NYU

CREPE:
A Convolutional Representation for Pitch Estimation

April 19, 2018

ICASSP 2018 Lecture Session AASP-L4.3: Music Signal Analysis and Processing

Jong Wook Kim, Justin Salamon, Peter Li, Juan Pablo Bello
Music and Audio Research Laboratory, New York University

Task: Monophonic Pitch Estimation

- Estimating the fundamental frequency of a monophonic sound recording.

→

```
...
9.010000,216.598864,0.733938
9.020000,219.406638,0.859176
...
```

- A long-standing topic in audio signal processing research
- A fundamental problem in understanding music and audio, with many aplications
  - A core component of melody extraction systems[1]
  - A method to generate pitch annotations in multi-track datasets[2]
  - Analyzing prosodic aspects such as intonations for speech analysis

[1] Juan Bosch and Emilia Gómez, “Melody extraction in symphonic classical music: a comparative study of mutual agreement between humans and algorithms,” in *Proceedings of the 9th Conference on Interdisciplinary Musicology (CIM14)*, 2014.

[2] Justin Salamon et al. “An analysis/synthesis framework for automatic f0 annotation ofmultitrackdatasets,” in *Proceedings of ISMIR*, 2017.

Background on Monophonic Pitch Estimation

A History of Heuristic Engineering Feature Extractor Functions

- Frequency-domain methods
  - Cepstrum[1]: IFT of log magnitude spectrum, SWIPE[2]: spectrum template matching
- Time-domain methods
  - $f_{ACF}(\tau) = \sum x_t x_{t+\tau}$, $f_{AMDF}(\tau) = \sum |x_t - x_{t_\tau}|$, $f_{DF}(\tau) = \sum (x_t - x_{t+\tau})^2$
  - YIN[3]: cumulative mean normalized difference function, $f_{YIN}(\tau) = f_{DF}(\tau) \big/ \sum_{j=1}^{\tau} f_{DF}(j)$
  - pYIN[4]: an extension to YIN based on probabilistic inference over YIN's threshold
- Claim: hand-crafted feature extractors did not solve the problem
  - Reported accuracies of existing algorithms near 100% are based on simplistic datasets
  - They still perform less than ideal in a dataset with diverse timbres, etc.
  - Should be able to benefit from data-driven methods, just like the other MIR tasks

[1] A Michael Noll, "Cepstrum pitch determination," *The journal of ASA, vol. 41, no. 2*, 1967.

[2] Arturo Camacho and John G Harris, "A sawtooth waveform inspired pitch estimator for speech and music," *The Journal of ASA, vol. 124, no. 3*, 2008

[3] Alain de Cheveigné and Hideki Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The Journal of ASA, vol. 111, no. 4*, 2002.

[4] Matthias Mauch and Simon Dixon, "pYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *Proceedings of ICASSP*, 2014.

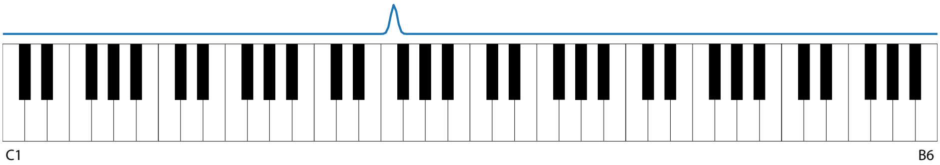
Deep Model Architecture

Layers	Filters	Kernel	Output	Note
Input			1024	normalized audio segment
Conv1D & MaxPool1D	1024	512	(128, 1024)	`strides=4` in Conv1D
Conv1D & MaxPool1D	128	64	(64, 64)	
Conv1D & MaxPool1D	128	64	(32, 64)	
Conv1D & MaxPool1D	128	64	(16, 64)	
Conv1D & MaxPool1D	256	64	(8, 64)	
Conv1D & MaxPool1D	512	64	(4, 512)	
Flatten & Dense	360		360	the pitch salience vector

- Input is a 1024-sample segment from 16 kHz recording (64 milliseconds)
- Each conv layer is followed by a batch normalization and a dropout of $p = 0.25$
- Using `padding="same"` and `pool_size=2` everywhere

Prediction Target: The Pitch Salience Representation

- The 360-dimensional output predicts the presence of pitch, inspired by [1]
  - Covers 6 octaves of notes, between C1 (32.7 Hz) and B6 (1975.5 Hz)
  - Gaussian curve centered at true pitch, with a stdev of 25 cents, as the ground-truth:



- Estimated pitch is then given as the (local) weighted average of the weights
- Optimization target: minimize the binary cross entropy:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{360} \left(-y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)\right)$$

  - Joint training of 360 binary classifiers, each detecting presence of certain pitch

[1] Rachel M Bittner et al. "Deep salience representations for f0 tracking in polyphonic music," in *Proceedings of ISMIR*, 2017.

Datasets and Evaluation

- For objective evaluation, we need a dataset with perfect pitch annotations
  - The only way to obtain such dataset is to synthesize data from known pitch curves
- The datasets:
  - **RWC-synth**: 6.16h of timbrally homogeneous audio, what pYIN[1] used for evaluation
  - **MDB-stem-Synth**: 15.36h of audio of 25 instruments, resynthesized from MedleyDB[2]
- 5-fold cross validation and artist-conditional splits
  - We report 5-fold cross-validation accuracies, with 60/20/20 train/validation/test split
  - Tracks from one artists have go to the same folds, to avoid cheating
- Reporting the following evaluation metrics, using `mir_eval`[3]:
  - **Raw Pitch Accuracy (RPA)**: proportion of frames for which pitch estimation is correct
  - **Raw Chroma Accuracy (RCA)**: same as above but for chroma, allowing octave errors

[1] Matthias Mauch and Simon Dixon, "pYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *Proceedings of ICASSP*, 2014.
[2] Rachel M Bittner et al. "Medleydb: A multitrack dataset for annotation-intensive mir research," in *Proceedings of ISMIR*, 2014.
[3] Colin Raffel et al. "mir_eval: A transparent implementation of common mir metrics," in Proceedings of ISMIR, 2014.

Results: Pitch and Chroma Accuracy

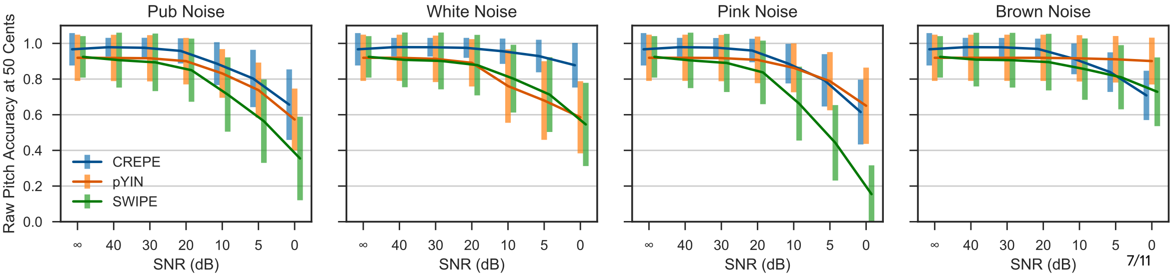
Dataset	Metric	CREPE	pYIN	SWIPE
RWC-synth	RPA	**0.999 ± 0.002**	0.990 ± 0.006	0.963 ± 0.023
	RCA	**0.999 ± 0.002**	0.990 ± 0.006	0.966 ± 0.020
MDB-stem-synth	RPA	**0.967 ± 0.091**	0.919 ± 0.129	0.925 ± 0.116
	RCA	**0.970 ± 0.084**	0.936 ± 0.092	0.936 ± 0.100

Dataset	Threshold	CREPE	pYIN	SWIPE
RWC-synth	50 cents	**0.999 ± 0.002**	0.990 ± 0.006	0.963 ± 0.023
	25 cents	**0.999 ± 0.003**	0.972 ± 0.012	0.949 ± 0.026
	10 cents	**0.995 ± 0.004**	0.908 ± 0.032	0.833 ± 0.055
MDB-stem-synth	50 cents	**0.967 ± 0.091**	0.919 ± 0.129	0.925 ± 0.116
	25 cents	**0.953 ± 0.103**	0.890 ± 0.134	0.897 ± 0.127
	10 cents	**0.909 ± 0.126**	0.826 ± 0.150	0.816 ± 0.165

Results: Noise Robustness

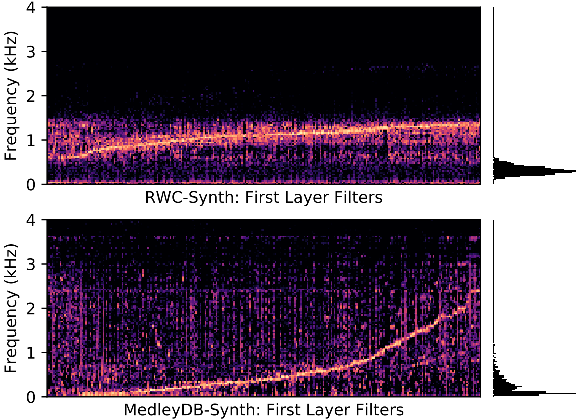
- Evaluated on degraded audio using Audio Degradation Toolbox[1] (ADT)
- CREPE performs well under various types of additive noise
  - except brown noise, for which pYIN performed better

[1] Matthias Mauch and Sebastian Ewert, “The audio degradation toolbox and its application to robustness evaluation,” in *Proceedings of ISMIR*, 2013.



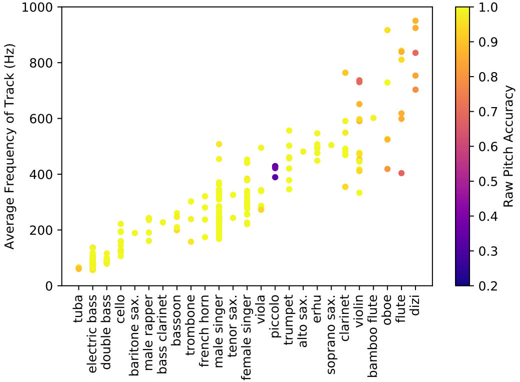
Results: First-Layer Filters

- The first-layer filters adapts to the timbre and the pitch distribution of the training dataset.
- When trained on a dataset with highly homogeneous timbre, the weights learn to differentiate the harmonics, rather than the F0.



On the Generalizability of the Model

- Pitch accuracies are correlated with the instruments and their frequency (⟶)
- Like any data-driven models, it only learns what it saw during training.
- Despite the artist-conditional splits, a model trained on one dataset doesn't tend to perform as well on other datasets.
- To make a general-purpose pitch estimator, it needs to be trained on a variety of datasets.



Try It!

- Python tool for running a pre-trained model: `https://github.com/marl/crepe/`
  - To install and run:

```
$ pip install crepe     # installs the CREPE package
$ crepe track.wav       # run pitch estimation on track.wav
```

- The script above produces:
  - a CSV file with estimated pitch and voicing confidence
  - a salience plot in a PNG file, and optionally as a Numpy format
- Trained on 6 different datasets to ensure generalizability
- An interactive demo: `https://marl.github.io/crepe/`

Conclusions and Future Work

- Presented a data-driven neural network model as a state of the art method
  - Runs directly on time-domain audio signal
  - Robust with heterogeneous timbre and additive noise
  - Stays highly accurate, even with 10 cents threshold
- Possible extensions to the model:
  - Temporal tracking of pitch curves
  - Data augmentation with:
    - pitch/phase shifts
    - various kinds of additive noise
  - Learnable pre-processing filter