
Deep Voice 2: Multi-Speaker Neural Text-to-Speech

Sercan Ö. Arik*
sercanarik@baidu.com

Gregory Diamos*
gregdiamos@baidu.com

Andrew Gibiansky*
gibianskyandrew@baidu.com

John Miller*
millerjohn@baidu.com

Kainan Peng*
pengkainan@baidu.com

Wei Ping*
pingwei01@baidu.com

Jonathan Raiman*
jonathanraiman@baidu.com

Yanqi Zhou*
zhouyanqi@baidu.com

Baidu Silicon Valley Artificial Intelligence Lab
1195 Bordeaux Dr. Sunnyvale, CA 94089

Abstract

We introduce a technique for augmenting neural text-to-speech (TTS) with low-dimensional trainable speaker embeddings to generate different voices from a single model. As a starting point, we show improvements over the two state-of-the-art approaches for single-speaker neural TTS: Deep Voice 1 and Tacotron. We introduce Deep Voice 2, which is based on a similar pipeline with Deep Voice 1, but constructed with higher performance building blocks and demonstrates a significant audio quality improvement over Deep Voice 1. We improve Tacotron by introducing a post-processing neural vocoder, and demonstrate a significant audio quality improvement. We then demonstrate our technique for multi-speaker speech synthesis for both Deep Voice 2 and Tacotron on two multi-speaker TTS datasets. We show that a single neural TTS system can learn hundreds of unique voices from less than half an hour of data per speaker, while achieving high audio quality synthesis and preserving the speaker identities almost perfectly.

1 Introduction

Artificial speech synthesis, commonly known as text-to-speech (TTS), has a variety of applications in technology interfaces, accessibility, media, and entertainment. Most TTS systems are built with a single speaker voice, and multiple speaker voices are provided by having distinct speech databases or model parameters. As a result, developing a TTS system with support for multiple voices requires much more data and development effort than a system which only supports a single voice.

In this work, we demonstrate that we can build all-neural multi-speaker TTS systems which share the vast majority of parameters between different speakers. We show that not only can a single model generate speech from multiple different voices, but also that significantly less data is required per speaker than when training single-speaker systems.

Concretely, we make the following contributions:

1. We present Deep Voice 2, an improved architecture based on Deep Voice 1 (Arik et al., 2017).
2. We introduce a WaveNet-based (Oord et al., 2016) spectrogram-to-audio neural vocoder, and use it with Tacotron (Wang et al., 2017) as a replacement for Griffin-Lim audio generation.

*Listed alphabetically.

- Using these two single-speaker models as a baseline, we demonstrate multi-speaker neural speech synthesis by introducing trainable speaker embeddings into Deep Voice 2 and Tacotron.

We organize the rest of this paper as follows. Section 2 discusses related work and what makes the contributions of this paper distinct from prior work. Section 3 presents Deep Voice 2 and highlights the differences from Deep Voice 1. Section 4 explains our speaker embedding technique for neural TTS models and shows multi-speaker variants of the Deep Voice 2 and Tacotron architectures. Section 5.1 quantifies the improvement for single speaker TTS through a mean opinion score (MOS) evaluation and Section 5.2 presents the synthesized audio quality of multi-speaker Deep Voice 2 and Tacotron via both MOS evaluation and a multi-speaker discriminator accuracy metric. Section 6 concludes with a discussion of the results and potential future work.

2 Related Work

We discuss the related work relevant to each of our claims in Section 1 in order, starting from single-speaker neural speech synthesis and moving on to multi-speaker speech synthesis and metrics for generative model quality.

With regards to single-speaker speech synthesis, deep learning has been used for a variety of sub-components, including duration prediction (Zen et al., 2016), fundamental frequency prediction (Ronanki et al., 2016), acoustic modeling (Zen and Sak, 2015), and more recently autoregressive sample-by-sample audio waveform generation (e.g., Oord et al., 2016; Mehri et al., 2016). Our contributions build upon recent work in entirely neural TTS systems, including Deep Voice 1 (Arik et al., 2017), Tacotron (Wang et al., 2017), and Char2Wav (Sotelo et al., 2017). While these works focus on building single-speaker TTS systems, our paper focuses on extending neural TTS systems to handle multiple speakers with less data per speaker.

Our work is not the first to attempt a multi-speaker TTS system. For instance, in traditional HMM-based TTS synthesis (e.g., Yamagishi et al., 2009), an average voice model is trained using multiple speakers’ data, which is then adapted to different speakers. DNN-based systems (e.g., Yang et al., 2016) have also been used to build average voice models, with i-vectors representing speakers as additional inputs and separate output layers for each target speaker. Similarly, Fan et al. (2015) uses a shared hidden representation among different speakers with speaker-dependent output layers predicting vocoder parameters (e.g., line spectral pairs, aperiodicity parameters etc.). For further context, Wu et al. (2015) empirically studies DNN-based multi-speaker modeling. More recently, speaker adaptation has been tackled with generative adversarial networks (GANs) (Hsu et al., 2017).

We instead use trainable speaker embeddings for multi-speaker TTS. The approach was investigated in speech recognition (Abdel-Hamid and Jiang, 2013), but is a novel technique in speech synthesis. Unlike prior work which depends on fixed embeddings (e.g. i-vectors), the speaker embeddings used in this work are trained jointly with the rest of the model from scratch, and thus can directly learn the features relevant to the speech synthesis task. In addition, this work does not rely on per-speaker output layers or average voice modeling, which leads to higher-quality synthesized samples and lower data requirements (as there are fewer unique parameters per speaker to learn).

In order to evaluate the distinctiveness of the generated voices in an automated way, we propose using the classification accuracy of a speaker discriminator. Similar metrics such as an “Inception score” have been used for quantitative quality evaluations of GANs for image synthesis (e.g., Salimans et al., 2016). Speaker classification has been studied with both traditional GMM-based methods (e.g., Reynolds et al., 2000) and more recently with deep learning approaches (e.g., Li et al., 2017).

3 Single-Speaker Deep Voice 2

In this section, we present Deep Voice 2, a neural TTS system based on Deep Voice 1 (Arik et al., 2017). We keep the general structure of the Deep Voice 1 (Arik et al., 2017), as depicted in Fig. 1 (the corresponding training pipeline is depicted in Appendix A). Our primary motivation for presenting an improved single-speaker model is to use it as the starting point for a high-quality multi-speaker model.

One major difference between Deep Voice 2 and Deep Voice 1 is the separation of the phoneme duration and frequency models. Deep Voice 1 has a single model to jointly predict phoneme duration

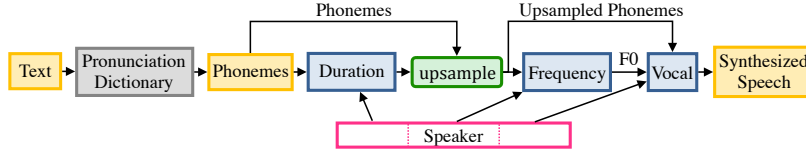


Figure 1: Inference system diagram: first text-phonemes dictionary conversion, second predict phoneme durations, third upsample and generate F_0 , finally feed F_0 and phonemes to vocal model.

and frequency profile (voicedness and time-dependent fundamental frequency, F_0). In Deep Voice 2, the phoneme durations are predicted first and then are used as inputs to the frequency model.

In the subsequent subsections, we present the models used in Deep Voice 2. All models are trained separately using the hyperparameters specified in Appendix B. We will provide a quantitative comparison of Deep Voice 1 and Deep Voice 2 in Section 5.1.

3.1 Segmentation model

Estimation of phoneme locations is treated as an unsupervised learning problem in Deep Voice 2, similar to Deep Voice 1. The segmentation model is convolutional-recurrent architecture with connectionist temporal classification (CTC) loss (Graves et al., 2006) applied to classify phoneme pairs, which are then used to extract the boundaries between them. The major architecture changes in Deep Voice 2 are the addition of batch normalization and residual connections in the convolutional layers. Specifically, Deep Voice 1’s segmentation model computes the output of each layer as

$$h^{(l)} = \text{relu} \left(W^{(l)} * h^{(l-1)} + b^{(l)} \right), \quad (1)$$

where $h^{(l)}$ is the output of the l -th layer, $W^{(l)}$ is the convolution filterbank, $b^{(l)}$ is the bias vector, and $*$ is the convolution operator. In contrast, Deep Voice 2’s segmentation model layers instead compute

$$h^{(l)} = \text{relu} \left(h^{(l-1)} + \text{BN} \left(W^{(l)} * h^{(l-1)} \right) \right), \quad (2)$$

where BN is batch normalization (Ioffe and Szegedy, 2015). In addition, we find that the segmentation model often makes mistakes for boundaries between silence phonemes and other phonemes, which can significantly reduce segmentation accuracy on some datasets. We introduce a small post-processing step to correct these mistakes: whenever the segmentation model decodes a silence boundary, we adjust the location of the boundary with a silence detection heuristic.²

3.2 Duration Model

In Deep Voice 2, instead of predicting a continuous-valued duration, we formulate duration prediction as a sequence labeling problem. We discretize the phoneme duration into log-scaled buckets, and assign each input phoneme to the bucket label corresponding to its duration. We model the sequence by a conditional random field (CRF) with pairwise potentials at output layer (Lample et al., 2016). During inference, we decode discretized durations from the CRF using the Viterbi forward-backward algorithm. We find that quantizing the duration prediction and introducing the pairwise dependence implied by the CRF improves synthesis quality.

3.3 Frequency Model

After decoding from the duration model, the predicted phoneme durations are upsampled from a per-phoneme input features to a per-frame input for the frequency model.³ Deep Voice 2 frequency

²We compute the smoothed normalized audio power as $p[n] = (x[n]^2 / x_{\max}^2) * g[n]$, where $x[n]$ is the audio signal, $g[n]$ is the impulse response of a Gaussian filter, x_{\max} is the maximum value of $x[n]$ and $*$ is one-dimensional convolution operation. We assign the silence phoneme boundaries when $p[n]$ exceeds a fixed threshold. The optimal parameter values for the Gaussian filter and the threshold depend on the dataset and audio sampling rate.

³Each frame is ensured to be 10 milliseconds. For example, if a phoneme lasts 20 milliseconds, the input features corresponding to that phoneme will be repeated in 2 frames. If it lasts less than 10 milliseconds, it is extend to a single frame.

model consists of multiple layers: firstly, bidirectional gated recurrent unit (GRU) layers (Cho et al., 2014) generate hidden states from the input features. From these hidden states, an affine projection followed by a sigmoid nonlinearity produces the probability that each frame is voiced. Hidden states are also used to make two separate normalized F_0 predictions. The first prediction, f_{GRU} , is made with a single-layer bidirectional GRU followed by an affine projection. The second prediction, f_{conv} , is made by adding up the contributions of multiple convolutions with varying convolution widths and a single output channel. Finally, the hidden state is used with an affine projection and a sigmoid nonlinearity to predict a mixture ratio ω , which is used to weigh the two normalized frequency predictions and combine them into

$$f = \omega \cdot f_{\text{GRU}} + (1 - \omega) \cdot f_{\text{conv}}. \quad (3)$$

The normalized prediction f is then converted to the true frequency F_0 prediction via

$$F_0 = \mu_{F_0} + \sigma_{F_0} \cdot f, \quad (4)$$

where μ_{F_0} and σ_{F_0} are, respectively, the mean and standard deviation of F_0 for the speaker the model is trained on. We find that predicting F_0 with a mixture of convolutions and a recurrent layer performs better than predicting with either one individually. We attribute this to the hypothesis that including the wide convolutions reduces the burden for the recurrent layers to maintain state over a large number of input frames, while processing the entire context information efficiently.

3.4 Vocal Model

The Deep Voice 2 vocal model is based on a WaveNet architecture (Oord et al., 2016) with a two-layer bidirectional QRNN (Bradbury et al., 2017) conditioning network, similar to Deep Voice 1. However, we remove the 1×1 convolution between the gated tanh nonlinearity and the residual connection. In addition, we use the same conditioner bias for every layer of the WaveNet, instead of generating a separate bias for every layer as was done in Deep Voice 1.⁴

4 Multi-Speaker Models with Trainable Speaker Embeddings

In order to synthesize speech from multiple speakers, we augment each of our models with a single low-dimensional speaker embedding vector per speaker. Unlike previous work, our approach does not rely on per-speaker weight *matrices* or *layers*. Speaker-dependent parameters are stored in a very low-dimensional vector and thus there is near-complete weight sharing between speakers. We use speaker embeddings to produce recurrent neural network (RNN) initial states, nonlinearity biases, and multiplicative gating factors, used throughout the networks. Speaker embeddings are initialized randomly with a uniform distribution over $[-0.1, 0.1]$ and trained jointly via backpropagation; each model has its own set of speaker embeddings.

To encourage each speaker’s unique voice signature to influence the model, we incorporate the speaker embeddings into multiple portions of the model. Empirically, we find that simply providing the speaker embeddings to the input layers does not work as well for any of the presented models besides the vocal model, possibly due to the high degree of residual connections present in the WaveNet and due to the difficulty of learning high-quality speaker embeddings. We observed that several patterns tend to yield high performance:

- **Site-Specific Speaker Embeddings:** For every use site in the model architecture, transform the shared speaker embedding to the appropriate dimension and form through an affine projection and a nonlinearity.
- **Recurrent Initialization:** Initialize recurrent layer hidden states with site-specific speaker embeddings.
- **Input Augmentation:** Concatenate a site-specific speaker embedding to the input at every timestep of a recurrent layer.
- **Feature Gating:** Multiply layer activations elementwise with a site-specific speaker embedding to render adaptable information flow.⁵

⁴We find that these changes reduce model size by a factor of ~ 7 and speed up inference by $\sim 25\%$, while yielding no perceptual change in quality. However, we do not focus on demonstrating these claims in this paper.

⁵We hypothesize that feature gating lets the model learn the union of all necessary features while allowing speaker embeddings to determine what features are used for each speaker and how much influence they will have on the activations.

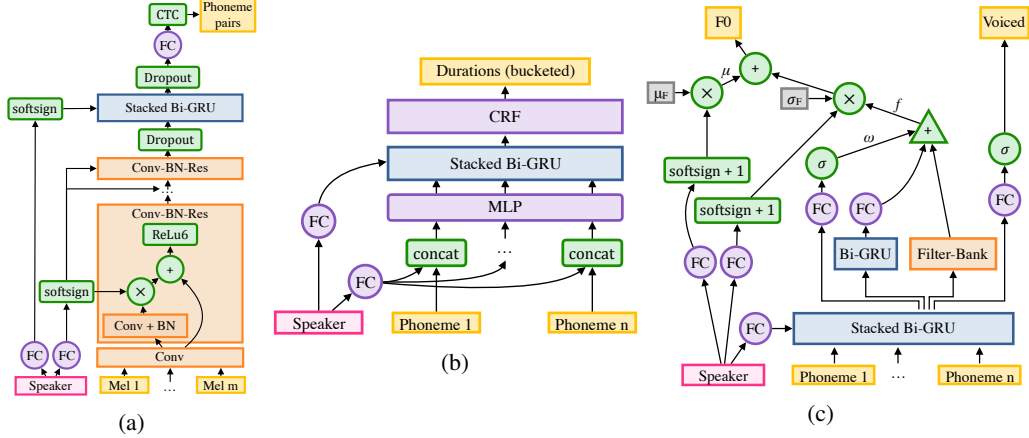


Figure 2: Architecture for the multi-speaker (a) segmentation, (b) duration, and (c) frequency model.

Next, we describe how speaker embeddings are used in each architecture.

4.1 Multi-Speaker Deep Voice 2

The Deep Voice 2 models have separate speaker embeddings for each model. Yet, they can be viewed as chunks of a larger speaker embedding, which are trained independently.

4.1.1 Segmentation Model

In multi-speaker segmentation model, we use feature gating in the residual connections of the convolution layers. Instead of Eq. (2), we multiply the batch-normalized activations by a site-specific speaker embedding:

$$h^{(l)} = \text{relu} \left(h^{(l-1)} + \text{BN} \left(W * h^{(l-1)} \right) \cdot g_s \right), \quad (5)$$

where g_s is a site-specific speaker embedding. The same site-specific embedding is shared for all the convolutional layers. In addition, we initialize each of the recurrent layers with a second site specific embedding. Similarly, each layer shares the same site-specific embedding, rather than having a separate embedding per layer.

4.1.2 Duration Model

The multi-speaker duration model uses speaker-dependent recurrent initialization and input augmentation. A site-specific embedding is used to initialize RNN hidden states, and another site-specific embedding is provided as input to the first RNN layer by concatenating it to the feature vectors.

4.1.3 Frequency Model

The multi-speaker frequency model uses recurrent initialization, which initializes the recurrent layers (except for the recurrent output layer) with a single site-specific speaker-embedding. As described in Section 3.3, the recurrent and convolutional output layers in the single-speaker frequency model predict a *normalized* frequency, which is then converted into the true F_0 by a fixed linear transformation. The linear transformation depends on the mean and standard deviation of observed F_0 for the speaker. These values vary greatly between speakers: male speakers, for instance, tend to have a much lower mean F_0 . To better adapt to these variations, we make the mean and standard deviation trainable model parameters and multiply them by scaling terms which depend on the speaker embeddings. Specifically, instead of Eq. (4), we compute the F_0 prediction as

$$F_0 = \mu_{F_0} \cdot (1 + \text{softsign} (V_\mu^T g_f)) + \sigma_{F_0} \cdot (1 + \text{softsign} (V_\sigma^T g_f)) \cdot f, \quad (6)$$

where g_f is a site-specific speaker embedding, μ_{F_0} and σ_{F_0} are trainable scalar parameters initialized to the F_0 mean and standard deviation on the dataset, and V_μ and V_σ are trainable parameter vectors.

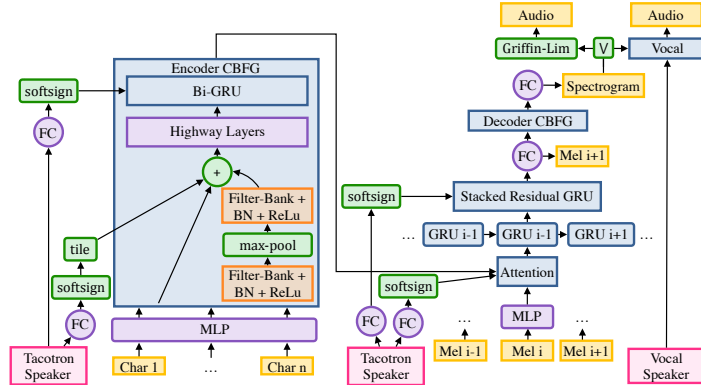


Figure 3: Tacotron with speaker conditioning in the Encoder CBHG module and decoder with two ways to convert spectrogram to audio: Griffin-Lim or our speaker-conditioned Vocal model.

4.1.4 Vocal Model

The multi-speaker vocal model uses only input augmentation, with the site-specific speaker embedding concatenated onto each input frame of the conditioner. This differs from the global conditioning suggested in [Oord et al. \(2016\)](#) and allows the speaker embedding to influence the local conditioning network as well.

Without speaker embeddings, the vocal model is still able to generate somewhat distinct-sounding voices because of the distinctive features provided by the frequency and duration models. Yet, having speaker embeddings in the vocal model increases the audio quality. We indeed observe that the embeddings converge to a meaningful latent space.

4.2 Multi-Speaker Tacotron

In addition to extending Deep Voice 2 with speaker embeddings, we also extend Tacotron ([Wang et al., 2017](#)), a sequence-to-sequence character-to-waveform model. When training multi-speaker Tacotron variants, we find that model performance is highly dependent on model hyperparameters, and that some models often fail to learn attention mechanisms for a small subset of speakers. We also find that if the speech in each audio clip does not start at the same timestep, the models are much less likely to converge to a meaningful attention curve and recognizable speech; thus, we trim all initial and final silence in each audio clip. Due to the sensitivity of the model to hyperparameters and data preprocessing, we believe that additional tuning may be necessary to obtain maximal quality. Thus, our work focuses on demonstrating that Tacotron, like Deep Voice 2, is capable of handling multiple speakers through speaker embeddings, rather than comparing the quality of the two architectures.

4.2.1 Character-to-Spectrogram Model

The Tacotron character-to-spectrogram architecture consists of a convolution-bank-highway-GRU (CBHG) encoder, an attentional decoder, and a CBHG post-processing network. Due to the complexity of the architecture, we leave out a complete description and instead focus on our modifications.

We find that incorporating speaker embeddings into the CBHG post-processing network degrades output quality, whereas incorporating speaker embeddings into the character encoder is necessary. Without a speaker-dependent CBHG encoder, the model is incapable of learning its attention mechanism and cannot generate meaningful output (see Appendix D.2 for speaker-dependent attention visualizations). In order to condition the encoder on the speaker, we use one site-specific embedding as an extra input to each highway layer at each timestep and initialize the CBHG RNN state with a second site-specific embedding.

We also find that augmenting the decoder with speaker embeddings is helpful. We use one site-specific embedding as an extra input to the decoder pre-net, one extra site-specific embedding as the initial attention context vector for the attentional RNN, one site-specific embedding as the initial decoder GRU hidden state, and one site-specific embedding as a bias to the tanh in the content-based attention mechanism.

Model	Samp. Freq.	MOS
Deep Voice 1	16 KHz	2.05 \pm 0.24
Deep Voice 2	16 KHz	2.96 \pm 0.38
Tacotron (Griffin-Lim)	24 KHz	2.57 \pm 0.28
Tacotron (WaveNet)	24 KHz	4.17 \pm 0.18

Table 1: Mean Opinion Score (MOS) evaluations with 95% confidence intervals of Deep Voice 1, Deep Voice 2, and Tacotron. Using the crowdMOS toolkit, batches of samples from these models were presented to raters on Mechanical Turk. Since batches contained samples from all models, the experiment naturally induces a comparison between the models.

4.2.2 Spectrogram-to-Waveform Model

The original Tacotron implementation in (Wang et al., 2017) uses the Griffin-Lim algorithm to convert spectrograms to time-domain audio waveforms by iteratively estimating the unknown phases.⁶ We observe that minor noise in the input spectrogram causes noticeable estimation errors in the Griffin-Lim algorithm and the generated audio quality is degraded. To produce higher quality audio using Tacotron, instead of using Griffin-Lim, we train a WaveNet-based neural vocoder to convert from linear spectrograms to audio waveforms. The model used is equivalent to the Deep Voice 2 vocal model, but takes linear-scaled log-magnitude spectrograms instead of phoneme identity and F_0 as input. The combined Tacotron-WaveNet model is shown in Fig. 3. As we will show in Section 5.1, WaveNet-based neural vocoder indeed significantly improves single-speaker Tacotron as well.

5 Results

In this section, we will present the results on both single-speaker and multi-speaker speech synthesis using the described architectures. All model hyperparameters are presented in Appendix B.

5.1 Single-Speaker Speech Synthesis

We train Deep Voice 1, Deep Voice 2, and Tacotron on an internal English speech database containing approximately 20 hours of single-speaker data. The intermediate evaluations of models in Deep Voice 1 and Deep Voice 2 can be found in Table 3 within Appendix A. We run an MOS evaluation using the crowdMOS framework (Ribeiro et al., 2011) to compare the quality of samples (Table 1). The results show conclusively that the architecture improvements in Deep Voice 2 yield significant gains in quality over Deep Voice 1. They also demonstrate that converting Tacotron-generated spectrograms to audio using WaveNet is preferable to using the iterative Griffin-Lim algorithm.

5.2 Multi-Speaker Speech Synthesis

We train all the aforementioned models on the VCTK dataset with 44 hours of speech, which contains 108 speakers with approximately 400 utterances each. We also train all models on an internal dataset of audiobooks, which contains 477 speakers with 30 minutes of audio each (for a total of \sim 238 hours). The consistent sample quality observed from our models indicates that our architectures can easily learn hundreds of distinct voices with a variety of different accents and cadences. We also observe that the learned embeddings lie in a meaningful latent space (see Fig. 4 as an example and Appendix D for more details).

In order to evaluate the quality of the synthesized audio, we run MOS evaluations using the crowdMOS framework, and present the results in Table 2. We purposefully include ground truth samples in the set being evaluated, because the accents in datasets are likely to be unfamiliar to our North American crowdsourced raters and will thus be rated poorly due to the accent rather than due to the model quality. By including ground truth samples, we are able to compare the MOS of the models with the ground truth MOS and thus evaluate the model quality rather than the data quality; however, the resulting MOS may be lower, due to the implicit comparison with the ground truth samples. Overall, we observe that the Deep Voice 2 model can approach an MOS value that is close to the ground truth, when low sampling rate and companding/expanding taken into account.

⁶Estimation of the unknown phases is done by repeatedly converting between frequency and time domain representations of the signal using the short-time Fourier transform and its inverse, substituting the magnitude of each frequency component to the predicted magnitude at each step.

Dataset	Multi-Speaker Model	Samp. Freq.	MOS	Acc.
VCTK	Deep Voice 2 (20-layer WaveNet)	16 KHz	2.87±0.13	99.9%
VCTK	Deep Voice 2 (40-layer WaveNet)	16 KHz	3.21±0.13	100 %
VCTK	Deep Voice 2 (60-layer WaveNet)	16 KHz	3.42±0.12	99.7%
VCTK	Deep Voice 2 (80-layer WaveNet)	16 KHz	3.53±0.12	99.9%
VCTK	Tacotron (Griffin-Lim)	24 KHz	1.68±0.12	99.4%
VCTK	Tacotron (20-layer WaveNet)	24 KHz	2.51±0.13	60.9%
VCTK	Ground Truth Data	48 KHz	4.65±0.06	99.7%
Audiobooks	Deep Voice 2 (80-layer WaveNet)	16 KHz	2.97±0.17	97.4%
Audiobooks	Tacotron (Griffin-Lim)	24 KHz	1.73±0.22	93.9%
Audiobooks	Tacotron (20-layer WaveNet)	24 KHz	2.11±0.20	66.5%
Audiobooks	Ground Truth Data	44.1 KHz	4.63±0.04	98.8%

Table 2: MOS and classification accuracy for all multi-speaker models. To obtain MOS, we use crowdMOS toolkit as detailed in Table 1. We also present classification accuracies of the speaker discriminative models (see Appendix E for details) on the samples, showing that the synthesized voices are as distinguishable as ground truth audio.

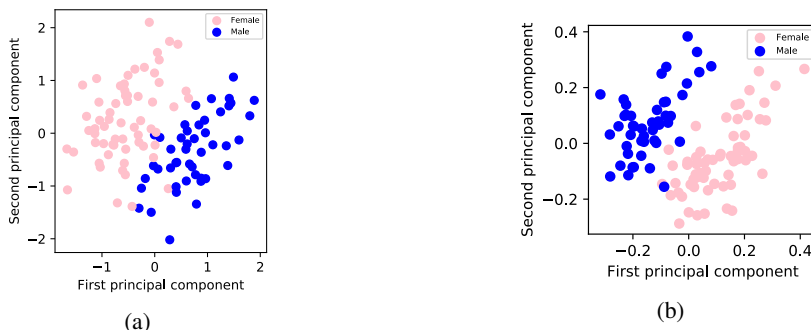


Figure 4: Principal components of the learned speaker embeddings for the (a) 80-layer vocal model and (b) character-to-spectrogram model for VCTK dataset. See Appendix D.3 for details.

A multi-speaker TTS system with high sample quality but indistinguishable voices would result in high MOS, but fail to meet the desired objective of reproducing the input voices accurately. To show that our models not only generate high quality samples, but also generate *distinguishable* voices, we also measure the classification accuracy of a speaker discriminative model on our generated samples. The speaker discriminative is a convolutional network trained to classify utterances to their speakers, trained on the same dataset as the TTS systems themselves. If the voices were indistinguishable (or the audio quality was low), the classification accuracy would be much lower for synthesized samples than it is for the ground truth samples. As we demonstrate in Table 2, classification accuracy demonstrates that samples generated from our models are as distinguishable as the ground truth samples (see Appendix E for more details). The classification accuracy is only significantly lower for Tacotron with WaveNet, and we suspect that generation errors in the spectrogram are exacerbated by the WaveNet, as it is trained with ground truth spectrograms.

6 Conclusion

In this work, we explore how entirely-neural speech synthesis pipelines may be extended to multi-speaker text-to-speech via low-dimensional trainable speaker embeddings. We start by presenting Deep Voice 2, an improved single-speaker model. Next, we demonstrate the applicability of our technique by training both multi-speaker Deep Voice 2 and multi-speaker Tacotron models, and evaluate their quality through MOS. In conclusion, we use our speaker embedding technique to create high quality text-to-speech systems and conclusively show that neural speech synthesis models can learn effectively from small amounts of data spread among hundreds of different speakers.

The results presented in this work suggest many directions for future research. Future work may test the limits of this technique and explore how many speakers these models can generalize to, how little data is truly required per speaker for high quality synthesis, whether new speakers can be added to a system by fixing model parameters and solely training new speaker embeddings, and whether the speaker embeddings can be used as a meaningful vector space, as is possible with word embeddings.

References

- O. Abdel-Hamid and H. Jiang. Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code. In *ICASSP*, 2013.
- S. O. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, J. Raiman, S. Sengupta, and M. Shoeybi. Deep voice: Real-time neural text-to-speech. In *ICML*, 2017.
- J. Bradbury, S. Merity, C. Xiong, and R. Socher. Quasi-recurrent neural networks. In *ICLR*, 2017.
- K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.
- Y. Fan, Y. Qian, F. K. Soong, and L. He. Multi-speaker modeling and speaker adaptation for DNN-based TTS synthesis. In *IEEE ICASSP*, 2015.
- A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006.
- C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang. Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. *arXiv:1704.00849*, 2017.
- S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.
- G. Lample, M. Ballesteros, K. Kawakami, S. Subramanian, and C. Dyer. Neural architectures for named entity recognition. In *Proc. NAACL-HLT*, 2016.
- C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu. Deep speaker: an end-to-end neural speaker embedding system. *arXiv preprint arXiv:1705.02304*, 2017.
- S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio. SampleRNN: An unconditional end-to-end neural audio generation model. *arXiv:1612.07837*, 2016.
- A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016.
- D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000.
- F. Ribeiro, D. Florêncio, C. Zhang, and M. Seltzer. Crowdmos: An approach for crowdsourcing mean opinion score studies. In *IEEE ICASSP*, 2011.
- S. Ronanki, O. Watts, S. King, and G. E. Henter. Median-based generation of synthetic speech durations using a non-parametric approach. *arXiv:1608.06134*, 2016.
- T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NIPS*, 2016.
- J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio. Char2wav: End-to-end speech synthesis. In *ICLR2017 workshop submission*, 2017.
- Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, et al. Tacotron: Towards end-to-end speech synthesis. In *Interspeech*, 2017.
- Z. Wu, P. Swietojanski, C. Veaux, S. Renals, and S. King. A study of speaker adaptation for DNN-based speech synthesis. In *Interspeech*, 2015.
- J. Yamagishi, T. Nose, H. Zen, Z.-H. Ling, T. Toda, K. Tokuda, S. King, and S. Renals. Robust speaker-adaptive hmm-based text-to-speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 2009.
- S. Yang, Z. Wu, and L. Xie. On the training of DNN-based average voice model for speech synthesis. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), Asia-Pacific*, 2016.
- H. Zen and H. Sak. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *IEEE ICASSP*, 2015.
- 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:1606.06061*, 2016.

Appendices

A Training Deep Voice 2

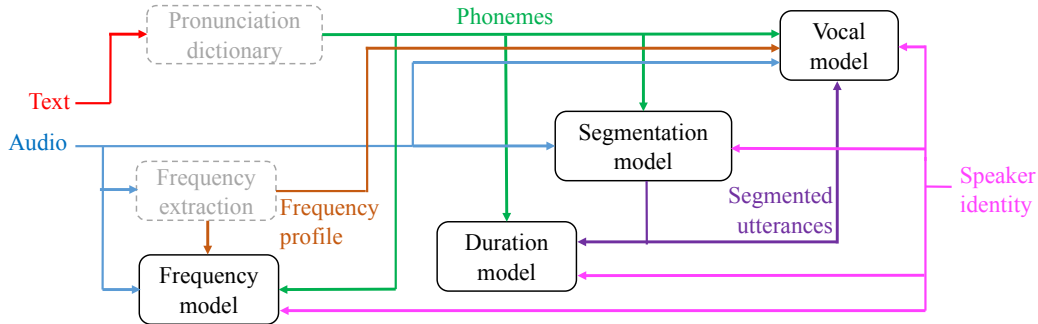


Figure 5: System diagram for training procedure for Deep Voice 2.

For convenience, we depict the training procedure for Deep Voice 2 in Fig. 5, similar to the figure in Arik et al. (2017). The pronunciation dictionary can be substituted by a grapheme-to-phoneme model as in Arik et al. (2017), to adapt for unseen words in the pronunciation dictionary.

For frequency extraction during training, similar to Arik et al. (2017), we use the Praat software, which may be substituted by another fundamental frequency and voicedness estimation algorithm.

The intermediate evaluation metrics for Deep Voice 2 models are compared to their Deep Voice 1 counterparts in Table 3. We obtain significant improvements in these evaluation metrics for all the models, but we also note that the overall synthesis quality cannot be fully quantified by these metrics.

Model	Evaluation metric	Deep Voice 1	Deep Voice 2
Segmentation	Phoneme pair error rate	7 %	0.1 %
Duration	Mean absolute error	38.0 ms	17.5 ms
Frequency	Mean absolute error	29.4 Hz	24.5 Hz

Table 3: Comparison of segmentation, duration, frequency models for Deep Voice 1 and Deep Voice 2, both trained with the same single-speaker data and evaluated as described in Arik et al. (2017).

B Model Hyperparameters

All hyperparameters of the models used in this paper are provided in Table 4.

To speed up the training of character-to-spectrogram model in Tacotron in our experiments, we added a penalty term in the form CTC loss (obtained from the attention hidden states) to the overall loss function. We do not have a clear conclusive evidence that it improves the overall audio quality but we observed faster convergence in some cases.

Learning rate is presented as a triple $\ell - r - s$, which means that the initial learning rate of ℓ was decayed by a factor of r every s iterations. All models use the Adam optimization technique (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\varepsilon = 10^{-8}$.

Convolutional layers are presented as $l \times, o, h \times w$ filters, which means that there are l convolutional layers, and o (output channels) filters in each layer. The filter size is $h \times w$, where height h is in frequency bins and width w is in time frames.

Any unspecified details or hyperparameters for Deep Voice 2 are identical to those from the best models used in the original implementation of Deep Voice 1 (Arik et al., 2017). Similarly, any unspecified details or hyperparameters for our Tacotron implementation are identical to those from the best models used in the original implementation of Tacotron (Wang et al., 2017).

		Single-Speaker	VCTK	Audiobooks
Segmentation	Number of MFCCs	40	40	40
	Convolutional layers	4×, 128 9 × 5 filters	4×, 64 9 × 5 filters	5×, 128 9 × 5 filters
	Recurrent layers (Bi-GRU)	4×, 512-wide	4×, 1024-wide	4×, 1024-wide
	Dropout keep probability	0.95	0.85	0.85
	Learning rate	$10^{-3} - 0.95 - 400$	$2 \cdot 10^{-4} - 0.95 - 1000$	$2 \cdot 10^{-4} - 0.95 - 2000$
	Silence threshold	0.1	0.1	0.1
	Gaussian width	0.25 ms	0.25 ms	0.25 ms
	Batch size	8	8	8
Duration	Speaker embedding size	N/A	16	32
	Fully-connected	2×, 256 units	4×, 256 units	4×, 256 units
	Recurrent layers (Bi-GRU)	4×, 256-wide	4×, 512-wide	4×, 512-wide
	Dropout keep probability	0.8	0.85	0.85
	Output buckets	100	250	300
	Learning rate	$3 \cdot 10^{-4} - 0.9 - 300$	$6 \cdot 10^{-4} - 0.9 - 400$	$3 \cdot 10^{-4} - 0.9 - 800$
	Batch size	128	32	32
	Speaker embedding size	N/A	16	32
Frequency	Hidden layers (Bi-GRU)	3×, 256-wide	3×, 512-wide	3×, 512-wide
	Output dimension	32	32	64
	Convolution widths	5, 10, 20	3, 6, 15, 30	6 9 18 35
	Learning rate	$10^{-3} - 0.9 - 300$	$4 \cdot 10^{-4} - 0.9 - 300$	$4 \cdot 10^{-4} - 0.9 - 300$
	Batch size	32	32	32
	Speaker embedding size	N/A	16	16
Vocal	Layers	60	20/40/60/80	80
	Learning rate	$10^{-3} - 0.9886 - 1000$	$10^{-3} - 0.9886 - 1000$	$10^{-3} - 0.9886 - 1000$
	Batch size	8	8	8
	Speaker embedding size	N/A	16	16
Character-to-spectrogram	Enc.-CBHG bank size	16	16	16
	Enc.-CBHG channels	128	128	128
	Enc.-CBHG recurrent size	128	128	128
	Enc.-CBHG highway layers	4	4	4
	Enc.-CBHG maxpool width	2	2	2
	Enc.-CBHG proj. sizes	128, 128	128, 128	128, 128
	Enc.-CBHG proj. width	3	3	3
	Decoder layers	3	3	3
	Dropout keep probability	0.5	0.8	0.8
	Attention size	128	256	512
	Attention state size	256	256	256
	Decoder prenet sizes	256, 128	256, 128	256, 128, 64
	Post-CBHG bank size	8	8	8
	Post-CBHG channels	128	512	512
	Post-CBHG conv. widths	3	3	3
	Post-CBHG recurrent size	128	256	256
	Post-CBHG highway layers	4	4	4
	Post-CBHG maxpool width	2	2	2
	Reduction factor	4	4	4
	CTC loss coefficient	0.01	0.01	0.01
Learning rate	$10^{-3} - 1 - \text{N/A}$	$10^{-3} - 0.95 - 3000$	$10^{-3} - 0.95 - 3000$	
Batch size	16	16	16	
Speaker embedding size	N/A	32	32	

Table 4: Model hyperparameters for all models presented in this paper.

C Training Time for Multi-speaker Models

We present the details of training time for multi-speaker models in Table 5. We use the same optimized WaveNet inference kernels described in Deep Voice 1. For detailed analysis and corresponding techniques to optimize the inference time, we refer the readers to (Arik et al., 2017).

Model	Hardware	Time per iteration	Number of iterations	Total time
Segmentation	1 Titan X GPU	1200 ms	90k	30 hours
Duration	1 Tesla K40 GPU	320 ms	60k	4.5 hours
Frequency	1 Tesla K40 GPU	1100 ms	25k	7.5 hours
Vocal (20 layer)	8 Titan X GPUs	280 ms	350k	27 hours
Vocal (40 layer)	8 Titan X GPUs	450 ms	350k	44 hours
Vocal (60 layer)	8 Titan X GPUs	600 ms	500k	83 hours
Vocal (80 layer)	8 Titan X GPUs	780 ms	500k	108 hours
Character-to-spectrogram	1 TitanX GPU	450 ms	200k	25 hours

Table 5: Training time details of multi-speaker Deep Voice 2 and Tacotron models on VCTK dataset.

D Interpretation of Learned Embeddings

In this section, we explore the consequences of speaker-dependent models on intermediate model outputs, model activations, and the distributions of the learned embeddings.

D.1 Speaker-Dependent Fundamental Frequency Profiles

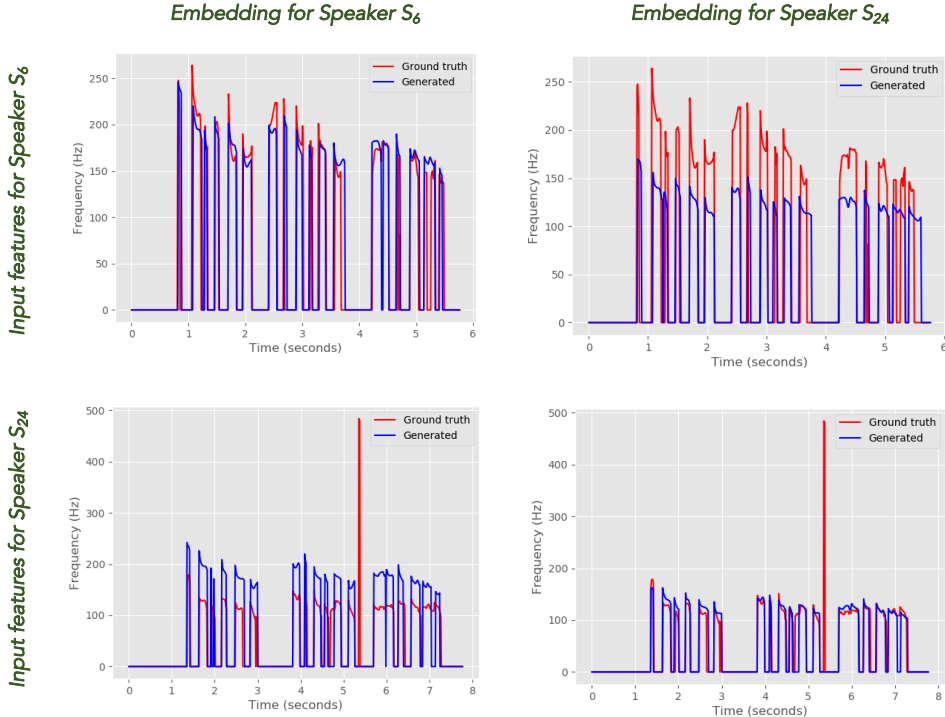


Figure 6: Time-dependent fundamental frequency profiles (collectively including the F_0 and voiced-ness) generated by the model when the inputs and embeddings correspond to Speaker S_6 or S_{24} . Speaker S_6 is a 23-year-old female with a Southern England accent and Speaker S_{24} is a 24-year-old male with an Indian accent. The pronounced sentence is "Six spoons of fresh snow peas five thick slabs of blue cheese and maybe a snack for her brother Bob". Spikes in the ground truth are caused by the estimation errors using the Praat software (Arik et al., 2017), and we observe that our frequency model can compensate them in some cases, learning from the other correct ground truth samples.

To demonstrate the significance of the speaker embeddings, we run inference for the frequency model with the speaker embedding vectors corresponding to the actual speaker and a different speaker. As shown in Fig. 6, while the input phoneme features are dominant in determining the overall shape of the fundamental frequency profiles, the actual values are highly speaker dependent. For example, when the speaker embedding vector of a male is substituted with the speaker embedding vector of a female, the output frequency profile would cause generation of a female voice despite all the other input features correspond to the correct male speaker.

D.2 Speaker-Dependent Attention Plots

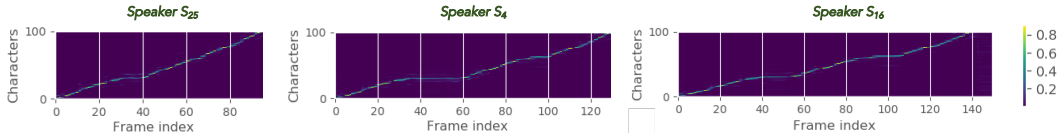


Figure 7: Attention plots for three speakers speaking at different speeds. Speaker S_{25} is a 22 year-old male with a Scottish accent, Speaker S_4 is a 23-year-old female with a Southern England accent, and Speaker S_{16} is a 22-year-old male with a London accent. The pronounced sentence is "Six spoons of fresh snow peas five thick slabs of blue cheese and maybe a snack for her brother Bob".

Fig. 7 shows the learned attention plots for three different speakers who talk at different speeds. It demonstrates that the modifications in the Tacotron encoder architecture are highly effective making the attention model speaker dependent such that different portions of the input text can be focused depending on the speech features of the speaker.

D.3 Principal Components of the Embeddings

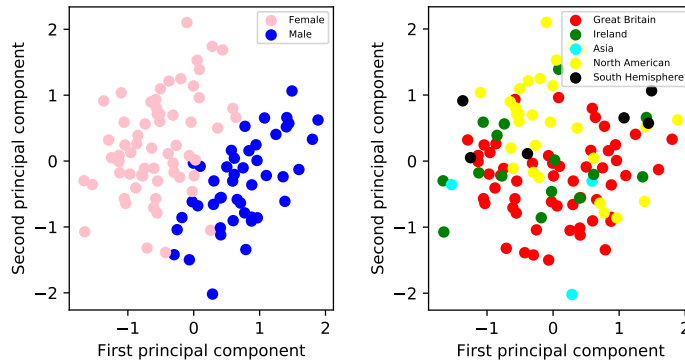


Figure 8: Principal components of the learned embeddings of the 80-layer vocal model, shown with the genders and regions of the speakers.

We explore the latent space of the learned speaker embeddings by visualizing them in a lower dimensional space. Fig. 8 and Fig. 9 show the first two principal components of the learned embeddings of the vocal model and character-to-spectrogram model respectively. Although they are initialized randomly and completely trained based on a loss function related to the generative quality, we can observe discriminative patterns in the learned embeddings. Gender of the speaker is the most apparent discriminative pattern in these learned embeddings that even a linear classifier fit on the shown two-dimensional space can classify the gender with a very high accuracy. Besides, we observe apparent discriminative patterns for the region of the speaker.⁷ In the two-dimensional space, especially Great Britain and North America regions seem highly separable.

⁷The regions are determined according to https://en.wikipedia.org/wiki/Regional_accents_of_English

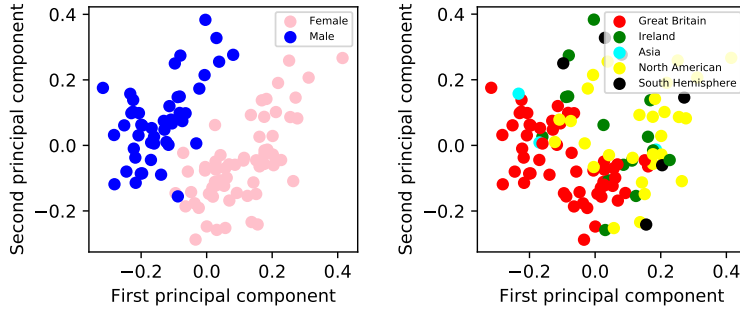


Figure 9: Principal components of the learned embeddings of the character-to-spectrogram model, shown with the genders and regions of the speakers.

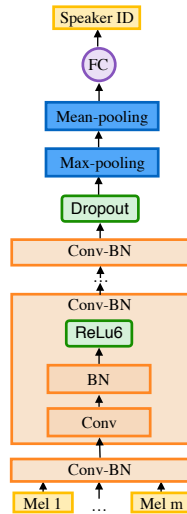


Figure 10: Architecture for speaker discriminator.

E Speaker Discriminative Model

To compute multi-speaker classification accuracy, we use a speaker discriminative model trained on the ground truth data set of multiple speakers. Although using another discriminator model such as Deep Speaker (Li et al., 2017) or other methods would also suffice, we choose to create our own deep learning based discriminative model. We note that our accuracy results on the test set are on par with the state-of-the-art speaker classification methods in the literature. Our architecture is depicted in Fig. 10. We use mel-frequency cepstral coefficients (MFCCs) computed after resampling the input to a constant sampling frequency. Then, we employ two-dimensional convolutional layers convolving over both time and cepstral frequency bands, with a relu nonlinearity clipped to a maximum of six after each convolutional layer. The last convolutional layer is followed by max-pooling layer. We then mean-pool over time for all utterance timesteps and apply a fully connected layer with a relu nonlinearity followed by a fully connected output layer with a softmax nonlinearity and cross-entropy classification loss. In order to avoid overfitting to the dataset, we apply dropout after every relu nonlinearity.

In order to demonstrate that the classification results are not sensitive to the choice of the hyperparameters of the discriminative model, we demonstrate the classification accuracy for other choices in this section. Hyperparameters for all the discriminator models are available in Table 7. Only the results for the models, $D3$ and $D8$, are presented in Table 2, as they yielded the highest validation set accuracy.

Param.	D1	D2	D3 (in Table 2)	D4
Audio resampling freq.	16 KHz	16 KHz	16 KHz	16 KHz
Number of MFCCs	20	20	80	80
Hop length	160	160	160	160
Convolution layers	5×, 32 2 × 10 filters	5×, 32 9 × 5	5×, 32 2 × 20	5×, 32 9 × 5
Maxpool width & stride	2	2	2	2
Fully connected size	16	16	32	32
Dropout probability	0.75	0.75	0.75	0.75
Learning rate	10 ⁻³ -0.95 - 1000	10 ⁻³ -0.95 - 1000	10 ⁻³ -0.95 - 1000	10 ⁻³ -0.95 -1000

Table 6: Hyperparameters for speaker discriminative models for VCTK dataset.

Param.	D5	D6	D7	D8 (in Table 2)
Audio resampling freq.	16 KHz	16 KHz	16 KHz	16 KHz
Number of MFCCs	20	20	80	80
Hop length	160	160	160	160
Convolution layers	3×, 32 9 × 5 filters	5× 32 2 × 10	7× , 32 9 × 5	5×, 32 2 × 10
Maxpool width & stride	2	2	2	2
Fully connected size	32	32	32	32
Dropout probability	0.75	0.75	0.75	0.75
Learning rate	10 ⁻³ -0.95 - 1000	10 ⁻³ -0.99 - 2000	10 ⁻³ -0.95 - 1000	10 ⁻³ -0.99 - 2000

Table 7: Hyperparameters for speaker discriminative models for Audiobook dataset.

Dataset	Multi-Speaker Model	D1	D2	D3 (in Table 2)	D4
VCTK	Deep Voice 2 (20-layer WaveNet)	97.87%	97.60%	99.92%	99.84%
VCTK	Deep Voice 2 (40-layer WaveNet)	98.56%	98.68%	100.00%	100.00%
VCTK	Deep Voice 2 (60-layer WaveNet)	98.56%	98.44%	99.68%	99.80%
VCTK	Deep Voice 2 (80-layer WaveNet)	99.06%	99.21%	99.96%	99.96%
VCTK	Tacotron (Griffin-Lim)	95.89%	96.24%	99.37%	99.60%
VCTK	Tacotron (20-layer WaveNet)	87.93%	85.19%	60.87%	65.43%
VCTK	Ground Truth Data	98.07 %	98.00%	99.74%	99.66%

Table 8: Classification accuracy using speaker discriminative models D1, D2, D3 and D4. The corresponding model hyperparameters are given in Table 6.

Dataset	Multi-Speaker Model	D5	D6	D7	D8 (in Table 2)
Audiobooks	Deep Voice 2 (80-layer WaveNet)	98.24%	95.66%	96.80%	97.42%
Audiobooks	Tacotron (Griffin-Lim)	96.93%	94.89%	92.24%	93.87%
Audiobooks	Tacotron (20-layer WaveNet)	83.36%	80.81%	60.00%	66.53%
Audiobooks	Ground Truth Data	96.30%	97.49%	98.39%	98.80%

Table 9: Classification accuracy using speaker discriminative models D5, D6, D7 and D8. The corresponding model hyperparameters are given in Table 7.