MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

MASTER TEHNOLOGII MULTIMEDIA

Propun nota: 10 (zece)
Mircea Giurgiu

ACP 2

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

Prenume Oscar GAL, I TM

Coordonator: Prof. dr. ing. Mircea GIURGIU
Mircea Giurgiu

2021

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

***Abstract*— The current paper presents an overview of how to implement a Text-To-Speech (TTS) technique based on a deep Convolutional Neural Network (CNN). The state-of-the-art techniques that are used in the present are mostly based on a Recurrent Neural Network (RNN), however training these types of neural networks often takes a very powerful computer, on a long period of type, typically taking about several days or weeks. Recently studies have shown that CNN-based TTS works faster and better than RNN-based techniques, because of high parallelizability. The object of this paper is to study if this type of TTS system, based on CNN is much more efficient. By the end of the paper, we should have a TTS system that was trained on our voice, in a relatively short period and also with a relatively small dataset.**

Keywords*— *Text-to-speech, deep learning, convolutional neural network, attention, sequence to sequence learning.

I. INTRODUCTION

Are there any use cases of text to speech based on convolutional neural networks? These systems nowadays can be seen almost anywhere. Starting from online casinos to phones with guidance for people with visual impediments, these types of systems helped humankind a lot. The growth of this branch of computer science was hugely impacted by the apparition of easily accessible trainable neural networks. The main reason for this paper is being a support for my dissertation paper, where will be described some bracelets that will help people with visual impediments navigate through their life easier.

The TTS traditional systems are not necessarily friendly and easy to use, and this is since they are made of many domain-specific modules, using hard-to-understand language and techniques especially if you haven't studied this branch of computer science before. For example, a typical TTS system is composed out of many modules e.g., a text analysis, an F_0 generator, a spectrum generator, a pause estimator, and a vocoder that synthesizes a waveform from these data, etc.

Let's take Tacotron as an example. This tool uses a black box system, this actually means that this piece of software unites internal building blocks into a single model, and directly connects the input and output; this type of techniques is mostly called 'end-to-end' learning. The software mentioned above, directly estimates a spectrogram from an input text, has achieved promising performance recently, without intensively engineered parametric models based on domain-specific knowledge. Tacotron, however, has a drawback that it exploits many recurrent units, which are quite costly to train, making it almost infeasible for ordinary labs without luxurious machines to study and extend it further. Indeed, some people tried to implement open clones of Tacotron, but they are struggling to reproduce the speech of satisfactory quality as clear as the original work.

The purpose of this paper is to show Deep Convolutional TTS (DCTTS), a novel, handy neural TTS, which is fully convolutional. The architecture is largely similar to Tacotron but is based on a fully convolutional sequence-to-sequence learning model similar to the literature We show this handy TTS actually works in a reasonable setting. The contribution of this article is twofold: 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. An idea to rapidly train the attention, which we call 'guided attention,' is also shown.

II. Fundamentals

A. Deep Learning

Deep learning is an artificial intelligence (AI) function that imitates the working of the human brain in processing data and creating patterns for use in decision making. Deep leaning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured of unlabeled. It is also known as deep neural learning of deep neural network.

Deep leaning has evolved in the last couple of years due to the raise of digital era, which has brought about an explosion of data, known simply as big data. However, the data, which normally is unstructured, is so vast that could take decades for humans to comprehend it and extract relevant information.

In the last couple of years, there was an increase of studies in the deep learning TTS systems, and some of the latest's studies has shown surprisingly clear results. The TTS systems based on deep neural networks include Zen's work in 2013 [4], there are also some studies based on RNN e.g. [5, 6, 7, 8], and recently proposed techniques e.g., Wave Net [9], Char2Wav [10], Deep voice [11, 12] and Tacotron [2]. The method that we are going to implement, and test is based on the technique developed by Tachibana [1]. The methods that is presented in their paper [1] is based on CNN and not RNN.

B. Sequence to sequence learning

Sequence to sequence learning (Seq2Seq) is about training models to convert sequences from one domain (e.g., sentences in English, text) to sequences in another domain (e.g., same sentence translated in French, spoken words) RNN-based sequence to sequence, however, has some disadvantages. One is that a vanilla encoder-decoder model cannot encode too long sequences into a fixed-length vector effectively. This problem has been resolved by a mechanism called 'attention' [13], and the attention mechanism now has become a standard idea in seq2seq learning techniques.

Another problem is that RNN typically requires much time to train, since it is less suitable for parallel computation using GPUs. In order to overcome this problem, several people proposed the use of CNN, instead of RNN, e.g. [12]. Some studies have shown that CNN-based alternative networks can be trained much faster, and can even outperform the RNN-based techniques.

Gehring et al. [3] recently united these two improvements of seq2seq learning. They proposed an idea how to use attention mechanism in a CNN-based seq2seq learning model, and showed that the method is quite effective for machine translation. Our proposed method, indeed, is based on the similar idea to the literature [3].

C. Basic knoledge of the audio spectogram

An audio waveform can be mutually converted to a complex spectrogram $Z = \{Z_{f,t}\} \in C^{F' \times T'}$ by linear maps called 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 $|Z| = \{ |Z_{f,t}| \}$, since it still has useful information for many purposes, and that |Z| is almost identical to Z in a sense that there exist many phase estimation (~ waveform synthesis) techniques from magnitude spectrograms, e.g. the famous Griffin&Lim algorithm. We always use RTISI-LA an online G&L, 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 $S \in R^{F' \times T'}$, (F << F'), by applying a mel filter-bank to |Z|. This is a standard dimension reduction technique in speech processing. In this paper, we also reduce the temporal dimensionality from T' to [T' /4] =: 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 $S \leftarrow (S/\max(S))^{\gamma}$.

D. Notation: Convolution and Highway Activation

In this paper, we denote 1D convolution layer [14] by a space saving notation $C^{o\leftarrow i}_{k*\delta}(X)$, where i is the sizes of input channel, o is the sizes 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 also denote the 1D deconvolution layer as $D^{o\leftarrow i}_{k*\delta}(X)$. The stride of deconvolution is always 2 in this paper. Let us write a layer composition operator as $\cdot <| \cdot$, and let us write networks like $F \triangleleft \mathrm{ReLU} \triangleleft \bar{G}(X) := \bar{F}(\mathrm{ReLU}(G(X)))$ and $(F \triangleleft G)^2(X) := F \triangleleft G \triangleleft F \triangleleft G(X)$, etc. ReLU is an element-wise activation function defined by ReLU(x) = max(x, 0).

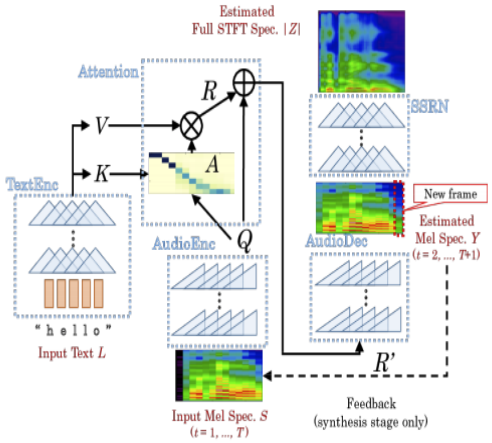

Fig. 1. Network architecture.

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

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

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

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

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

III. Proposed Network

Our DCTTS model consists of two networks: (1) Text2Mel, which synthesize a mel spectrogram from an input text, and (2) Spectrogram Super-resolution Network (SSRN), which convert a coarse mel spectrogram to the full STFT spectrogram. Fig. 1 shows the overall architecture of the proposed method.

A. Text2Mel: Text to Mel Spectogram 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 TextEnc first encodes the input sentence $L = [l_1, \ldots, l_N] \in \mathrm{Char}^N$ consisting of N characters, into the two matrices $K, V \in R^{d\times N}$. On the other hand, the network AudioEnc encodes the coarse mel spectrogram $S(= S_{1:F,1:T}) \in R^{F\times T}$, of previously spoken speech, whose length is T, into a matrix $Q \in R^{d\times T}$.

$$(K, V) = \mathrm{TextEnc}(L) \quad (1)$$
$$Q = \mathrm{AudioEnc}(S1{:}F,1{:}T) \quad (2)$$

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

$$A = \mathrm{softmaxn\text{-}axis}(K^T Q/\sqrt{d}) \quad (3)$$

$A_{nt} \sim 1$ implies that the module is looking at n-th character l_n at the time frame t, and it will look at l_n or l_{n+1} or characters around them, at the subsequent time frame t + 1. Whatever, let us expect those are encoded in the n-th column of V . Thus a seed $R \in R^{d\times T}$, decoded to the subsequent frames $S_{1:F,2:T+1}$, is obtained as

$$R = \mathrm{Att}(Q, K, V) := V A. \text{ (Note: matrix product.)} \quad (4)$$

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 concatenated matrix $R' \in R^{2d\times T}$ is decoded by

the Audio Decoder module to synthesize a coarse mel spectrogram,

$$Y_{1:F,2:T+1} = \text{AudioDec}(R') \quad (5)$$

The result $Y_{1:F,2:T+1}$ is compared with the temporally-shifted ground truth $S_{1:F,2:T+1}$, by a loss function $L_{spec}(Y_{1:F,2:T+1}|S_{1:F,2:T+1})$, and the error is back-propagated to the network parameters. The loss function was the sum of L1 loss and the binary divergence D_{bin},

$$D_{bin}(Y|S) := E_{ft}[-S_{ft} \log Y_{ft} - (1 - S_{ft}) \log(1 - Y_{ft})] = E_{ft}[-S_{ft}\hat{Y}_{ft} + \log(1 + \exp \hat{Y}_{ft})] \quad (6)$$

where $\hat{Y}_{ft} = \text{logit}(Yft)$. Since the binary divergence gives a nonvanishing gradient to the network, $\partial D_{bin}(Y|S)/\partial \hat{Y}_{ft} \propto Y_{ft} - S_{ft}$, it is advantageous in gradient-based training. It is easily verified that the spectrogram error is non-negative, $L_{spec}(Y|S) = D_{bin}(Y|S) + E[|Y_{ft} - S_{ft}|] \geq 0$, and the equality holds iff $Y = S$.

Our networks are fully convolutional, and are not dependent on any recurrent units. Instead of RNN, we sometimes take advantages of dilated convolution [32, 13, 24] to take long contextual information into account. The top equation of Fig. 2 is the content of TextEnc. It consists of the character embedding and the stacked 1D non-causal convolution. A previous literature used a heavier RNN-based component named 'CBHG,' but we found this simpler network also works well. AudioEnc and AudioDec, shown in Fig. 2, are composed of 1D causal convolution layers with Highway activation. These convolution should be causal because the output of AudioDec is feedbacked to the input of AudioEnc in the synthesis stage.

IV. Code Implementation

The first step in order to obtain the desired TTS based on CNN should be the installing the dependencies and downloading the data set of vocals.

```
import os
from os.path import exists, join, expanduser
project_name = "pytorch-dc-tts"
if not exists(project_name):
  ! git clone --quiet
https://github.com/tugstugi/{project_name}
  ! cd {project_name} && pip install -q -r
requirements.txt
```

Then we should prepare the models

```
import sys
sys.path.append(project_name)

import warnings
warnings.filterwarnings("ignore")  # ignore warnings
in this notebook

import numpy as np
import torch

from tqdm import *
import IPython
from IPython.display import Audio

from hparams import HParams as hp
from audio import save_to_wav
from models import Text2Mel, SSRN
from datasets.lj_speech import vocab, idx2char,
get_test_data

torch.set_grad_enabled(False)
text2mel = Text2Mel(vocab)
text2mel.load_state_dict(torch.load("ljspeech-
text2mel.pth").state_dict())
text2mel = text2mel.eval()
ssrn = SSRN()
ssrn.load_state_dict(torch.load("ljspeech-
ssrn.pth").state_dict())
ssrn = ssrn.eval()
```

Sentences to synthesize

```
SENTENCES = [
   "The birch canoe slid on the smooth planks.",
   "Glue the sheet to the dark blue background."]
```

Synthetize on CPU

```
for i in range(len(SENTENCES)):
   sentence = SENTENCES[i]
   normalized_sentence = "".join([c if c.lower() in
vocab else '' for c in sentence])
```

```
    print(normalized_sentence)
        sentences = [normalized_sentence]
    max_N = len(normalized_sentence)
    L = torch.from_numpy(get_test_data(sentences,
max_N))
    zeros = torch.from_numpy(np.zeros((1, hp.n_mels,
1), np.float32))
    Y = zeros
    A = None

    for t in range(hp.max_T):
      _, Y_t, A = text2mel(L, Y,
monotonic_attention=True)
      Y = torch.cat((zeros, Y_t), -1)
      _, attention = torch.max(A[0, :, -1], 0)
      attention = attention.item()
if L[0, attention] == vocab.index('E'):  # EOS
          break

    _, Z = ssrn(Y)

    Z = Z.cpu().detach().numpy()
    save_to_wav(Z[0, :, :].T, '%d.wav' % (i + 1))
          IPython.display.display(Audio('%d.wav' % (i
+ 1), rate=hp.sr))
```

REFERENCES

[1] Hideyuki Tachibana, Katsuya Uenoyama – "EFFICIENTLY TRAINABLE TEXT-TO-SPEECH SYSTEM BASED ON DEEP CONVOLUTIONAL NETWORKS WITH GUIDED ATTENTION"

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

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

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

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

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

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

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

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

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

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

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

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

[14] 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.