Fully-Convolutional Network for Pitch Estimation of Speech Signals

Luc Ardaillon, Axel Roebel

Fully-Convolutional Network for Pitch Estimation of Speech Signals

Luc Ardaillon, Axel Roebel

UMR 9912 STMS (IRCAM / Sorbonne University / CNRS), Paris, France

luc.ardaillon@ircam.fr, axel.roebel@ircam.fr

Abstract

The estimation of fundamental frequency (F_0) from audio is a necessary step in many speech processing tasks such as speech synthesis, that require to accurately analyze big datasets, or real-time voice transformations, that require low computation times. New approaches using neural networks have been recently proposed for F_0 estimation, outperforming previous approaches in terms of accuracy. The work presented here aims at bringing some more improvements over such CNN-based state-of-the-art approaches, especially when targeting speech data. More specifically, we first propose to use the recent *PaN* speech synthesis engine in order to generate a high-quality speech database with a reliable ground truth F_0 annotation. Then, we propose 3 variants of a new fully-convolutional network (FCN) architecture that are shown to perform better than other similar data-driven methods, with a significantly reduced computational load making them more suitable for real-time purposes.

Index Terms: speech analysis, F_0 estimation, CNN

1. Introduction

Fundamental frequency (F_0) estimation is a necessary step in many speech processing tasks such as TTS or singing synthesis, for which it is often necessary to analyze big databases. However, analysis errors are likely to induce undesirable artifacts and manual correction is difficult for such big datasets. Until recently, all approaches used to be based on digital signal processing and heuristics, including methods based on the auto-correlation function like YIN [1] or PYIN [2], or template matching with the spectrum of a sawtooth signal as in SWIPE [3], among many others.

But recently, new data-driven approaches using various kinds of neural networks have been proposed for monophonic F_0 estimation [4], [5], [6], [7], and for multi-pitch [8] or main melody estimation [9], [10], outperforming previous results. A recent trend, since the success of the wavenet model [11], is towards end-to-end systems using the raw signal's waveform as input to the network. Following this trend, several neural networks for monophonic F_0 analysis have been proposed, like the CREPE model [6], or the ones proposed in [4] and [7], using the signal's waveform as input to a CNN, an MLP, or a RNN, respectively. A particular advantage of such end-to-end approaches is that once the model is trained, no parametrisation is required from the user.

A particular use case for F_0 estimation is in real-time applications and audio plugins for voice transformations [12] requiring low computation times. But recent approaches like [6] are too computationally heavy, and some more improvements are thus still necessary in order to make such methods compatible with real-time constraints, without reducing the accuracy.

Building up on previous studies, this paper aims at bringing some improvements over data-driven state-of-the-art approaches for monophonic pitch estimation of speech sound, both in terms of accuracy and efficiency. Following the methodology described in [6], summarized in section 2, we propose a new fully-convolutional neural network architecture that will be described in Section 3. Then, in order to train this model we propose to use the recent *PaN* speech synthesis engine [13, Section 3.5.2] to build a high-quality speech database with a reliable ground truth, as detailed in Section 4. Finally, we conducted some evaluations to compare the performances of our models to that of the CREPE and SWIPE algorithms, both in terms of accuracy and computational efficiency, that will be detailed in Section 5.

2. Method

For our purposes, we adopt a similar approach to the one described in [6] for CREPE, with a few adaptations. In this approach, the F_0 estimation is seen as a classification task, using a CNN that takes a raw waveform as input and outputs a vector of probabilities of the F_0 to belong to each possible output pitch classes. In our case, those 486 pitch classes correspond to a division of the [30-1000] Hz range into steps of 12.5 cents, assuming that such a range covers all possible pitch values for vocal sounds (including soprano singing and para-linguistic sounds like baby cries). Note that 30Hz is very low for voice, but might occur occasionally (e.g. in fry mode). The resolution has been increased compared to [6] (12,5 cents instead of 25), assuming that it might help to get more accurate results.

For training, the target vector is gaussian-blurred according to equation 1, with a standard deviation of 25 cents, such that the activation values y_i of the bins of pitch c_i (in cents) surrounding a ground truth frequency c_{true} decay accordingly (similarly to [6]).

$$y_i = \exp(-\frac{(c_i - c_{true})^2}{2 \cdot 25^2}) \tag{1}$$

The network is then trained to minimize the binary cross-entropy between the target and predicted vectors.

From the predicted vector $\hat{y}$, the resulting pitch estimate, in cents, is then computed following equation 2, similarly to what is done in CREPE[1].

$$\hat{c} = \frac{\sum_{i=I-4}^{I+4} \hat{y}_i c_i}{\sum_{i=I-4}^{I+4} \hat{y}_i} \quad , \quad I = argmax_i(\hat{y}_i) \tag{2}$$

This value can then be converted back from cents to Hz to obtain the final F_0.

3. Network architecture

The CREPE architecture proposed in [6] is a rather simple CNN architecture composed of 6 convolutional layers followed each by max pooling and relu activations, and a final dense layer with

[1] https://github.com/marl/crepe

sigmoid activations that produces the prediction vector. However, this architecture has several drawbacks which will be discussed in the following.

A first drawback is related to the zero-padding applied to the input of each convolutional layer (mode "same" in keras), which is necessary in order to keep the same size between the input and output of the layer. This may appear as a minor issue when using small filters compared to the size of the input (as is often the case in image processing for instance). But in the CREPE architecture, the filters used are quite long (e.g. 512 samples in the first layer for an input of size 1024), and this border effect is thus not negligible. In particular, in the layers 3 to 6 of the CREPE architecture, the size of the filters (64) is superior to the temporal dimension of their inputs (8 to 64, depending on the layer). Thus, most of the convolution operations in these layers results in multiplications with zeros, from which one would not expect the model to benefit, and which also raises the number of computations. To overcome this problem, a solution is to use only 'valid' convolutions, for which no padding is applied. However, in this case the output of a convolution is smaller than its input, and the network architecture needs to be adapted to take this into account.

A second limitation is due to the final dense layer of CREPE which can only accept fixed-sized inputs, which implies to do the prediction on a frame basis. For this purpose, the input sound is split into overlapping frames that are fed individually to the network which predicts one F_0 value for each frame (the degree of overlap depending on the expected time resolution of the produced F_0 curve). Each convolution is thus computed several times on the overlapping parts of each succeeding frames sharing the same values, which implies some redundancy in the computation.

Instead of doing the prediction on a frame basis, we thus propose here to use a Fully-Convolutional Network (FCN) architecture. For this purpose, it is possible to replace the final dense layer by an equivalent convolutional one whose length is equal to the temporal dimension of the input tensor, and whose number of filters is equal to the size of the expected ouput vector (in our case 486) [14]. Since a convolutional layer does not require a fixed-sized input, this configuration allows running the convolutions only one time on the whole input signal instead of frame-wise, thereby saving a lot of calculation. Then, the network will not output a single vector, but a 3D tensor containing the output prediction vectors at each temporal step, with the pitch classes arranged along the depth dimension (as illustrated in figure 1).

Beside this fully-convolutional approach, one way to reduce the computation load of a CNN is to reduce the number of parameters (i.e. the number of layers, the number of filters in each layer, or the size of those filters). In the implementation of CREPE, it is suggested to use less filters to reduce the size of the model. But it may also be beneficial to try reducing the size of the filters and number of layers.
In the CREPE architecture, the filters in the first layer are particularly long (512), such that they cover at least 1 full period of the signal (1 period of a 30Hz sinusoid at 16 kHz is 500 samples). However, there is no evidence that this is necessary, and the network might be able to learn as well using smaller filters. In addition to reducing the number of filters in each layer, we thus also propose to reduce their size.

Finally, the CREPE model is trained on 16 kHz audio. However, only a few harmonics are necessary to infer a pitch. Reducing the sampling rate may also help to reduce the necessary computation, by getting read of unnecessary details in the signal. As we limit the target pitch range to [30-1000] Hz, we chose to reduce the sampling rate to 8 kHz, which still allows to preserve at least the 4 first harmonics (including the fundamental).

Based on all those considerations, we propose here a new fully-convolutional architecture, described in figure 1, that incorporates all the previously-exposed proposals. The proposed model is composed of 7 convolutional layers. Max pooling is included only after the first 3 convolutional layers (including more max pooling layers after the 3rd one would require to use a much bigger input size, which is undesirable for latency). Each layer is followed by batch normalization [15], and its output is passed through relu activations, except for the last layer for which sigmoid activations are used. The number of filters in each layer and their sizes are detailed in figure 1. All convolutional layers are applied with a stride of 1. The minimal input to the network to obtain 1 F_0 value is a 1953-samples excerpt from an audio signal sampled at 8 kHz. Note that because no padding is applied before the convolutions, the temporal dimension is reduced after each convolutional layer even without max pooling. The relation between the output size l_{out} and input size l_{in} of a convolutional layer and the size of the filters l_{fil} is $l_{out} = l_{in} - l_{fil} + 1$. The choice of the input size of the network has thus been determined according to this relation, after the network architecture (number and positions of the convolutional and max pooling layers, and size of the filters) has been fixed. Because of the size reduction implied by the 3 pooling layers, the time step between each output prediction is equal to 8 samples ($= 2^3$). The temporal resolution of the prediction (1ms) is thus greatly increased compared to a frame-based prediction.

However, this model imposes a minimum latency of a half input size duration, i.e. $\frac{1953/2}{8000} \approx 0.122s$, which is quite big for real-time purposes. In order to reduce this latency, we propose two alternative models based on the same architecture. The first proposed alternative is to remove the 2^{nd} convolutional layer in figure 1 (including the max pooling). This alternative model requires an input of size ≥ 929, which reduces the latency to $\frac{929/2}{8000} \approx 0.058s$. The second alternative is to keep the same architecture as in figure 1, but reducing the size of the filters in all layers to 32. In this case, the minimum input size is 993 (latency of $\frac{993/2}{8000} \approx 0.062s$). We call those 3 models respectively *FCN-1953*, *FCN-993*, and *FCN-929* according to their respective input sizes.

With the proposed architectures, the number of parameters has been reduced from about $22.2 \cdot 10^6$ for CREPE to about $12.3 \cdot 10^6$ for FCN-1953 and FCN-929, and $6.7 \cdot 10^6$ for FCN-993, while getting similar or better results in our evaluations, as will be described in section 5. Moreover, the proposed fully-convolutional approach allows to significantly reduce the computation time. The proposed architectures have been implemented in keras, and a python implementation with pre-trained models are made available online[2].

4. Database

4.1. Data generation

For the purpose of proper training and objective evaluations, we need a database with a perfect ground truth F_0 annotation. Though many databases of voice recordings are available, obtaining a reliable F_0 ground truth is less straightforward, as ex-

[2]https://github.com/ardaillon/FCN-f0

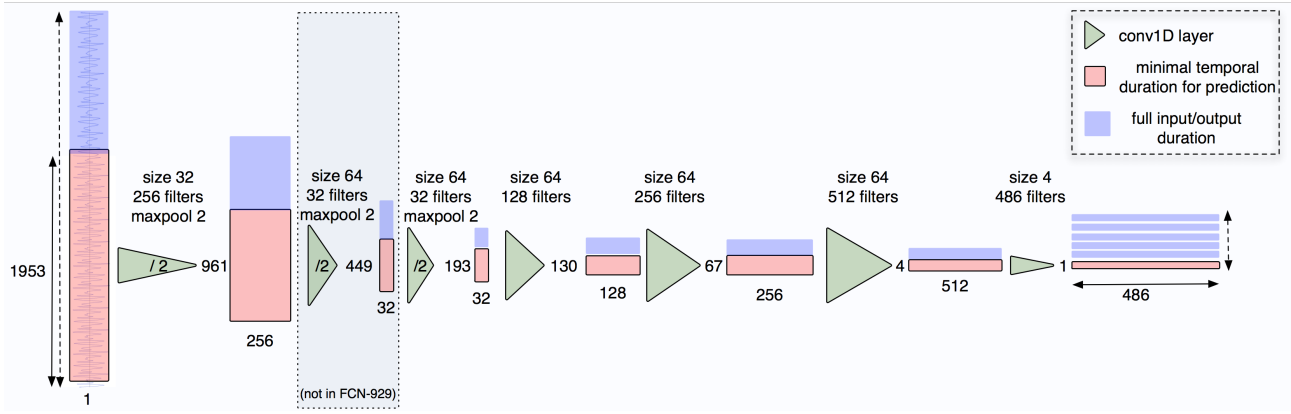

Figure 1: *Architecture of the proposed fully-convolutional network FCN-1953. All convolutions are valid with stride 1. For the alternative model FCN-929, the 2nd layer is removed and the input size is 929. For the alternative model FCN-993, all filters are of size 32 except in the last layer, and the input size is 993.*

isting estimators are likely to make errors, and manual correction is not precise and not applicable to big databases.

For this purpose, a first approach, proposed in [7], [16], consists in using a specific database, such as the PTDB-TUG database [17], that includes laryngograph recordings on which one of the already available F_0 estimators is applied to obtain a good estimation of the ground truth. (However, our experience with one such database revealed that the laryngograph recordings where sometimes too corrupted to obtain a reliable ground truth).

A second solution, proposed in [6], [18], consists in using synthesis to have a perfect control over the F_0 of the resulting signals. In this case, a database of real recordings is first analyzed using a pre-existing estimator to get annotations that will be used as a ground truth, but that are likely to contain errors. Then, instead of correcting these annotations, the sound is re-synthesized based on this ground truth F_0 annotation, such that the resynthesis perfectly matches the corresponding F_0 track, including the potential errors.

Since proper laryngograph recordings are rarely available and given that the synthesis approach allows more control over the ground truth, we chose this second approach. However, the synthesis method used in [6] is purely harmonic, using a bank of oscillators, and thus doesn't include any unvoiced component. Though this approach may be acceptable for some musical instruments, it seems oversimplistic for vocal sounds, and is likely to result into too homogeneous and unrealistic data that doesn't account well for all the complexity and variability of vocal sounds.

Many available vocoders allow to synthesize speech with a good quality. It seems thus benefitial to use one of them in order to improve the quality of the dataset for speech. We propose here to use the recent PaN vocoder for this purpose [13, Section 3.5.2]. Basically, the PaN model uses the target F_0 values to generate a sequence of pulses based on the LF model of glottal source [19]. In order to model various voice qualities, a constant random Rd value has been chosen in the range [0.3-2.7] for each synthesized file. Then, the vocal tract filter is applied on the pulses. Finally, the unvoiced component, separated from the original sound, is added to the voiced source to generate the complete signal. Since the separation of the voiced and unvoiced components is not perfect, some sinusoids may sometimes remain in the unvoiced residual. In order to avoid this problem, the phases in the residual signal are randomized.

4.2. Dataset description

To create our own dataset, we first merged the BREF [20] and TIMIT [21] datasets. Then, the CREPE algorithm [6] with the provided pre-trained model has been used to estimate the F_0 accross the whole database. Finally, each sound has been re-synthesized using the PaN synthesis engine. This ends up into a total of 11616 short sentences spoken by many male and female voices in English and French languages. Then, in order to better span the targeted range of F_0 values ([30-1000] Hz), the same dataset is also generated one octave lower and one octave higher. Files that contain F_0 values outside of the target range are excluded from the training and evaluation data. The sounds have been first generated at a 16 kHz sampling rate, and then subsampled to 8 kHz, as required by our models.

5. Evaluation

5.1. Methodology

For evaluation purposes, we split our database into 3 subsets for training, validation and evaluation (with the proportions 60/20/20). Since the same synthesis are replicated at 3 different sets of transpositions in the database, the same split is applied to them, so that the content of the samples present in the 3 subsets are completely different, independently of their pitch, and a similar pitch distribution is selected for the 3 subsets.

With this methodology, we compare the results of our models against that of the CREPE and the SWIPE algorithms. For the CREPE model, we evaluate both the pre-trained model provided by the authors ("CREPE"), and another version re-trained in similar conditions than our models ("CREPE-speech"). Note that the CREPE and SWIPE algorithm are trained and evaluated on the 16 kHz database, while 8 kHz is used for our models.
In [6], the CREPE algorithm has been shown to outperform both the SWIPE and pYIN algorithms. However, this evaluation has only been conducted on synthetic data. In order to attest that the results are not biased by the synthetic quality of our database, we thus also evaluate the results on a manually-annotated subset of 100 real recordings taken from the test split.

The results are evaluated in terms of Raw Pitch Accuracy (RPA) using the mir_eval [22] python package, with 25, 50, and

Table 1: *Raw Pitch Accuracy values (mean and std) on the synthesized and manually corrected databases*

	FCN-1953	**FCN-993**	**FCN-929**	**CREPE**	**CREPE-speech**	**SWIPE**
PaN-synth (25 cents)	93.62 ± 3.34%	**94.31 ± 3.15%**	93.50 ± 3.43%	77.62 ± 9.31%	86.92 ± 8.28%	84.56 ± 11.68%
PaN-synth (50 cents)	98.37 ± 1.62%	**98.53 ± 1.54%**	98.27 ± 1.73%	91.23 ± 6.0%	97.27 ± 2.09%	93.1 ± 7.26%
PaN-synth (200 cents)	**99.81 ± 0.64%**	99.79 ± 0.65%	99.77 ± 0.73%	95.65 ± 5.17%	99.25 ± 1.07%	97.51 ± 4.9%
manual (50 cents)	88.32 ± 6.33%	88.57 ± 5.77%	**88.88 ± 5.73%**	87.03 ± 7.35%	88.45 ± 5.70%	85.93 ± 7.62%
manual (200 cents)	97.35 ± 3.02%	97.31 ± 2.56%	**97.36 ± 2.51%**	92.57 ± 5.22%	96.63 ± 2.91%	95.03 ± 4.04%

Table 2: *Minimum latency and mean computation time over 100 trials for a 2.51s sound, on a GPU and single core CPU*

	FCN-1953	**FCN-993**	**FCN-929**	**CREPE**	**SWIPE**
latency	0.122s	0.062s	0.058s	**0.032s**	0.128s
GPU	0.016s	**0.010**s	0.021s	0.092s	X
CPU	1.65s	0.89s	3.34s	14.79s	**0.63s**

200 cents thresholds. Note that 50 cents corresponds to about only 3Hz for a F_0 of 100Hz, while the width of a sinusoid on the spectrogram display used for manual correction is $\geq$ 15Hz.

5.2. Training procedure

All models are trained using mini-batch gradient descent with the Adam optimizer [23], with mini-batches composed of 32 examples randomly selected from the training set. An initial learning rate of 0.0002 is used, and a reduction by a factor 0.75 is applied when the validation loss doesn't decrease for 5 epochs, where an epoch consists of 500 batches, with a minimum value 0.0000025. The training is stopped when the validation accuracy has not improved for 32 epochs. Note that while an FCN can accept an input of arbitrary length, the input segments used for the training stage remain of fixed size, such that the temporal dimension of the output tensor is always equal to 1 during training. The input segments are normalized by their mean and variance prior to be fed to the network for training. (However, during inference, since the size of the input is not fixed, each sample is normalized individually by the mean and variance computed over a sliding window). Unlike in [6], we did not use dropout for training our models.

5.3. Results

The obtained RPA values (with mean and variance) are shown in table 1, both for the synthetic speech test split ("PaN-synth") and the manually-corrected subset of real recordings ("manual"). As can be seen, our models always perform better than the SWIPE and CREPE algorithms (except for the manual 50 cents case where the variance is the largest). The difference is particularly important with the 25 cents threshold, wich is probably partly due to the adapted target range and finer output resolution compared to CREPE. Regarding CREPE, the important difference of results between the default model and the one trained on our database ("CREPE-speech") outlines the importance of using an appropriate database according to the targeted data for training. The 3 variants of the proposed architecture gave relatively similar results in all cases.

Besides the accuracy, we also compared the computation time for the different approaches. The mean computation time over 100 analysis of a sound of duration 2.51s for each algorithm, both on a GPU and on a single core CPU, are reported in table 2, along with the minimum latency (half of input length divided by sampling rate. The biggest window size used by the SWIPE algorithm is 4096). We can see that both our model FCN-1953 and FCN-993 could be applied in real-time even on a single core CPU and FCN-993 allows reducing the computation time by a factor 16 compared to the CREPE algorithm. However, although the FCN-929 model has the lowest latency, it is also about 2 times slower than FCN-1953. This is due to the removal of the 2nd max pooling layer in this case, which implies a finer output time resolution of 4 samples instead of 8, and thus about twice more computations. While both models obtain similar accuracy, it is thus more efficient to use smaller filters rather than removing one layer of the network to reduce the latency. (However, complementary experiments showed that results start to degrade when the filters get too small.)

About the manual database, one can observe that the results are much lower than on the synthetic one for all algorithms (including swipe which is not data-driven), when using a threshold of 50 cents for the evaluation. However, they are already much better when evaluating with a 200 cents threshold. This may be explained by the inaccuracy of the manual correction which doesn't allow obtaining a very precise ground truth. Another possible explanation for those differences is that some files contain rough voice quality which is not present in the training database, and for which the ground truth F_0 is ambiguous.
Finally, those results reveal that our FCN models (along with "CREPE-speech") tend to give much more consistent result across all the evaluation data, compared to SWIPE that is better for high-pitched sounds than for low-pitched ones, according to complementary analysis.

In overall, the best alternative in terms of accuracy, speed and latency would be the FCN-993 model.

6. Conclusion

We presented a new Fully-Convolutional Network (FCN) architecture for monophonic F_0 estimation. An evaluation showed that our model ouperformed the CREPE and SWIPE approaches on synthetic and real speech data, in terms of accuracy. Furthermore, thanks to the fully-convolutional approach and the reduction of the model size, the computation time of our proposed models is significantly reduced compared to CREPE, making it applicable in real-time. However, there is still room for improvements to further reduce the latency. Further investigations would be necessary in order to evaluate the robustness of our model to different types of noise, and compare its results with other state-of-the-art models like [7]. While a voiced/unvoiced decision could be made by setting a threshold on the maximum value of the output vector, this approach did not work robustly in our experiments. It would thus be better if the network could output directly a voicing decision additionally to the f0 values, which should be investigated in future work. Finally, another remaining problem is in the case of rough sounds containing subharmonics, where the ground truth F_0 is ambiguous. One possible approach to avoid this ambiguity would be to detect the pulse positions instead of the F_0 directly.

7. References

[1] 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.

[2] M. Mauch and S. Dixon, "PYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2014, pp. 659–663.

[3] 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. [Online]. Available: http://asa.scitation.org/doi/10.1121/1.2951592

[4] P. Verma and R. W. Schafer, "Frequency estimation from waveforms using multi-layered neural networks," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 08-12-Sept, pp. 2165–2169, 2016.

[5] K. Han and D. L. Wang, "Neural network based pitch tracking in very noisy speech," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 22, no. 12, pp. 2158–2168, 2014. [Online]. Available: http://web.cse.ohio-state.edu/~wang.77/papers/Han-Wang.taslp14.pdf

[6] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A Convolutional Representation for Pitch Estimation," 2018. [Online]. Available: http://arxiv.org/abs/1802.06182

[7] A. Kato and T. Kinnunen, "Waveform to Single Sinusoid Regression to Estimate the F0 Contour from Noisy Speech Using Recurrent Deep Neural Networks," 2018. [Online]. Available: http://arxiv.org/abs/1807.00752

[8] R. M. Bittner, B. Mcfee, J. Salamon, P. Li, and J. P. Bello, "Deep Salience Representations for F0 Estimation in Polyphonic Music," *Ismir*, pp. 23–27, 2017. [Online]. Available: https://bmcfee.github.io/papers/ismir2017_salience.pdf

[9] D. Basaran, S. Essid, and G. Peeters, "Main melody extraction with source-filter nmf and crnn," in *ISMIR*, 2018, pp. 82–89. [Online]. Available: http://ismir2018.ircam.fr/doc/pdfs/273_Paper.pdf

[10] G. Doras, P. Esling, and G. Peeters, "On the use of u-net for dominant melody estimation in polyphonic music," in *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*. IEEE, 2019, pp. 66–70.

[11] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: a generative model for raw audio," *arXiv*, 2016.

[12] Flux, "Ircam Trax v3 audio plugin," https://www.flux.audio/project/ircam-trax-v3/.

[13] L. Ardaillon, "Synthesis and expressive transformation of singing voice," Ph.D. dissertation, EDITE; UPMC-Paris 6 Sorbonne Universités, 2017.

[14] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation."

[15] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[16] A. Kato and T. Kinnunen, "A Regression Model of Recurrent Deep Neural Networks for Noise Robust Estimation of the Fundamental Frequency Contour of Speech," 2018. [Online]. Available: http://arxiv.org/abs/1805.02958

[17] G. Pirker, M. Wohlmayr, S. Petrik, and F. Pernkopf, "A Pitch Tracking Corpus with Evaluation on Multipitch Tracking Scenario," no. August, pp. 1509–1512, 2011.

[18] J. Salamon, R. M. Bittner, J. Bonada, J. J. Bosch, E. Gomez, and juan pablo Bello, "An Analysis/Synthesis Framework for Automatic F0 Annotation of Multitrack Datasets," *ISMIR, International Society for Music Information Retrieval Conference*, pp. 0–7, 2017. [Online]. Available: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/164_Paper.pdf

[19] G. Fant, J. Liljencrants, and Q. Lin, "A four-parameter model of glottal flow," *STL-QPSR*, vol. 26, no. 4, pp. 1–13, 1985.

[20] J. L. Gauvain, L. F. Lamel, and M. Eskenazi, "Design Considerations and Text Selection for BREF, a large French Read-Speech Corpus," *1st International Conference on Spoken Language Processing, ICSLP*, no. January 2013, pp. 1097–1100, 1990. [Online]. Available: http://www.limsi.fr/~lamel/kobe90.pdf

[21] V. Zue, S. Seneff, and J. Glass, "Speech Database Development at MIT : TIMIT and Beyond," vol. 9, pp. 351–356, 1990.

[22] C. Raffel, B. Mcfee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "mir_eval: A Transparent Implementation of Common MIR Metrics," *Proc. of the 15th International Society for Music Information Retrieval Conference*, pp. 367–372, 2014. [Online]. Available: http://www.ee.columbia.edu/~dpwe/pubs/RaffMHS14-mireval.pdf

[23] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," pp. 1–15, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980