EFFICIENTLY TRAINABLE TEXT-TO-SPEECH SYSTEM BASED ON DEEP CONVOLUTIONAL NETWORKS WITH GUIDED ATTENTION

Hideyuki Tachibana, Katsuya Uenoyama
PKSHA Technology, Inc.
Bunkyo, Tokyo, Japan
h_tachibana@pkshatech.com

Shunsuke Aihara
Independent Researcher
Shinjuku, Tokyo, Japan
aihara@argmax.jp

ABSTRACT

This paper describes a novel text-to-speech (TTS) technique based on deep convolutional neural networks (CNN), without use of any recurrent units. Recurrent neural networks (RNN) have become a standard technique to model sequential data recently, and this technique has been used in some cutting-edge neural TTS techniques. However, training RNN components often requires a very powerful computer, or a very long time, typically several days or weeks. Recent other studies, on the other hand, have shown that CNN-based sequence synthesis can be much faster than RNN-based techniques, because of high parallelizability. The objective of this paper is to show that an alternative neural TTS based only on CNN alleviate these economic costs of training. In our experiment, the proposed Deep Convolutional TTS was sufficiently trained overnight (15 hours), using an ordinary gaming PC equipped with two GPUs, while the quality of the synthesized speech was almost acceptable.

Index Terms— Text-to-speech, deep learning, convolutional neural network, attention, sequence-to-sequence learning.

1. INTRODUCTION

Text-to-speech (TTS) is getting more and more common recently, and is getting to be a basic user interface for many systems. To further promote the use of TTS in various systems, it is significant to develop a manageable, maintainable, and extensible TTS component that is accessible to speech non-specialists, enterprising individuals and small teams.

Traditional TTS systems, however, are not necessarily friendly for them, since they are typically composed of many domain-specific modules. For example, a typical parametric TTS system is an elaborate integration of many modules e.g. a text analyzer, an F_0 generator, a spectrum generator, a pause estimator, and a vocoder that synthesize a waveform from these data, etc.

Deep learning [1] may integrate these internal building blocks into a single model, and connects the input and the output directly. This type of technique is sometimes called 'end-to-end' learning. Although such a technique is sometimes criticized as 'a black box,' an end-to-end TTS system named Tacotron [2], which directly estimates a spectrogram from an input text, has achieved promising performance recently, without using hand-engineered parametric models based on domain-specific knowledge. Tacotron, however, has a drawback that it uses many recurrent units which are quite costly to train. It is almost infeasible for ordinary labs who do not have luxurious machines to study and extend it further. In fact, some people tried to implement open clones of Tacotron [3, 4, 5, 6], but they are struggling to reproduce the speech of satisfactory quality as clear as the original results.

The purpose of this paper is to show Deep Convolutional TTS (DCTTS), a novel fully convolutional neural TTS. The architecture is largely similar to Tacotron [2], but is based on a fully convolutional sequence-to-sequence learning model similar to the literature [7]. We show that this fully convolutional TTS actually works in a reasonable setting. The contribution of this article is twofold: (1) Propose a fully CNN-based TTS system which can be trained much faster than an RNN-based state-of-the-art neural TTS system, while the sound quality is still acceptable. (2) An idea to rapidly train the attention module, which we call 'guided attention,' is also shown.

1.1. Related Work

1.1.1. Deep Learning and TTS

Recently, deep learning-based TTS systems have been intensively studied, and surprisingly high quality results are obtained in some of recent studies. The TTS systems based on deep neural networks include Zen's work [8], the studies based on RNN e.g. [9, 10, 11, 12], and recently proposed techniques e.g. WaveNet [13, sec. 3.2], Char2Wav [14], DeepVoice1&2 [15, 16], and Tacotron [2].

Some of them tried to reduce the dependency on hand-engineered internal modules. The most extreme technique in this trend would be Tacotron [2], which depends only on mel and linear spectrograms, and not on any other speech features e.g. F_0. Our method is close to Tacotron in a sense that it depends only on these spectral representations of audio signals.

Most of the existing methods above use RNN, a natural technique of time series prediction. An exception is WaveNet, which is fully convolutional. Our method is also based only on CNN but our usage of CNN would be different from WaveNet, as WaveNet is a kind of a vocoder, or a back-end, which synthesizes a waveform from some conditioning information that is given by front-end components. On the other hand, ours is rather a front-end (and most of back-end processing). We use CNN to synthesize a spectrogram, from which a simple vocoder can synthesize a waveform.

1.1.2. Sequence to Sequence (seq2seq) Learning

Recently, recurrent neural networks (RNN) have become a standard technique for mapping a sequence to another sequence, especially in the field of natural language processing, e.g. machine translation [17, 18], dialogue system [19, 20], etc. See also [1, sec. 10.4].

RNN-based seq2seq, however, has some disadvantages. Firstly, a vanilla encode-decoder model cannot encode too long sequence into a fixed-length vector effectively. This problem has been resolved by a mechanism called 'attention' [21], and the attention mechanism now has become a standard idea of seq2seq learning techniques; see also [1, sec. 12.4.5.1].

Another problem is that RNN typically requires much time to train, since it is less suited for parallel computation using GPUs. To overcome this problem, several researchers proposed the use of CNN instead of RNN, e.g. [22, 23, 24, 25, 26]. Some studies have shown that CNN-based alternative networks can be trained much faster, and sometimes can even outperform the RNN-based techniques.

Gehring et al. [7] recently united these two improvements of seq2seq learning. They proposed an idea on how to use attention mechanism in a CNN-based seq2seq learning model, and showed that the method is quite effective for machine translation. The method we proposed is based on the similar idea to the literature [7].

2. PRELIMINARY

2.1. Basic Knowledge of the Audio Spectrograms

An audio waveform and a complex spectrogram $Z = \{Z_{ft}\} \in \mathbb{C}^{F' \times T'}$ are mutually transformed by linear maps called Short-term Fourier Transform (STFT) and inverse STFT, where F' and T' denote the number of frequency bins and temporal bins, respectively. It is common to consider only the magnitude $|Z| = \{|Z_{ft}|\} \in \mathbb{R}^{F' \times T'}$, since it still has useful information for many purposes, and that $|Z|$ and Z are almost the same in the sense that there exist many phase estimation (i.e., $|Z|$ to Z estimation, and therefore, waveform synthesis) techniques, e.g. the famous Griffin&Lim algorithm (GLA) [27]; see also e.g. [28]. We always use RTISI-LA [29], an online GLA, to synthesize a waveform. In this paper, we always normalize STFT spectrograms as $|Z| \leftarrow (|Z|/\max(|Z|))^\gamma$, and convert back $|Z| \leftarrow |Z|^{\eta/\gamma}$ when we finally need to synthesize the waveform, where γ, η are pre- and post-emphasis factors.

It is also common to consider a mel spectrogram $S \in \mathbb{R}^{F \times T'}$, $(F \ll F')$, obtained by applying a mel filter-bank to $|Z|$. This is a standard dimension reduction technique for speech processing. In this paper, we also reduce the temporal dimensionality from T' to $\lceil T'/4 \rceil =: T$ by picking up a time frame every four time frames, to accelerate the training of Text2Mel shown below. We also normalize mel spectrograms as $S \leftarrow (S/\max(S))^\gamma$.

2.2. Notation: Convolution and Highway Activation

In this paper, we denote the 1D convolution layer [30] by a space saving notation $\mathsf{C}^{o \leftarrow i}_{k \star \delta}(X)$, where i is the size of input channel, o is the size of output channel, k is the size of kernel, δ is the dilation factor, and an argument X is a tensor having three dimensions (*batch, channel, temporal*). The stride of convolution is always 1. Convolution layers are preceded by appropriately-sized zero padding, whose size is suitably determined by a simple arithmetic so that the length of the sequence is kept constant. Let us similarly denote the 1D *deconvolution* layer as $\mathsf{D}^{o \leftarrow i}_{k \star \delta}(X)$. The stride of deconvolution is always 2 in this paper. Let us write a layer composition operator as $\cdot \triangleleft \cdot$, and let us write networks like $\mathsf{F} \triangleleft \mathsf{ReLU} \triangleleft \mathsf{G}(X) := \mathsf{F}(\mathsf{ReLU}(\mathsf{G}(X)))$, and $(\mathsf{F} \triangleleft \mathsf{G})^2(X) := \mathsf{F} \triangleleft \mathsf{G} \triangleleft \mathsf{F} \triangleleft \mathsf{G}(X)$, etc. ReLU is an element-wise activation function defined by $\mathsf{ReLU}(x) = \max(x, 0)$.

Convolution layers are sometimes followed by a Highway Network [31]-like gated activation, which is advantageous in very deep networks: $\mathsf{Highway}(X; \mathsf{L}) = \sigma(H_1) \odot \mathsf{ReLU}(H_2) + (1 - \sigma(H_1)) \odot X$, where H_1, H_2 are tensors of the same shape as X, and are output as $[H_1, H_2] = \mathsf{L}(X)$ by a layer L. The operator $\odot$ is the element-wise product, and σ is the element-wise sigmoid function. Hereafter let us denote $\mathsf{HC}^{d \leftarrow d}_{k \star \delta}(X) := \mathsf{Highway}(X; \mathsf{C}^{2d \leftarrow d}_{k \star \delta})$.

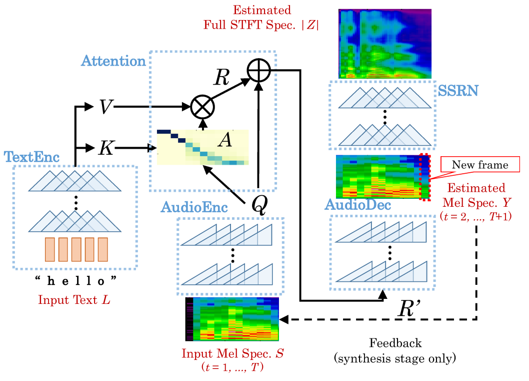

Fig. 1. Network architecture.

$$\mathsf{TextEnc}(L) := (\mathsf{HC}^{2d \leftarrow 2d}_{1 \star 1})^2 \triangleleft (\mathsf{HC}^{2d \leftarrow 2d}_{3 \star 1})^2 \triangleleft (\mathsf{HC}^{2d \leftarrow 2d}_{3 \star 27} \triangleleft \mathsf{HC}^{2d \leftarrow 2d}_{3 \star 9} \triangleleft \mathsf{HC}^{2d \leftarrow 2d}_{3 \star 3} \triangleleft \mathsf{HC}^{2d \leftarrow 2d}_{3 \star 1})^2 \triangleleft \mathsf{C}^{2d \leftarrow 2d}_{1 \star 1} \triangleleft \mathsf{ReLU} \triangleleft \mathsf{C}^{2d \leftarrow e}_{1 \star 1} \triangleleft \mathsf{CharEmbed}^{e\text{-dim}}(L).$$

$$\mathsf{AudioEnc}(S) := (\mathsf{HC}^{d \leftarrow d}_{3 \star 3})^2 \triangleleft (\mathsf{HC}^{d \leftarrow d}_{3 \star 27} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3 \star 9} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3 \star 3} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3 \star 1})^2 \triangleleft \mathsf{C}^{d \leftarrow d}_{1 \star 1} \triangleleft \mathsf{ReLU} \triangleleft \mathsf{C}^{d \leftarrow d}_{1 \star 1} \triangleleft \mathsf{ReLU} \triangleleft \mathsf{C}^{d \leftarrow F}_{1 \star 1}(S).$$

$$\mathsf{AudioDec}(R') := \sigma \triangleleft \mathsf{C}^{F \leftarrow d}_{1 \star 1} \triangleleft (\mathsf{ReLU} \triangleleft \mathsf{C}^{d \leftarrow d}_{1 \star 1})^3 \triangleleft (\mathsf{HC}^{d \leftarrow d}_{3 \star 1})^2 \triangleleft (\mathsf{HC}^{d \leftarrow d}_{3 \star 27} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3 \star 9} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3 \star 3} \triangleleft \mathsf{HC}^{d \leftarrow d}_{3 \star 1}) \triangleleft \mathsf{C}^{d \leftarrow 2d}_{1 \star 1}(R').$$

$$\mathsf{SSRN}(Y) := \sigma \triangleleft \mathsf{C}^{F' \leftarrow F'}_{1 \star 1} \triangleleft (\mathsf{ReLU} \triangleleft \mathsf{C}^{F' \leftarrow F'}_{1 \star 1})^2 \triangleleft \mathsf{C}^{F' \leftarrow 2c}_{1 \star 1} \triangleleft (\mathsf{HC}^{2c \leftarrow 2c}_{3 \star 1})^2 \triangleleft \mathsf{C}^{2c \leftarrow c}_{1 \star 1} \triangleleft (\mathsf{HC}^{c \leftarrow c}_{3 \star 3} \triangleleft \mathsf{HC}^{c \leftarrow c}_{3 \star 1} \triangleleft \mathsf{D}^{c \leftarrow c}_{2 \star 1})^2 \triangleleft (\mathsf{HC}^{c \leftarrow c}_{3 \star 3} \triangleleft \mathsf{HC}^{c \leftarrow c}_{3 \star 1}) \triangleleft \mathsf{C}^{c \leftarrow F}_{1 \star 1}(Y).$$

Fig. 2. Details of each component. For notation, see section 2.2.

3. PROPOSED NETWORK

Since some literature [2, 32] suggest that the staged synthesis from low- to high-resolution has advantages over the direct synthesis of high-resolution data, we synthesize the spectrograms using the following two networks. (1) Text2Mel, which synthesizes a mel spectrogram from an input text, and (2) Spectrogram Super-resolution Network (SSRN), which synthesizes a full STFT spectrogram from a coarse mel spectrogram. Fig. 1 shows the overall architecture.

3.1. Text2Mel: Text to Mel Spectrogram Network

We first consider to synthesize a coarse mel spectrogram from a text. This is the main part of the proposed method. This module consists of four submodules: Text Encoder, Audio Encoder, Attention, and Audio Decoder. The network $\mathsf{TextEnc}$ first encodes the input sentence $L = [l_1, \dots, l_N] \in \mathsf{Char}^N$ consisting of N characters, into the two matrices K (key), V (value) $\in \mathbb{R}^{d \times N}$, where d the dimension of encoded characters. On the other hand, the network $\mathsf{AudioEnc}$ encodes the coarse mel spectrogram $S = S_{1:F,1:T} \in \mathbb{R}^{F \times T}$, of speech of length T, into a matrix Q (query) $\in \mathbb{R}^{d \times T}$.

$$(K, V) = \mathsf{TextEnc}(L), \quad Q = \mathsf{AudioEnc}(S_{1:F,1:T}). \tag{1}$$

An attention matrix $A \in \mathbb{R}^{N \times T}$, defined as follows, evaluates how strongly the n-th character l_n and the t-th mel spectrum $S_{1:F,t}$ are related.

$$A = \mathsf{softmax}_{n\text{-axis}}(K^\mathsf{T} Q/\sqrt{d}). \tag{2}$$

$A_{nt} \sim 1$ implies that the module is focusing on n-th character l_n at time t, and it will focus on l_n or l_{n+1} (or others nearby), at the subsequent time $t+1$. Whatever, let us expect those are encoded in n-th column of V. Thus a matrix $R \in \mathbb{R}^{d \times T}$, which is the 'seed' of the subsequent mel spectra $S_{1:F,2:T+1}$, is obtained as

$$R = \mathsf{Att}(Q, K, V) := VA. \quad \text{(Note: matrix product.)} \tag{3}$$

The resultant R is concatenated with the encoded audio Q, as $R' = [R, Q]$, because we found it beneficial in our pilot study. Then, the Audio Decoder module estimates a mel spectrogram from the seed matrix $R' \in \mathbb{R}^{2d \times T}$ as follows,

$$Y_{1:F,2:T+1} = \mathsf{AudioDec}(R'). \tag{4}$$

The resultant $Y_{1:F,2:T+1}$ needs to approximate the temporally-shifted ground truth $S_{1:F,2:T+1}$. The error is evaluated by a loss function $\mathcal{L}_{\text{spec}}(Y_{1:F,2:T+1}|S_{1:F,2:T+1})$, and is back-propagated to the network parameters. The loss function was the sum of L1 loss and a function $\mathcal{D}_{\text{bin}}$ which we call binary divergence,

$$\begin{aligned}\mathcal{D}_{\text{bin}}(Y|S) &:= \mathbb{E}_{ft}\left[-S_{ft}\log\frac{Y_{ft}}{S_{ft}} - (1-S_{ft})\log\frac{1-Y_{ft}}{1-S_{ft}}\right] \\ &= \mathbb{E}_{ft}[-S_{ft}\hat{Y}_{ft} + \log(1+\exp\hat{Y}_{ft})] + \text{const.}, \end{aligned} \tag{5}$$

where $\hat{Y}_{ft} = \text{logit}(Y_{ft})$. Since the gradient is non-vanishing, i.e., $\partial\mathcal{D}_{\text{bin}}(Y|S)/\partial\hat{Y}_{ft} \propto Y_{ft} - S_{ft}$, it is advantageous for gradient-based training. It is easily verified that the spectrogram error is non-negative, $\mathcal{L}_{\text{spec}}(Y|S) = \mathcal{D}_{\text{bin}}(Y|S) + \mathbb{E}[|Y_{ft} - S_{ft}|] \geq 0$, and the equality holds iff $Y = S$.

3.1.1. Details of TextEnc, AudioEnc, and AudioDec

The networks are fully convolutional, and are not dependent on any recurrent units. In order to take into account the long contextual information, we used *dilated convolution* [33, 13, 24] instead of RNN.

The top equation of Fig. 2 is the content of TextEnc. It consists of a character embedding and several 1D *non-causal* convolution layers. In the literature [2] an RNN-based component named 'CBHG' was used, but we found this convolutional network also works well. AudioEnc and AudioDec, shown in Fig. 2, are composed of 1D *causal* convolution layers. These convolution should be causal because the output of AudioDec is feedbacked to the input of AudioEnc at the synthesis stage.

3.2. Spectrogram Super-resolution Network (SSRN)

We finally synthesize a full spectrogram $|Z| \in \mathbb{R}^{F' \times 4T}$, from the obtained coarse mel spectrogram $Y \in \mathbb{R}^{F \times T}$, using the spectrogram super-resolution network (SSRN). Upsampling frequency from F to F' is fairly simple. We can achieve that by increasing the convolution channels of a 1D convolutional network. Upsampling of the temporal axis is not done the same way. Instead, we quadruple the length of the sequence from T to $4T = T'$, by applying 'deconvolution' layers of stride size 2 twice.

The bottom equation of Fig. 2 shows SSRN. In this paper, all convolutions of SSRN are non-causal, since we do not consider online processing. The loss function is the same as Text2Mel: the sum of L1 distance and $\mathcal{D}_{\text{bin}}$, defined by (5), between the synthesized spectrogram $\mathsf{SSRN}(S)$ and the ground truth $|Z|$.

4. GUIDED ATTENTION

4.1. Guided Attention Loss: Motivation, Method and Effects

In general, an attention module is quite costly to train. Therefore, if there is some prior knowledge, incorporating them into the model may be a help to alleviate the heavy training. We show that the following simple method helps to train the attention module.

In TTS, the possible attention matrix A lies in the very small subspace of $\mathbb{R}^{N \times T}$. This is because of the rough correspondence of the order of the characters and the audio segments. That is, when one reads a text, it is natural to assume that the text position n progresses

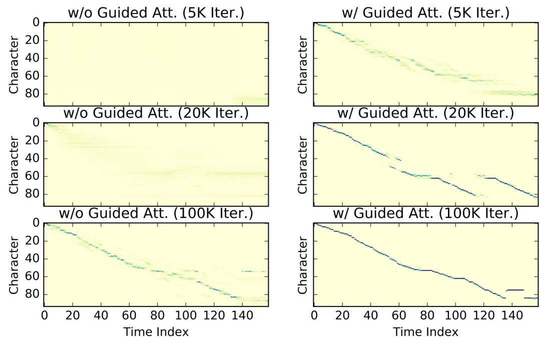

Fig. 3. Comparison of the attention matrix A, trained with and without the guided attention loss $\mathcal{L}_{\text{att}}(A)$. (Left) Without, and (Right) with the guided attention. The test text is *"icassp stands for the international conference on acoustics, speech, and signal processing."* We did not use the heuristics described in section 4.2.

nearly linearly to the time t, i.e., $n \sim at$, where $a \sim N/T$. This is a noticeable difference of TTS from other seq2seq learning techniques such as machine translation, where an attention module needs to resolve the word alignment between two languages that have very different syntax, e.g. English and Japanese.

Based on this idea, we introduce 'guided attention loss,' which prompts the attention matrix A to be 'nearly diagonal,' $\mathcal{L}_{\text{att}}(A) = \mathbb{E}_{nt}[A_{nt}W_{nt}]$, where $W_{nt} = 1 - \exp\{-(n/N - t/T)^2/2g^2\}$. In this paper, we set $g = 0.2$. If A is far from diagonal (e.g., reading the characters in the random order), it is strongly penalized by the loss function. This subsidiary loss function is simultaneously optimized with the main loss $\mathcal{L}_{\text{spec}}$ with the equal weight.

Although this measure is based on quite a rough assumption, it improved the training efficiency. In our experiment, if we added the guided attention loss to the objective, the term began decreasing only after ~100 iterations. After ~5K iterations, the attention became roughly correct, not only for training data, but also for new input texts. On the other hand, without the guided attention loss, it required much more iterations. It began learning after ~10K iterations, and it required ~50K iterations to look at roughly correct positions, but the attention matrix was still vague. Fig. 3 compares the attention matrix, trained with and without guided attention loss.

4.2. Forcibly Incremental Attention at the Synthesis Stage

At the synthesis stage, the attention matrix A sometimes fails to focus on the correct characters. Typical errors we observed were (1) skipping several letters, and (2) repeating a same word twice or more. In order to make the system more robust, we heuristically modified the matrix A to be 'nearly diagonal,' by a simple rule as follows. We observed this method sometimes alleviated such failures.

Let n_t be the position of the character which is read at time t, i.e., $n_t = \text{argmax}_n A_{nt}$. Comparing the current position n_t and the previous position n_{t-1}, if the difference $n_t - n_{t-1}$ is outside of the range $-1 \leq n_t - n_{t-1} \leq 3$, the current attention is forcibly set as $A_{nt} = \delta_{n,n_{t-1}+1}$ (Kronecker delta), to increment the attention target as $n_t = n_{t-1} + 1$.

5. EXPERIMENT

5.1. Experimental Setup

We used LJ Speech Dataset [34] to train the networks. This is a public domain speech dataset consisting of ~13K pairs of text and speech, without phoneme-level alignment, ~24 hours in total. These speech data have a little reverberation. We preprocessed the

Table 1. Parameter Settings.

Sampling rate of audio signals	22050 Hz
STFT window function	Hanning
STFT window length and shift	1024 ($\sim$46.4 [ms]), 256 ($\sim$11.6[ms])
STFT spectrogram size $F' \times 4T$	$513 \times 4T$ (T depends on audio clip)
Mel spectrogram size $F \times T$	$80 \times T$ (T depends on audio clip)
Dimension e, d and c	128, 256, 512
ADAM parameters $(\alpha, \beta_1, \beta_2, \varepsilon)$	$(2 \times 10^{-4}, 0.5, 0.9, 10^{-6})$
Minibatch size	16
Emphasis factors (γ, η)	(0.6, 1.3)
RTISI-LA window and iteration	100, 10
Character set, Char	`a-z,.'-` and `Space` and `NULL`

Table 2. Comparison of MOS (95% confidence interval), training time, and iterations (Text2Mel/SSRN), of an open Tacotron [5] and the proposed method (DCTTS). The digits with * were excerpted from the repository [5].

Method	Iteration	Time	MOS (95% CI)
Open Tacotron [5]	877K*	12 days*	2.07 ± 0.62
DCTTS	20K/ 40K	$\sim$2 hours	1.74 ± 0.52
DCTTS	90K/150K	$\sim$7 hours	2.63 ± 0.75
DCTTS	200K/340K	$\sim$15 hours	2.71 ± 0.66
DCTTS	540K/900K	$\sim$40 hours	2.61 ± 0.62

texts by spelling out some of abbreviations and numeric expressions, decapitalizing the capitals, and removing less frequent characters not shown in Table 1, where `NULL` is a dummy character for zero-padding.

We implemented our neural networks using Chainer 2.0 [35]. We trained the models using a household gaming PC equipped with two GPUs. The main memory of the machine was 62GB, which is much larger than the audio dataset. Both GPUs were NVIDIA GeForce GTX 980 Ti, with 6 GB memories.

For simplicity, we trained Text2Mel and SSRN independently and asynchronously using different GPUs. All network parameters were initialized using He's Gaussian initializer [36]. Both networks were trained by the ADAM optimizer [37]. When training SSRN, we randomly extracted short sequences of length $T = 64$ for each iteration to save memory usage. To reduce the disk access, we reduced the frequency of creating the snapshot of parameters to only once per 5K iterations. Other parameters are shown in Table 1.

As it is not easy for us to reproduce the original results of Tacotron, we instead used a ready-to-use model [5] for comparison, which seemed to produce the most reasonable sounds in the open implementations. It is reported that this model was trained using LJ Dataset for 12 days (877K iterations) on a GTX 1080 Ti, newer GPU than ours. Note, this iteration is still much less than the original Tacotron, which was trained for more than 2M iterations.

We evaluated mean opinion scores (MOS) of both methods by crowdsourcing on Amazon Mechanical Turk using crowdMOS toolkit [38]. We used 20 sentences from *Harvard Sentences* List 1&2. The audio data were synthesized using five methods shown in Table 2. The crowdworkers rated these 100 clips from 1 (Bad) to 5 (Excellent). Each worker is supposed to rate at least 10 clips. To obtain more responses of higher quality, we set a few incentives shown in the literature. The results were statistically processed using the method shown in the literature [38].

5.2. Result and Discussion

In our setting, the training throughput was $\sim$3.8 minibatch/s (Text2Mel) and $\sim$6.4 minibatch/s (SSRN). This implies that we can iterate the updating formulae of Text2Mel 200K times in 15 hours. Fig. 4 shows an example of attention, synthesized mel and full spectrograms, after 15 hours training. It shows that the method can almost correctly focus on the correct characters, and synthesize quite clear spectrograms. More samples are available at the author's web page.[1]

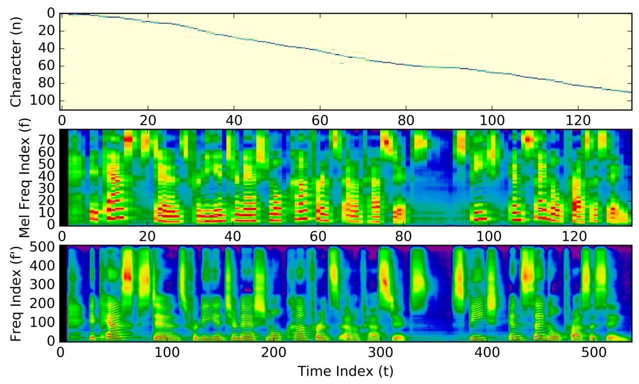

Fig. 4. (Top) Attention, (middle) mel, and (bottom) linear STFT spectrogram, synthesized by the proposed method, after 15 hours training. The input text is *"icassp stands for the international conference on acoustics, speech and signal processing."* (90 chars)

In our crowdsourcing experiment, 31 subjects evaluated our data. After the automatic screening by the toolkit [38], 560 scores rated by 6 subjects were selected for final statistics calculation. Table 2 compares the performance of our proposed method (DCTTS) and an open Tacotron. Our MOS (95% confidence interval) was 2.71 ± 0.66 (15 hours training) while the Tacotron's was 2.07 ± 0.62. Although it is not a strict comparison since the frameworks and the machines were different, it would be still concluded that our proposed method is quite rapidly trained to the satisfactory level compared to Tacotron.

Note that the MOS were below the level reported in [2]. The reasons may be threefold: (1) the limited number of iterations, (2) SSRN needs to synthesize the spectrograms from less information than [2], and (3) the reverberation of the training data.

6. SUMMARY AND FUTURE WORK

This paper described a novel TTS technique based on deep convolutional neural networks, and a technique to train the attention module rapidly. In our experiment, the proposed Deep Convolutional TTS was trained overnight ($\sim$15 hours), using an ordinary gaming PC equipped with two GPUs, while the quality of the synthesized speech was almost acceptable.

Although the audio quality is far from perfect yet, it may be improved by tuning some hyper-parameters thoroughly, and by applying some techniques developed in the deep learning community.

We believe this method will encourage further development of the applications based on speech synthesis. We can expect that this simple neural TTS may be extended to many other purposes e.g. emotional/non-linguistic/personalized speech synthesis, etc., by further studies. In addition, since a neural TTS has become lighter, the studies on more integrated speech systems e.g. some multimodal systems, may have become more feasible. These issues should be worked out in the future.

7. ACKNOWLEDGEMENT

The authors would like to thank the OSS contributors and the data creators (LibriVox contributors and @keithito).

[1] `https://github.com/tachi-hi/tts_samples`

8. REFERENCES

[1] I. Goodfellow et al., *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[2] Y. Wang et al., "Tacotron: Towards end-to-end speech synthesis," in *Proc. Interspeech*, 2017, arXiv:1703.10135.

[3] A. Barron, "Implementation of Google's Tacotron in TensorFlow," 2017, Available at GitHub, `https://github.com/barronalex/Tacotron` (visited Oct. 2017).

[4] K. Park, "A TensorFlow implementation of Tacotron: A fully end-to-end text-to-speech synthesis model," 2017, Available at GitHub, `https://github.com/Kyubyong/tacotron` (visited Oct. 2017).

[5] K. Ito, "Tacotron speech synthesis implemented in TensorFlow, with samples and a pre-trained model," 2017, Available at GitHub, `https://github.com/keithito/tacotron` (visited Oct. 2017).

[6] R. Yamamoto, "PyTorch implementation of Tacotron speech synthesis model," 2017, Available at GitHub, `https://github.com/r9y9/tacotron_pytorch` (visited Oct. 2017).

[7] J. Gehring et al., "Convolutional sequence to sequence learning," in *Proc. ICML*, 2017, pp. 1243–1252, arXiv:1705.03122.

[8] H. Zen et al., "Statistical parametric speech synthesis using deep neural networks," in *Proc. ICASSP*, 2013, pp. 7962–7966.

[9] Y. Fan et al., "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *Proc. Interspeech*, 2014, pp. 1964–1968.

[10] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *Proc. ICASSP*, 2015, pp. 4470–4474.

[11] S. Achanta et al., "An investigation of recurrent neural network architectures for statistical parametric speech synthesis.," in *Proc. Interspeech*, 2015, pp. 859–863.

[12] Z. Wu and S. King, "Investigating gated recurrent networks for speech synthesis," in *Proc. ICASSP*, 2016, pp. 5140–5144.

[13] A. van den Oord et al., "WaveNet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[14] J. Sotelo et al., "Char2wav: End-to-end speech synthesis," in *Proc. ICLR*, 2017.

[15] S. Arik et al., "Deep voice: Real-time neural text-to-speech," in *Proc. ICML*, 2017, pp. 195–204, arXiv:1702.07825.

[16] S. Arik et al., "Deep voice 2: Multi-speaker neural text-to-speech," in *Proc. NIPS*, 2017, arXiv:1705.08947.

[17] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[18] I. Sutskever et al., "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014, pp. 3104–3112.

[19] O. Vinyals and Q. Le, "A neural conversational model," in *Proc. Deep Learning Workshop, ICML*, 2015.

[20] I. V. Serban et al., "Building end-to-end dialogue systems using generative hierarchical neural network models.," in *Proc. AAAI*, 2016, pp. 3776–3784.

[21] D Bahdanau et al., "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR 2015, arXiv:1409.0473*, 2014.

[22] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, 2014, pp. 1746–1752, arXiv:1408.5882.

[23] X. Zhang et al., "Character-level convolutional networks for text classification," in *Proc. NIPS*, 2015, arXiv:1509.01626.

[24] N. Kalchbrenner et al., "Neural machine translation in linear time," *arXiv:1610.10099*, 2016.

[25] Y. N. Dauphin et al., "Language modeling with gated convolutional networks," *arXiv:1612.08083*, 2016.

[26] J. Bradbury et al., "Quasi-recurrent neural networks," in *Proc. ICLR 2017*, 2016.

[27] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Trans. ASSP*, vol. 32, no. 2, pp. 236–243, 1984.

[28] P. Mowlee et al., *Single Channel Phase-Aware Signal Processing in Speech Communication: Theory and Practice*, Wiley, 2016.

[29] X. Zhu et al., "Real-time signal estimation from modified short-time fourier transform magnitude spectra," *IEEE Trans. ASLP*, vol. 15, no. 5, 2007.

[30] Y. LeCun and Y. Bengio, "The handbook of brain theory and neural networks," chapter Convolutional Networks for Images, Speech, and Time Series, pp. 255–258. MIT Press, 1998.

[31] R. K. Srivastava et al., "Training very deep networks," in *Proc. NIPS*, 2015, pp. 2377–2385.

[32] H. Zhang et al., "StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proc. ICCV*, 2017, pp. 5907–5915, arXiv:1612.03242.

[33] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. ICLR*, 2016.

[34] K. Ito, "The LJ speech dataset," 2017, Available at `https://keithito.com/LJ-Speech-Dataset/` (visited Sep. 2017).

[35] S. Tokui et al., "Chainer: A next-generation open source framework for deep learning," in *Proc. Workshop LearningSys, NIPS*, 2015.

[36] K. He et al., "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. ICCV*, 2015, pp. 1026–1034, arXiv:1502.01852.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR 2015*, 2014, arXiv:1412.6980.

[38] F. Ribeiro et al., "CrowdMOS: An approach for crowdsourcing mean opinion score studies," in *Proc ICASSP*, 2011, pp. 2416–2419.