Université de Liège

Faculté des Sciences Appliquées

Automatic Multispeaker Voice Cloning

Author:
Corentin Jemine

Supervisor:
Prof. Gilles Louppe

Academic year 2018 - 2019

Graduation studies conducted for obtaining the Master's degree in Data Science by Corentin Jemine

Abstract

Recent advances in deep learning have shown impressive results in the domain of text-to-speech. To this end, a deep neural network is usually trained using a corpus of several hours of professionally recorded speech from a single speaker. Giving a new voice to such a model is highly expensive, as it requires recording a new dataset and retraining the model. A recent research introduced a three-stage pipeline that allows to clone a voice unseen during training from only a few seconds of reference speech, and without retraining the model. The authors share remarkably natural-sounding results, but provide no implementation. We reproduce this framework and open-source the first public implementation of it. We adapt the framework with a newer vocoder model, so as to make it run in real-time.

Contents

1 Introduction

Deep learning models have become predominant in many fields of applied machine learning. text-to-speech (TTS), the process of synthesizing artificial speech from a text prompt, is no exception. Deep models that would produce more natural-sounding speech than the traditional concatenative approaches begun appearing in 2016. Much of the research focus has been since gathered around making these deep models more efficient, more natural, or training them in an end-to-end fashion. Inference has come from being hundreds of times slower than real-time on GPU (van den Oord et al., 2016) to possible in real-time on a mobile CPU (Kalchbrenner et al., 2018). As for the quality of the generated speech, Shen et al. (2017) demonstrate near human naturalness. Interestingly, speech naturalness is best rated with subjective metrics; and comparison with actual human speech leads to the conclusion that there might be such a thing as "speech more natural than human speech". In fact, some argue that the human naturalness threshold has already been crossed (Shirali-Shahreza and Penn, 2018).

Datasets of professionally recorded speech are a scarce resource. Synthesizing a natural voice with a correct pronunciation, lively intonation and a minimum of background noise requires training data with the same qualities. Furthermore, data efficiency often remains one of the shortcomings of deep learning. Training a common text-to-speech model such as Tacotron (Wang et al., 2017) typically requires tens of hours of speech. Yet the ability of generating speech with any voice is attractive for a range of applications be they useful or merely a matter of customization. Research has led to frameworks for voice conversion and voice cloning. They differ in that voice conversion is a form of style transfer on a speech segment from a voice to another, whereas voice cloning consists in capturing the voice of a speaker to perform text-to-speech on arbitrary inputs.

While the complete training of a single-speaker TTS model is technically a form of voice cloning, the interest rather lies in creating a fixed model that is able to incorporate newer voices with little data. The common approach is to condition a TTS model trained to generalize to new speakers on an embedding of the voice to clone (Arik et al., 2017, 2018; Jia et al., 2018). The embedding is low-dimensional and derived by a speaker encoder model that takes reference speech as input. This approach is typically more data efficient than training a separate TTS model for each speaker, in addition to being orders of magnitude faster and less computationally expensive. Interestingly, there is a large discrepancy between the duration of reference speech needed to clone a voice among the different methods, ranging from half an hour per speaker to only a few seconds.

Our objective is to achieve a powerful form of voice cloning. The resulting framework must be able to operate in a zero-shot setting, that is, for speakers unseen during training. It should incorporate a speaker's voice with only a few seconds of reference speech. These desired results are shown to be fulfilled by (Jia et al., 2018).

Their results are impressive[1], but not backed by any public implementation. We reproduce their framework and make our implementation open-source[2]. In addition, we integrate the model of (Kalchbrenner et al., 2018) in the framework to make it run in real-time, i.e. to generate speech in a time shorter or equal to the duration of the produced speech. add a word about our results

The structure of this document goes as follows. We begin with a short introduction on TTS methods that involve machine learning. Follows a review of the evolution of the state of the art for TTS and for voice cloning. We then present the work of (Jia et al., 2018) along with our own implementation. experiments, toolbox

2 A review of text-to-speech methods in machine learning

2.1 Statistical parametric speech synthesis

Statistical parametric speech synthesis (SPSS) refers to a group of data-driven TTS methods that emerged in the late 90s. In SPSS, the relationship between the features computed on the input text and the output acoustic features is learned by a statistical generative model (called the acoustic model). A complete SPSS framework thus also includes a pipeline to extract features from the text to synthesize, as well as a system able to reconstruct an audio waveform from the acoustic features produced by the acoustic model (such a system is called a vocoder). Unlike the acoustic model, these two parts of the framework may be entirely engineered and make use of no statistical methods. While modern deep TTS models are usually not referred to as SPSS, the SPSS pipeline as depicted in figure 1 applies just as well to those newer methods.

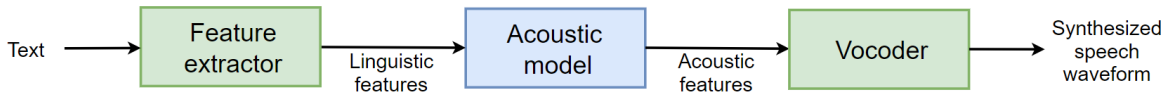

Figure 1: The general SPSS pipeline. The blue box is purely a statistical model while the green boxes can be engineered processes or/and statistical models.

The role of the feature extractor is to provide data that is more indicative of what the speech produced by the model is expected to sound like. Speech is a complex process, and directly feeding characters to a weak acoustic model will prove not to be effective. Providing additional features from natural language processing (NLP) techniques may greatly reduce the extent of the task to be learned by the acoustic model. It may however result in trade-offs when it comes to naturalness, especially for rare or unknown words. Indeed, manually engineered heuristics do not quite fully

[1] `https://google.github.io/tacotron/publications/speaker_adaptation/index.html`
[2] repo link

characterize all intricacies of spoken language. For this reason, feature extraction can also be done with trained models. The line between the feature extractor and the acoustic model can then become blurry, especially for deep models. In fact, a tendency that is common across all areas where deep models have overtaken traditional machine learning techniques is for feature extraction to consist of less heuristics, as models become able to operate at higher levels of abstraction.

A common feature extraction technique is to build frames that will integrate surrounding context in a hierarchical fashion. For example, a frame at the syllable level could include the word that comprises it, its position in the word, the neighbouring syllables, the phonemes that make up the syllable, ... The lexical stress and accent of individual syllables can be predicted by a statistical model such as a decision tree. To encode prosody, a set of rules such as ToBI (Beckman and Elam, 1997) can be used. Ultimately, there remains a work of feature engineering to present a frame as a numerical object to the model, e.g. categorical features are typically encoded using a one-hot representation.

One could wonder why the acoustic model does not directly predict an audio waveform. Audio happens to be difficult to model: it is a particularly dense domain and audio signals are typically highly nonlinear. A representation that brings out features in a more tractable manner is the time-frequency domain. Spectrograms are smoother and much less dense than their waveform counterpart. They also have the benefit of being two-dimensional, thus allowing models to better leverage spatial connectivity. Unfortunately, a spectrogram is a lossy representation of the waveform that discards the phase. There is no unique inverse transformation function, and deriving one that produces natural-sounding results is not trivial. When referring to speech, this generative function is called a vocoder. The choice of the vocoder is an important factor in determining the quality of the generated audio.

Talk about evaluation metrics (MOS and A/B testing)?

2.2 Evolution of the state of the art in text-to-speech

The state of the art in SPSS has for long remained a hidden Markov model (HMM) based framework (Tokuda, 2013). This approach, laid out in figure 2, consists in clustering the linguistic features extracted from the input text with a decision tree, and to train a HMM per cluster (Yoshimura et al., 1999). The HMMs are tasked to produce a distribution over spectrogram coefficients, their derivative, second derivative and a binary flag that indicates which parts of the generated audio should contain voice. With the maximum likelihood parameter generation algorithm (MLPG) (Tokuda et al., 2000), spectrogram coefficients are sampled from this distribution and eventually fed to the MLSA vocoder (Imai, 1983). It is possible to modify the voice generated by conditioning the HMMs on a speaker or tuning the generated speech parameters with adaptation or interpolation techniques (Yoshimura et al., 1997). Note that, while this framework used to be state of the art for SPSS, it was still inferior in terms of the

naturalness of the generated speech compared to the well-established concatenative approaches.

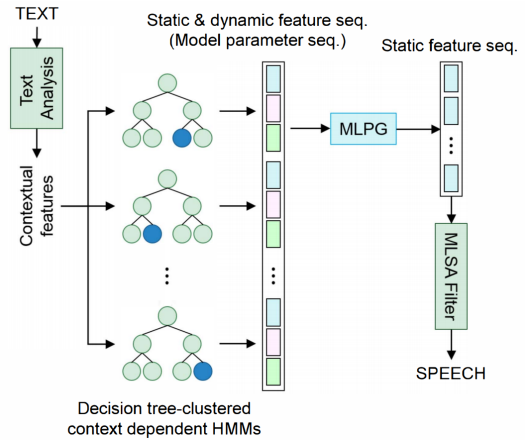

Figure 2: The general HMM-based TTS pipeline. Figure extracted from (Hashimoto et al., 2015).

Method	MOS
HMM+MLPG	3.08 ($\pm$0.12)
HMM+DNN	2.86 ($\pm$0.12)
DNN+MLPG	**3.53 ($\pm$0.12)**
DNN+DNN	3.17 ($\pm$0.12)

Table 1: MOS of the different methods explored in (Hashimoto et al., 2015). The first line is the HMM-based framework. For the second and fourth line, the MLPG algorithm is replaced by a fully-connected neural network.

Improvements to this framework were later brought by feed-forward deep neural networks (DNN), as a result of progress in both hardware and software. Zen et al. (2013) proposes to replace entirely the decision tree-clustered HMMs in favor of a DNN. They argue for better data efficiency as the training set is no longer fragmented in different clusters of contexts. They demonstrate improvements over the speech quality with a number of parameters similar to that of the HMM-based approach. Later researches corroborate these findings (Qian et al., 2014; Hashimoto et al., 2015). The MOS of different model combinations tried by (Hashimoto et al., 2015) are reported in Table 1

(Fan et al., 2014) support that RNNs make natural acoustic models as they are able to learn a compact representation of complex and long-span functions. As RNNs are fit to generate temporally consistent series, the static features can directly be determined by the acoustic model, alleviating the need for dynamic features and MLPG. They compare networks of bidirectional LSTMs against the HMM and DNN based approaches described previously. Their A/B testing results are conclusive, we report them in figure 3.

Figure 3: A/B testing of the models. Hybrid_B is a network with fully-connected and bidirectional LSTM layers. Figure extracted from (Fan et al., 2014).

The coming of WaveNet (van den Oord et al., 2016) made a substantial breakthrough in TTS. WaveNet is a deep convolutional neural network that, for a raw

audio waveform, models the distribution of a single sample given all previous ones. It is thus possible to directly generate audio by predicting samples one at a time in an autoregressive fashion. WaveNet leverages stacks of one-dimensional dilated convolutions with a dilation factor increasing exponentially with the layer depth, allowing for the very large receptive field and the strong nonlinearity needed to model raw audio. Conditioning the model on linguistic features is required to perform TTS. WaveNet acts thus both as an acoustic model and as a vocoder. Note that without the local conditioning, a trained WaveNet generates sound alike the training data but without structure or semantics (essentially babbling). The authors compare WaveNet to an older parametric approach and to a concatenative approach, the results are reported in Figure 4. The parametric approach is an LSTM-based system while the other is an HMM-driven unit selection concatenative system (not detailed in this document). Notice how the results vary between US English and Mandarin Chinese, showing that TTS performance is not language agnostic.

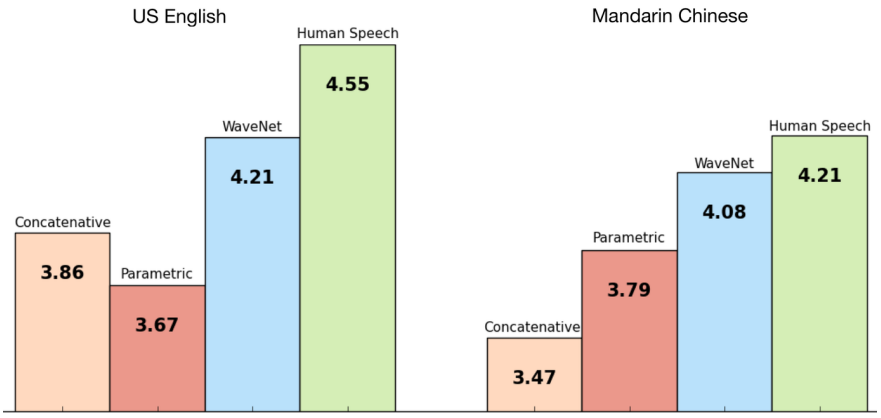

Figure 4: MOS of WaveNet's performance compared with a parametric and concatenative approach, as well as with natural speech.[3]

Follows Tacotron (Wang et al., 2017), a sequence-to-sequence model that produces a spectrogram from a sequence of characters alone, further reducing the need for domain expertise. In this framework, the vocoder is the Griffin-Lim algorithm. Tacotron uses an encoder-decoder architecture where, at each step, the decoder operates on a weighted sum of the encoder outputs. This attention mechanism, described in (Bahdanau et al., 2014), lets the network decide which steps of the input sequence are important with respect to each step of the output sequence. Tacotron achieves a MOS of 3.85 on a US English dataset, which is more than the 3.69 score obtained in the parametric approach of (Zen et al., 2016) but less than the 4.09 score obtained by the concatenative approach of (Gonzalvo et al., 2016). The authors mention that Tacotron is merely a step towards a better framework. Subsequently, Tacotron 2 is published (Shen et al., 2017). The architecture of Tacotron 2 remains that of an encoder-decoder with attention although several changes to the type of layers are made. The main difference with Tacotron is the use of a modified WaveNet as vocoder. On the same dataset, Tacotron 2 achieves a MOS of 4.53, which compares to the 4.58

[3]Figure extracted from `https://deepmind.com/blog/wavenet-generative-model-raw-audio/`

for human speech (the difference is not statistically significant), achieving the all-time highest MOS for TTS. With A/B testing, Tacotron 2 was found to be only slightly less preferred on average than ground truth samples. These ratings are shown in figure 5.

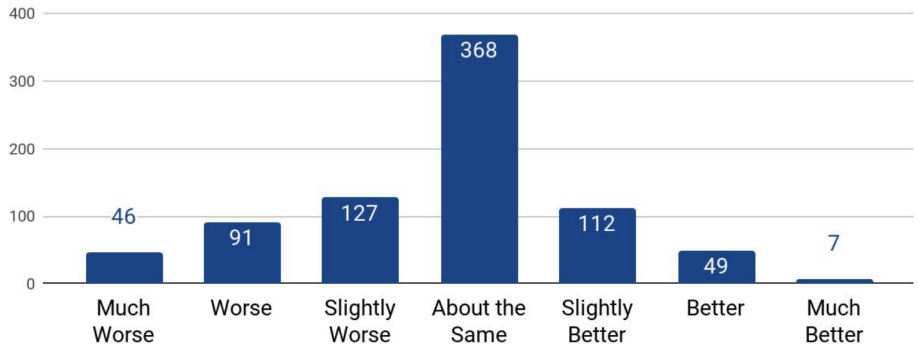

Figure 5: Preference ratings between Tacotron 2 and ground truth samples. There are 800 ratings from 100 items. The labels are expressed with respect to Tacotron 2. Figure extracted from (Shen et al., 2017).

Efficient neural audio synthesis?
The plot summarizing everything

3 Related voice cloning methods

4 Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis

4.1 Overview

Our solution to real-time voice cloning is largely based on (Jia et al., 2018) (referred to as SV2TTS throughout this document). It describes an approach to zero-shot voice cloning that only requires 5 seconds of reference speech. This paper is only one of the many publications from the Tacotron series[4] authored at Google. Interestingly, the SV2TTS paper does not bring much innovation of its own, rather it is based on three major earlier works from Google: the GE2E loss (Wan et al., 2017), Tacotron (Wang et al., 2017) and WaveNet (van den Oord et al., 2016). The complete framework is a three-stage pipeline, where the steps correspond to the models listed in order previously. Many of the current TTS tools and functionalities provided by Google, such as the Google assistant[5] or the Google cloud services[6], make use of these same

[4] https://google.github.io/tacotron/
[5] https://assistant.google.com/
[6] https://cloud.google.com/text-to-speech/

models. While there are many open-source reimplementations of these models online, there is none of the SV2TTS framework to our knowledge (as of May 2019).

The three stages of the framework are as follows:

- A speaker encoder that derives an embedding from the short utterance of a single speaker. The embedding is a meaningful representation of the voice of the speaker, such that similar voices are close in latent space. This model is described in (Wan et al., 2017) (referred as GE2E throughout this document) and (Heigold et al., 2015).
- A synthesizer that, conditioned on the embedding of a speaker, generates a spectrogram from a text. This model is the popular Tacotron 2 (Shen et al., 2017) without WaveNet (which is often referred to as just Tacotron due to its similarity to the first iteration).
- A vocoder that infers an audio waveform from the spectrograms generated by the synthesizer. The authors used WaveNet (van den Oord et al., 2016) as a vocoder, effectively reapplying the entire Tacotron 2 framework.

At inference time, the speaker encoder is fed a short reference utterance of the speaker to clone. It generates an embedding that is used to condition the synthesizer, and a text processed as a phoneme sequence is given as input to the synthesizer. The vocoder takes the output of the synthesizer to generate the speech waveform. This is illustrated in figure 6.

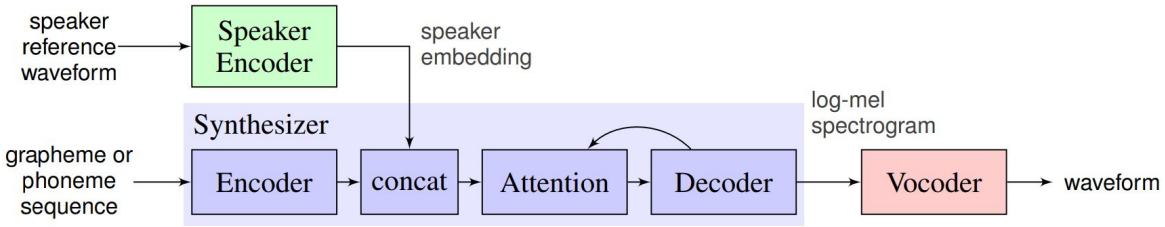

Figure 6: The SV2TTS framework during inference. The blue blocks represent a high-level view of the Tacotron architecture modified to allow conditioning on a voice. Figure extracted from (Jia et al., 2018).

A particularity of the SV2TTS framework is that all models can be trained separately and on distinct datasets. For the encoder, one seeks to have a model that is robust to noise and able to capture the many characteristics of the human voice. Therefore, a large corpus of many different speakers would be preferable to train the encoder, without any strong requirement on the noise level of the audios. Additionally, the encoder is trained with the GE2E loss which requires no labels other than the speaker identity. With GE2E, the task to be learned by the model is a speaker verification task, which by itself has little to do with voice cloning. However, the task is stipulated in way that the network will output an embedding that is a

meaningful representation of the voice of the speaker. This embedding is suitable for conditioning the synthesizer on a voice, hence the name of the paper: "Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis". For the datasets of the synthesizer and the vocoder, transcripts are required and the quality of the generated audio can only be as good as that of the data. Higher quality and annotated datasets are thus required, which often means they are smaller in size.

4.2 Problem definition

Consider a dataset of utterances grouped by their speaker. We denote the jth utterance of the ith speaker as $\mathbf{u}_{ij}$. Utterances are in the waveform domain. We denote by $\mathbf{x}_{ij}$ the log-mel spectrogram of the utterance $\mathbf{u}_{ij}$. A log-mel spectrogram is a deterministic, non-invertible (lossy) but derivable function that extracts speech features from a waveform, so as to handle audio in a more tractable fashion in machine learning.

The encoder $\mathcal{E}$ computes the embedding $\mathbf{e}_{ij} = \mathcal{E}(\mathbf{x}_{ij}; \mathbf{w}_{\mathcal{E}})$ corresponding to the utterance $\mathbf{u}_{ij}$, where $\mathbf{w}_{\mathcal{E}}$ are the parameters of the encoder. Additionally, the authors define a speaker embedding as the centroid of the embedding of the speaker's utterances:

$$\mathbf{c}_i = \frac{1}{n} \sum_{j}^{n} \mathbf{e}_{ij} \tag{1}$$

The synthesizer $\mathcal{S}$, parametrized by $\mathbf{w}_{\mathcal{S}}$, is tasked to approximate $\mathbf{x}_{ij}$ given $\mathbf{c}_i$ and $\mathbf{t}_{ij}$, the transcript of utterance $\mathbf{u}_{ij}$. We have $\hat{\mathbf{x}}_{ij} = \mathcal{S}(\mathbf{c}_i, \mathbf{t}_{ij}; \mathbf{w}_{\mathcal{S}})$. In our implementation, we directly use the utterance embedding rather than the speaker embedding (we motivate this choice in section 4.4), giving instead $\hat{\mathbf{x}}_{ij} = \mathcal{S}(\mathbf{u}_{ij}, \mathbf{t}_{ij}; \mathbf{w}_{\mathcal{S}})$.

Finally, the vocoder $\mathcal{V}$, parametrized by $\mathbf{w}_{\mathcal{V}}$, is tasked to approximate $\mathbf{u}_{ij}$ given $\hat{\mathbf{x}}_{ij}$. We have $\hat{\mathbf{u}}_{ij} = \mathcal{V}(\hat{\mathbf{x}}_{ij}; \mathbf{w}_{\mathcal{V}})$.

One could train this framework in an end-to-end fashion with the following objective function:

$$min_{\mathbf{w}_{\mathcal{E}}, \mathbf{w}_{\mathcal{S}}, \mathbf{w}_{\mathcal{V}}} L_{\mathcal{V}}(\mathbf{u}_{ij}, \mathcal{V}(\mathcal{S}(\mathcal{E}(\mathbf{x}_{ij}; \mathbf{w}_{\mathcal{E}}), \mathbf{t}_{ij}; \mathbf{w}_{\mathcal{S}}); \mathbf{w}_{\mathcal{V}}))$$

Where $L_{\mathcal{V}}$ is a loss function in the waveform domain. This approach has drawbacks:

- It requires training all three models on a same dataset, meaning that this dataset would ideally need to meet the requirements for all models: a large number of speakers for the encoder but at the same time transcripts for the synthesizer and a low level noise for the synthesizer and somehow an average noise level for the encoder (so as to be able to handle noisy input speech). These conflicts are problematic and would lead to training models that could perform better if trained separately on distinct datasets. Specifically, a small dataset will likely lead to poor generalization and thus poor zero-shot performance.

- The convergence of the combined model could be very hard to reach. In particular, the Tacotron synthesizer typically takes a significant time before producing correct alignments (see ?).

An evident way of addressing the second issue is to separate the training of the synthesizer and of the vocoder. Assuming a pretrained encoder, the synthesizer can be trained to directly predict the mel spectrograms of the target audio:

$$min_{\mathbf{w}_\mathcal{S}} L_\mathcal{S}(\mathbf{x}_{ij}, \mathcal{S}(\mathbf{e}_{ij}, \mathbf{t}_{ij}; \mathbf{w}_\mathcal{S}))$$

Where $L_\mathcal{S}$ is a loss function in the time-frequency domain.

The vocoder is then trained directly on the spectrograms. Note that both the approaches of training on ground truth spectrograms or on synthesizer-generated spectrograms are valid (see ?). The latter requires a pretrained synthesizer.

$$min_{\mathbf{w}_\mathcal{V}} L_\mathcal{V}(\mathbf{u}_{ij}, \mathcal{V}(\mathbf{x}_{ij}; \mathbf{w}_\mathcal{V})) \quad \text{or} \quad min_{\mathbf{w}_\mathcal{V}} L_\mathcal{V}(\mathbf{u}_{ij}, \mathcal{V}(\hat{\mathbf{x}}_{ij}; \mathbf{w}_\mathcal{V}))$$

Remains the optimization of the speaker encoder. Unlike the synthesizer and the vocoder, the encoder does not have labels to be trained on. The task is lousily defined as producing "meaningful" embeddings that characterize the voice in the utterance. One could conceive of a way to train the speaker encoder as an autoencoder, but it would require the corresponding upsampling model to be made aware of the text to predict. Either the dataset is constrained to a same sentence, either one needs transcripts and the upsampling model is the synthesizer. In both cases the quality of the training is impaired by the dataset and unlikely to generalize well. Fortunately, the GE2E loss (Wan et al., 2017) brings a solution to this problem and allows to train the speaker encoder independently of the synthesizer. This is described in section 4.3.

While all parts of the framework are trained separately, there is still the requirement for the synthesizer to have embeddings from a trained encoder and for the vocoder to have mel spectrograms from a trained synthesizer (if not training on ground truth spectrogram). Figure 7 illustrates how each model depends on the previous one for training. The speaker encoder needs to generalize well enough to produce meaningful embeddings on the dataset of the synthesizer; and even when trained on a common dataset, it still has to be able to operate in a zero-shot setting at inference time.

4.3 Speaker encoder

The encoder model and its training procedure are described over several papers (Jia et al., 2018; Wan et al., 2017; Heigold et al., 2015). We reproduced this model with a PyTorch implementation of our own. We synthesize the parts that are pertinent to SV2TTS as well as our choices of implementation.

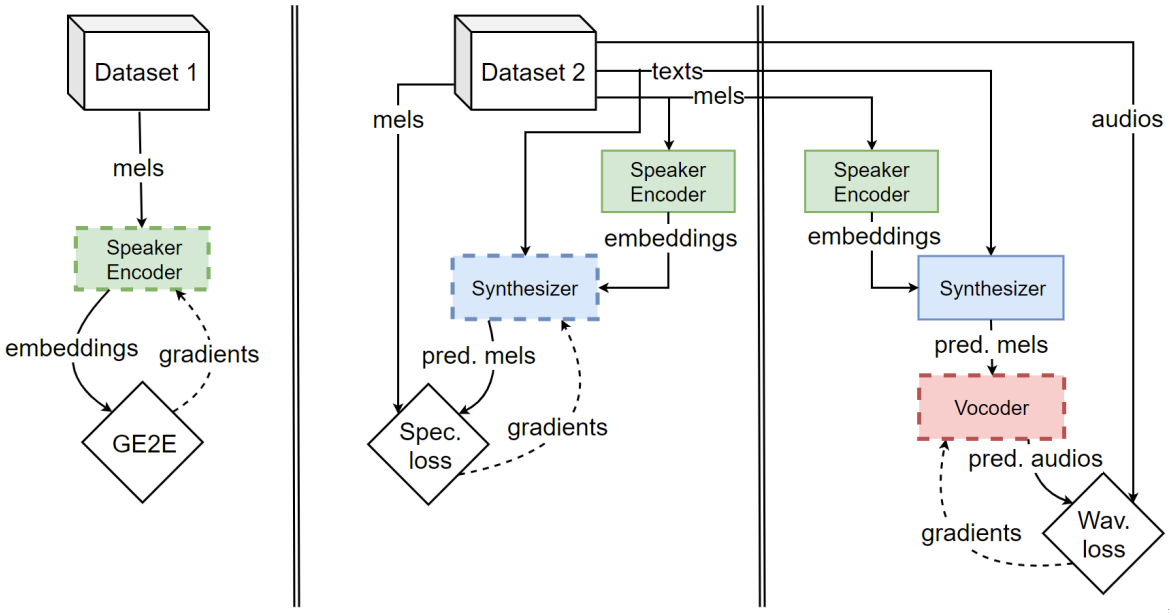

Figure 7: The sequential three-stage training of SV2TTS (following our choices of implementation). Models with solid contour lines are frozen. Note that the mel spectrograms fed to the speaker encoder and those used as target for the synthesizer are created with different parameters.

4.3.1 Model architecture

The model is a 3-layer LSTM with 768 hidden nodes followed by a projection layer of 256 units. While there is no reference in any of the papers as to what a projection layer is, the intuition is that it is simply a 256 outputs fully-connected layer per LSTM that is repeatedly applied to every output of the LSTM. When we first implemented the speaker encoder, we directly used 256 units LSTM layers instead, for the sake of quick prototyping, simplicity and for a lighter training load. This last part is important, as the authors have trained their own model for 50 million steps (although on a larger dataset), which is technically difficult for us to reproduce. We found this smaller model to perform extremely well, and we haven't found the time to train the larger version later on. move to experiments?

The inputs to the model are 40-channels log-mel spectrograms with a 25ms window width and a 10ms step. The output is the L2-normalized hidden state of the last layer, which is a vector of 256 elements. Our implementation also features a ReLU layer before the normalization, with the goal in mind to make embeddings sparse and thus more easily interpretable.

4.3.2 Generalized End-to-End loss

The speaker encoder is trained on a speaker verification task. Speaker verification is a typical application of biometrics where the identity of a person is verified through their

voice. A template is created for a person by deriving their speaker embedding (see equation 1) from a few utterances. This process is called enrollment. At runtime, a user identifies himself with a short utterance and the system compares the embedding of that utterance with the enrolled speaker embeddings. Above a given similarity threshold, the user is identified. The GE2E loss simulates this process to optimize the model.

At training time, the model computes the embeddings $\mathbf{e}_{ij}$ $(1 \leq i \leq N, 1 \leq j \leq M)$ of M utterances of fixed duration from N speakers. A speaker embedding $\mathbf{c}_i$ is derived for each speaker: $\mathbf{c}_i = \frac{1}{M}\sum_j^M \mathbf{e}_{ij}$ $(1 \leq i \leq N)$. The similarity matrix $\mathbf{S}_{ij,k}$ is the result of the two-by-two comparison of all embeddings $\mathbf{e}_{ij}$ against every speaker embedding $\mathbf{c}_k$ $(1 \leq k \leq N)$ in the batch. This measure is the scaled cosine similarity:

$$\mathbf{S}_{ij,k} = w \cdot cos(\mathbf{e}_{ij}, \mathbf{c}_k) + b = w \cdot \mathbf{e}_{ij} \cdot ||\mathbf{c}_k||_2 + b \tag{2}$$

where w and b are learnable parameters. This entire process is illustrated in figure 8. From a computing perspective, the cosine similarity of two L2-normed vectors is simply their dot product, hence the rightmost hand side of equation 2. An optimal model is expected to output high similarity values when an utterance matches the speaker $(i = k)$ and lower values elsewhere $(i \neq k)$. To optimize in this direction, the loss is the sum of row-wise softmax losses.

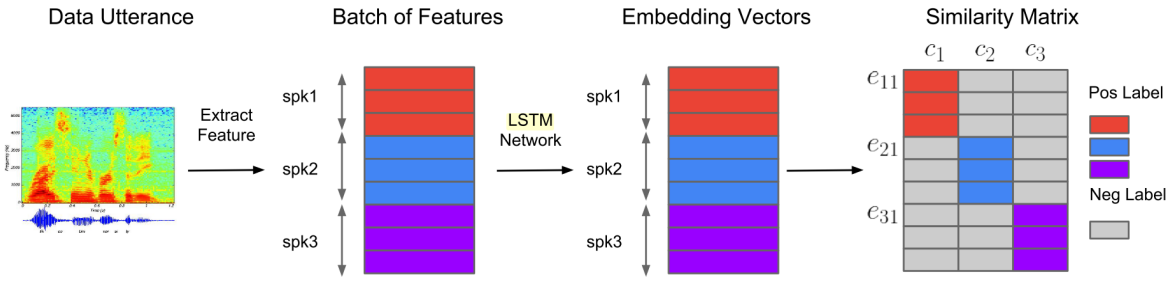

Figure 8: The construction of the similarity matrix at training time. This figure is extracted from (Wan et al., 2017).

Note that each utterance $\mathbf{e}_{ij}$ is included in the centroid $\mathbf{c}_i$ of the same speaker when computing the loss. This creates a bias towards the correct speaker independently of the accuracy of the model. To prevent this, an utterance that is compared against its own speaker's embedding will be removed from the computation of the speaker embedding. The similarity matrix is then defined as:

$$\mathbf{S}_{ji,k} = \begin{cases} w \cdot \cos(\mathbf{e}_{ij}, \mathbf{c}_i^{(-j)}) + b & \text{if} \quad i = k \\ w \cdot \cos(\mathbf{e}_{ij}, \mathbf{c}_k) + b & \text{otherwise.} \end{cases} \tag{3}$$

where the exclusive centroids $\mathbf{c}_i^{(-j)}$ are defined as:

$$\mathbf{c}_i^{(-j)} = \frac{1}{M-1}\sum_{\substack{m=1 \\ m \neq j}}^{M} \mathbf{e}_{im} \tag{4}$$

The fixed duration of the utterances in a training batch is of 1.6 seconds. These utterances are partial utterances sampled from the longer complete utterances in the dataset. While the model architecture is able to handle inputs of variable length, it is reasonable to expect that it performs best with utterances of the same duration as those seen in training. Therefore, at inference time an utterance is split in segments of 1.6 seconds overlapping by 50%, and the encoder forwards each segment individually. The resulting outputs are averaged then normalized to produce the utterance embedding. This is illustrated in figure 9. Curiously, the authors of SV2TTS advocate for 800ms windows at inference time but still 1.6 seconds ones during training. We prefer to keep 1.6 seconds for both, as is done in GE2E.

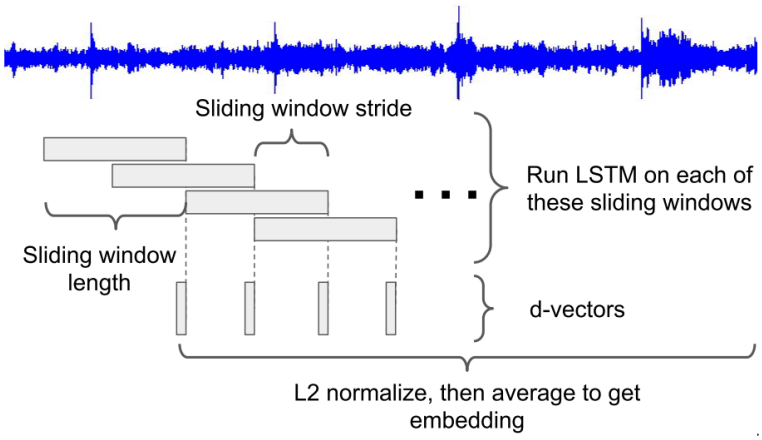

Figure 9: Computing the embedding of a complete utterance. The d-vectors are simply the unnormalized outputs of the model. This figure is extracted from (Wan et al., 2017).

The authors use $N = 64$ and $M = 10$ as parameters for the batch size. When enrolling a speaker in a practical application, one should expect to have several utterances from each user but likely not an order of magnitude above that of 10, so this choice is reasonable. As for the number of speakers, it is good to observe that the time complexity of computing the similarity matrix is $O(N^2M)$. Therefore this parameter should be chosen not too large so as to not slow down substantially the training, as opposed to simply picking the largest batch size that fits on the GPU. It is still of course possible to parallelize multiple batches on the same GPU while synchronizing the operations across batches for efficiency. We found it particularly important to vectorize all operations when computing the similarity matrix, so as to minimize the number of GPU transactions.

4.3.3 Experiments

To avoid segments that are mostly silent when sampling partial utterances from complete utterances, we use the webrtcvad[7] python package to perform Voice Activity

[7] `https://github.com/wiseman/py-webrtcvad`

Detection (VAD). This yields a binary flag over the audio corresponding to whether or not the segment is voiced. We perform a moving average on this binary flag to smooth out short spikes in the detection, which we then binarize again. Finally, we perform a dilation on the flag with a kernel size of $s + 1$, where s is the maximum silence duration tolerated. The audio is then trimmed of the unvoiced parts. We found the value $s = 0.2$s to be a good choice that retains a natural speech prosody. This process is illustrated in figure 10. A last preprocessing step applied to the audio waveforms is normalization, to make up for the varying volume of the speakers in the dataset.

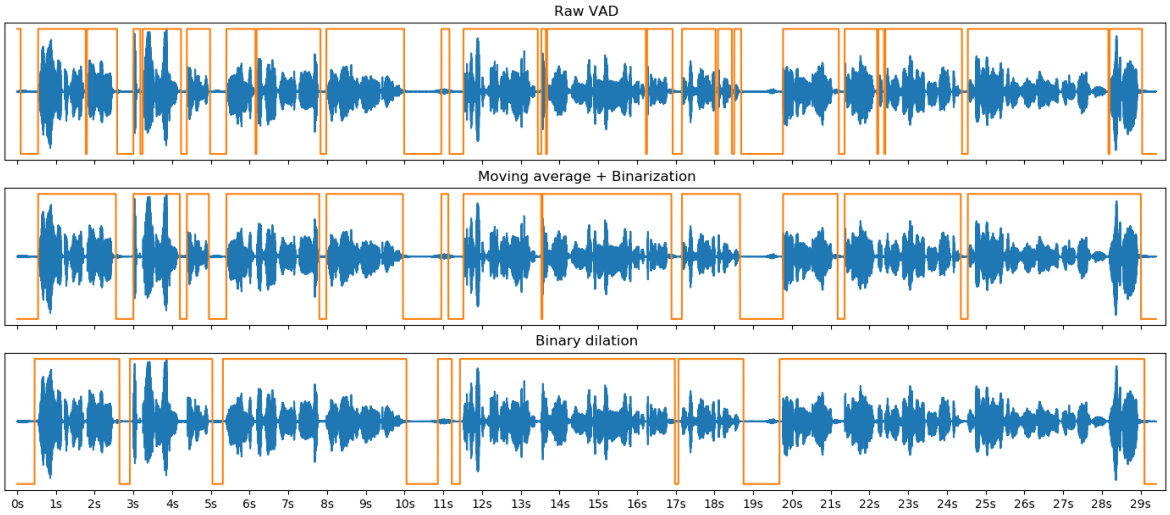

Figure 10: The steps to silence removal with VAD, from top to bottom. The orange line is the binary voice flag where the upper value means that the segment is voiced, and unvoiced when lower.

The authors combined several noisy datasets to make for a large corpus of speech of quality similar to what is found in the wild. These datasets are LibriSpeech (Panayotov et al., 2015), VoxCeleb1 (Nagrani et al., 2017), VoxCeleb2 (Chung et al., 2018) and an internal dataset, to which we do not have access. LibriSpeech is a corpus of audiobooks making up for 1000 hours of audio from 2400 speakers, split equally in two sets "clean" and "other". The clean set is supposedly made up of cleaner speech than the other set, even though some parts of the clean set still contain a lot of noise (Zen et al., 2019). VoxCeleb1 and VoxCeleb2 are made up from audio segments extracted from youtube videos of celebrities (often in the context of an interview). VoxCeleb1 has 1.2k speakers, while VoxCeleb2 has about 6k. Both these datasets have non-English speakers. We used heuristics based on the nationality of the speaker to filter non-English ones out of the training set in VoxCeleb1, but couldn't apply those same heuristics to VoxCeleb2 as the nationality is not referenced in that set. Note that it is unclear without experimentation as to whether having non-English speakers hurts the training of the encoder (the authors make no note of it either). All these datasets are sampled at 16kHz.

The authors test different combinations of these datasets and observe the effect on the quality of the embeddings. They adjust the output size of LSTM model (the size

of the embeddings) to 64 or 256 according to the number of speakers. They evaluate the subjective naturalness and similarity with ground truth of the speech generated by a synthesizer trained from the embeddings produced by each model. They also report the equal error rate of the encoder on speaker verification, which we discuss later in this section. These results can be found in Table 2.

Training Set	Speakers	Embedding Dim	Naturalness	Similarity	SV-EER
LS-Clean	1.2K	64	3.73 ± 0.06	2.23 ± 0.08	16.60%
LS-Other	1.2K	64	3.60 ± 0.06	2.27 ± 0.09	15.32%
LS-Other+VC	2.4K	256	3.83 ± 0.06	2.43 ± 0.09	11.95%
• LS-Other+VC+VC2	8.4K	256	3.82 ± 0.06	2.54 ± 0.09	10.14%
Internal	18K	256	4.12 ± 0.05	3.03 ± 0.09	5.08%

Table 2: Training of the speaker encoder on different datasets, from (Jia et al., 2018). LS is LibriSpeech and VC is VoxCeleb. The synthesizers are trained on LS-Clean and evaluated on a test set. The line with a bullet is the implementation we aim to reproduce.

These results indicate that the number of speakers is strongly correlated with the good performance of not only the encoder on the verification task, but also of the entire framework on the quality of the speech generated and on its ability to clone a voice. The small jump in naturalness, similarity and EER gained by including VoxCeleb2 could indicate that the variation of languages is hurting the training. The internal dataset of the authors is a proprietary voice search corpus from 18k English speakers. The encoder trained on this dataset performs significantly better, however we only have access to public datasets. We thus proceed with LibriSpeech-Other, VoxCeleb1 and VoxCeleb2.

We train the speaker encoder for one million steps. To monitor the training we report the EER and we observe the ability of the model to cluster speakers. We periodically sample a batch of 10 speakers with 10 utterances each, compute the utterance embeddings and projecting them in a two-dimensional space with UMAP (McInnes and Healy, 2018). As embeddings of different speakers are expected to be further apart in the latent space than embeddings from the same speakers, it is expected that clusters of utterances from a same speaker form as the training progresses. We report our UMAP projections in figure 11, where this behaviour can be observed.

As mentioned before, the authors have trained their model for 50 million steps on their proprietary dataset. While both our dataset and our model are smaller, our model still hasn't converged at 1 million steps. The loss decreases steadily with little variance and can still decrease more, but we are bound by the time.

loss plot + test set? + EER

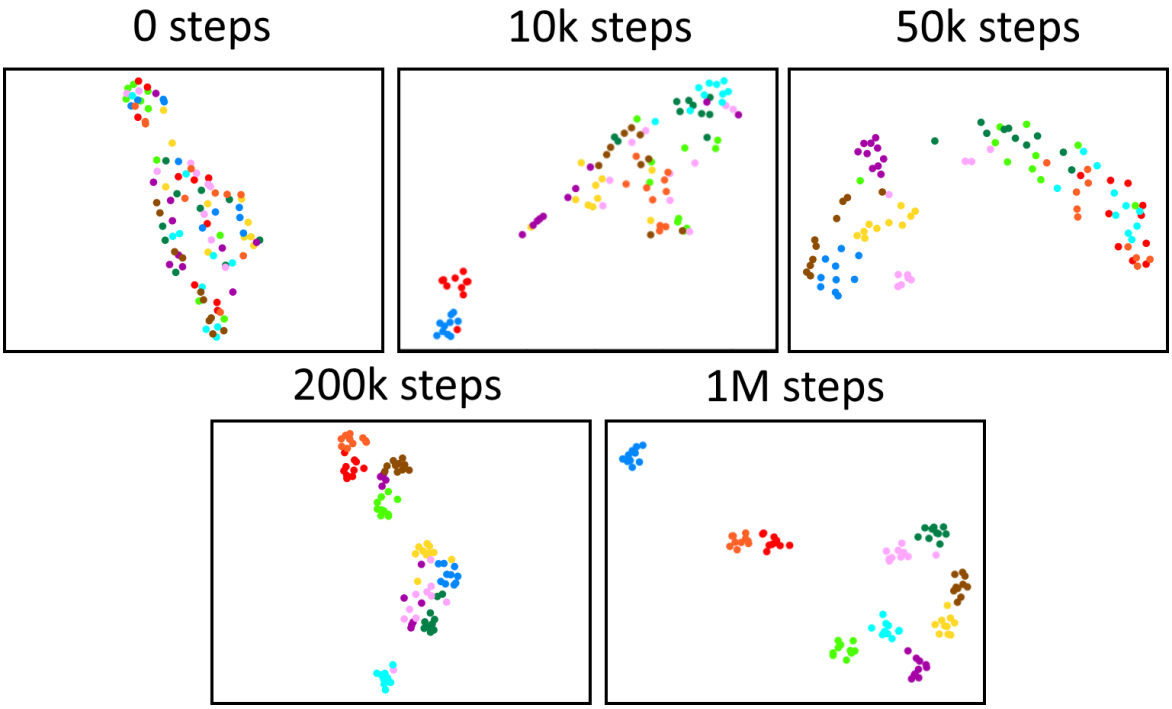

Figure 11: UMAP projections of utterance embeddings from randomly selected batches from the train set at different iterations of our model. Utterances from the same speaker are represented by a dot of the same color. We specifically omit to pass labels to UMAP, so the clustering is entirely done by the model.

4.4 Synthesizer

The synthesizer architecture is that of Tacotron 2 without Wavenet (van den Oord et al., 2016). We use an open-source implementation[8] of Tacotron 2 from which we strip Wavenet and implement the modifications added by SV2TTS.

4.4.1 Model architecture

We briefly present the top-level architecture of the modified Tacotron 2 without Wavenet (which we'll refer to as simply Tacotron). For further details, we invite the reader to take a look at its originating paper Shen et al. (2017) as well as the Tacotron 1 paper (Wang et al., 2017).

Tacotron is a recurrent sequence-to-sequence model that predicts a mel spectrogram from text. It features an encoder-decoder structure (not to be mistaken with the speaker encoder of SV2TTS) that is bridged by a location-sensitive attention mechanism Chorowski et al. (2015). Individual characters from the text sequence are first embedded as vectors. Convolutional layers follow, so as to increase the information span of a single encoder frame. These frames are passed through a bidirectional LSTM

[8] `https://github.com/Rayhane-mamah/Tacotron-2`

to produce the encoder output frames. This is where SV2TTS brings a modification to the architecture: a speaker embedding is concatenated to every frame output by the encoder. The attention mechanism attends to the encoder output frames to generate the decoder input frames. Each decoder input frame is concatenated with the previous decoder frame output passed through a pre-net, making the model autoregressive. This concatenated vector goes through two unidirectional LSTM layers and is projected to produce a single mel spectrogram frame. Another projection of the same vector to a scalar allows the network to predict that it should stop generating frames by producing a value above a threshold. The entire sequence of frames is passed through a residual post-net before it becomes the generated mel spectrogram. This architecture is represented in figure 12.

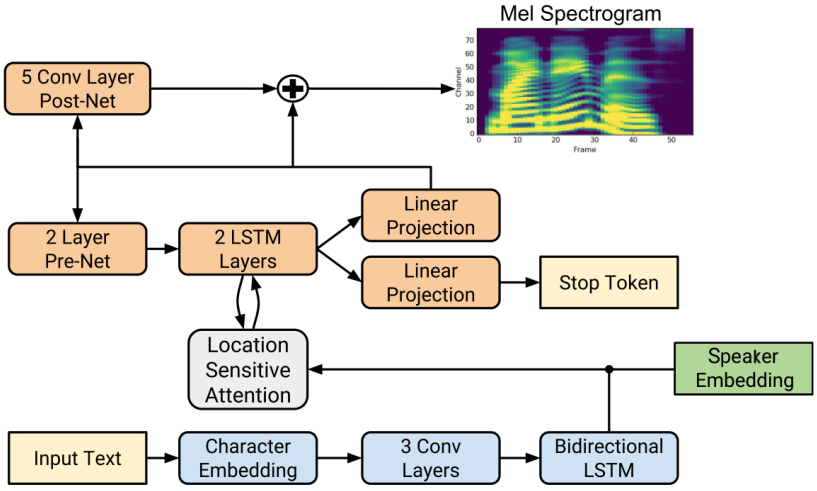

Figure 12: The modified Tacotron architecture. The blue blocks correspond to the encoder and the orange ones to the decoder. This figure was extracted from (Shen et al., 2017) and modified.

A noteworthy property of presenting the embedding at every encoder step is that it allows to change a voice through a sentence, e.g. by morphing a voice to another with a linear interpolation between their respective embedding.

The target mel spectrograms for the synthesizer present more features than those used for the speaker encoder. They are computed from a 50ms window with a 12.5ms step and have 80 channels. The input texts are not processed for pronunciation in our implementation.

4.4.2 Experiments

In SV2TTS, the authors consider two datasets for training both the synthesizer and the vocoder. These are LibriSpeech-Clean which we have mentioned earlier and VCTK[9]

[9] `https://homepages.inf.ed.ac.uk/jyamagis/page3/page58/page58.html`

which is a corpus of only 109 native English speakers recorded with professional equipment. The speech in VCTK is sampled at 48kHz and downsampled to 24kHz in their experiments, which is still higher than the 16kHz of LibriSpeech. They find that a synthesizer trained on LibriSpeech generalizes better than on VCTK when it comes to similarity but not naturalness. They assess this by training the synthesizer on one set, and testing it on the other. These results are in Table 3. We decided to work with the dataset that would offer the best voice cloning similarity on unseen speakers, and therefore picked LibriSpeech. We have also tried using the newer LibriTTS (Zen et al., 2019) dataset created by the Tacotron team. This dataset is a cleaner version of the whole LibriSpeech corpus with noisy speakers pruned out, a higher sampling rate of 24kHz and the punctuation that LibriSpeech lacks. Unfortunately, the synthesizer could not produce meaningful alignments on this dataset for reasons we ignore. We kept the original LibriSpeech dataset instead.

Synthesizer Training Set	Testing Set	Naturalness	Similarity
VCTK	LibriSpeech	4.28 ± 0.05	1.82 ± 0.08
LibriSpeech	VCTK	4.01 ± 0.06	2.77 ± 0.08

Table 3: Cross-dataset evaluation on naturalness and speaker similarity for unseen speakers. This table is extracted from (Jia et al., 2018)

Following the preprocessing recommendations of the authors, we use an Automatic Speech Recognition (ASR) model to force-align the LibriSpeech transcripts to text. We found the Montreal Forced Aligner[10] to perform well on this task. We've also made a cleaner version of these alignments public[11] to save some time for other users in need of them. With the audio aligned to the text, we can split utterances on silences. This helps the synthesizer to converge, both because of the removal of silences in the target spectrogram, but also due to the reduction of the median duration of the utterances in the dataset as shorter sequences offer less room for timing errors. We ensure that utterances are not shorter than 1.6 seconds, the duration of partial utterances used for training the encoder, and not longer than 11.25 seconds so as to save GPU memory for training. The distribution of the length of the utterances in the dataset is plotted in figure 13. Isolating the silences with force-aligning the text to the utterances additionally allows to create a profile of the noise for all utterances of the same speaker. We found a Fourier-analysis based noise removal algorithm to perform well on this task, but unfortunately could not reimplement this algorithm in our preprocessing pipeline.

In SV2TTS, the embeddings used to condition the synthesizer at training time are speaker embeddings. We argue that utterance embeddings of the same target utterance make for a more natural choice. At inference time, utterance embeddings are used. While the space of utterance and speaker embeddings is the same, speaker

[10] `https://montreal-forced-aligner.readthedocs.io/en/latest/`

[11] `https://github.com/CorentinJ/librispeech-alignments`

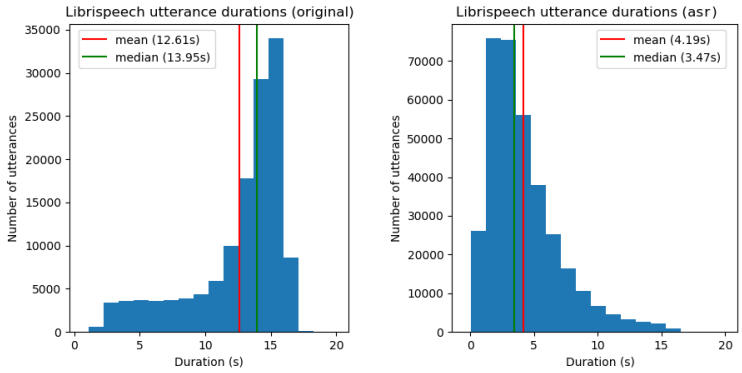

Figure 13: Histogram of the duration of the utterances in LibriSpeech-Clean before (left) and after (right) splitting utterances on silences.

embeddings are not L2-normalized. This difference in domain should be small and have little impact, as the authors agreed when we asked them about it. However, they do not mention how many utterance embeddings are used to derive a speaker embedding. One would expect that all utterances available should be used; but with a larger number of utterance embeddings, the average vector (the speaker embedding) will further stray from its normalized version. Furthermore, the authors mention themselves that there are often large variations of tone and pitch within the utterances of a same speaker in the dataset, as they mimic different characters (see SV2TTS appendix B). Utterances have lower intra-variation as their scope is limited to a sentence at most. Therefore, the embedding of an utterance is expected to be a more accurate representation of the voice in the utterance than the embedding of the speaker. This holds if the utterance is long enough than to produce a meaningful embedding. While the "optimal" duration of reference speech was found to be 5 seconds, the embedding is shown to be already meaningful with only 2 seconds of reference speech (see table 4). We believe that with utterances no shorter than the duration of partial utterances, the utterance embedding should be sufficient for a meaningful capture of the voice, hence we used utterance embeddings for training the synthesizer.

	1 sec	2 sec	3 sec	5 sec	10 sec
Naturalness (MOS)	4.28 ± 0.05	4.26 ± 0.05	4.18 ± 0.06	4.20 ± 0.06	4.16 ± 0.06
Similarity (MOS)	2.85 ± 0.07	3.17 ± 0.07	3.31 ± 0.07	3.28 ± 0.07	3.18 ± 0.07
SV-EER	17.28%	11.30%	10.80%	10.46%	11.50%

Table 4: Impact of duration of the reference speech utterance. Evaluated on VCTK. This table is extracted from (Jia et al., 2018).

training, gta, parameters

Unfortunately, as is also the case for the vocoder, it is difficult to provide any quantitative assessment of the performance of the model. We can observe that the model is producing correct outputs through informal listening tests, but evaluating

it would require us to setup subjective score polls to derive the MOS. While most authors we referred to could do so, this is beyond our possibilities. In the case of the synthesizer however, one can also verify that the alignments generated by the attention module make sense.

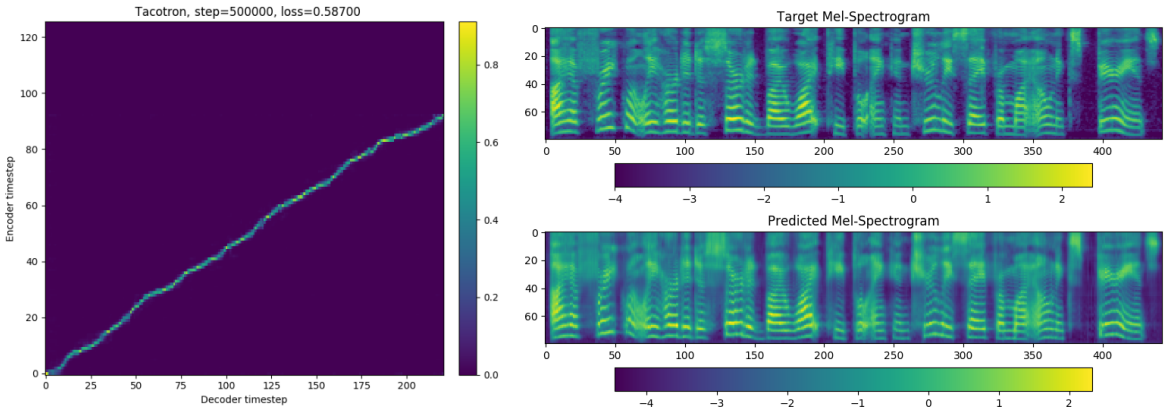

Figure 14: (left) Example of alignment between the encoder steps and the decoder steps. (right) Comparison between the predicted spectrogram (with GTA) and the ground truth spectrogram.

4.5 Vocoder

References

Sercan Arik, Gregory Diamos, Andrew Gibiansky, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech, 2017.

Sercan O. Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples, 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Mary E. Beckman and Gayle Ayers Elam. Guidelines for tobi labelling, 03 1997.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, KyungHyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *CoRR*, abs/1506.07503, 2015. URL `http://arxiv.org/abs/1506.07503`.

J. S. Chung, A. Nagrani, and A. Zisserman. Voxceleb2: Deep speaker recognition. In *INTERSPEECH*, 2018.

Y Fan, Yuang Qian, Feng-Long Xie, and Frank Soong. Tts synthesis with bidirectional lstm based recurrent neural networks. pages 1964–1968, 01 2014.

Xavi Gonzalvo, Siamak Tazari, Chun-an Chan, Markus Becker, Alexander Gutkin, and Hanna Silen. Recent advances in google real-time hmm-driven unit selection synthesizer. In *Interspeech*, pages 2238–2242, 2016.

K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda. The effect of neural networks in statistical parametric speech synthesis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4455–4459, April 2015. doi: 10.1109/ICASSP.2015.7178813.

Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer. End-to-end text-dependent speaker verification. *CoRR*, abs/1509.08062, 2015. URL `http://arxiv.org/abs/1509.08062`.

S. Imai. Cepstral analysis synthesis on the mel frequency scale. In *ICASSP '83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 93–96, April 1983. doi: 10.1109/ICASSP.1983.1172250.

Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez-Moreno, and Yonghui Wu. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *CoRR*, abs/1806.04558, 2018. URL `http://arxiv.org/abs/1806.04558`.

Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis, 2018.

Leland McInnes and John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. 02 2018.

A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, April 2015. doi: 10.1109/ICASSP.2015.7178964.

Y. Qian, Y. Fan, W. Hu, and F. K. Soong. On the training aspects of deep neural network (dnn) for parametric tts synthesis. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3829–3833, May 2014. doi: 10.1109/ICASSP.2014.6854318.

Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, R. J. Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. *CoRR*, abs/1712.05884, 2017. URL `http://arxiv.org/abs/1712.05884`.

S. Shirali-Shahreza and G. Penn. Mos naturalness and the quest for human-like speech. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 346–352, Dec 2018. doi: 10.1109/SLT.2018.8639599.

K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. Speech parameter generation algorithms for hmm-based speech synthesis. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, volume 3, pages 1315–1318 vol.3, June 2000. doi: 10.1109/ICASSP.2000.861820.

Yoshihiko; Toda Tomoki; Zen Heiga; Yamagishi Junichi; Oura Keiichiro Tokuda, Keiichi; Nankaku. Speech synthesis based on hidden markov models. *Proceedings of the IEEE*, 101, 05 2013. doi: 10.1109/JPROC.2013.2251852.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016. URL `http://arxiv.org/abs/1609.03499`.

Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification, 2017.

Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: A fully end-to-end text-to-speech synthesis model. *CoRR*, abs/1703.10135, 2017. URL `http://arxiv.org/abs/1703.10135`.

Takayoshi Yoshimura, Takashi Masuko, Keiichi Tokuda, Takao Kobayashi, and Tadashi Kitamura. Speaker interpolation in hmm-based speech synthesis system. In *EUROSPEECH*, 1997.

Takayoshi Yoshimura, Keiichi Tokuda, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Simultaneous modeling of spectrum, pitch and duration in hmm-based speech synthesis. In *EUROSPEECH*, 1999.

H. Zen, A. Senior, and M. Schuster. Statistical parametric speech synthesis using deep neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7962–7966, May 2013. doi: 10.1109/ICASSP.2013.6639215.

Heiga Zen, Yannis Agiomyrgiannakis, Niels Egberts, Fergus Henderson, and Przemyslaw Szczepaniak. Fast, compact, and high quality LSTM-RNN based statistical parametric speech synthesizers for mobile devices. *CoRR*, abs/1606.06061, 2016. URL `http://arxiv.org/abs/1606.06061`.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. *CoRR*, abs/1904.02882, 2019. URL `http://arxiv.org/abs/1904.02882`.