Analysing Deep Learning-Spectral Envelope Prediction Methods for Singing Synthesis

Frederik Bous
UMR STMS - IRCAM
CNRS, Sorbonne University
Paris, France
frederik.bous@ircam.fr

Axel Roebel
UMR STMS - IRCAM
CNRS, Sorbonne University
Paris, France
axel.roebel@ircam.fr

***Abstract*—We conduct an investigation on various hyper-parameters regarding neural networks used to generate spectral envelopes for singing synthesis. Two perceptive tests, where the first compares two models directly and the other ranks models with a mean opinion score, are performed. With these tests we show that when learning to predict spectral envelopes, 2d-convolutions are superior over previously proposed 1d-convolutions and that predicting multiple frames in an iterated fashion during training is superior over injecting noise to the input data. An experimental investigation whether learning to predict a probability distribution vs. single samples was performed but turned out to be inconclusive. A network architecture is proposed that incorporates the improvements which we found to be useful and we show in our experiments that this network produces better results than other stat-of-the-art methods.**

***Index Terms*—Singing synthesis, spectral envelopes, deep learning**

I. Introduction

Singing synthesis is concerned with generating audio that sounds like a human singing voice from a musical description such as midi or sheet music. We observe that the human voice has one of the greatest varieties of possible sounds and the human ear is trained to distinguish the smallest differences in human voices. Compared with acoustic instruments, singing not only incorporates melody and articulation, but also text. Compared with speech, singing requires special treatment of the fundamental frequency f_0 as well as timing, which must be aligned to match melody and rhythm respectively. However, due to its similarity to speech synthesis, more precisely text-to-speech (tts), many methods from tts may also be applied to singing synthesis. For years concatenative methods [1], [2] dominated both fields [3]–[6]. While these techniques yield fairly decent results, they are inflexible and the underlying parametric speech models usually treat all parameters independently, which poses difficulties with coherency of the parameters. However, today fast computation on gpus and large databases allow us treating all parameters at once in a single model with neural networks and they have already been successfully applied to text-to-speech applications in the past years:

The system of [7] uses recurrent neural networks to model the statistic properties needed for their concatenative synthesis. WaveNet [8] goes further and models the raw audio, rather than concatenating existing audio or using a vocoder. Shortly after that, end-to-end systems like Tacotron [9] and Deep Voice [10] were developed which create raw audio from input on phoneme level or even character level. The authors of [11] used the architecture of [8] to learn input data for a parametric singing synthesizer.

While WaveNet processes data that is inherently one dimensional (i. e., raw audio), spectral envelopes are generated in [11]. There the input data is thus multidimensional, the authors use 60 parameters to represent the spectral envelopes. This changes the nature of the data and former strong points of WaveNet may loose importance whereas some weaknesses may have a more significant impact. This has motivated our investigation into alternative network topologies and training strategies which finally has lead to an improved synthesis model.

We found that, contradicting the assumptions in [11], 2d-convolutions yield better perceived audio while reducing the required number of trainable parameters. We also observe that learning by predicting multiple frames successively is superior to learning with additive noise at the input. A clear benefit from predicting parametric distributions rather than samples explicitly could not be found. As a result we propose our own network for predicting spectral envelopes.

The paper is structured as follows: we will first introduce our network in section II and discuss its differences to existing systems in section III. The experimental setup will be explained in section IV and we present the results from our perceptive test in section V

II. Proposed Network Architecture

We aim to build a system for composers and professionals that wish to use synthetic singing voice in their compositions and applications. While making application easy by automating obvious decisions, there should be as much ability to tweak all kinds of parameters as possible. Therefore end-to-end systems like Tacotron 2 [9], where only the raw text is used as input and raw audio comes out as output and all other properties are only implicitly included in the model, if at all, do not fit our needs.

The role of the fundamental frequency f_0 is a very different in singing synthesis as compared to speech synthesis. In

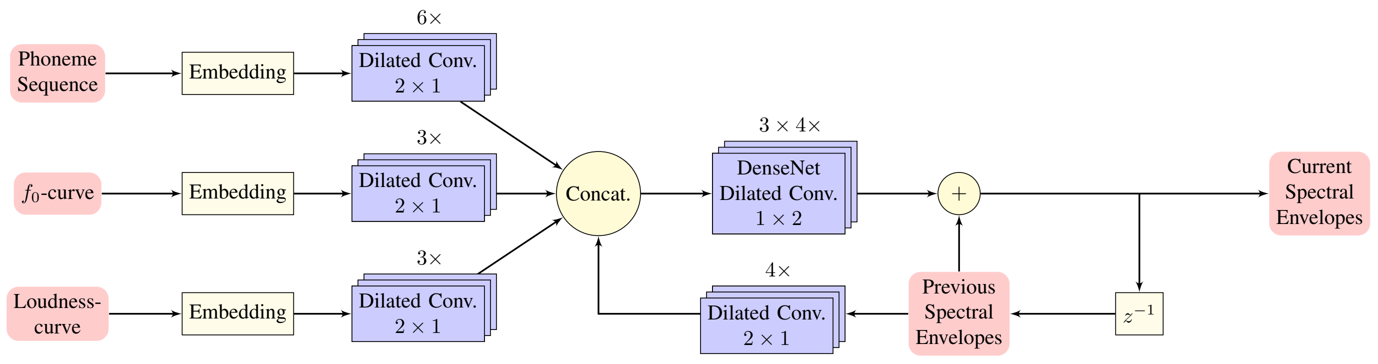

Fig. 1. Schematic layout of the network. Blue stacks denote stacks of layers, where $3 \times 4\times$ means *three stacks of four layers each*, $6\times$ means *one stack with six layers*. In each stack the dilation rate is doubled in each layer starting with a dilation rate of 1. The block z^{-1} denotes a delay of one time step. Concatenation is done in the feature dimension.

speech, the f_0-curve follows only few constraints but needs to be coherent with the other parameters. Learning it implicitly makes sense for end-to-end text-to-speech application as it does not carry much information, but coherence with other parameters is important. In singing, the f_0-curve is the parameter responsible for carrying the melody but it carries also musical style and emotion [12]. It is therefore important to model it explicitly, which can be achieved with, e. g., B-splines [13], to still be able to tweak it by hand to fit the needs of the particular application.

While systems like WaveNet [8] operate on raw audio, these architectures require very large datasets, which are currently not available for singing voice. This is for one due to less funding and for the other that recording proper singing requires even more work, as professional singers can not sing as long in one session as a professional speaker could speak.

In our application we use a vocoder model for singing voice synthesis. We use an improved version of the SVLN vocoder [14], [15], that is used to create singing voice from the modified parametric representation of the singing signal stored in a singing voice database. In this context we aim to use a neural network to provide spectral envelopes that fit the local context (phoneme, F0, loudness) to the SVLN vocoder, that would then be responsible to generate the corresponding source signal.

A. Input data

Training data has been obtained from our own dataset of singing voice, which was originally created for a concatenative singing synthesizer. It consists of about 90 minutes of singing from a tenor voice. From this database we extract the spectral envelopes as well as the phoneme sequences, f_0-curves and loudness-curves as control parameters.

Phonemes were aligned with [16] and manually adjusted. Spectral envelopes are extracted from the audio with an improved version of the *true envelope* method [17]. The loudness curve is extracted by using a very simple implementation of the loudness model of Glasberg et al. [18], the f_0-curve is extracted by the pitch estimator of [19]. All data is given with a sampling rate of 200Hz (5ms step size).

The spectral envelopes are represented by 60 log Mel-frequency spectral coefficients [20], such that we can treat the spectral envelope sequence as a 2d spectrogram with Mel-spaced frequency bins. To obtain 60 bins but to keep a good resolution in the lower bins, we consider only frequencies up to 8kHz.

B. Spectral Envelope Generation

The spectral envelopes are generated by a recursive convolutional neural network. The network predicts one spectral envelope at a time by using the previous spectral envelopes as well as a window of the phoneme, f_0 and loudness values of previous, current and next time steps. We thus let the network see a large context of control parameters from both future and past and thus allow it to create its own encoding.

The architecture is inspired by [9] and [10]. These systems use a neural network to create Mel-spectra, which are then converted to raw audio by either the Griffin-Lim algorithm or a vocoder. However, since we model the f_0 curve separately and do not encode it in the output, we can use a much simpler model.

Since all input parameters are given with the same rate, there is no need for attention. Only an encoding network of the control parameters, a pre-net for the previous spectral envelopes and a frame reconstruction network remain. In all parts we use blocks of dilated convolutions with exponentially growing dilation rate [8], but additionally to dilated convolutions in time direction (as used in WaveNet and [11] and which we shall call (2×1)-dilated convolutions) we also use dilated convolution in the frequency direction ((1×2)-dilated convolutions).

We can summarise the architecture as follows (cf. Fig. 1):

- Input the envelopes from the last 2^{n_e} time steps and 2^{n_i} phoneme-values, f_0-values and loudness-values (where n_i is different for each parameter) from a window of previous, current and next time steps around the current time step. The phonemes are mapped to 60 frequency bins by an embedding layer, f_0-values and loudness-values are mapped to 60 frequencies by outer products with trainable 60 dimensional vectors.

- For each input parameter use a stack of n_i dilated convolution layers of (2×1)-convolutions and dilation rate $(2^l, 1)$ in layer l (starting with $l = 0$). No zero padding is done here. The convolution for the envelopes is causal, the other convolutions are non-causal (with a symmetric receptive field of past and future values).
- After the time-convolutions, the time-dimension is now one, while the frequency dimension remained 60 for each input parameter. We concatenate all outputs from the time-convolutions along the feature dimension.
- The new frame is generated from the concatenation by several stacks of dilated convolution in the frequency-direction and with DenseNet skip-connections and bottleneck layers [21]. We use three stacks of four layers and use zero padding to keep the 60 frequency dimensions.
- The final output is produced by a (1×1) convolution with one filter and adding the result to the previous frame. We thus only learn the difference from the previous frame to the next frame.

The convolutions in time direction use 24 filters each, while the convolutions in frequency direction use 32 filters with bottleneck layers with 96 filters. All layers use rectified linear units as activation functions.

C. Training

The number of layers for the stacks of dilated convolutions in time-direction is 4 for the spectral envelopes, 6 for the phonemes and 3 for both the f_0 and loudness.

We train the model using the adam optimizer [22] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, just like in the original paper, but with an initial learning rate or $5 \cdot 10^{-4}$ and decay rate of $1 - 1 \cdot 10^{-5}$ per update (batch). We feed the network with minibatches consisting of 16 samples each chosen from random locations.

The loss is obtained as a simple mean squared error (mse) of the log amplitudes of the individual frequency bins. Other error functions like the mean absolute error or Sobolev norms (sums of L^p norms and L^p norms over its derivatives) were also considered but we found that results did not differ significantly.

III. Differences with Existing Models

A. 2d vs. 1d Convolutions

The authors of [11] claim that “the translation invariance that 2d convolutions offer is an undesirable property for the frequency dimension”. Although in fact we do not expect to see every formant in each frequency bin with equal probability, formants can be found at different frequency locations. To be able to reduce the formants representation, we need to be able to shift the filters in time *and* frequency.

To prove our claim, we build a 2d version of the WaveNet-style network from [11] and compare it to the original version to show that it yields in fact better audio quality.

The 2d version of [11] replaces (2) dilated 1d convolutions with dilation rates of 2^l with (2×3) dilated 2d convolutions with dilation rate of $(2^l, 2^l)$. We reduce the number of filters dramatically so that we now have less trainable parameters (about one third) as compared to the original model but still more features per time step.

B. Predicting Distributions

It is common practice in prediction to learn to predict distributions rather than samples. Distributions allow modelling data that is uncertain or noisy. In the case of WaveNet [8] the system models a time series that is a mix of a periodic signal and coloured noise. The coloured noise cannot be modelled by a deterministic system and therefore predicting a distribution and sampling from it is necessary.

Reference [11] use the WaveNet architecture to generate not audio, but spectral envelopes. Their system predicts parameters of a constrained Gaussian mixture to generate an independent parametric probability distribution for each frequency bin of the spectral envelope. However there are some very important differences between raw audio and spectral envelopes: raw audio (as modelled by WaveNet) has only one dimension per time step while spectral envelopes are modelled (here) with 60 frequency bins. Raw audio is rapidly changing, contains oscillations and coloured noise, while spectral envelopes are not oscillating, slowly changing and not noisy.

Since one time step of spectral envelopes contains 60 frequency bins, it is impossible to model all correlations of all frequency bins. This is typically not necessary, as correlations between frequency bins that are far apart can be assumed to be insignificant. Nevertheless, there are correlations between neighbouring frequency bins that cannot be neglected, if the goal is to model the actual probability distribution of the spectral envelopes. Generating independent parametric distributions for each frequency bin F_i (as is done by [11]) must either assume that the frequency bins are independent (which they are not) or in fact yield an approximation of the true distribution by the conditional expectations $\tilde{F}_i = \mathbb{E}(F_i | \{F_j : j \neq i\})$. This is however the uninteresting part of the distribution. The conditional expectation $\tilde{F}_i$ describes the independent noise in each frequency band while multiple possible positions of formants are not modelled at all.

Since spectral envelopes are not noisy, we believe that it is not necessary at all, to predict probability distributions. Our approach generates a spectral envelope directly.

C. Stability by iterated prediction

One problem with recursive models is stability. During prediction the error accumulates over time and once strayed too much from the path, there is no way to recover, because the system is in a state which it has never seen during training. It is also worth noting, that the envelopes do not change much during phonemes, but change more rapidly during a phoneme change.

To learn to make good predictions over a long time, a typical approach is to add noise to the input envelopes to simulate envelopes that have been previously predicted improperly or predicted properly, but were not contained in the training set. However the noise level needs to be very high and thus reduces the quality of the training data (Reference [11] suggests a noise

TABLE I
THE DIFFERENT MODELS THAT WERE TRAINED FOR THE PERCEPTIVE TESTS.

Name	Architecture	Conv.	Loss	Data Augmentation
BB1	Blaauw & Bonada	1-d	CGM[a]	noise
BB2	Blaauw & Bonada	2-d	CGM	noise
MSE	Bous & Roebel	2-d	MSE[b]	iterated
CGM	Bous & Roebel	2-d	CGM	iterated
iter	Bous & Roebel	2-d	MSE	iterated
noise	Bous & Roebel	2-d	MSE	noise

[a]constrained Gaussian mixture from [11]
[b]mean squared error

level of 20% of the value range). Instead, we enforce stability by iteratively predicting dozens of frames for each batch and applying the loss function to all predicted envelopes. This way we force the network to consider long term evolution and recover from prediction errors that are more likely to occur.

IV. Experimental Setup

To support our claims from Section III and to show that our network works well, we have conducted two perceptive tests with several different models: a direct comparison and a mos-test.

We train the networks on our singing database [3] consisting of roughly 1000 short phrases, and additional recordings of various pitches, loudnesses and crescendi, as well as short excerpts from real songs, from a single tenor voice, totalling about 90 minutes of singing voice. We split these recordings into training and testing files, where for each model we use the same training and testing files.

To regenerate the spectral envelopes with models that predict a probability distribution, we use the *constrained Gaussian mixture* from [11] with a generation temperature of $\tau = 0$ to minimise sampling noise.

To obtain raw audio we resynthesize the testing files with the SVLN vocoder [14], [15] by replacing the original spectral envelopes with the regenerated envelopes. Extrapolation for frequencies above 8 kHz is achieved by constant extrapolation with the value from the last frequency bin. We also include resynthesis with ground truth envelopes by resynthesizing the testing files without replacing the envelopes, thus resulting in a vocoder round trip. This procedure ensures that differences in the audio are exclusively due to differences in the spectral envelopes that are used, and not to the use of the vocoder itself.

Two evaluate each of the proposed changes we perform a direct comparison of two models, that differ only with respect to the single hyper-parameter subject to testing. Given our three modifications we evaluate

- the use of 2d versus 1d convolution by means of comparing our reimplementation of [11] and our modification as described in Section III-A,
- the advantage of modelling predictions as probability distributions by means of comparing a model trained with mse-loss with another trained to maximise the log-likelihood of the distribution of predicted samples,
- iterated training by means of comparing a model that was trained with a single prediction with noise of 12db standard deviation added to the input log-spectrum (the 12db for the noise were found to work best among the values that were tested), and another model that trained recursively performing 24 iterated predictions without any noise was added to the input.

TABLE II
PERCEPTIVE TEST RESULTS OF DIRECT COMPARISON. THE PREFERENCE IS GIVEN TOWARDS THE LEFT MODEL, I. E., A POSITIVE NUMBER IMPLIES A PREFERENCE TOWARDS THE FIRST MODEL. THE p-VALUE IS THE RESULT OF A ONE-SIDED t-TEST.

Comparison	Preference (French)	p-Value (French)	Preference (all)	p-Value (all)
BB2 vs. BB1	$+1.44$	0.03%	$+0.94$	0.00%
CGM vs. MSE	$+0.00$	50.00%	$+0.32$	3.73%
iter vs. noise	$+0.78$	1.51%	$+0.48$	0.26%

TABLE III
PERCEPTIVE TEST RESULTS FOR MEAN OPINION SCORES (MOS) WITH 5% CONFIDENCE INTERVALS.

Model	Mos (French)	Mos (all)
iter	3.15 ± 0.38	3.51 ± 0.20
noise	3.11 ± 0.33	3.45 ± 0.19
BB1	2.77 ± 0.38	2.96 ± 0.24
BB2	3.11 ± 0.46	3.51 ± 0.23
Ground truth	3.56 ± 0.53	3.61 ± 0.25

To identify if the hyper-parameter is useful for overall quality, participants of our test were given the same phrase from both models and were asked to give a preference from -3 to 3.

The mean opinion score has been measured by asking the participants to rank the given phrases on a scale from 1 to 5, where 1 was the worst and 5 was the best. Each participant was given the same phrase from all five models, but the phrases may differ for each participant. The models we used are summarised in Table I. For the mos test the following models were used: the two models from the 2d/1d comparison (BB1 and BB2), the two models from the iterated vs. training with input noise comparison (iter and noise), and a resynthesis with ground truth envelopes.

The survey was carried out online. We received 31 submissions from various backgrounds. Of those 31 submissions, 9 were from native French speakers.

V. Results

Preferences of native French speakers are listed separately because the phrases were in French language. We can see that native French speakers were more critical (apparent in the mos test, cf. Table III). This may be because native French speakers could additionally consider the pronunciation, and pronunciation may still not be as good (in the feed back it was actually mentioned that the singing voices seem to have kind of an accent).

A. Comparison Test

Table II shows the results from the comparison test. The "Preference" column contains the mean of the preference values that were submitted towards the left model, i. e., in the comparison a vs. b positive values mean that a was preferred, negative values mean that b was preferred. The "p-Value" column contains the p-value of the one sided Student-t-test, i. e., the probability that, the data was generated under the alternative hypothesis ("the right model was better or equal").

There is a very clear preference towards the use of 2-d convolutions among both native French speakers and all participants in total. The p-value of 0.00% actually means that the p-value was below 0.005%, which was rounded down to 0. Also a strong preference was given towards the iterated training method. No clear preference could be deduced for the choice of loss function. While a slight (and significant) preference was given by all participants in total, no preference was found among native French speakers. Incidentally the preference values add up to zero, however there were submissions with both negative and positive preference.

B. Mean Opinion Score Test

Table III shows the results from the MOS test. The given values are the mean of the submitted scores plus/minus half of the 5% confidence interval obtained by a two-sided Student-t-test. The preferences are not as clear as compared to the comparison test. While the relative preferences cannot be accepted with a p-value of 5% among native French speakers, the preference against the state of the art BB1 is supported by the preferences of all participants.

The confidence intervals are rather large due to the admittedly small number of participations. The inferior conclusiveness of the MOS test can be explained by its design: During the MOS test the participants were exposed to the (almost) same recording five times. While they might have heard some differences among the individual versions, they were much more inclined to put them in the same category because they were still very similar, than in the comparison test, where they were explicitly asked to favour one recording over the other.

VI. Conclusions

In this paper we introduced a neural network architecture that is able to generate spectral envelopes for singing synthesis using a vocoder model. We showed in perceptive tests that the modifications we made with respect to the state-of-the-art method are useful in improving the perceptive result. In particular we showed that 2d convolutions are beneficial in modelling spectral envelopes and iteratively predicting multiple frames during training is superior to simply injecting noise at the input. An investigation whether predicting probability distributions rather than single samples was also carried out, but no benefit could be found when evaluating among native French speakers.

VII. Acknowledgments

We like to thank Merlijn Blaauw for helpful discussions supporting our implementation of [11]

References

[1] E. Moulines and F. Charpentier, "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones," *Speech communication*, vol. 9, no. 5-6, pp. 453–467, 1990.

[2] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, vol. 1. IEEE, 1996, pp. 373–376.

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

[4] J. Bonada, M. Umbert, and M. Blaauw, "Expressive singing synthesis based on unit selection for the singing synthesis challenge 2016." in *INTERSPEECH*, 2016, pp. 1230–1234.

[5] H. Kenmochi and H. Ohshita, "Vocaloid-commercial singing synthesizer based on sample concatenation," in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[6] X. Gonzalvo, S. Tazari, C.-a. Chan, M. Becker, A. Gutkin, and H. Silen, "Recent advances in google real-time hmm-driven unit selection synthesizer." in *Interspeech*, 2016, pp. 2238–2242.

[7] H. Zen, Y. Agiomyrgiannakis, N. Egberts, F. Henderson, and P. Szczepaniak, "Fast, compact, and high quality lstm-rnn based statistical parametric speech synthesizers for mobile devices," *arXiv preprint arXiv:1606.06061*, 2016.

[8] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio." in *SSW*, 2016, p. 125.

[9] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," *arXiv preprint arXiv:1712.05884*, 2017.

[10] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep voice 3: 2000-speaker neural text-to-speech," *arXiv preprint arXiv:1710.07654*, 2017.

[11] M. Blaauw and J. Bonada, "A neural parametric singing synthesizer modeling timbre and expression from natural songs," *Applied Sciences*, vol. 7, no. 12, p. 1313, 2017.

[12] L. Ardaillon, C. Chabot-Canet, and A. Roebel, "Expressive control of singing voice synthesis using musical contexts and a parametric f0 model," in *Interspeech 2016*, vol. 2016, 2016, pp. 1250–1254.

[13] L. Ardaillon, G. Degottex, and A. Roebel, "A multi-layer f0 model for singing voice synthesis using a b-spline representation with intuitive controls," in *Interspeech 2015*, 2015.

[14] G. Degottex, P. Lanchantin, A. Roebel, and X. Rodet, "Mixed source model and its adapted vocal-tract filter estimate for voice transformation and synthesis," *Speech Communication*, vol. 55, no. 2, pp. 278–294, 2013.

[15] S. Huber and A. Roebel, "On glottal source shape parameter transformation using a novel deterministic and stochastic speech analysis and synthesis system," in *Proc InterSpeech*, 2015.

[16] P. Lanchantin, A. C. Morris, X. Rodet, and C. Veaux, "Automatic phoneme segmentation with relaxed textual constraints." in *Proc. of The International Conference on Language Resources and Evaluation*, 2008.

[17] A. Röbel, F. Villavicencio, and X. Rodet, "On cepstral and all-pole based spectral envelope modeling with unknown model order," *Pattern Recognition Letters, Special issue on Advances in Pattern Recognition for Speech and Audio Processing*, vol. 28, no. 6, pp. 1343–1350, 2007.

[18] B. R. Glasberg and B. C. Moore, "A model of loudness applicable to time-varying sounds," *Journal of the Audio Engineering Society*, vol. 50, no. 5, pp. 331–342, 2002.

[19] A. Camacho, "Swipe: A sawtooth waveform inspired pitch estimator for speech and music," Ph.D. dissertation, University of Florida, 2007. [Online]. Available: http://www.kerwa.ucr.ac.cr:8080/handle/10669/536

[20] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai, "Mel-generalized cepstral analysis-a unified approach to speech spectral estimation," in *Third International Conference on Spoken Language Processing*, 1994.

[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.