow + pescador
- ADAM optimizer
- Early stopping @20, learning rate reduction @10
  - Determined by decoder loss
- 8 seconds per patch
- 32 patches ber batch
- 1024 batches per epoch

Inter-root confusions



Confusions primarily toward P4/P5

Inversion estimation

- For each detected chord segment
  - Find the most likely bass note
  - If that note is within the detected quality, predict it as the inversion
- Implemented in the crema package
- Inversion-sensitive metrics ~1% lower than inversion-agnostic

Pitches as chroma