Understanding the features of voice and using it to synthesize speech

Baris Ozkuslar
Computer Engineering
Middle East Technical University
Ankara, Turkey
ozkuslar.baris@gmail.com

***Abstract*—The goal of this project is understanding the intrinsic properties of a human voice and using it to synthesize new speech with the properties of the given voice. In order to achieve this goal, state of the art speech recognition and speech generation models are fused together to create a speech synthesizer that can adapt itself and imitate a given voice.**

I. Introduction

I worked on a speech synthesis tool that is able to imitate the vocal features of a given voice input. The model takes a sample voice mel spectrogram and a sentence as input, and outputs a spectrogram which includes the utterance of the input sentence while mimicking the input voice. In my method, I construct a two level model which makes use of a previously published speech recognition and speech synthesis models.

There are multiple uses of this model. First of all, it can be used to generate speech for anyone, given that there exist a voice sample for the person, and the person's voice features is not too far from the voices in the training data. After training, the model can be modified a bit to recognize who is the person speaking. It can also be used to modify features of the voice of a speech. I focused on the first use in this project.

Example synthesis results is available here [1]. The quality of voice outputs is worse than state of the art results. I aim for this project to be a proof of concept that shows it is possible to imitate a voice with only one pass from the network, without additional training to fit the new speaker voice.

II. Previous Work

In the recent years, speech synthesis has been a very popular topic in machine learning field, and multiple successful models have been proposed.

Tacotron [2] is a single speaker speech synthesis model which directly generates raw spectrogram from text. Tacotron is end to end, text to spectrogram, but it's limited with the speaker given in the training.

WaveNet [3] is a multi speaker speech synthesis model which generates audio waveform from text features, can be conditioned on speaker identity(speakers should be in the training data). WaveNet can be trained on voices of many people, and it can learn to generate a speech with a chosen voice. However, it's still limited by the number of speakers given in the training data. It takes linguistic features as input, not text.

Char2Wav [4] is a speech synthesis model with two step prediction. First step is vocoder parameters prediction and second step is audio waveform prediction. The training of these two steps is separate. It has a preliminary paper with no complete version. It has an open source implementation published by the authors.

Char2Wav paper doesn't give details on imitating voices not in training data, but the website lyrebird.ai-built by the same authors-seems successful at generating a speech from a text, while imitating the given sample voice. But the quality of the generated speeches with 1 minute sample voice is worse than state of the art single or multispeaker models that directly generate speech for voices in training data.

VoiceLoop [5] has been proposed as another voice imitation model. The model takes a speaker embedding and an input sentence(as a list of phonemes) and outputs vocoder features which can be converted to waveform by WORLD vocoder. In training, embedding vector for each speaker is learned and kept in a lookup table. After training, a new speaker can be fit by keeping all network parameters fixed and only optimizing the embedding for the speaker. VoiceLoop has the best results I could find in the task of imitating unknown speakers.

These two methods reduce the time to imitate a new speaker, but they still require re-training for the new person. The model weights don't change, but they are training with the given person's sample voices to train an embedding for the person.

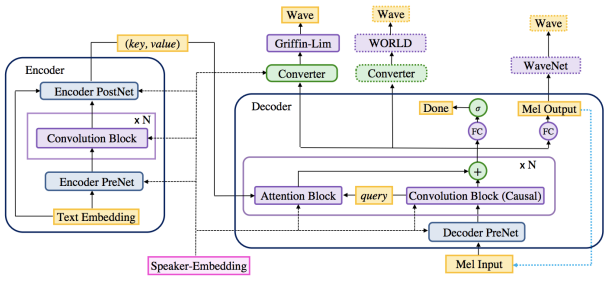

Fig. 1. Architecture of Deep Voice 3

Deep Voice 3 [6] is a multispeaker speech synthesis model.

It takes text as input and outputs mel spectrogram for the predicted voice. It's shown to memorize 2000 speakers to synthesize speech for each of them.

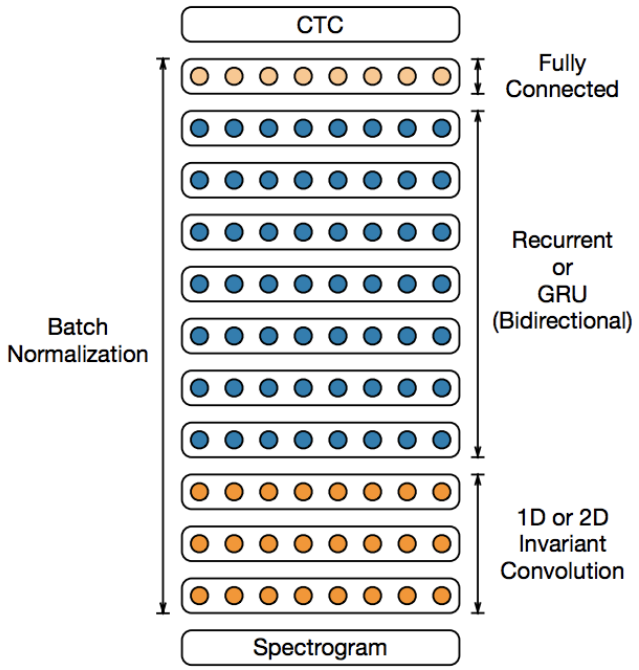

Fig. 2. Architecture of Deep Speech 2

Deep Speech 2 [7], differently from the models above, is a speech recognition model. It takes voice input in mel spectrogram format and outputs the sentence prediction.

III. Model Architecture

For the speech synthesis from a sample voice task, I construct an encoder-decoder model as shown in Figure 3. Initially, there is a decoder which takes the sample voice mel spectrogram as input and converts it to an embedding which represents the features of the sample voice. Decoder architecture is a modified version of Deep Speech 2, which was adapted from an open source implementation. [8]

This embedding and the input sentence is then fed to the next stage of the model, the decoder. Decoder takes these two inputs and outputs a linear spectrogram. Decoder architecture is a modified version of Deep Voice 3 model, which was also adapted from an open source implementation. [9]

There were a few modifications to both models. For Deep Speech 2 model, the softmax layer at the end was removed. Normally, the model outputs a prediction for each timestep. In my model, only the last recurrent output of Deep speech model was used. This output is actually the latent vector in Figure 3. Model was simplified a bit by decreasing RNN hidden size from 768 to 100. Size of the latent vector is 10.

Parameters of Deep Voice 3 model wasn't modified. Deep Voice model has a speaker vector which is originally used for multispeaker synthesis. Instead of inputting speaker id into this vector as in the original Deep Voice method, my model inputs the latent vector into here.

This separated two level architecture has its benefits. It's inspired by the autoencoders. The sample voice is compressed to a latent vector and this is the only information decoder knows about the sample voice. This constraint forces encoder to learn the features of human voice in order to fit as much knowledge about the voice as possible into low dimensional latent space. At the same time, decoder is also forced to learn what to interpret from the input latent vector so that it can synthesize speech with the same style as input voice.

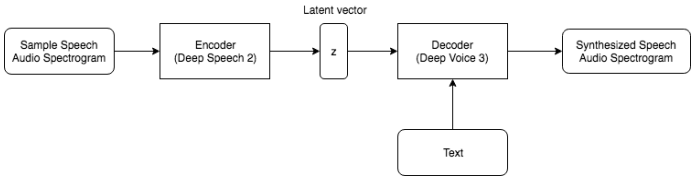

Fig. 3. Architecture of the model

IV. Training

Training process is outlined in Figure 4. VCTK Corpus is used for the training. Training set contains sentence-spectrogram pairs for each speaker. At each step in the training, there are two inputs, text(sentence) and voice spectrogram, and one output, synthesized speech spectrogram. Input voice and sentence is sampled independently from the training set. Spectrogram of the sampled sentence spoken by the same person who speaks in the input voice is found and kept in memory to be used in optimization. With these inputs, one forward pass is executed in the model. Loss is calculated by comparing output spectrogram with the spectrogram of the input sentence. Same loss is used from the Deep Voice 3 implementation. This loss is composed of sub-losses: loss for attention module, loss for spectrogram outputs and done(indicating speech is over) binary loss.

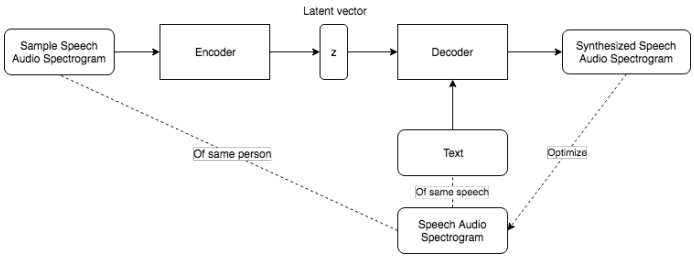

Fig. 4. Training process

Decoder model takes two inputs, latent vector and text. At the start of the training, it's much easier for model to disregard latent vector input and focus on text input, because encoder is also not trained and it's very likely to output noise from a sample voice. This resembles the issue in GAN training [10]: because the generator is not optimized yet and it's just outputting noise at the start, discriminator's job is very

easy, until later when generator starts to catch up. In speech synthesis scenario, I found that encoder is not able to ”catch up”, so decoder only cares about the text, which results in the model always giving the same output for the same sentence, no matter the voice sample input. In other words, the model could generate speech, but not with the expected voice to be imitated. To get around this issue, encoder is trained more than the decoder. This may be regarded as a hyperparameter for this model, specifying the ratio of encoder to decoder training. This hyperparameter is analogous to k hyperparameter in GAN.

At each step, the decision whether to only train the encoder or train both decoder and encoder is given. If whole model is to be trained, gradients are calculated and the whole model is updated according to the gradients. If only the encoder is to be trained, gradients are calculated, decoder weights are frozen and only encoder weights are updated. Adam optimization method is used to train the model. Encoder and full model optimizers do not share state, they are independent.

V. Sound processing

Wav files in VCTK contains lots of long silences, which affects training char-level seq2seq models. Voice files have been scanned by a script which prepares phoneme alignments for all utterances. This preparation step takes around 20 hours for the VCTK dataset. Then, voice files are preprocessed by cutting the silences at the beginning and end of the voices by checking these phoneme timestamps. Low decibel(< 25) parts are also cut. The resultant spectrograms for each voice are ready to be used for training.

VI. Result and Discussion

In the initial testing and debugging process, dataset was not split to training, validation and test set. So VCTK Corpus was fully used in training. Some of the example results were synthesized by using voice inputs from this dataset. Some example results were synthesized by using voice inputs found from internet which are outside training dataset. Those are not preprocessed and they contain background noise, they are not of the same quality overall. Nevertheless, model performs similarly on any of the voice inputs, showing that it can imitate the voices outside training dataset.

Quality of outputs are worse than state of the art results. I did not find the need to conduct an objective survey to measure quality of results, because the difference is obvious. However, the model is still able to differentiate voice inputs and create different speeches for a different speakers, even though it contains some static noise and the voice information is not fully captured. Pitch information is obviously captured, but more vague information like accent is not captured by the model.

Training of the model took 3-4 days on a GTX1070 machine to reach an acceptable point. 1.5M steps(16 samples/step) were when the model could differentiate between speeches and was stable enough. This number could be lowered down quite a lot, just by tweaking the encoder to decoder train ratio hyperparameter.

This model was trained only once to a point where it can successfully synthesize different speeches for different speakers. Only one input sentence was used for encoder, and no hyperparameter tuning was made yet. It’s possible that the model can be improved to a point where it can generate quality outputs. Ideas to achieve that is discussed in Future Work section.

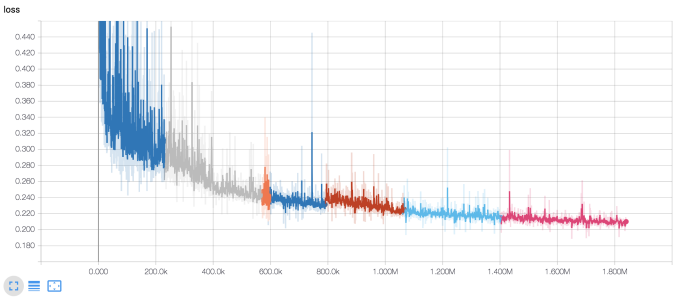

Fig. 5. Loss

In Figure 5, it can be seen that it takes the loss a very long time(600K-800K steps) to get to a stable point. Around step 800K, encoder training ratio was increased in order to force encoder to learn more aggressively. As a result, loss increases a bit, after a while, it starts to fall even lower. The effect of increasing encoder training ratio can be seen in Figure 6 and 7 as well.

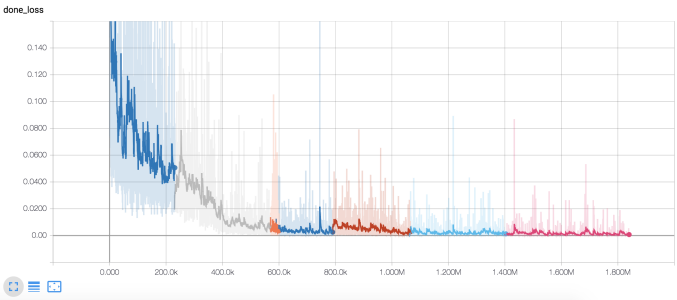

Fig. 6. Done loss

Looking at Figure 6, model could learn where to stop fairly fast. But even if the model is correct on stopping, it may have problems on finishing a sentence. In comparison to alignments in Figure 8, alignments in Figure 9 is showing that model is not always attending in a stable manner. These alignments may correspond to a speech stopping in the middle of a sentence and continue to output the same sound until the end of the spectrogram.

VII. Future Work

There are a lot of work to be done to improve the quality of the synthesized speeches. I will talk about them briefly here.

- In training, encoder to decoder training ratio was updated manually by checking generated speech qualities. A nicer way to do this may be employing a strategy which gradually

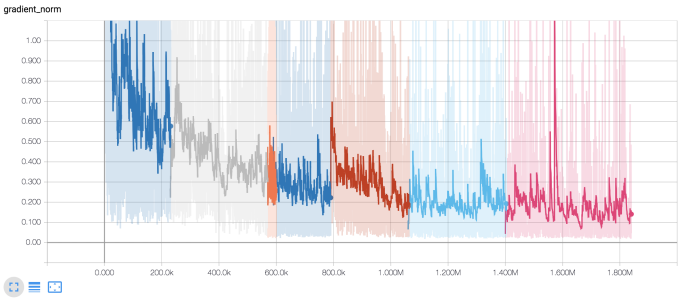

Fig. 7. Gradient Norm

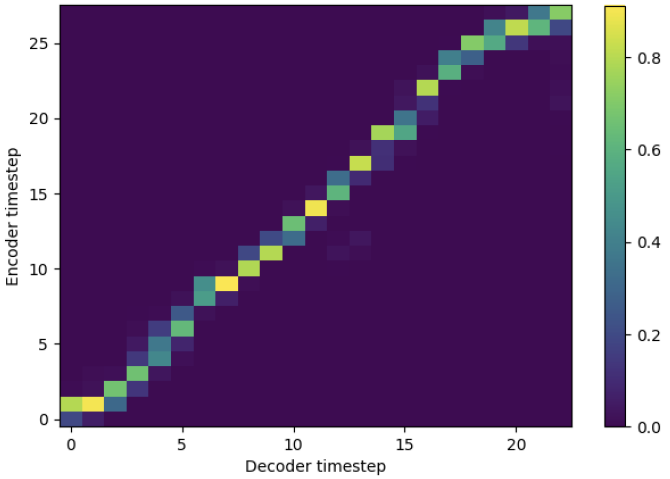

Fig. 8. Alignment Example 1

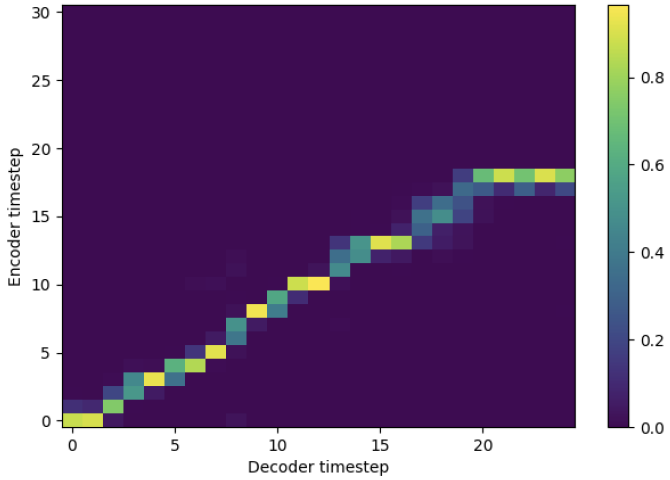

Fig. 9. Alignment Example 2

increases the ratio, as a result increasing the learning rate of encoder more and more.

- Deep Voice 3 and Deep Speech 2 are two complex models. Connecting them one after the other may have resulted in very low gradients in the first layers of the encoder model. Checking the norm of the gradient for these layers and finding a way to overcome this problem(eg. residual connections) may be beneficial, and it may even eliminate the need for the encoder to decoder train ratio hyperparameter.

- Currently, model was trained with only one sample voice input. Joining multiple sentences of the same speaker and giving the joint voice as input to model may be effective, both in training and testing. It's discussed in VoiceLoop paper that longer sample inputs are much more effective in their results.

- This dataset contains speeches from approximately 100 people, each of them with fewer than 400 sentences. Finding a bigger and more diverse dataset may be very helpful.

VIII. Conclusion

It seems that it's possible to imitate speech even with one sentence sampled outside training data. Quality of the generated voice is not on par with state of the art speech synthesis models, but these are the results of a single training without much tuning. This model and training process can be improved to generate better sounding, better imitating speech real time.

References

[1] Baris Ozkuslar. Synthesized speech examples from the model in this paper. https://soundcloud.com/ozkuslar/sets/speech-synthesis, 2018.

[2] Yuxuan Wang et al. Tacotron: Towards end-to-end speech synthesis. *arXiv:1703.10135*, 2017.

[3] Aaron van den Oord et al. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

[4] Jose Sotelo et al. Char2wav: End-to-end speech synthesis. *ICLR 2017*, 2017.

[5] Yaniv Taigman et al. Voiceloop: Voice fitting and synthesis via a phonological loop. *arXiv:1707.06588*, 2017.

[6] Wei Ping et al. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. *arXiv:1710.07654*, 2017.

[7] Dario Amodei et al. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv:1512.02595*, 2015.

[8] Sean Naren. Pytorch implementation of deep speech 2. https://github.com/SeanNaren/deepspeech.pytorch, 2017.

[9] Ryuichi Yamamoto. Pytorch implementation of deep voice 3. https://github.com/r9y9/deepvoice3_pytorch, 2017.

[10] Ian J. Goodfellow et al. Generative adversarial networks. *arXiv:1406.2661*, 2014.