Voice Converter Using DeepSpeech and Tacotron

Sreenithy Chandran
1213391684
Arizona State University

Satyajit Giri
1213037993
Arizona State University

November 5, 2019

Abstract

Usually Voice Converters use a set of training and testing example pairs to train a Machine Learning model like a Gaussian Mixture Model (GMM) or Deep Neural Network (DNN). In many edge devices or applications, it is impractical to acquire this data. In our project we propose a method to remove the need to collect audio samples from the user by combining a DeepMind implementation with a Tacotron implementation. Consequently, we are able to convert input speaker's audio signal to text and used this text to synthesize speech in the voice that Tacotron was trained on. In addition, we show that our proposed pipeline also significantly reduces the need to train on the output user's voice. Last of all, we incorporate the initial project proposal of reducing model size and computational cost by pruning the model, quantizing the weights and creating heap maps.

1 Introduction

Traditionally text to speech synthesis involves a multiple stage process, including text analysis front end, an acoustic model and an audio synthesis model. Towards End-to-End speech synthesis was (Tacotron) (4) was created by researchers at Google to replace the complex multiple stage approach to speech synthesis with an end to end system powered by deep learning. The first Tacotron led to an in increase the mean opinion score on US English and was substantially faster than other state of the art text to speech algorithms.

Similar to Tacotron, DeepSpeech (2) aims to remove the complicated speech recognition pipeline. It's input is a series of overlapping and windowed frames from which spectrogram are taken. Each spectrogram is passed through a series of dense layers to extract good features which are then shared in a Bi-directional Recurrant Neural Network (BRNN) that looks at relationships between frames to determine which character is most likely to occur at that frame.

We believe that the next evolution of text to speech algorithms will be minimizing the algorithms such that it leads to a decrease in memory space and increase in speed. Most deep learning platforms provide tools that help you do this. In particular, TensorFlow Lite has multiple methods to trim, prune, quantize and memory map models robustly.

Finally, we explore various resources and source code to Voice Conversion. We opt into Sprocket's (7) GMM model based approach because it yields good results even with only 30 input audio voices and 30 output audio voices. We have replaced the input audio voice with Tacotron synthesized audio samples, thus removing the need to have the an input user say the same contents in the audio clips of the output user.

2 Division of Work

We roughly divide the project into four stages: Automatic Speech Recognition (ASR), Text to Speech Synthesis (TTS), reducing model weights and increasing model speeds, and finally Voice to Voice Conversion. While both team members were actively involved in both stages: Satyajit was responsible for ASR and model minimization whereas Sreenithy was responsible for TTS and Voice to

Voice Conversion.

3 Automatic Speech Recognition Using DeepSpeech

3.1 Traditional ASR

Speech Recognition is a challenging problem that was traditionally solved by using a multistage process shown in Figure 1. It consists of a Acoustic Model that converts speech to phonemes. Then we pass the phones through a phoneme model that assigns probability of that sound corresponding to a certain character. Finally, these characters are passed through a language model that predicts how likely certain words and letters are to exist together to produce the final text. This multistage process requires expert knowledge at each stage. For example, for just the feature extractor block, many signal processing and speech experts have spent decades trying to determine the best features to describe a speech signal. Consequently often a traditional ASR pipeline is susceptible to overfitting and is not robust to speaker variation, voice variation and noise.

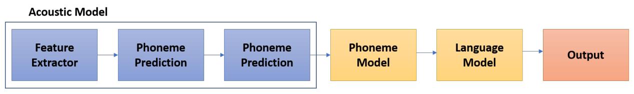


Figure 1: Pipeline of the traditional multistage approach to ASR

3.2 DeepSpeech

In 2014, researchers at Baidu, introduced the concept of DeepSpeech that attempts to replace this multistage pipeline with one end to end deep learning based model (2). We see in Figure 2 that the model starts by taking overlapping frames of the Mel frequency spectrogram of the speech signal. These frames are then passed through either layers of convolution or fully connected layers, where the output is similar to the features that come out of the previous Phoneme Model. Finally, this features are passed through a BRNN, where relationships between features in adjacent frames are studied to finally predict what character at that location would lead to such a combination of features.

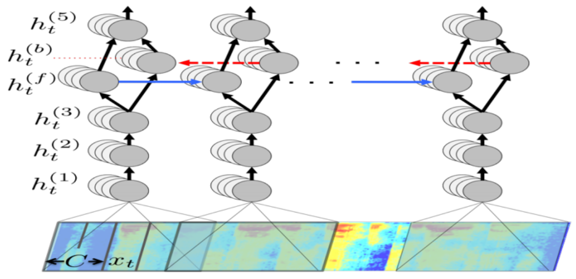


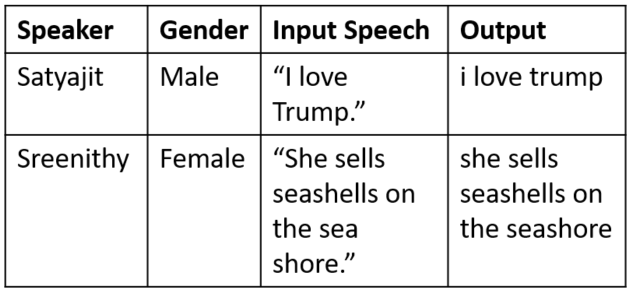
Figure 2: DeepSpeech Model Architecture

3.3 Implementation

We cloned the Mozilla's DeepSpeech repository from github (8) and implemented on Tensorflow. We used a pretrained model, but quantized, trimmed, and memory mapped the model. We observed that the model worked great even after beind reduced for inference. One disappointment was there was no way to implement this model in real time.

3.4 Results

Our results are highlighted in the figure below [Figure 3]. They show that the model was highly accurate and even could correctly identify words from tongue twisters like "she sells seashells on the seashore".



Speaker	Gender	Input Speech	Output
Satyajit	Male	"I love Trump."	i love trump
Sreenithy	Female	"She sells seashells on the sea shore."	she sells seashells on the seashore

Figure 3: DeepSpeech Sample outputs

4 Text to Speech Synthesis Using Tacotron

Tacotron, an integrated end-to-end generative TTS model that takes a character sequence as input and outputs the corresponding spectrogram. Tacotron does not need hand engineered linguistic features or complex components such as an HMM aligner. It can be trained from scratch with random initialization. The backbone of Tacotron is a seq2seq model with attention.

4.1 Model Architecture

At a high-level, this model takes characters as input and produces spectrogram frames, which are then converted to waveforms. We describe the components of it below:
CBFG Module
CBHG consists of a bank of 1-D convolutional filters, followed by highway networks and a bidirectional gated recurrent unit (GRU). CBHG is a powerful module for extracting representations from sequences.
Encoder
The goal of the encoder is to extract robust sequential representations of text. The input to the encoder is a character sequence, where each character is represented as a one-hot vector and embedded into a continuous vector. We then apply a set of non-linear transformations, collectively called a pre-net, to each embedding. A CBHG module transforms the prenet outputs into the final encoder representation used by the attention module.
Decoder
A content-based tanh attention decoder is used, where a stateful recurrent layer produces the attention query at each decoder time step. We concatenate the context vector and the attention RNN cell output to form the input to the decoder RNNs. The output of the decoder is a linear scale spectrogram which is passed through the Griffin Lim Reconstruction to give a waveform.

4.2 Preprocessing data for training Tacatron

For preprocessing the data the following steps are to be adopted

- Obtain $audio, text$ pairs for training
- Get the Linear Scale Spectogram
- Obtain the Mel Scale Spectogram

4.3 Results

We used the trained weights of the tacotron model and quantise it using the memory mapped method, this is then used for inference. We observed that the TTS model works well for short sentences. Even tongue twisters that are hard for people to speak is synthesised clearly however, for very sentences containing more than 10 word the model fails.

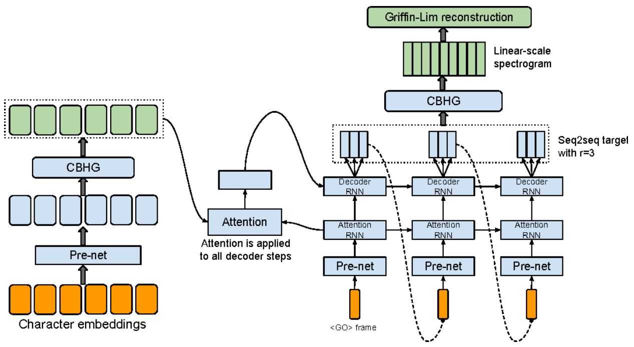


Figure 4: Model architecture. The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech.

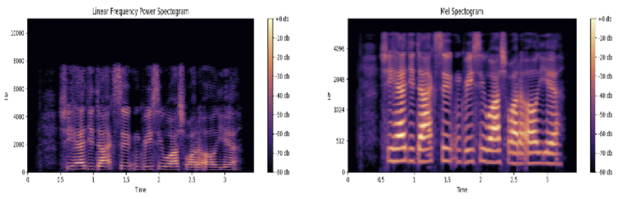


Figure 5: Linear power spectrum and Mel spectrum of a frame

5 Reducing Model Size and Increasing Speed for Production

5.1 Pruning and Trimming

Generally pruning refers to exploring redundancy in neural network after training and removing redundant connections. We propose to prune redundancy between channels of a convolution filter by ranking the effect of each layer on our performance metric. We will remove channels that have little impact on accuracy thus reducing the computational cost and size of the model.

5.2 Quantization

Quantization was a method developed to bring deep learning capabilities to devices at the edge. Edge devices have several characteristics: they have smaller computational abilities and are constrained in power and memory. For these reasons, most Deep Neural Networks (DNNs), which require significant memory to store the models and large amounts of processing capabilities become unsuitable for these edge devices. One method to reduce a neural network is quantization, where we take convolution layers populated with 32 bit floating numbers and quantize it to 8 bits integers.

Krishnamoorthi highlighted the major benefits, designs and approaches to quantization in (3). We highlight the primary benefits below:

- **Model Size** Model sizes are reduced by a factor of 4.
- **Accuracy** Accuracy drops by 2% from quantization.
- **Speed** A quantized model is 2 to 3 times faster.

5.3 Memory Mapping

Traditionally, when we load a model, we allocate an area of memory on the heap and then copy bytes from disk into it. When we memory map a model we exploit mechanisms in the OS and instead the entire contents of a file appear directly in memory. Some benefits of memory mapping are highlighted below:

- **Model Size** Speeds loading
- **Accuracy** Reduces paging (increases performance)

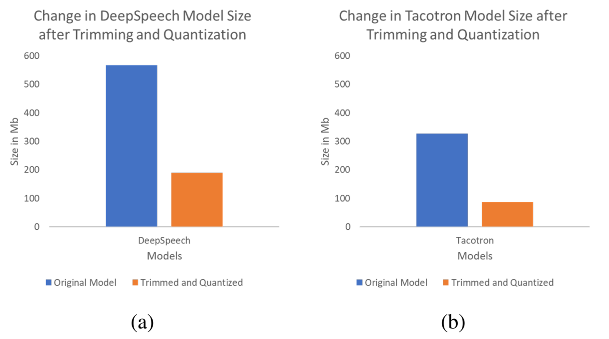


Figure 6: Three times reduction in DeepSpeech and Tacotron size

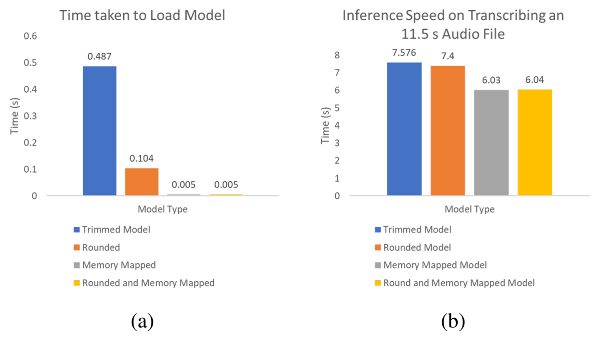


Figure 7: Both load time and inference have been reduced

- **Speed** Does not count towards RAM budget for your app

5.4 Results

After getting the checkpoint model in Tensorflow, we first trimmed and quantized both the DeepSpeech model and the Tacotron model (Figure 6a, 6b). In both cases we observed an almost 3x reduction in models size. This closely matches the reduction of 4x that was predicted by Krishnamoorthy.

We then took the trimmed model and then memory mapped it. We noticed significant speed ups both in loading the model and running forward inference when we used the memory mapped model.

6 Voice Conversion Model

Statistical voice conversion (VC)() is a technique to convert specific non- or paralinguistic information while keeping linguistic information unchanged, and speaker conversion has been studied as a typical application of VC for a few decade. In order to overcome the need for having large dataset for source and target speakers each saying the same sentences we make obtain the source wave file pass it through the Deep Speech model and then pass the obtained sentence as the input to the Tacotron model which gives the output file in the Tacotron trained voice. Having obtained this it is then used as the input during inference for a voice conversion model that has already been trained for the original Target speaker voice and Tacaotron voice to get the sentence as spoken by the target speaker. The voice conversion model used here is GMM trained and we make use of the sprocket library.

6.1 GMM-based VC methods in sprocket

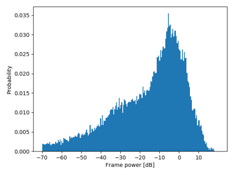
In this section, we describe details of a GMM-based VC method using parallel speech utterances of the source and target speakers (i.e., a parallel dataset), focusing on two typical methods: 1) maximum likelihood parameter generation (MLPG) considering the global variance (GV) based on the GMM 2) vocoder-free VC using the log-spectral differential (DIFFVC) which have been implemented in sprocket.

6.2 Conversion process

For the conversion process, the acoustic features of the source speaker are converted into those of the target speaker using the trained GMM. As the acoustic feature to be converted in sprocket, F0 and the mel-cepstrum are used. Other factors such as the aperiodicity, speaking rate, the temporal structure of the F0 trajectory, and the power trajectory are retained as those of the source voice. Note that arbitrary utterances of the source speaker can be converted into those of the target speaker.

6.3 Training process

Figure shows the training process of the GMM-based VC method. For the training process, the GMM-based VC method carries on following steps: 0) preparation of the parallel speech dataset, 1) acoustic feature extraction, 2) calculation of acoustic feature statistics, 3) time alignment between the source and target feature vectors, and 4) GMM modeling


(a)

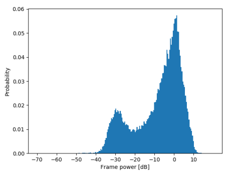

(b)

Figure 8: Power histogram of a)Tacotron speaker b)Obama voice

6.4 Results

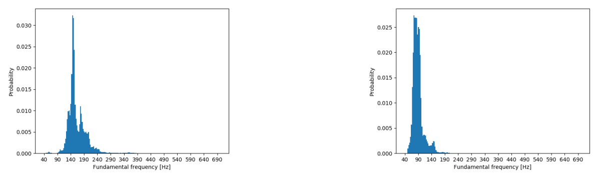
Inorder to train the voice conversion model between Tacatron speech and Obama speech. We downloaded a Obama speech and the waveforms are stored in not a single waveform file but several waveform files by dividing into several utterances of about 5 seconds each. In all we have 30 samples for training Then we obtain the waveforms of the tacotron with the same sentences.

Setting of the speaker-dependent parameters

The F0 range is a representative speaker-dependent parameter for acoustic feature extraction. For normal speech, the F0 range is relatively well approximated as a unimodal distribution. In the F0 extraction process, F0 values of the double and half harmonics are sometimes extracted owing to the analysis errors, and these errors significantly degrade the sound quality of the converted voice. To avoid such errors, the F0 range for each speaker is specified in accordance with the F0 histograms. Based on the figure 3), we manually change the values of minimum F0 and maximum F0 of each speaker

Setting of the pair-dependent parameters

Because the number of mixture components of the GMM strongly affects the conversion quality, it should be carefully set in accordance with the number of training utterances. We use the number of mixture components as 8 when the number of training utterances is 30



(a) (b)

Figure 9: F0 histogram of a)Tacotron speaker b)Obama voice

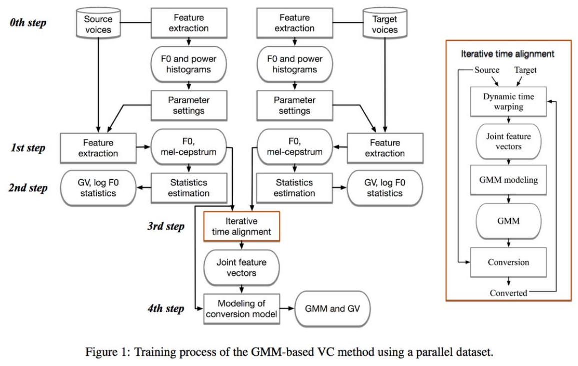


Figure 10: Training process of the GMM-based VC method using a parallel dataset.

7 Results

The first step of the voice converter pipline is to obtain the speech of sample X and pass it through the Deep Speech model and then pass the obtained sentence as the input to the Tacotron model which gives the output file in the Tacotron trained voice. Having obtained this it is then used as the input during inference for a voice conversion model that has already been trained for the original Target speaker voice and Tacaotron voice to get the sentence as spoken by the target speaker. All the converted voice samples are available in the .zip file.

8 Conclusion

We initially set out to reduce model size and increase inference speed of a Tacotron to make it more suitable for edge devices. We achieved that fairly quickly using Tensorflow commands. We then used the DeepSpeech model to recognize speech as text, a Tacotron model to convert the text into a common synthetic voice, and trained a sprocket model to convert that synthetic voice to Obama's. Our output can definitely by identified as synthetic, but it is undeniable that it is Obama-like. We believe that using a better TTS model (we suggest the Tacotron 2) would lead to better outputs. We also think that the Sprocket was trained on simplistic features like MFCCs and can be expanded to include prosody based features to make the audio sound more realistic.

References

[1] K. Kobayashi, T. Toda, S. Nakamura, "F0 transformation techniques for statistical voice conversion with direct waveform modification with spectral differential," Proc. IEEE SLT, pp. 693-700, Dec. 2016

[2] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Y. Ng., "Deep Speech: Scaling up end-to-endspeech recognition", arXiv:1412.5567v2 [cs.CL] 19 Dec 2014

[3] K. R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," arXiv:1806.08342v1

[4] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss,N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le,Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, Tacotron:Towards end-to-end speech synthesis, in Proc. Interspeech,Aug. 2017, pp. 40064010.

[5] Kobayashi, Kazuhiro, and Tomoki Toda. "sprocket: Open-source voice conversion software." Proc. Odyssey 2018 The Speaker and Language Recognition Workshop. 2018.

[6] https://github.com/keithito/tacotron

[7] https://github.com/k2kobayashi/sprocket

[8] https://github.com/mozilla/DeepSpeech