DeepPitch: Wide-Range Monophonic Pitch Estimation using Deep Convolutional Neural Networks

Lloyd Watts
June 14, 2018

Abstract – Pitch Estimation is an important problem in Machine Hearing, with application in Music Information Retrieval, Speech Analysis, and Auditory Scene Analysis. In this paper, we propose DeepPitch, a Deep Convolutional Neural Network (DCNN) approach that operates on a high-resolution log-frequency spectrum (9 taps/semitone), with one DCNN operating as a binary Pitch Activity Detector, and another DCNN estimating the Pitch Value at 9 taps/semitone resolution. Most Pitch Estimation algorithms such as pYIN, CREPE, RAPT, etc., use a restricted range of output pitch values, compared to the wide range of human pitch perception. DeepPitch was designed to cover a wide pitch range of 27.5 Hz to 4.186 kHz, corresponding to the full range of a piano. The training and test data are based on a 107-minute set of 44.1 kHz audio samples including speech (male, female, child), musical instruments (piano, guitar, bass, violin), synthetic test signals (sine waves, missing fundamental examples, sirens), and bandlimited speech and piano. Performance of the algorithm is 99.75% accuracy on Pitch Activity Detection, 96.92% on Pitch Value.

1. Introduction

Pitch Estimation, or fundamental frequency (f0) estimation, of a monophonic audio signal is an important problem in Music Information Retrieval [1, 2, 3], Speech Analysis [4], and Auditory Scene Analysis [5], and monophonic Pitch Estimation is a necessary prerequisite for the development of polyphonic Pitch Estimation. Many front-end signal representations have been used in many different Pitch Estimation algorithms, including linear-frequency spectral frames (used by harmonic sieve methods [6, 7]), cepstrum [8], autocorrelation function (ACF) [9], average magnitude difference function (AMDF) [10], normalized cross-correlation function (NCCF, used by RAPT [11] and PRAAT [12]), and cumulative mean normalized difference function (CMNDF, used by YIN [13] and pYIN [14]), cochlear model [15], and waveform samples (used by CREPE [16]).

In this paper, we present DeepPitch, a monophonic Pitch Estimation method using a high-resolution log-frequency spectrum as the front-end signal representation, and two Deep Convolution Neural Networks (DCNNs), one for Pitch Activity Detection, and one for Pitch Value Estimation. The inputs and the outputs of the Pitch Estimation DCNN are on the same high-resolution log-frequency scale (9 taps/semitone), so the DCNN has a direct mapping from inputs to outputs, and a translation-invariant convolutional approach is a natural fit to the harmonic comb patterns which regularly appear in the spectral inputs. The training data was chosen to adapt the system to a broad set of stimuli, covering the full 27.5 Hz to 4.186 kHz range of a piano: speech (male, female, child), musical instruments (piano, bass, guitar, violin), synthetic test signals (sine waves, missing fundamental examples, sirens), and bandlimited speech and piano.

2. Architecture

The DeepPitch architecture is shown in Figure 1.

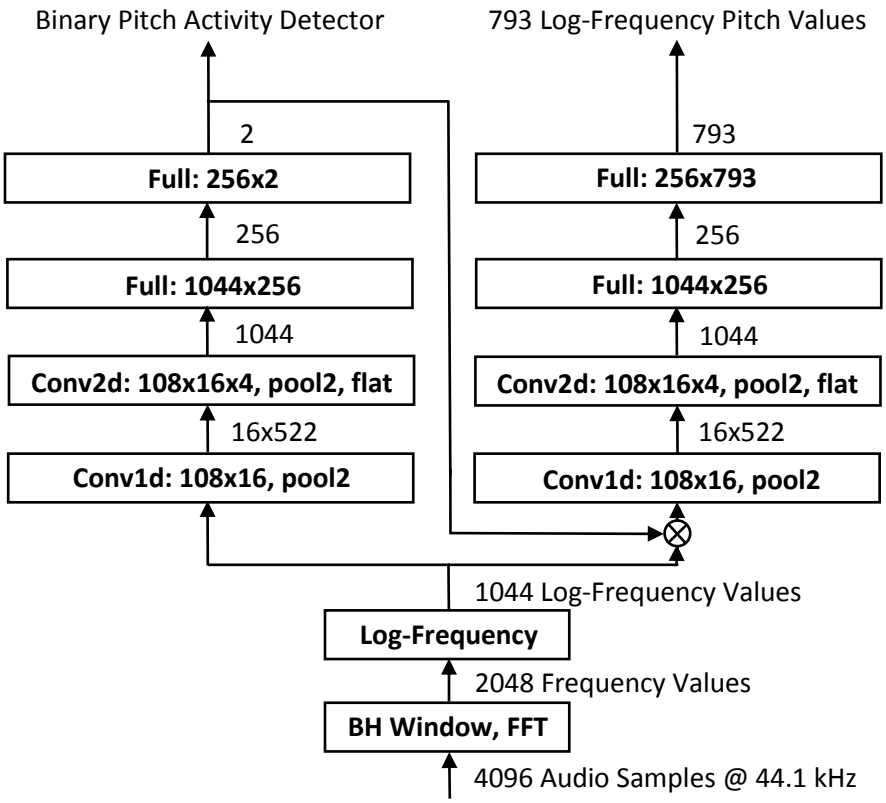

Figure 1: The DeepPitch Architecture.

The input to the system is a frame of 4096 Audio Samples, sampled at 44.1 kHz sampling rate, and hopped by 441 samples, for a frame rate of 100Hz. This frame of samples is processed by a Blackman-Harris Window and a 4096-point FFT, leading to a 2048-point magnitude spectrum on a linear frequency scale from 0 to 22.1 kHz. The 1044-point Log-Frequency spectrum is computed by resampling the linear-frequency spectrum using cubic interpolation onto a log-frequency scale from 27.5 Hz to 22.1 kHz, at a resolution of 9 taps/semitone.

The Pitch Activity Detector DCNN consists of two convolutional layers and two fully connected layers. The first convolutional layer uses 16 1-dimensional kernels of size 108 frequency values (spanning one octave = 12 semitones x 9 taps/semitone). This convolution operates on the 1044 frequency values and produces 16x1044 outputs, which are then rectified (Rectified Linear Units, or RELU) and max-pooled by 2 down to 16x522. The second convolutional layer uses 4 2-dimensional kernels of size 16x108 (spanning two octaves, because of the previous pooling). This convolution produces 4x522 outputs, which are then rectified (RELU) and max-pooled by 2 down to 4x261, and then flattened to 1044 outputs. The third layer is fully connected with 1044 inputs and 256 rectified (RELU) outputs, and is followed by a dropout layer [17] with dropout probability 0.5. The fourth layer is fully connected with 256 inputs and 2 outputs, with no rectification, and a SoftMax, representing a binary Pitch Activity Detection.

The Pitch Value Estimator DCNN consists of two convolutional layers and two fully connected layers. The first convolutional layer uses 16 1-dimensional kernels of size 108 frequency values (spanning one octave = 12 semitones x 9 taps/semitone). This convolution operates on the 1044 frequency values and produces 16x1044 outputs, which are then rectified (RELU) and max-pooled by 2 down to 16x522. The second convolutional layer uses 4 2-dimensional kernels of size 16x108 (spanning two octaves, because of the previous pooling). This convolution produces 4x522 outputs, which are then rectified (RELU) and max-pooled by 2 down to 4x261, and then flattened to 1044 outputs. The third layer is fully connected with 1044 inputs and 256 rectified linear (RELU) outputs, and is followed by a dropout layer with dropout probability 0.5. The fourth layer is fully connected with 256 inputs and 793 outputs, with no rectification, and a SoftMax, representing a probability distribution of 793 candidate pitch values, covering the 88-semitone range of a piano at 9 taps/semitone (plus one unused tap).

Both networks are trained to minimize the binary cross entropy between the target vector and the predicted vector. This loss function is optimized using the ADAM optimizer [18], with learning rate 0.0001. The best performing model is selected after training until the validation accuracy no longer improves. This architecture and the training procedures are implemented using TensorFlow [19].

3. Experiments

3.1. Data Sets

The Core Data Set is 8 minutes and 40.04 seconds (520.04 seconds) of audio, or 52004 frames at 100 frames/second. As shown in Figure 2, the audio contains speech (male, female, child), full scales played on musical instruments (piano, guitar, bass, violin), sine wave scale, test signals (missing fundamentals, sirens), stressed speech, bandlimited speech, and bandlimited piano notes from low register. The ground truth targets for the speech samples were produced using get_f0 (RAPT) from WaveSurfer software [20], and then manually corrected where obvious octave errors had occurred. The ground truth targets for the musical instruments were generated along with the data using a custom MIDI data set. The audio signals and ground truth targets for the sine wave scale and test signals were produced using custom Mathematica software. The ground truth target pitch values are shown in green in Figure 1. Note that the pitch values for the musical scales are active at every musical semitone (i.e. chromatic scales, standard A440 tuning), and the pitch values for the missing fundamental samples were created at every 6 semitones.

This Core Data Set was originally intended as a test set for a hand-crafted pitch estimation algorithm. It is inadequate for use as a training/validation/test set for a Machine Learning algorithm because of its very irregular histogram of target values, as shown in Figure 3(a). In that histogram, the distributions of the Speech samples, musical instruments (spikes every semitone), and missing fundamental samples (spikes every 6 semitones) can easily be seen. To rectify this problem, an Extended Data Set was created by pitch-shifting the speech and musical instrument log-frequency spectral values through a range of ±4 frequency taps (±½ semitone at 9 taps/semitone), and similarly pitch-shifting the missing fundamental values through a range of ±27 frequency taps (±3 semitones at 9 taps/semitone), thus filling in the unrepresented frequency taps. The Extended Data Set is 643,480 frames long, representing 6,434.8 seconds of audio, or 107.25 minutes. The improved histogram of the Extended Data Set is shown in Figure 3(b).

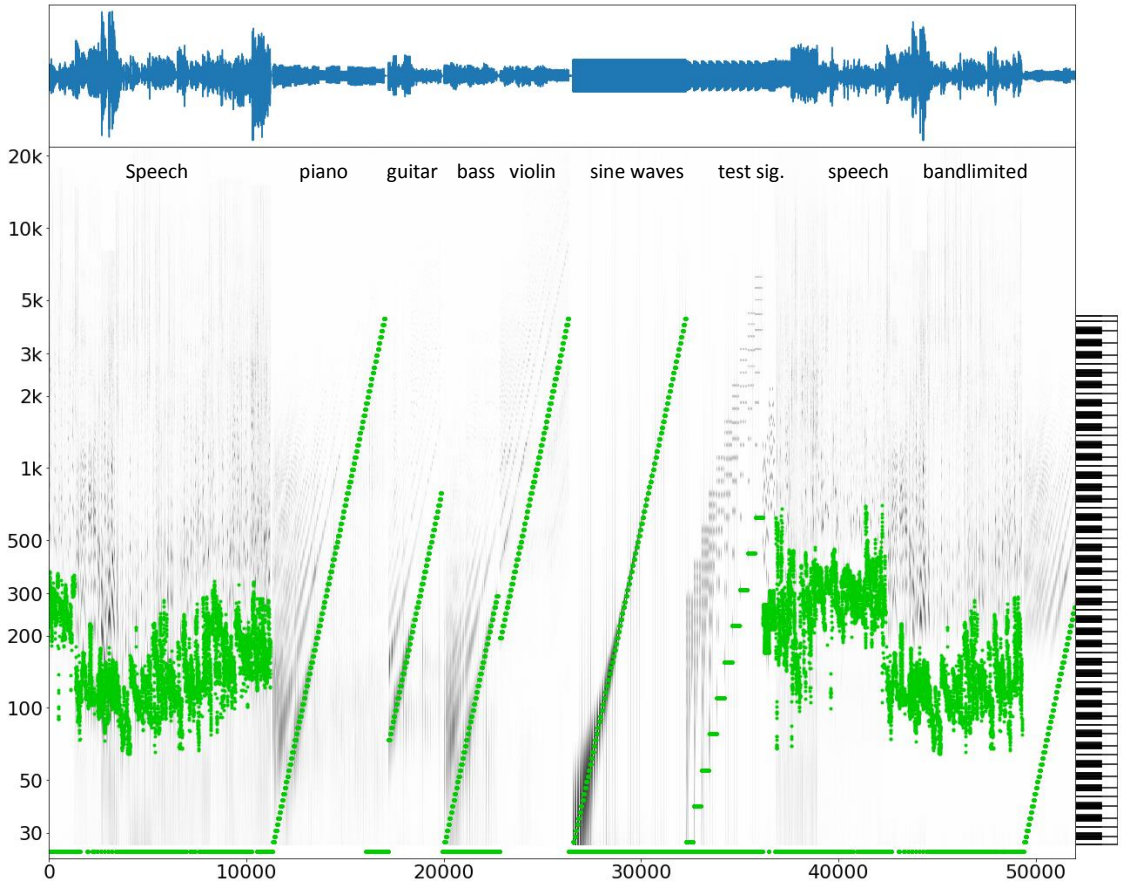

Figure 2: The Core Data Set (520 seconds).

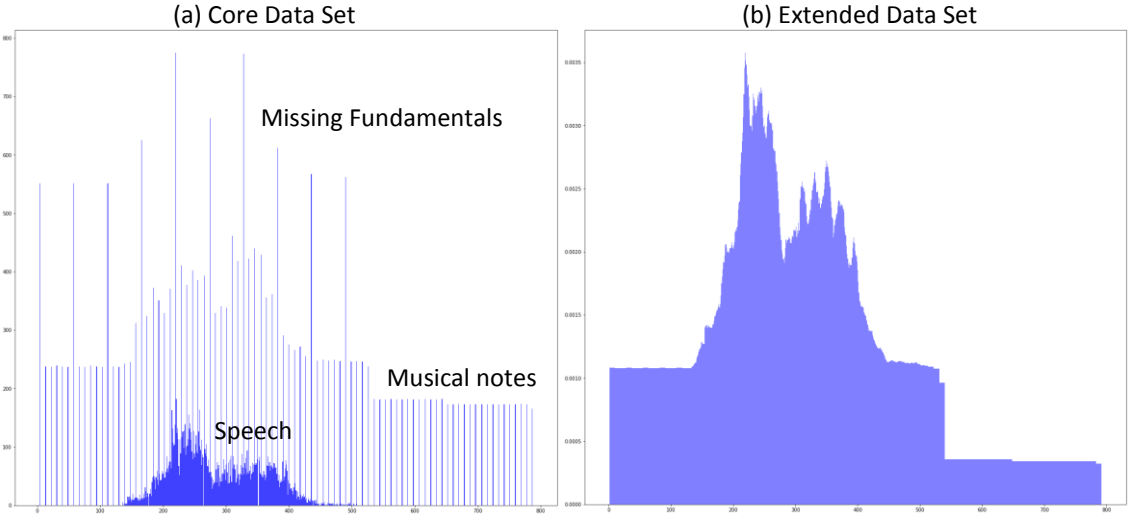

Figure 3: Pitch Target Value Histograms.

The Pitch Estimator was trained only on frames which were classified as pitched in the ground truth data. This subset of the Extended Data Set contained 510,849 frames, representing 5,108.5 seconds of audio, or 85.14 minutes.

3.2. Methodology

We trained the model using a simple 70/10/20 train/validation/test split. During training, a pitch estimate was considered correct if it exactly matched the target pitch at 9 taps/semitone resolution (11 cent accuracy). We trained the networks until the accuracy on the Validation Set stopped improving. For evaluation of the performance of the pitch estimator, an estimate was considered correct if it matched the target pitch within ±4 frequency taps (± ½ semitone at 9 taps/semitone, or ± 50 cents).

Once we had established good performance of the network, we did some moderate hyper-parameter tuning, decreasing the number of neurons in each layer to find a reduced-complexity solution with very little degradation in performance. The reported configuration is the result of this complexity reduction. Quadrupling the number of neurons in each layer relative to the reported configuration improves the accuracy by about 0.1%.

3.3. Results

The accuracy of the Pitch Activity Detector on the Test Set was 99.75%. The accuracy of the Pitch Estimator on the Test Set was 96.92%.

An example of an output of the DeepPitch algorithm on a male speech sample is shown in Figure 4. Figure 5 shows the DeepPitch output on the full Core Data Set.

4. Discussion and Conclusions

In this paper, we presented DeepPitch, a Deep Convolutional Neural Network architecture for monophonic pitch estimation operating on log-frequency spectral inputs. The parameters of the design and the data set used for training and testing were chosen to match the broad capabilities of the Human Auditory Pathway, i.e. wide input frequency range (20Hz-22kHz), wide output pitch range (27.5Hz-4.186kHz), high output pitch resolution on logarithmic frequency scale (9 taps/semitone), and to be able to handle a wide range of input signal types (male, female and child speech, musical instruments with wide pitch ranges, test signals (sine waves, missing fundamental examples, sirens), stressed speech, and bandlimited speech and musical instruments. Most pitch algorithms are designed for, and tested on, specialized subsets of those signal types (e.g., adult speech only (RAPT) [11], or synthesized musical instruments only (CREPE) [16]).

DeepPitch also is designed to work only on the current frame of audio data, which means that it is a low-latency design, compared to other algorithms such as pYIN [14] and RAPT [11], which use Hidden Markov Models with Viterbi Decoding to provide smoothed pitch tracks, but at the cost of large and variable decoding latency.

We see DeepPitch as a foundational component in a Machine Hearing system modeled on the Human Auditory Pathway as described by Watts [21]. In this architecture, the inputs to the Pitch Estimator are the log-frequency-scale spectral energies (from the Cochlea and MultiPolar Cells), and this log-frequency spectral representation is preserved throughout the higher stages of the Human Auditory Pathway – this

is one reason why it was important to use a log-frequency spectral representation as the input to DeepPitch. The output of the Pitch Estimator is used as a signal in its own right in multiple contexts (pitch, melody, chord in music; pitch and prosody for speech) [21], and as a grouping cue for Auditory Scene Analysis [5], stream separation and noise robustness.

To fully deliver on the potential of DeepPitch in the context of the broader Human Auditory Pathway model, it will be necessary to extend DeepPitch to the general-purpose Polyphonic case. An early attempt at this was reported in [15], using a hand-crafted fixed Harmonic Sieve approach on a logarithmic frequency scale, however its performance was limited by octave errors in speech, and poor pitch estimation performance for extreme pitches (i.e. very low- or very high-pitched musical notes). The current DeepPitch architecture solves those problems, as can be seen in Figure 5, and we believe it could now be extended to the general-purpose Polyphonic case.

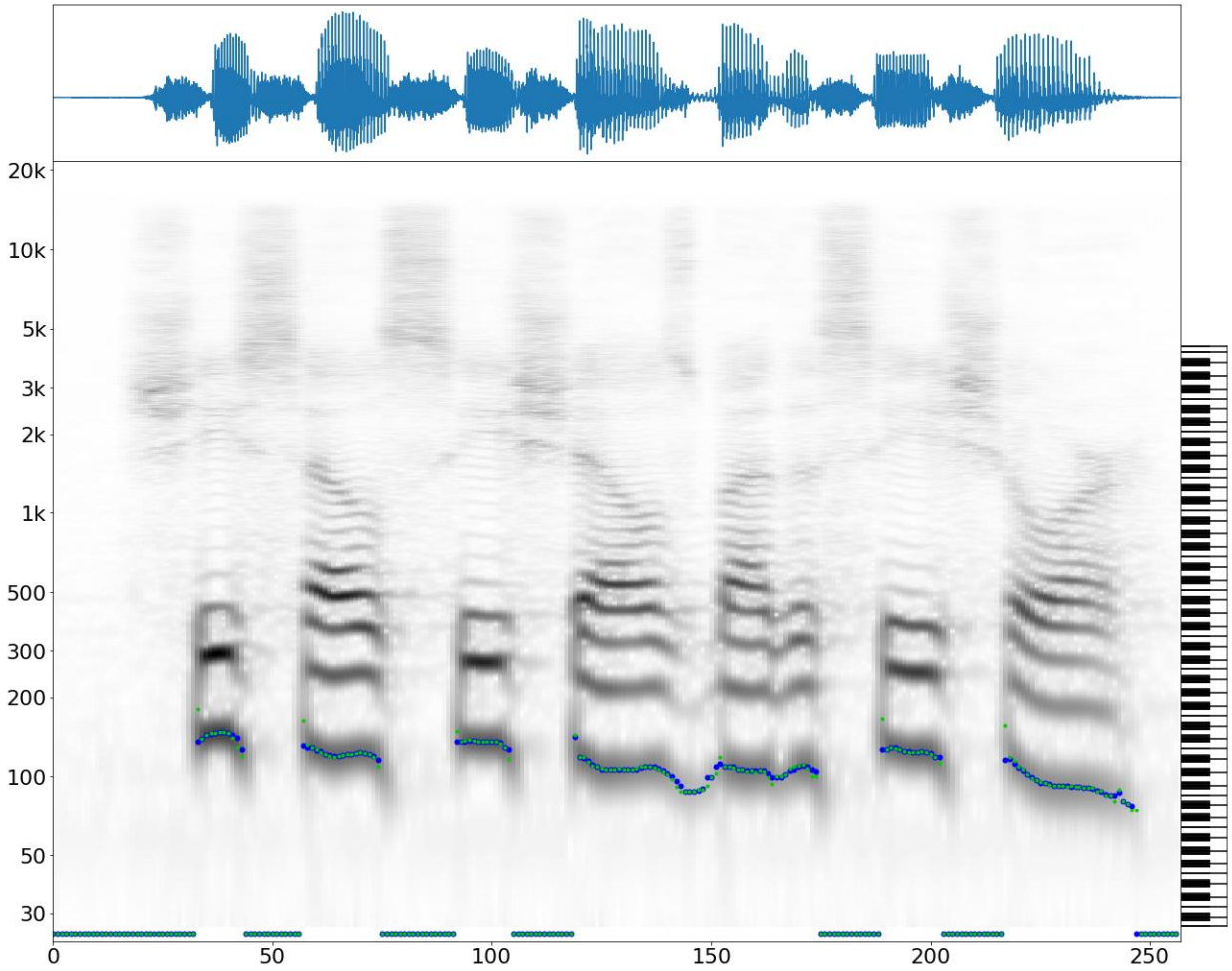

Figure 4: DeepPitch Output for male speech "She sells sea shells by the sea shore". Green dots are ground truth Pitch values. Blue dots are Pitch Estimates from the DeepPitch algorithm.

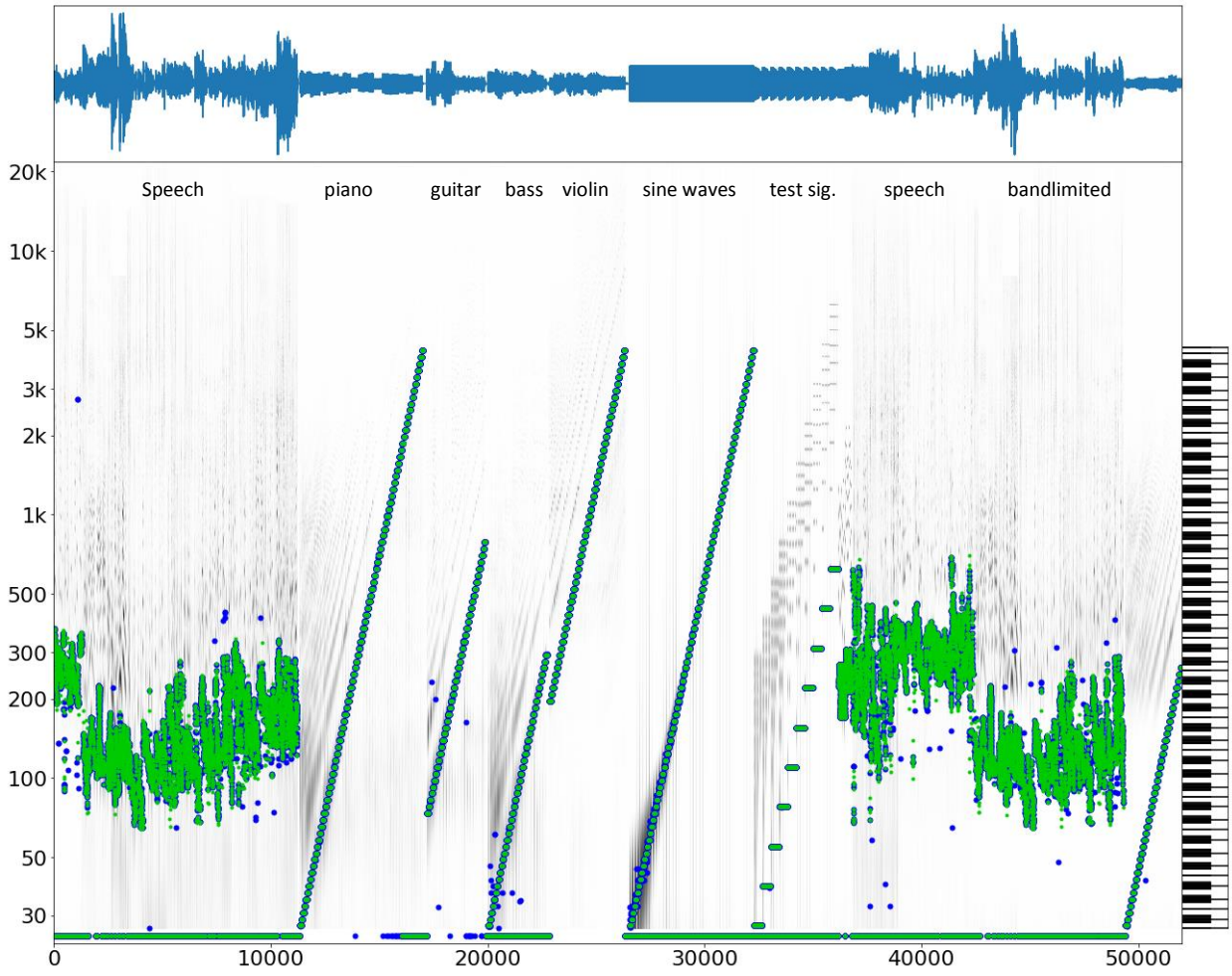

Figure 5: DeepPitch Output for the Core Data Set. Green dots are ground truth Pitch values. Blue dots are Pitch Estimates from the DeepPitch algorithm.

References

[1] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research.," in Proceedings of the 15th ISMIR Conference, 2014, vol. 14, pp. 155–160.

[2] 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.

[3] Matthias Mauch, Chris Cannam, Rachel Bittner, George Fazekas, Justin Salamon, Jiajie Dai, Juan Bello, and Simon Dixon, "Computer-aided melody note transcription using the tony software: Accuracy and efficiency," in Proceedings of the First International Conference on Technologies for Music Notation and Representation, 2015.

[4] Maria Luisa Zubizarreta, Prosody, focus, and word order, MIT Press, 1998.

[5] Bregman, A. (1990), Auditory Scene Analysis, MIT Press.

[6] Goldstein, J. (1973). ‘‘An optimum processor theory for the central formation of pitch of complex tones,’’ J. Acoust. Soc. Am. 54, 1496–1516.
[7] Terhardt, E. (1974). ‘‘Pitch consonance and harmony,’’ J. Acoust. Soc. Am. 55, 1061–1069.
[8] Michael Noll, “Cepstrum pitch determination,” The journal of the acoustical society of America, vol. 41, no. 2, pp. 293–309, 1967.
[9] John Dubnowski, Ronald Schafer, and Lawrence Rabiner, “Real-time digital hardware pitch detector,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 24, no. 1, pp. 2–8, 1976.
[10] Myron Ross, Harry Shaffer, Andrew Cohen, Richard Freudberg, and Harold Manley, “Average magnitude difference function pitch extractor,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 22, no. 5, pp. 353–362, 1974.
[11] David Talkin, “A robust algorithm for pitch tracking (rapt),” Speech Coding and Synthesis, 1995.
[12] Paul Boersma, “Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound,” in Proceedings of Institute of Phonetic Sciences, 1993, vol. 17, pp. 97–110.
[13] Alain De Cheveigné and Hideki Kawahara, “Yin, a fundamental frequency estimator for speech and music,” The Journal of the Acoustical Society of America, vol. 111, no. 4, pp. 1917–1930, 2002.
[14] Matthias Mauch and Simon Dixon, “pYIN: A fundamental frequency estimator using probabilistic threshold distributions,” in Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on. IEEE, 2014, pp. 659–663.
[15] L. Watts, "Commercializing Auditory Neuroscience", in *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2006 Symposium*, published by National Academy of Engineering, 2007, pp. 5-14.
[16] Jong Wook Kim, Justin Salamon, Peter Li, Juan Pablo Bello, “CREPE: A Convolutional Representation for Pitch Estimation”, arXiv:1802.06182.
[17] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.
[18] Diederik Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” in Proceedings of the International Conference on Learning Representations (ICLR), 2015.
[19] TensorFlow: https://www.tensorflow.org/.
[20] WaveSurfer: https://sourceforge.net/projects/wavesurfer/
[21] L. Watts, "Reverse-Engineering the Human Auditory Pathway", in J. Liu et al. (Eds.): *IEEE WCCI 2012, LNCS 7311,* pp. 47–59, 2012.