FusedF0: Improving DNN-based F0 Estimation by Fusion of Summary-Correlograms and Raw Waveform Representations of Speech Signals

Eray Eren, Lee Ngee Tan, Abeer Alwan

Department of Electrical and Computer Engineering, University of California, Los Angeles, USA

erayeren@g.ucla.edu, leengeetan@ucla.edu, alwan@ee.ucla.edu

Abstract

DSP-based F0 estimation algorithms, such as multi-band summary-correlogram (MBSC), are robust to noisy speech. Recent studies show that mapping from raw waveform segments into F0 estimates by DNNs can outperform DSP-based methods in F0 estimation. However, generalization and noise robustness of DNNs have not been fully addressed previously. We propose a hybrid DSP and DNN based approach to F0 estimation. Key contributions include: (a) a modified version of MBSC that is substantially faster than the original algorithm while maintaining the accuracy of F0 estimates; (b) a method for fusing DSP features with raw waveform representations using a DNN architecture to obtain noise-robust F0 estimation; (c) demonstrating that auxiliary DSP features improve generalization with a relatively small number of DNN parameters. On the PTDB-TUG database, the proposed algorithm outperforms the MBSC and CREPE DNN baselines (including optimized versions) for clean and noisy speech at 20, 10, and 0 dB SNR.

Index Terms: pitch estimation, summary-correlogram fusion, noise-robust, deep learning

1. Introduction

Fundamental frequency (F0) estimation of a speech signal is the process of extracting the periodicity from voiced speech segments. F0 estimation is important for many speech processing applications such as automatic speech recognition (ASR) [1], text-to-speech (TTS) synthesis [2], speech enhancement [3], speaker verification [4], and voice separation [5].

Conventional F0 estimation approaches have utilized digital speech processing-based (DSP) heuristics. These DSP-based approaches can be divided into 3 main categories: time domain processing [6, 7, 8, 9], frequency domain processing [10, 11, 12, 13], and time-frequency domain processing [14, 15].

Time domain methods like RAPT [6] and Kaldi F0 extraction [7] utilize the normalized cross-correlation (NCCF) function for F0 estimation. On the other hand, YIN [8] and PYIN [9] make use of the cumulative mean normalized difference function for F0 estimation in the time domain. Additionally, the subharmonic-to-harmonic ratio (SHR) is the basis for the frequency domain F0 extractor in [10]. Another frequency domain F0 extractor in [11], utilizes the summation of residual harmonics (SRH). The frequency domain F0 extractor SWIPE [12] estimates F0 by matching the input signal spectrum with a sawtooth waveform spectrum by cosine similarity, and SWIPE' [12], a variant of SWIPE, utilizes only the first and prime harmonics of the signals. A recent study [13], makes use of a high-resolution frequency domain transformation to extract F0 estimates. Time-frequency domain methods like multi-band summary-correlogram (MBSC) [15], or [14] separate a speech signal into various bands, and apply time domain processing in each band separately.

Unlike conventional DSP-based approaches, deep learning-based approaches learn the mapping function from the data to F0 estimates. These methods differ in both the input representation that they use, and in the neural model architecture. The recurrent JDC model [16], and the convolutional GIO model [17] use spectrogram inputs to estimate fundamental frequencies. Another convolutional neural model, SPICE [18], uses constant-Q transforms (CQT) as inputs, and estimates F0 in a self-supervised fashion. Furthermore, the convolutional CREPE model [19] processes segmented raw waveforms instead of their frequency domain representations to obtain F0s. Deep neural network (DNN) methods that process raw waveforms, such as the widely used state-of-the-art (SOTA) CREPE model, have been shown to outperform several DSP baselines such as SWIPE for F0 estimates for both clean and noisy conditions if the DNNs are trained with adequate and sufficient training samples. On the other hand, SOTA DSP methods like MBSC are accurate and very robust to noise. To the best of our knowledge, there has been no comparison of these recent deep learning-based architectures with MBSC for noisy speech. This can be due to the fact that the MBSC algorithm is slow. The CREPE study reported robustness for different noisy conditions. However, raw waveform processing in the initial layers of DNNs are susceptible to a known vanishing gradient problem, which can adversely affect noise-robustness [20].

We propose a hybrid approach that fuses noise-robust auxiliary DSP representations and raw waveform representations to obtain F0 estimates using deep learning. For fusing noise-robust auxiliary DSP features, we use intermediate features, summary-correlograms (SCs), from a modified version of MBSC. The modified version of MBSC processes input waveforms much faster than the original algorithm, enabling the fusion in our proposed DNN architecture. We show that the proposed fusion network, FusedF0, can outperform both the SOTA DSP baselines and raw waveform processing DNN, CREPE, for different noise conditions using the PTDB-TUG database [21].

The paper is organized as follows. Section 2 introduces the modified MBSC algorithm and the proposed FusedF0 network for F0 estimation. In Section 3, we describe the experimental setup, the dataset, the baselines and the F0 estimation metrics used in the experiments. In Section 4, we report and discuss the evaluation of the proposed architecture against the SOTA DSP and DNN baselines. Lastly, in Section 5, we summarize our contributions and discuss future work.

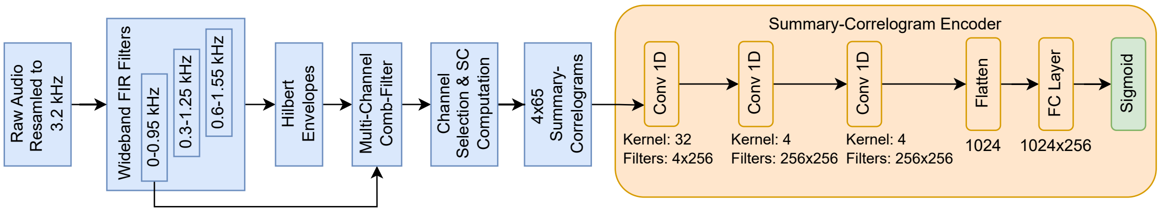

Figure 1: *The efficiency-improved summary-correlogram (SC) extraction process of the modified MBSC algorithm (in blue) and the summary-correlogram encoder (SCEnc) network (in orange). "Kernel" indicates the corresponding kernel size, and "Filter" indicates the corresponding number of input and output channels for the convolutional layers.*

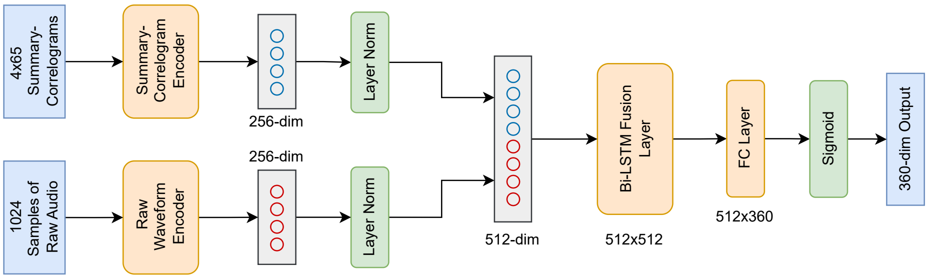

Figure 2: *FusedF0 architecture.*

2. Model and Architecture

2.1. Improving the efficiency of MBSC

While MBSC achieves high F0 accuracy in quiet and in noise, its computational complexity is high. We attempt first to reduce the algorithm's complexity while maintaining F0 estimation accuracy. Figure 1 shows the SC extraction process and the summary-correlogram encoder (SCEnc) architecture. SCs are extracted in a way that is similar to the MBSC algorithm: first, wideband FIR filters are implemented, and the Hilbert transform is used to obtain Hilbert envelopes. Then, multichannel comb-filters are used to enhance peaks in the spectrum. These enhanced peaks are selected based on the harmonic-to-subharmonic (HSR) ratio, and SCs are calculated to obtain the final SC features.

We modified the algorithm to make it faster. First, we resample the raw audio waveforms to 3.2 kHz (from 8 kHz), reduce the FFT size to 512 (from 8192) and the number of bands to 3 (from 4); and use 64-point (instead of 32) FIR filters with 950 Hz bandwidth (instead of 1 kHz) and 650 Hz overlap (instead of 200 Hz). These changes lead to a much faster SC extraction for MBSC while retaining performance, as will be shown in Section 4. SC calculations are based on the output of the first band in addition to the Hilbert envelopes from the 3 frequency bands. These 4 different 65-dimensional SCs are concatenated to form the final 4×65 dimensional SC feature vector X_{sc} which is used in FusedF0. That is, the band dimension is 4 and the correlogram dimension is 65.

The feature vector, X_{sc}, is used in the modified MBSC pitch extractor to obtain the multi-band SC computations, pitch estimates, and voicing detection [15].

2.2. System Architecture

The FusedF0 architecture uses two different neural encoders: a summary-correlogram encoder (SCEnc) and a raw waveform encoder (RawEnc). The SCEnc maps SCs, which are extracted from the faster modified MBSC, into 256-dimensional vectors. Similarly, the RawEnc maps segmented samples of the raw audio files into 256-dimensional vectors. After layer normalizations (LN), these encoded vectors are concatenated (Cat); and a bidirectional LSTM (BiLSTM) layer is used for the fusion of the encoded vectors (512-dimensional). Finally, a fully-connected (FC) layer, followed by sigmoidal activations, maps fused vectors into 360-dimensional outputs. The overview of FusedF0 architecture is shown in Figure 2. The overall network is characterized by the following equations:

$$X_{cat} = Cat(LN(SCEnc(X_{sc})), LN(RawEnc(X_{raw}))), \quad (1)$$

$$X_{fused} = BiLSTM(X_{cat}), \quad (2)$$

$$X_{out} = Sigmoid(FC(X_{fused})), \quad (3)$$

where X_{sc} indicates 4×65 dimensional SC features, X_{raw} indicates 1024-dimensional raw waveform features, X_{cat} stands for 512-dimensional concatenated vectors, X_{fused} denotes 512-dimensional vectors after Bi-LSTM fusion layer, and X_{out} denotes the 360-dimensional outputs.

SCEnc, shown in Figure 1, utilizes 1-dimensional convolutions (Conv 1D). The first Conv 1D layer uses a stride of 2 on the correlogram dimension, and the band dimensions are used as the input channels of the first layer. Furthermore, the size of the first Conv 1D layer kernel is 32. The output channels of all the Conv 1D layers in the SCEnc are set to 256. The second and third Conv 1D layers use a kernel size of 4 with a stride of 1. All the Conv 1D layers are max pooled with a max pool kernel of 2 after the ReLU [22] activations followed by batch normalizations [23]. We use a dropout [24] rate of 0.25 after these layers. After the Conv 1D layers, the band and the correlogram dimensions are flattened (i.e.,, the $4x65$ outputs are made into a vector). The flattened dimension is mapped by an FC layer with sigmoid activations to obtain the final outputs of SCEnc.

We use Crepe-Tiny [19], which uses one eighth the number of filters in each convolutional layer compared to the full

CREPE model size, as the RawEnc. However, we use the output FC layer of the Crepe-Tiny, followed by sigmoid activations, with 256-dimensional output instead of a 360-dimensional output. The frame (or the segment) shift for both SCEnc and RawEnc are 10 msec long.

Similar to [19, 25, 26], FusedF0 network outputs are represented on the logarithmic scale with equal sized bins, and the corresponding scales of these bins for the 360-dimensional output vector X_{out} are smoothed with the Gaussian kernel smoother used in [19, 25] to form a smoothed output vector $\hat{Y}$. Aiming for a better fit to speech signals, unlike the above mentioned studies which focused on singing voices and music, we cover a frequency range from 30 to 400 Hz, and divide this range into 360 equal sized bins, which corresponds to 12.5-cent intervals, on the logarithmic scale. The following equations:

$$c(f) = 1200 \cdot \log_2 \frac{f}{10} \,, \tag{4}$$

$$\hat{c} = \frac{\sum_{m-4}^{m+4} \hat{y}_i c_i}{\sum_{m-4}^{m+4} \hat{y}_i}, \quad m = \underset{i}{\operatorname{argmax}}\, \hat{y}_i \,, \tag{5}$$

$$f = 2^{\hat{c}/1200} \cdot 10 \,, \tag{6}$$

are used for converting the frequency scale f in Hz to the logarithmic scale $c(f)$; calculating the local weighted average $\hat{c}$ of smoothed values $\hat{y}_i$ with the closest logarithmic frequency bins c_i; and converting back to the frequency scale in Hz, respectively.

3. Experimental setup

Trained models were optimized using the Adam optimizer [27] with a learning rate of 0.0002, and decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.98$. Binary cross-entropy was used as the loss function for the optimization. The hyperparameters of the FusedF0 architecture were optimized using the validation set. We used early stopping to prevent overfitting. Since the validation loss metrics did not improve, the Crepe-Opt (full) model was trained for 32k iterations; the Crepe-Med-Opt and the proposed FusedF0 models were trained for 40k iterations; and the Crepe-Tiny-Opt was trained for 51k iterations. We used an NVIDIA Tesla A100 GPU for the experiments.

For the RawEnc in our architecture, we investigated the use of different CREPE variant sizes, but our experimental findings reveal that the performance of the fusion algorithm degrades with increasing CREPE variant size. Since the summary correlogram encoder does not require many parameters, we argue that using too large of a RawEnc leads to overfitting to raw waveform features.

3.1. Database, baselines and evalution metrics

We trained and evaluated the proposed FusedF0 model on the PTDB-TUG [21] database, which has ground truth labels from laryngograph signals. The PTDB-TUG database contains 10 female and 10 male native English speakers. It consists of 236 speech recordings for each speaker ($\sim$ 0.5 hours per speaker) and the corresponding ground truth F0s. We excluded the female speaker F10 and the male speaker M10 from our training set in order to evaluate the average test performance of seen and unseen speakers. For each of the 18 seen speakers, 140 utterances were used for the training set, 46 utterances were used for the validation set, and 46 utterances were used for the test set. Similarly, for each of the unseen speakers F10 and M10, 46 utterances were included in the test set.

All samples in the dataset were downsampled to 16 kHz. The proposed model and the baselines were tested on clean and noisy speech signals. We added babble noise to the clean speech recordings using the noise files from the NOISEX-92 [28] noise corpus at 20, 10, and 0 dB SNR. For the training and validation sets, only the clean samples were used.

To objectively compare with the DNN F0 baseline CREPE [19], we trained CREPE on this database using the output logarithmic frequency scale conversions described in Section 2 with the same 30 Hz to 400 Hz frequency range. To examine the effect of the model size of this baseline, we used 3 different sizes: full, medium, and tiny. We denote these models that were optimized for speech signals and trained on the PTDB-TUG database as Crepe-Opt (full), Crepe-Med-Opt (medium), Crepe-Tiny-Opt (tiny). We also evaluated the original CREPE (full) model (without Viterbi smoothing), which was trained on many datasets (RWC-Synth [9], MIR-1k [29], MDB-stem-synth [30], MedlyDB [31], Nsynth [32], and Bach10 [33]).

In addition to the modified MBSC (MBSCv2), we evaluated MBSC [15] and SWIPE' [12] since they have been shown to be highly effective DSP-based F0 estimation algorithms in the literature.

Moreover, for all the neural models, the modified MBSC's voicing detector was used in determining voiced/unvoiced (V/UV) decisons. After the V/UV decisions, F0 estimates were made and truncated between [60, 400 Hz] during inference.

For the evaluation metrics, we calculated mean absolute error (MAE) and gross pitch error (GPE) [11] using the voiced segments. These metrics were calculated with following equations:

$$MAE = \frac{\sum_{fr=0}^{N_{tot_v}} |f_{pred} - f_{ref}|}{N_{tot_v}} \,, \tag{7}$$

$$GPE = \frac{N_{err_v}}{N_{tot_v}} \cdot 100 \,, \tag{8}$$

where f_{pred} and f_{ref} denote predicted and reference frequencies, respectively. The MAE metric is calculated for each voiced frame fr, and the total number of voiced frames is denoted by N_{tot_v}. A threshold of 10% is for the GPE metric, and N_{err_v} indicate the number of voiced frames where the relative prediction errors are above this threshold.

4. Results and Discussion

4.1. Efficiency of the modified MBSC

We modified the original MBSC model proposed in [15] with the changes introduced in Section 2.1. In order to evaluate the efficiency of the modified version (MBSCv2) with respect to MBSC, latencies of the MBSCv2 and MBSC algorithms for different length input speech signals are shown in Figure 3. For these experiments, we used an Intel Core i7-4700HQ CPU with 2.40GHz clock speed. As can be seen from this figure, after the modifications, we achieve approximately 90% reduction in the latencies while maintaining similar performance as shown in Tables 1 and 2. Even though we achieve improvements on the efficiency of the SC extraction process, its latency is below real-time.

4.2. Performance comparison with DNN and DSP baselines

We evaluated the proposed FusedF0 algorithm, and compared it to both SOTA DNN and DSP baselines. For all comparisons, we used the PTDB-TUG test set described in Section 3.1.

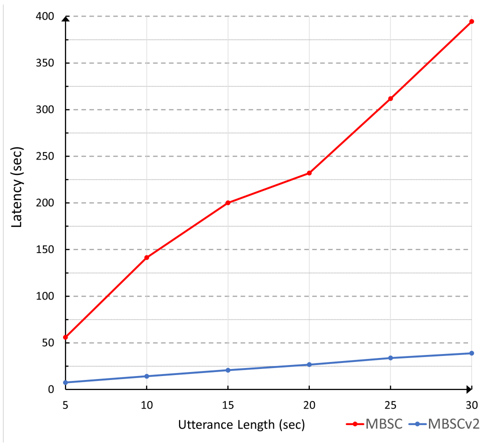

Figure 3: *Plot of latency (y-axis, in seconds) with respect to different utterance lengths (x-axis, in seconds) for the MBSC [15] and the modified MBSC (MBSCv2) algorithms. MBSCv2 is shown in blue, and MBSC is shown in red.*

Table 1 shows the MAEs and Table 2 shows the GPEs for the clean and noisy speech signals. FusedF0 outperforms the baselines in all cases. Although CREPE models trained on PTDB-TUG can outperform DSP baselines for the clean and the noisy speech signals at 20 dB and 10 dB SNR, FusedF0 significantly outperforms those models in terms of the MAE and GPE (%) metrics as shown in the tables. FusedF0's performance is even more impressive at 0 dB SNR compared to the best DNN model, Crepe-Med-Opt; FusedF0 has a GPE of 5.81 and an MAE of 5.66 compared to Crepe-Med-Opt, which has a GPE of 9.47 GPE and an MAE of 8.57 MAE. This reinforces our hypothesis that fusion of SC features can increase the robustness of DNN models. Additionally, in Table 2, MBSC and MBSCv2 slightly perform better than FusedF0 at 0 dB although FusedF0 outperforms these for the clean, 20 dB, and 10 dB speech in terms of GPE.

Furthermore, in Table 1, we observe that Crepe-Opt, Crepe-Med-Opt, and Crepe-Tiny-Opt outperform the original CREPE (full) [19] model, which was not trained on PTDB-TUG. This makes sense since DNN-based models usually require training on data with a similar distribution to the test data. Moreover, Crepe-Med-Opt model performs the best among the Crepe-based baselines. This could be attributed to the size of model parameters. Since the PTDB-TUG database is a relatively small dataset, using the 22M full Crepe model, Crepe-Opt, may not result in an ideal inductive bias, although validation losses are used for early stopping in the experiments. Similarly, Crepe-Tiny-Opt baseline can be considered too small for this database. We reported the average performance of the seen and unseen speakers in Tables 1 and 2. Individually: for the best performing DNN baseline, Crepe-Med-Opt, the GPEs (%) of the seen speakers are 0.79 (clean), 1.2 (20 dB), 1.38 (10 dB), and 9.05 (0 dB); while the GPEs of the unseen speakers are 3.99 (clean), 4.18 (20 dB), 4.46 (10 dB), and 9.89 (0 dB). For the proposed method, FusedF0, the GPEs of the seen speakers are 0.74 (clean), 0.83 (20 dB), 1.17 (10 dB), and 4.94 (0 dB); while the GPEs of the unseen speakers are 3.14 (clean), 3.07 (20 dB), 4.19 (10 dB), and 6.68 (0 dB). This demonstrates that FusedF0 can outperform the best-performing DNN baseline for both seen and unseen speakers, especially on noisy test conditions.

As described in Section 2, the SCEnc uses SC features, which are intermediate representations from the modified MBSC, and the RawEnc is based on Crepe-Tiny. Since FusedF0 outperforms the DNN and DSP baselines, including Crepe-based models and the MBSCv2 model, we conclude that fusion of SCs and raw waveform representations has a synergetic effect.

Table 1: *Mean absolute errors (MAEs) of the proposed FusedF0 model and baselines for clean and noisy (babble) speech at 20, 10, and 0 dB SNR.*

Model	**Params**	**Trained on**	**Clean**	**20dB**	**10dB**	**0db**
CREPE	22M	Many	6.65	8.05	8.87	13.3
Crepe-Opt	22M	PTDB-TUG	3.32	3.69	4.07	11.2
Crepe-Med-Opt	5.9M	PTDB-TUG	2.91	3.12	3.49	8.57
Crepe-Tiny-Opt	0.5M	PTDB-TUG	3.51	3.7	4.18	12.25
SWIPE'	-	-	9.72	10.4	12.4	19.9
MBSC	-	-	5.42	5.43	5.52	5.68
MBSCv2	-	-	5.22	5.48	5.78	5.91
FusedF0	5.9M	PTDB-TUG	**2.41**	**2.45**	**3.17**	**5.66**

Table 2: *Gross pitch errors (%) of the proposed FusedF0 model and baselines for clean and noisy (babble) speech at 20, 10, and 0 dB SNR.*

Model	**Params**	**Trained on**	**Clean**	**20dB**	**10dB**	**0dB**
CREPE	22M	Many	6.09	6.54	7.41	10.9
Crepe-Opt	22M	PTDB-TUG	2.75	3.19	3.56	12.1
Crepe-Med-Opt	5.9M	PTDB-TUG	2.39	2.69	2.92	9.47
Crepe-Tiny-Opt	0.5M	PTDB-TUG	2.84	3.22	3.54	13.02
SWIPE'	-	-	5.78	5.98	6.69	8.41
MBSC	-	-	5.37	5.47	5.54	**5.55**
MBSCv2	-	-	5.3	5.57	5.71	5.59
FusedF0	5.9M	PTDB-TUG	**1.94**	**1.95**	**2.68**	5.81

5. Conclusion

We propose a novel hybrid DSP and DNN F0 tracking architecture by fusing summary-correlogram representations and raw waveform representations of speech signals. We also modified the MBSC algorithm to make it faster with a 90% reduction in latency while maintaining F0 accuracy. We extracted the SCs used in FusedF0 in a similar fashion to the modified MBSC. We showed that the proposed FusedF0 outperforms strong DSP and DNN baselines such as MBSC and CREPE with a relatively small number of network parameters. Furthermore, we demostrated the robustness of FusedF0 on noisy speech signals, and showed that auxiliary SC features used in neural FusedF0 can improve F0 errors by up to 38% compared to the best performing DNN baseline.

We may further improve the architecture efficiency by dividing the input utterance into smaller chunks for SC calculation. Additionally, F0 annotation of a larger dataset like LibriSpeech [34] can be a significant future direction for a better F0 evaluation.

6. Acknowledgements

This work was supported in part by the NSF.

7. References

[1] J. Guglani and A. Mishra, “Automatic speech recognition system with pitch dependent features for punjabi language on kaldi toolkit,” *Applied Acoustics*, vol. 167, p. 107386, 2020.

[2] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T. Liu, “Fastspeech 2: Fast and high-quality end-to-end text to speech,” in *ICLR*. OpenReview.net, 2021.

[3] H. Schröter, T. Rosenkranz, A. N. Escalante-B., and A. K. Maier, “LACOPE: latency-constrained pitch estimation for speech enhancement,” in *INTERSPEECH*. ISCA, 2021, pp. 656–660.

[4] S. K. Sarangi, M. Sahidullah, and G. Saha, “Optimization of data-driven filterbank for automatic speaker verification,” *Digit. Signal Process.*, vol. 104, p. 102795, 2020.

[5] T. Nakano, K. Yoshii, Y. Wu, R. Nishikimi, K. W. E. Lin, and M. Goto, “Joint singing pitch estimation and voice separation based on a neural harmonic structure renderer,” in *WASPAA*. IEEE, 2019, pp. 160–164.

[6] D. Talkin and W. B. Kleijn, “A robust algorithm for pitch tracking (rapt),” *Speech coding and synthesis*, vol. 495, p. 518, 1995.

[7] P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, “A pitch extraction algorithm tuned for automatic speech recognition,” in *ICASSP*. IEEE, 2014, pp. 2494–2498.

[8] A. De Cheveigné and H. 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.

[9] M. Mauch and S. Dixon, “PYIN: A fundamental frequency estimator using probabilistic threshold distributions,” in *ICASSP*. IEEE, 2014, pp. 659–663.

[10] X. Sun, “Pitch determination and voice quality analysis using subharmonic-to-harmonic ratio,” in *ICASSP*. IEEE, 2002, pp. 333–336.

[11] T. Drugman and A. Alwan, “Joint robust voicing detection and pitch estimation based on residual harmonics,” in *INTERSPEECH*. ISCA, 2011, pp. 1973–1976.

[12] A. Camacho and J. G. Harris, “A sawtooth waveform inspired pitch estimator for speech and music,” *The Journal of the Acoustical Society of America*, vol. 124, no. 3, pp. 1638–1652, 2008.

[13] Y. Liu, P. Wu, A. W. Black, and G. K. Anumanchipalli, “A fast and accurate pitch estimation algorithm based on the pseudo wigner-ville distribution,” *CoRR*, vol. abs/2210.15272, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.15272

[14] G. Hu and D. L. Wang, “A tandem algorithm for pitch estimation and voiced speech segregation,” *IEEE Trans. Speech Audio Process.*, vol. 18, no. 8, pp. 2067–2079, 2010.

[15] L. N. Tan and A. Alwan, “Multi-band summary correlogram-based pitch detection for noisy speech,” *Speech Commun.*, vol. 55, no. 7-8, pp. 841–856, 2013.

[16] S. Kum and J. Nam, “Joint detection and classification of singing voice melody using convolutional recurrent neural networks,” *Applied Sciences*, vol. 9, no. 7, p. 1324, 2019.

[17] X. Sun, X. Liang, Q. He, B. Zhu, and Z. Ma, “GIO: A timbre-informed approach for pitch tracking in highly noisy environments,” in *ICMR ’22: International Conference on Multimedia Retrieval, Newark, NJ, USA, June 27 - 30, 2022*. ACM, 2022, pp. 480–488.

[18] B. Gfeller, C. H. Frank, D. Roblek, M. Sharifi, M. Tagliasacchi, and M. Velimirovic, “SPICE: self-supervised pitch estimation,” *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 1118–1128, 2020.

[19] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, “Crepe: A convolutional representation for pitch estimation,” in *ICASSP*. IEEE, 2018, pp. 161–165.

[20] M. Ravanelli and Y. Bengio, “Speaker recognition from raw waveform with sincnet,” in *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018*. IEEE, 2018, pp. 1021–1028.

[21] G. Pirker, M. Wohlmayr, S. Petrik, and F. Pernkopf, “A pitch tracking corpus with evaluation on multipitch tracking scenario,” in *INTERSPEECH*. ISCA, 2011, pp. 1509–1512.

[22] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. Omnipress, 2010, pp. 807–814.

[23] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, vol. 37. JMLR.org, 2015, pp. 448–456.

[24] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[25] S. Singh, R. Wang, and Y. Qiu, “Deepf0: End-to-end fundamental frequency estimation for music and speech signals,” in *ICASSP*. IEEE, 2021, pp. 61–65.

[26] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, “Deep salience representations for f0 estimation in polyphonic music,” in *ISMIR*, 2017, pp. 63–70.

[27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.

[28] A. Varga and H. J. M. Steeneken, “Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems,” *Speech Commun.*, vol. 12, no. 3, pp. 247–251, 1993.

[29] C. Hsu and J. R. Jang, “On the improvement of singing voice separation for monaural recordings using the MIR-1K dataset,” *IEEE Trans. Speech Audio Process.*, vol. 18, no. 2, pp. 310–319, 2010.

[30] J. Salamon, R. M. Bittner, J. Bonada, J. J. Bosch, E. Gómez Gutiérrez, and J. P. Bello, “An analysis/synthesis framework for automatic f0 annotation of multitrack datasets,” in *Hu X, Cunningham SJ, Turnbull D, Duan Z. ISMIR 2017 Proceedings of the 18th International Society for Music Information Retrieval Conference; 2017 Oct 23-27; Suzhou, China.[Suzhou]: ISMIR; 2017.* International Society for Music Information Retrieval (ISMIR), 2017.

[31] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, “Medleydb: A multitrack dataset for annotation-intensive MIR research,” in *ISMIR*, 2014, pp. 155–160.

[32] J. H. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, “Neural audio synthesis of musical notes with wavenet autoencoders,” in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 1068–1077.

[33] Z. Duan, B. Pardo, and C. Zhang, “Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions,” *IEEE Trans. Speech Audio Process.*, vol. 18, no. 8, pp. 2121–2133, 2010.

[34] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An ASR corpus based on public domain audio books,” in *ICASSP*. IEEE, 2015, pp. 5206–5210.