Deep Neural Networks on Text-to-Speech Synthesis

Panagiotis Tsagkaratos

Thesis submitted in partial fulfillment of the requirements for the *Masters' of Science degree in Information Systems and Services*.

University Of Piraeus
Department of Digital Systems
Piraeus University Campus, 185 34 Piraeus, Greece

Thesis Advisor: Prof. *Michael Filippakis*

2022

Deep Neural Networks on Text-to-Speech Synthesis

Abstract

Text-to-speech (TTS) synthesis is the automatic conversion of written text to spoken language. TTS systems play an important role in natural human-computer interaction. Concatenative speech synthesis and statistical parametric speech synthesis were the prominent methods used for decades. In the era of Deep learning, TTS systems have dramatically improved the quality of synthetic speech. The aim of this work was the comparison of [1] with the latest development in the field of TTS and suggesting improvements. The neural network architecture of Tacotron-2 is used for speech synthesis directly from text. The system is composed of a recurrent sequence-to-sequence feature prediction network that maps character embeddings to acoustic features, followed by a modified WaveNet model acting as a vocoder to synthesize time-domain waveforms from the predicted acoustic features. Developing TTS systems for any given language is a significant challenge and requires large amount of high quality acoustic recordings.

Βαθιά Νευρωνικά Δίκτυα Για Τη Σύνθεση Ομιλίας Από Κείμενο

Περίληψη

Η σύνθεση ομιλίας από κείμενο (TTS) είναι η αυτόματη μετατροπή του γραπτού λόγου σε προφορικό. Τα συστήματα σύνθεσης ομιλίας από κείμενο παίζουν σημαντικό ρόλο στη διάδραση ανθρώπου-υπολογιστή. Η συνενωτική σύνθεση ομιλίας και η στατιστική παραμετρική σύνθεση ομιλίας ήταν οι μέθοδοι που εφαρμόστηκαν για δεκαετίες. Στην εποχή της Βαθιάς Μάθησης, τα συγκεκριμένα συστήματα έχουν βελτιώσει δραματικά την ποιότητα της συνθετικής ομιλίας. Ο στόχος αυτής της εργασίας είναι η σύγκριση του [1] με τις τελευταίες εξελίξεις στον τομέα TTS και παράλληλα, η παράθεση προτάσεων για περαιτέρω βελτίωση του. Η αρχιτεκτονική νευρωνικού δικτύου του Tacotron-2 χρησιμοποιείται για σύνθεση ομιλίας κατευθείαν από κείμενο. Το σύστημα αποτελείται από ένα αναδρομικό από-ακολουθία-σε-ακολουθία δίκτυο πρόβλεψης χαρακτηριστικών, που αντιστοιχίζει ενσωματώσεις χαρακτήρων σε φασματογράμματα κλίμακας Μελ που ακολουθείται από ένα τροποποιημένο μοντέλο WaveNet, που λειτουργεί ως συνθεσάϊζερ ομιλίας για να συνθέσει κυματομορφές στο πεδίο του χρόνου από αυτά τα ακουστικά χαρακτηριστικά. Η ανάπτυξη συστημάτων σύνθεσης ομιλίας από κείμενο για μια δεδομένη γλώσσα είναι μια σημαντική πρόκληση και απαιτεί μεγάλη ποσότητα ηχογραφήσεων υψηλής ποιότητας.

Acknowledgements

I would like to express my sincere gratitude to my advisor and excellent teacher Prof. Michael Filippakis for the support, motivation, enthusiasm and the opportunity he gave me to explore the field of TTS. Furthermore, I would like to thank Dr. Poulou Marilena for the assistance in supervising the analysis of the data, her contribution to the experimental part of the dissertation and her helpful comments in the research analysis. Also, I must express my profound gratitude to my family for the encouragement and the time that we spend together. Lastly, I thank my friends and students from the Master for their support throughout the duration of the program.

Contents

Chapter 1

Introduction

Speech Synthesis is described as the artificial production of human speech. Its aim is to create intelligible and natural audio which is indistinguishable from human recordings. A speech synthesizer is a computer system used for this purpose. Speech synthesis has applications that can be used by people with a wide range of disabilities (screen readers for people with visual impairment), or people with dyslexia. Other significant applications of speech synthesis are used for entertainment productions such as games and animations and for creation of educational tools for foreign languages. Speech Synthesis combined with Speech Recognition allows interaction with mobile devices via natural language processing interfaces.

Text-To-Speech (TTS) synthesis is called the automatic conversion of written to spoken language. The input is text and the output is speech waveform [2]. A TTS system is divided into two parts. The first part is called the front-end and converts text into linguistic specifications. The second part is called the backend and uses these specifications to generate a waveform. These two modules are conventionally constructed independently. The linguistic specification comprises whatever factors might affect the acoustic realisation of the speech sounds making up the utterance. Many tasks performed by the front end (e.g. predicting pronunciation from spelling) are quite specific to one language or one family of languages [3]. The text-analysis module is trained using text corpora and often includes statistical models to analyze text, e.g., the phrasing boundary, accent, and POS. The waveform generation module, on the other hand, is trained using a labeled speech database. In statistical parametric synthesis, this module includes acoustic models.

In the era of Deep Learning, end-to-end neural networks, which are the state of the art for speech recognition tasks, have become highly competitive with the conventional TTS systems. An end-to-end TTS system can be trained on pairs without hand-crafted feature engineering. All the modules are trained together to optimize a global performance criterion, without manual integration of separately trained modules. It allows rich conditioning on various attributes such as, speaker, language, sentiment, etc. It is more robust than a multi-component system. It has the potential for transfer learning and it can adapt to new data. It may be trained on huge amount of often noisy data found in the real world.

Neural TTS systems proposed nowadays, are: Deep Voice 1 [41], Deep Voice 2 [40], Deep Voice 3 [42], Tacotron [23], Tacotron 2 [26], Char2Wav [43], VoiceLoop (Taigman et al., 2018), and ClariNet [42]. Deep Voice 1 and 2 retain the traditional TTS pipeline, which has separate graphemeto-phoneme, phoneme duration, frequency, and waveform synthesis models. In contrast, Tacotron, Deep Voice 3, and Char2Wav employ the attention based sequence-to-sequence models [44]. These models depend on a traditional vocoder, or a separately trained neural vocoder to convert the predicted spectrogram to raw audio. ClariNet, which is based on Deep Voice 3, seems to be the first text-to-wave neural architecture for TTS. Tacotron (1, 2) take characters as input and produce spectrogram frames, which are then converted to waveforms. Tacotron 1 produces linear-scale spectrograms and uses the Griffin-Lim algorithm to synthesize waveform from the predicted spectrogram. Tacotron 2 produces mel-scaled spectrograms, which are used as local conditioning to a WaveNet vocoder. In contrast to Tacotron 1, Tacotron 2 uses simpler building blocks, using vanilla LSTM and convolutional layers in the encoder and decoder instead of CBHG stacks and GRU recurrent layers.

1.1 History

In the past decades, the mainstream techniques were concatenative and parametric speech synthesis. Both of them had complex pipelines, required a lot of resource and manpower. They also were language specific.

An exemplar-based speech synthesis system simply stores the speech corpus itself. The stored speech data are labelled so that appropriate parts of them can be found, extracted and then concatenated during the synthesis phase. The most prominent exemplar based technique and one of the dominant approaches to speech synthesis is Unit-Selection (Fig. 1). In this technique, units from a large speech database are selected according to how well they match a specification and how well they join together. The specification and the units are completely described by a feature structure, which can be any mixture of linguistic and acoustic features. The quality of output derives directly from the quality of the recordings and it appears that the larger the database, the better the coverage. Commercial systems have exploited these techniques to bring about a new level of synthetic speech. However, these techniques limit the output speech to the same style as that in the original recordings. In addition, recording large databases with variations is very difficult and costly.

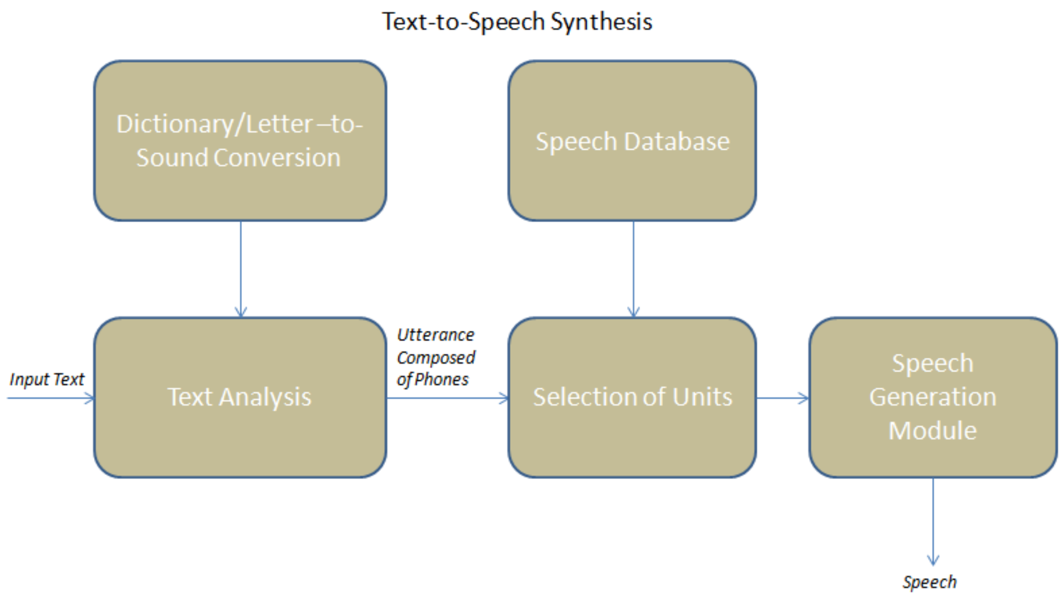


Figure 1: Modules of a unit selection TTS system.

A model-base speech synthesis system fits a model to the speech corpus (during the training phase) and stores this model. Due to the presence of noise and unpredictable factors in speech training data the models are usually statistical. Statistical Parametric speech synthesis [2] has also grown in popularity over the last years, in contrast to the selection of actual instances of speech. These models don't use stored exemplars. They describe the parameters of models using statistics (e.g. means and variances of probability density functions) which capture the distribution of parameter values found in the training data (Fig 1.2).

Most of the advantages of statistical parametric synthesis against unit-selection synthesis are related to its flexibility due to the statistical modelling process. Some of these advantages are:

- Transforming voice characteristics, speaking styles and emotions (by transforming its model parameters).
- Multilingual support (only the contextual factors to be used depend on each language).
- Small footprint compared with unit-selection (because statistics of acoustic models are stored rather than the multi-templates of the speech units). So statistical parametric speech synthesis seemed to be suitable for embedded applications.

The quality of speech produced by the initial statistical parametric systems was significantly lower than this of unit-selection systems. Three factors degrade the quality of speech of SPSS: a) the vocoder, b) the acoustic modelling accuracy and c) the over-smoothing of the synthesized speech parameters. Researchers tried to improve all these factors. The most frustrating task was the improvement of the vocoder. After decades of research the proposed vocoders [4], [5] introduced little or no improvement compared to straight vocoder [6] which existed since the early days of SPSS [7]. On the other hand, the invention of more advanced statistical models, such as the trajectory HMMs, leaded to smoother speech trajectories and increased user opinion scores [8].

Finally, the most rewarding task was the addressing of over-smoothing. Toda applied the Global-Variance (GV) heuristic, which is a post-processing algorithm that scales the variance of the trajectories of the synthesized speech parameters to be equal to the variance of the original speech [9]. As a result, the naturalness of the synthesized speech was greatly improved and approached the quality of the unit-selection systems but did not surpassed it. Up to the begging of 2016, the best quality TTS systems were either unit-selection or hybrid systems where the global structure of the trajectories is created by statistical models and the segments of actual instances of speech were chosen from a database.

The Hidden Markov Models (HMMs) has been proven a powerful model of speech [10]. An important reason for this, is the availability of effective and efficient learning algorithms (Expectation-Maximization algorithm that is used for the parameter estimation) [11, 12, 13]. However, there are several assumptions in HMMs that do not apply to the properties of speech.

Disadvantages of HMM synthesis are:

- The speech has to be generated by a parametric model, so no matter how naturally the models generate parameters, the final quality is very much dependent on the model used.
- Even with the dynamic constraints, the models generate somewhat 'safe' observations and fail to generate some of the more interesting and delicate phenomena in speech.
- The assumptions such as the observations are conditionally independent given the state sequence.
- The speech statistics of each state do not change dynamically.
- The Markov assumption itself: that the probability of being in a given state at time t only depends on the state at time t − 1, is inappropriate for speech sounds where dependencies often extend through several states.

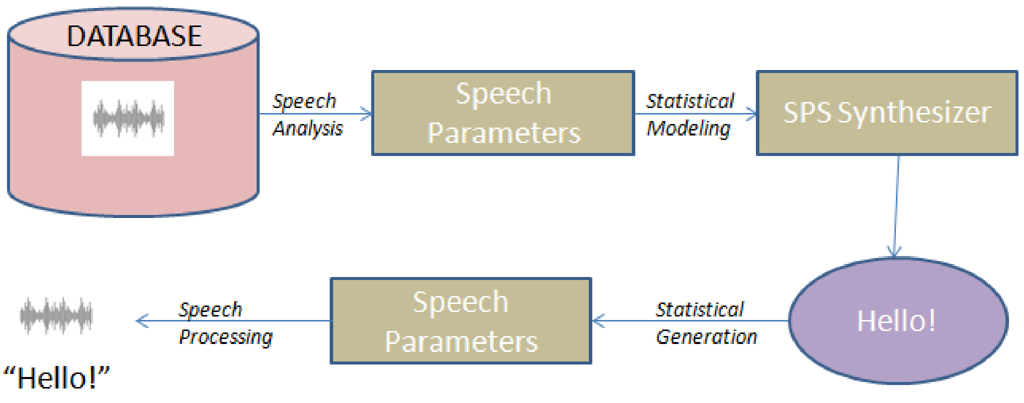


Figure 2: Statistical parametric speech synthesis**.**

Trajectory HMMs [8] can alleviate two limitations of the standard HMM, which are (i) piece-wise constant statistics within a state and (ii) conditional independence assumption of state output probabilities, without increasing the number of model parameters. In the same paper, a Viterbi-type training algorithm based on the maximum likelihood criterion was also derived. The performance of the trajectory HMM was evaluated both in speech recognition and synthesis. In a speaker-dependent continuous speech recognition experiment, the trajectory HMM achieved an error reduction over the corresponding standard HMM. Subjective listening test results showed that the introduction of the trajectory HMM improved the naturalness of synthetic speech. The formulation of the trajectory HMM is closely related to the technique for speech parameter generation from the standard HMM [7] in which the speech parameter sequence is determined so as to maximize its output probability for the standard HMM under the constraints between the static and dynamic features [30]. While the speech parameter generation algorithm was derived to construct HMM-based speech synthesizers which can synthesize speech with various voice characteristics the generation algorithm was also applied to speech recognition. Statistical parametric speech synthesis can benefit from a stronger model of the speech signal, with perhaps a more explicit representation of the physical and linguistic specification from text.

Linear Dynamical Models (LDMs) , also known as Kalman filter models, had been proposed to explicitly capture the dynamics of speech [15] [16] [17]. LDMs are probabilistic, state space models, which explicitly model some of the dynamics of speech and introduce the continuity and context dependence needed for good quality synthesis. These models have also the property that can be trained via the EM algorithm in a maximum likelihood framework. LDMs can produce a smoother trajectory of the synthesized speech (closer to the natural).

1.2 Analysis of related work

In [1], they implemented a TTS system which can generate natural speech for a variety of speakers in a data efficient manner. The system is composed of three independently trained neural networks, a recurrent speaker encoder, which computes a fixed dimensional vector from a speech signal, a sequence-to-sequence synthesizer based on Tacotron 2, which predicts a mel spectrogram from a sequence of grapheme or phoneme inputs, conditioned on the speaker embedding vector, and an autoregressive WaveNet vocoder, which converts the spectrogram into time domain waveforms. The goal was to generate speech audio in the voice of different speakers, including those unseen during training. In addition to this, a zero-shot learning setting is addressed, where a few seconds of untranscribed reference audio from a target speaker is used to synthesize new speech in that speaker’s voice, without updating any model parameters. It is known that in order to properly train a TTS model, a wide variety of data is needed, from a number of different speakers and a long duration of high quality recordings, without backround noise. However, in this paper, the approach is to decouple speaker modeling from speech synthesis by independently training a speakerdiscriminative embedding network that captures the space of speaker characteristics and simultaneously train a high quality TTS model on a smaller dataset conditioned on the representation learned by the first network. It is suggested that decoupling the networks enables them to be trained on independent data, which reduces the need to obtain high quality multispeaker training data. The speaker embedding network is trained on a speaker verification task to determine if two different utterances were spoken by the same speaker. In contrast to the subsequent TTS model, this network is trained on untranscribed speech containing reverberation and background noise from a large number of speakers. In fact, the synthesis network is trained on 1.2K speakers whereas the encoder is trained on a much larger set of 18K speakers, which seems to improve adaptation quality, and further enable synthesis of completely novel speakers by sampling from the embedding prior.

Two public datasets were used for training the speech synthesis and vocoder networks. VCTK contains 44 hours of clean speech from 109 speakers, the majority of which have British accents. It is split into three subsets: train, validation (containing the same speakers as the train set) and test (containing 11 speakers held out from the train and validation sets). LibriSpeech consists of the union of the two “clean” training sets, comprising 436 hours of speech from 1,172 speakers, sampled at 16 kHz. The majority of speech is US English, however since it is sourced from audio books, the tone and style of speech can differ significantly between utterances from the same speaker. As in the original dataset, there is no punctuation in transcripts. The speaker sets are completely disjoint among the train, validation, and test sets. Many recordings in the LibriSpeech clean corpus contain noticeable environmental and stationary background noise. A preprocessing technique to remove backround noise was only used on the synthesis target; the original noisy speech was passed to the speaker encoder.

One evaluation metric was speech naturalness. They compared the naturalness of synthesized speech using synthesizers and vocoders trained on VCTK and LibriSpeech, on seen and unseen data. The VCTK model achieved a higher Mean Opinion Score (MOS) than LibriSpeech, mainly because of the lack of punctuation in transcripts in LibriSpeech, which makes it difficult for the model to learn to pause naturally, and the higher level of background noise compared to VCTK, despite the denoising preprocessing. However, MOS was about the same on seen and unseen speakers, even higher on the LibriSpeech model.

Another evaluation metric was speech similarity, where the scores for the VCTK model tend to be higher than those for LibriSpeech, most likely due to the wider degree of within-speaker variation in LibriSpeech, and background noise level in the dataset. However, on unseen data, the proposed model obtains lower similarity between ground truth and synthesized speech, suggesting that some nuances like prosody are lost. The speaker encoder is trained only on North American accented speech. As a result, accent mismatch constrains the performance on speaker similarity on VCTK. Lastly, in order to generalize to out of domain speakers, they used synthesizers trained on VCTK and LibriSpeech to synthesize speakers from the other dataset. Results showed that the LibriSpeech model synthesized VCTK speakers with significantly higher speaker similarity than the VCTK model is able to synthesize LibriSpeech speakers. The better generalization of the LibriSpeech model suggests that training the synthesizer on only 100 speakers is insufficient to enable high quality speaker transfer.

In addition to the previous metric, they evaluated the ability of a limited speaker verification system to distinguish synthetic from real speech, based on a different training set of 28M utterances from 113K speakers for the training of speaker encoder. Results showed that the VCTK model performed much worse than LibriSpeech, but in general, that the proposed model could generate speech that resembles the target speaker, but not well enough to be confusable with a real speaker. Nevertheless, the experiments showed that speaker transfer quality could be significantly improved by increasing the amount of speaker encoder training data. Lastly, the model is able to generate realistic speech from fictitious speakers that are dissimilar from the training set, implying that the model has learned to utilize a realistic representation of the space of speaker variation.

Chapter 2

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

2.1 Overview

Our approach to real-time voice cloning is largely based on [1] (referred to as SV2TTS throughout this document) and [19]. It describes a framework for zero-shot voice cloning that solely needs 5 seconds of reference speech. This paper is merely one of the many publications from the Tacotron series authored at Google. Apparently, the SV2TTS paper doesn’t bring abundant innovation of its own, rather it is supported by three major earlier works from Google: the GE2E loss [22], Tacotron [23] and WaveNet [24]. The whole 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 or the Google cloud services , make use of these same models. The three stages of the framework are as follows:

• A speaker encoder that derives associate degree embedding from the short auditory communication of a single speaker. The embedding could be a meaningful illustration of the voice of the speaker, such as similar voices square are close in latent space. This model is described in [22] (referred as GE2E throughout this document) and [25].

• A synthesizer that, conditioned on the embedding of a speaker, generates a spectrogram from text. This model is the standard Tacotron 2 [26] without WaveNet (which is usually mentioned as simply Tacotron because of its similarity to the primary iteration).

• A vocoder that infers associate degree audio wave shape from the spectrograms generated by the synthesizer. The authors used WaveNet [24] as a vocoder, effectively reapplying the whole Tacotron 2 framework.

At logical thinking time, the speaker encoder is fed a brief reference auditory communication of the speaker to clone. It generates associate degree embedding that's used to condition the synthesizer, and a text processed as a speech sound sequence is given as input to the synthesizer. The vocoder takes the output of the synthesizer to come up with the speech wave shape.

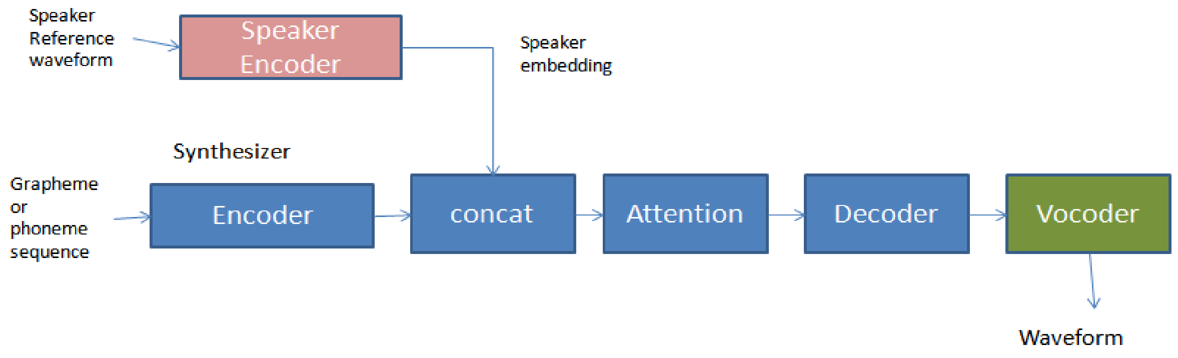


Figure 3: The SV2TTS framework during inference. The blue blocks represent a high-level view of the Tacotron architecture modified to allow conditioning on a voice.

A feature of the SV2TTS framework is that each one of all models used, will be trained separately and on distinct datasets. For the encoder, one seeks to own a model that is strong to noise and able to capture the various characteristics of the human voice. Therefore, an outsized corpus of the many completely different speakers would be preferred to train the encoder, without any robust demand on the background level of the audios. Additionally, the encoder is trained with the GE2E loss which needs no labels other than the speaker identity. With GE2E, the task to be learned by the model could be a speaker verification task, that by itself has very little to try to to with voice biological research. However, the task is stipulated in a way that the network can output an embedding that's a meaningful illustration of the voice of the speaker. This embedding is appropriate for conditioning the synthesizer on a voice, therefore the name of the paper: “Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis”. For the datasets of the synthesizer and also the vocoder, transcripts are required and also the quality of the generated audio can solely be nearly as good as that of the data. Higher quality and annotated datasets are thus needed, which regularly means that they're smaller in size.

In the following segment, we formally outline the task that SV2TTS aims to solve and within the remaining subsections, we present the 3 parts of the framework as described in [1], our implementation of each and therefore the results of our experiments.

2.2 Problem definition

Consider a dataset of utterances classified by their speaker. We denote the *j*th utterance of the *i*th speaker as $\mathbf{u}_{ij}$. Utterances are in the waveform domain. We denote by x_{ij} the log-mel spectrogram of the vocalization $\mathbf{u}_{ij}$. A log-mel spectogram is a settled, non-invertible (lossy) function that extracts speech characteristics from a waveform, so as to handle speech in machine learning. The encoder ***E*** computes the embedding $\mathbf{e}_{ij} = \mathbf{E}(\mathbf{x}_{ij} ; \mathbf{w}_E)$ similar to the utterance $\mathbf{u}_{ij}$, where $\mathbf{w}_E$ are the parameters of the encoder. In addition, the authors outline a speaker embedding as the centroid of the embeddings of the speaker’s utterances:

$$c_i = \frac{1}{n}\sum_{j=1}^{n} e_{ij} \qquad (1)$$

The synthesizer ***S***, parametrized by $\boldsymbol{w}_S$, is tasked to approximate $\mathbf{x}_{ij}$ given $\mathbf{c}_i$ and $\mathbf{t}_{ij}$, the transcript of utterance $\mathbf{u_{ij}}$. We have $\mathbf{x_{ij}} = \boldsymbol{S}(\mathbf{c_i}, \mathbf{t_{ij}}; \mathbf{w_S})$. In our implementation, we directly use the utterance embedding rather than the speaker embedding), giving instead have $\mathbf{x_{ij}} = \boldsymbol{S}(\mathbf{u_{ij}}, \mathbf{t_{ij}}; \mathbf{w_S})$.

Finally, the vocoder $\boldsymbol{V}$, parametrized by $\mathbf{w_V}$, is tasked to approximate $\mathbf{u_{ij}}$ given $\mathbf{x_{ij}}$. We have $\mathbf{u_{ij}} = \boldsymbol{V}(\mathbf{x_{ij}}; \mathbf{w_V})$. One could train this framework in an end-to-end fashion with the following objective function:

$$\min_{w_E, w_S, w_V} L_V(u_{ij}, V(S(E(x_{ij}; w_E), t_{ij}; w_S); w_V)) \quad (2)$$

Where L_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: an outsized range of speakers for the encoder but also, transcripts for the synthesizer. A low level noise for the synthesizer and somehow a mean background level for the encoder (in order to be able to handle abuzz input speech). These conflicts are problematic and would lead to training models that would perform better if trained individually on distinct datasets. Specifically, a small dataset can probably lead to poor generalization and therefore poor zero-shot performance.

Another disadvantage is that the convergence of the combined model might be extremely arduous to achieve. Indeed, the Tacotron synthesizer might take a long time to establish a number of correct alignments. An evident approach of addressing the second issue is to separate the training of the synthesizer and of the vocoder. Assuming a pretrained encoder, the synthesizer will be trained to directly predict the mel spectrograms of the target audio:

$$\min_{w_S} L_S(x_{ij}, S(e_{ij}, t_{ij}; w_S)) \quad (3)$$

Where L_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. The latter requires a pretrained synthesizer.

$$\min_{w_V} L_V(u_{ij}, V(x_{ij}; w_V)) \quad (4)$$

Remains the improvement of the speaker encoder. Unlike the synthesizer and also the vocoder, the encoder doesn't have labels to be trained on. The task is lousily outlined as producing “meaningful” embeddings that characterize the voice within the vocalization. One might imagine some way to train the speaker encoder as associate autoencoder, but it would need the corresponding upsampling model to be 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 results of the coaching are largely affected by the dataset and unlikely to generalize well. Fortunately, the GE2E loss [22] brings an answer to this issue and allows to train the speaker encoder independently of the synthesizer.

While all components of the framework are trained individually, there's still the need for the synthesizer to own embeddings from a trained encoder and for the vocoder to own mel spectrograms from a trained synthesizer (if not training on ground truth spectrogram). However, every model depends on the previous one for training. The speaker encoder has to generalize well enough to produce meaningful embeddings on the dataset of the synthesizer, and even once trained on a standard dataset, it still needs to be able to operate during a zero-shot setting at inference time.

2.3 Speaker encoder

The encoder model and its training procedure are described over several papers [1] [22] [25]. We reproduced this model with the implementation of [19], where they synthesized the parts that are outlined in SV2TTS as well as personal choices of implementation.

2.3.1 Model architecture

The model is a 3-layer LSTM with 768 hidden nodes followed by a projection layer of 256 units. While there's no reference in any of the papers on what a projection layer is, our intuition is that it's merely a 256 outputs fully-connected layer per LSTM that is repeatedly applied to each output of the LSTM. When we initially enforced the speaker encoder, we directly used 256 units LSTM layers instead, for the sake of fast prototyping, simplicity and for a lighter training load. This last half is vital, as the authors have trained their own model for fifty million steps (although on a bigger dataset), which is technically difficult for us to reproduce. We found this model to perform significantly well. 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 integration conjointly options a ReLU layer before the normalization, with the goal in mind to create embeddings sparse and thus more understandable.

2.3.2 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 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. This entire process is illustrated in Figure 4. From a computing perspective, the cosine similarity of two L2-normed vectors is simply their dot product. An optimal model is expected to output high similarity values when an utterance matches the speaker (i = k) and lower values elsewhere (i 6= k). To optimize in this direction, the loss is the sum of row-wise softmax losses.

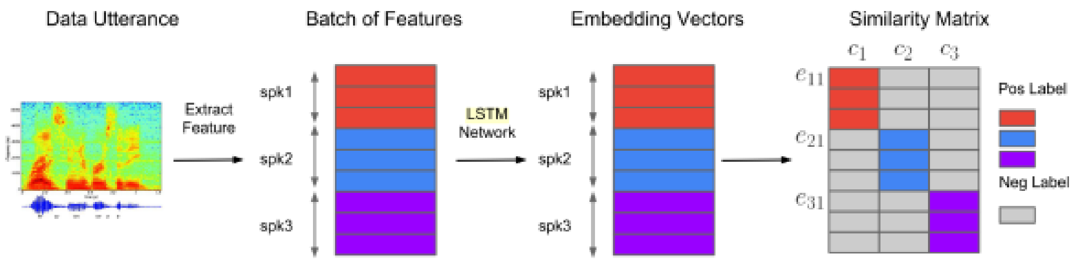


Figure 4: The construction of the similarity matrix at training time. This figure is extracted from [22].

Note that each utterance $\boldsymbol{e}_{ij}$ is included in the centroid $\boldsymbol{c}_i$ of the same speaker when calculating the loss. This creates a bias towards the correct speaker independently of the accuracy of the model; and the authors argue that it also leaves room for trivial solutions. To prevent this, an utterance that is compared against its own speaker's embedding will be removed from the speaker embedding.

The fixed duration of the utterances in a training batch is approximately 1.6 seconds. These 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 5. 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.

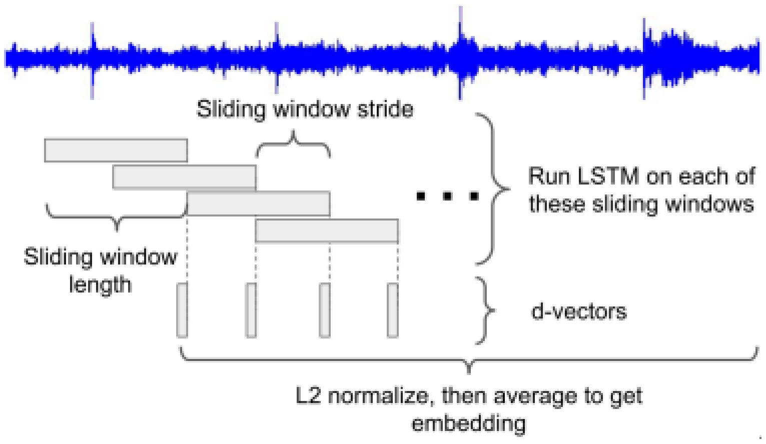

Sliding window stride
Run LSTM on each of these sliding windows
Sliding window length
d-vectors
L2 normalize, then average to get embedding

Figure 5: Computing the embedding of a complete utterance. The d-vectors are simply the unnormalized outputs of the model. This figure is extracted from [22].

The authors use N = sixty four and M = ten as parameters for the batch size. When enrolling a speaker in a practical exposure, one ought to expect to have many utterances from every user however not of a magnitude higher than of ten, so this choice is reasonable. As for the quantity of speakers, it's optimal to state that the time complexity of computing the similarity matrix is $O(N^2M)$, thus this parameter should be chosen high enough in order to not slow down significantly the training, as opposed to merely selecting the largest batch size that fits on the GPU. It's still of course attainable to set multiple batches on the same GPU while synchronizing the operations across batches for efficiency. We found it vital to vectorize all operations once computing the similarity matrix, so as to minimize the number of GPU transactions.

2.4 Synthesizer

The synthesizer is Tacotron 2 without Wavenet [24]. An open-source implementation of Tacotron 2 is used from which Wavenet is excluded, and impelemented the modifications added by SV2TTS.

The top-level architecture of the modified Tacotron 2 without Wavenet (which we'll refer to as simply Tacotron) is briefly presented below. For further details, we invite the reader to take a look at the Tacotron papers [26] [23].

Tacotron is a continual sequence-to-sequence model that predicts a mel spectrogram from text. It integrates an encoder-decoder structure that is bridged by a location-sensitive attention mechanism [27]. Individual characters from the text sequence are initially embedded as vectors. Convolutional layers follow, so as to increase the span of a single encoder frame. These frames are passed through a bidirectional LSTM to produce the encoder output frames. This is where SV2TTS offers a modification to the architecture: a speaker embedding is concatenated to every frame that is output by the Tacotron 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 before being projected to a single mel spectrogram frame. Another projection of the same vector to a scalar allows the network to predict on its own that it should stop generating frames by emitting a value above a certain threshold. The entire sequence of frames is passed through a residual post-net before it becomes the mel spectrogram. This architecture is represented in Figure 6.

The target mel spectrograms for the synthesizer offer 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 this implementation, and the characters are fed as they are. There are a few preprocessing techniques however: replacing abbreviations and numbers by their complete textual form, forcing all characters to ASCII, normalizing whitespaces and making all characters lowercase. Punctuation could be used, but isn't present in the datasets used.

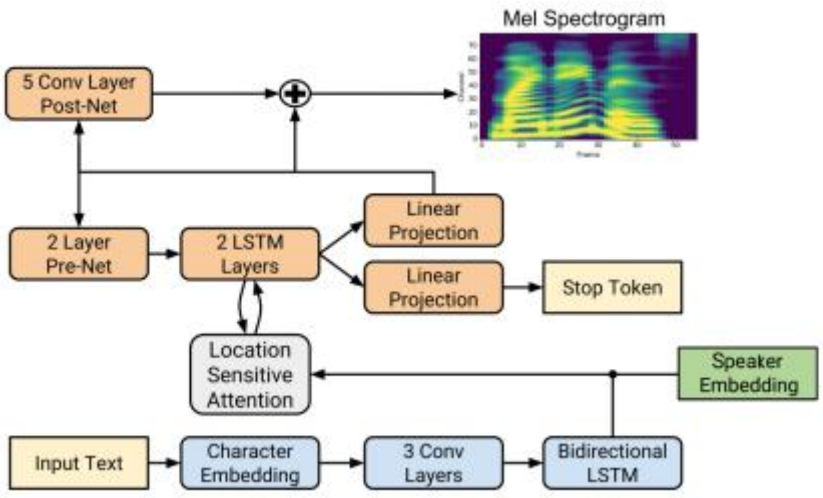


Figure 6: The modified Tacotron architecture. The blue blocks correspond to the encoder and the orange ones to the decoder. This figure was extracted from [26].

2.5 Vocoder

WaveNet is the vocoder In SV2TTS and in Tacotron2. WaveNet has a significant role in deep learning with audio since its release and remains state of the art when it comes to voice naturalness in TTS. It is however also known for being one of the slowest practical deep learning tools at inference time. Several papers along the time, brought improvements on that aspect to bring the generation near real-time or faster than real-time, e.g. [24] [28] [18], with no or next to no hit to the quality of the generated speech. Nevertheless, WaveNet remains the vocoder in SV2TTS as the authors are not mainly concerned about the speed of execution and because Google's own WaveNet implementation with various improvements already generates at 8000 samples per second [18]. This is in contrast with the "standard" WaveNet which generates at 172 steps per second at best [24].[18] proposes a simple scheme for describing the inference speed of autoregressive models. Given a target vector u with |**u**| samples to predict, the total time of inference T(**u**) can be decomposed as:

$$T(u) = |u| \sum_{i=1}^{N} (c(op_i) + d(op_i)) \qquad (5)$$

where *N* is the number of matrix-vector products (∝ the number of layers) that is needed to produce one sample, c(op_i) is the computation time of layer *i* and d(op_i) is the overhead of the computation (typically I/O operations) for layer *i*. Note that standard sampling rates for speech include 16kHz, 22.05kHz and 24kHz (while music is normally sampled at 44.1kHz), meaning that for just 5 seconds of audio |**u**| is near to 100,000 samples. The standard WaveNet architecture accounts for three stacks of 10 residual blocks of two layers each, heading to *N* = 60.

WaveRNN, the model described in [18] improves on WaveNet by not only reducing the contribution from *N* but also from **u**, *c(op_i)* and *d(op_i)*. The vocoder model we use is an open source implementation that is based on WaveRNN but presents a number of different design choices made by github user fatchord. We'll refer to this architecture as the "alternative WaveRNN".

Regarding architecture, all 60 convolutions from WaveNet are substituted by a single GRU layer [29]. The authors state that the high non-linearity of a GRU layer alone is close enough to compare to the complexity of the entire WaveNet model. Indeed, they report a MOS of 4.51 ± 0.08 for Wavenet and 4.48 ± 0.07 for their best WaveRNN model. The inputs to the model are the GTA mel spectrogram generated by the synthesizer, with the ground truth audio as target. At training time the model predicts fixed-size waveform segments. The forward pass of WaveRNN is implemented with only N = 5 matrix-vector products in a coarse-fine scheme where the lower 8 bits (coarse) of the target 16 bits sample are predicted first and then used to condition the prediction of the higher 8 bits (fine). The prediction is over the parameters of a distribution from which the output is sampled.

The authors aim to improve on the factors c(op_i) and d(op_i) by implementing the sampling operation as a custom GPU operation. We do not adopt the same approach. They also sparsify the network with the pruning strategy from [32] [33].This method gradually prunes weights during training, in contrast to more classical pruning algorithms that operate between numerous trainings. The algorithm creates a binary mask over the weights that indicates whether the weight should be forced set to 0 or stay as is. The proportion of zero weights compared to the total number of weights in the network is called sparsity. Results from [32] [33] indicate that large networks with sparsity levels between 90% and 95%

significantly outperform their dense versions. The authors of WaveRNN additionally state that $c(op_i)$ is proportional to the number of nonzero weights.

Finally, $|\mathbf{u}|$ is improved with batched sampling, with the utterance divided in segments of specified length while the generation is done in parallel over all segments. To save some context between the end of a segment and the beginning of the subsequent one, a small section of the end of a segment is repeated at the beginning of the next one. This process is called folding. The model then forwards the folded segments. To retrieve the unfolded tensor, the overlapping sections of consecutive segments are merged by a cross-fade. In the present thesis, we use batched sampling with the alternative WaveRNN, with a segment length of 8000 samples and an overlap length of 400 samples. With these parameters, a folded batch of size 2 will yield a bit more than 1 second of audio for 16kHz speech.

In this thesis, we use the alternative WaveRNN and we rely on the source code and on the diagram of the author (Figure 7) to comprehend its inner workings. At every single training step, a mel spectrogram and its corresponding waveform are split in the same number of segments. The inputs to the model are the spectrogram segment t in order to predict and the waveform segment $t - 1$. The model is expected to have as a result the waveform segment t of identical length. The mel spectrogram goes through an upsampling network to correspond with the length of the target waveform, while the number of mel channels remains the same. In addition, a Resnet-like model uses the spectrogram as input to generate features that will condition the layers throughout the transformation of the mel spectrogram to a waveform. The resulting vector is repeated to address the length of the waveform segment. This conditioning vector is then divided equally four ways along the channel dimension, and the first part is concatenated with the upsampled spectrogram and with the waveform segment of the previous timestep. The resulting vector goes through excessive transformation with skip connections: first two GRU layers then a dense layer. Between each step, the conditioning vector is concatenated with the intermediate waveform. Finally, two dense layers produce a distribution over discrete values that correspond to a 9-bit encoding of mu-law companded audio.

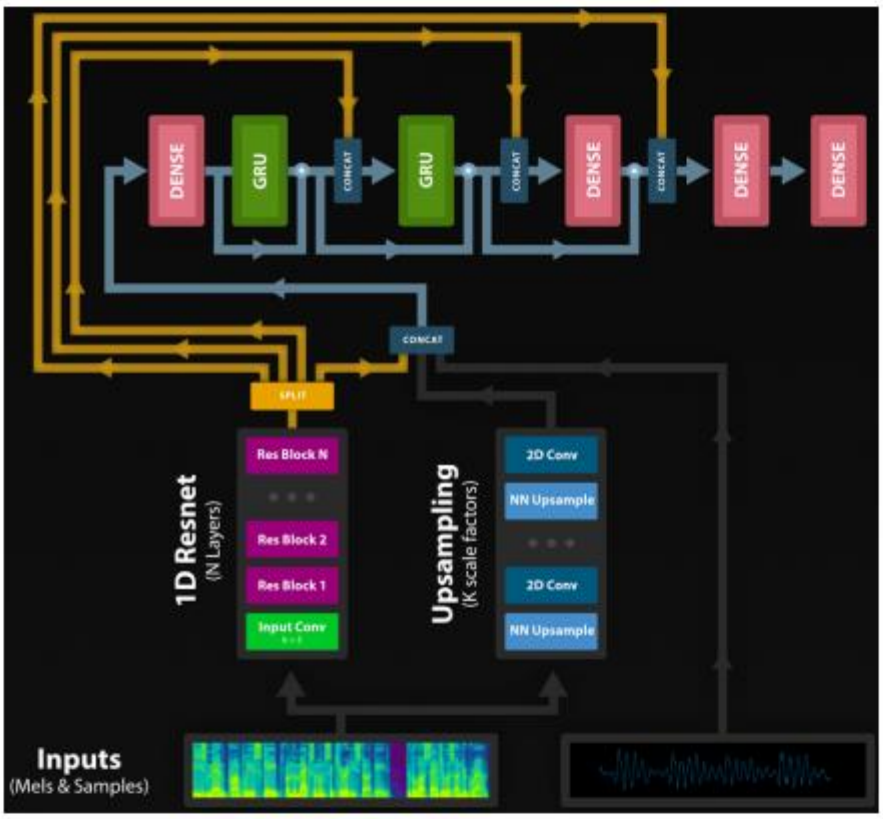


Figure 7: The alternative WaveRNN architecture. This figure is extracted from [45].

Chapter 3

Experiments

As the framework of SV2TTS is basically divided in three main components, it is vital to understand that there are several changes that we can apply in the training of the encoder, synthesizer and vocoder, and there is no right or wrong combination of modifications that can guarantee a natural output. However, as the encoder demands a significant amount of data in order to be trained efficiently, and as a result, a long training time and resources, we select the initial model trained in [19] and experiment with the synthesizer and the vocoder. In the following section 3.1, we outline a number of preprocessing steps and describe the training process of the encoder.

3.1 Experiments with the encoder

Aiming to avoid segments that are mostly silent when sampling partial utterances from complete utterances, the python package webrtcvad8 is used to perform Voice Activity Detection (VAD). This yields a binary flag over the audio discriminating whether or not the segment is voiced. A moving average is performed on this binary flag to smooth out short spikes in the detection, following an additional binarization. Lastly, a dilation is conducted 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. The outcome of s = 0.2s seems to be an effective choice that retains a natural speech prosody. This process is illustrated in Figure 8. A last preprocessing step applied to the audio waveforms is normalization, which is needed to make up for the varying volume of the speakers in the dataset.

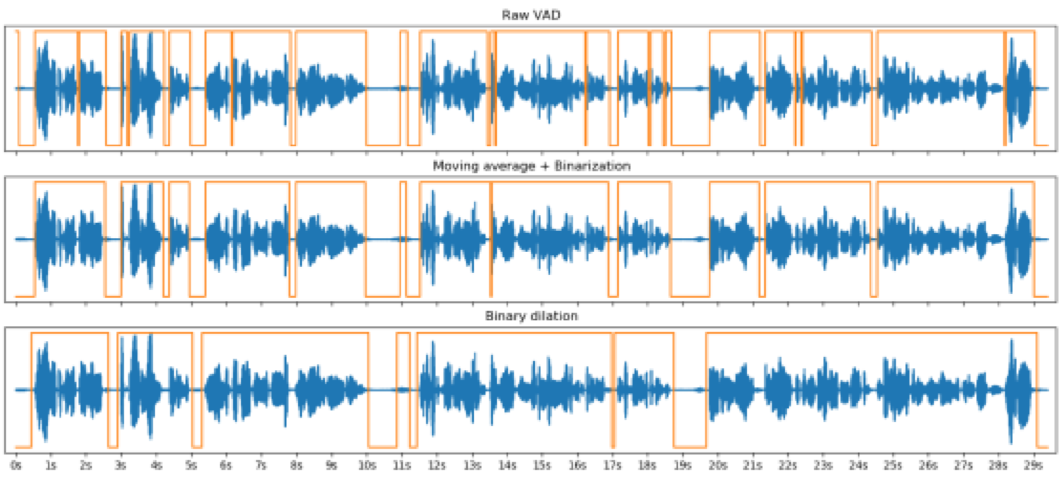
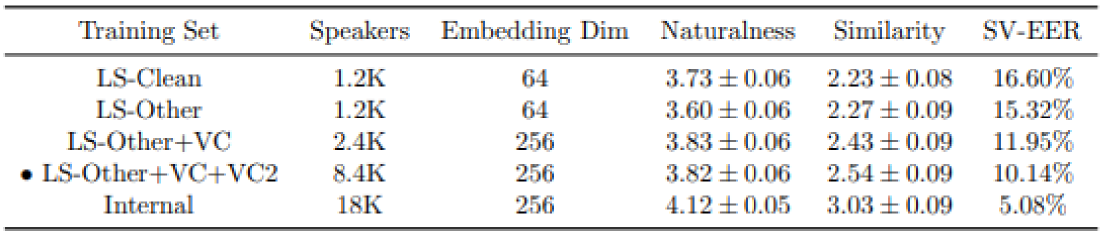


Figure 8: The steps to silence removal with VAD, from top to bottom. The orange line is the binary voice flag where the upper value depicts that the segment is voiced, and unvoiced when lower. This figure is extracted from [20].

The authors of SV2TTS used several noisy datasets to make for a large corpus of speech of remarkable quality. These datasets are LibriSpeech [35], VoxCeleb1 [34], VoxCeleb2 [36] and an internal dataset, which is not publicly available. LibriSpeech is a corpus of audiobooks consisting of 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 appear as noisy [37]. VoxCeleb1 and VoxCeleb2 are made up from audio segments extracted from youtube videos of celebrities (often in the context of a podcast or interview). VoxCeleb1 has 1.2k speakers, while VoxCeleb2 has approximately 6k. Both these datasets have nonEnglish speakers. Regarding VoxCeleb1, there is some filtering that can be done concerning the nationality of the speaker, in order to filter non-English ones out of the training set, however those same heuristics cannot be applied to VoxCeleb2 as the nationality is not referenced in that set. It is vital to understand that it is not clear without experimentation as to whether having non-English speakers hurts the training of the encoder (as the authors have no reference of it either). All these datasets are sampled at 16kHz.

The authors test different combinations of these datasets and evaluate 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 measure 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. These results can be found in Table 1.



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 1: Training of the speaker encoder on different datasets, from [1]. 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 that is adopted in the training of our encoder.

These results are an indicator that the number of speakers is highly correlated with the efficient 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 little jump in naturalness, similarity and EER gained by including VoxCeleb2 could possibly indicate that the variation of languages is not favorable to the training. Regarding the internal dataset of the authors, it is a proprietary voice search corpus from 1k English speakers. The encoder trained on this dataset performs significantly better, however this is a dataset we don’t have access to, so the datasets which are used for the training are with LibriSpeech-Other, VoxCeleb1 and VoxCeleb2.

The speaker encoder is trained for one million steps. A batch of 10 speakers with 10 utterances each is periodically sampled, following the computation of the utterance embeddings and projecting them in a two-dimensional space with UMAP [38]. 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. The UMAP projections are shown in Figure 9, where this behavior can be observed.

As mentioned earlier, the authors of SV2TTS have trained their model for 50 million steps, however due to resources and time, this is not possible for us. The resulting model of [19] yields very strong results nonetheless with a test set EER at 4.5%, while the authors for the same set have shown a value of 10.14% with 50 times more steps.

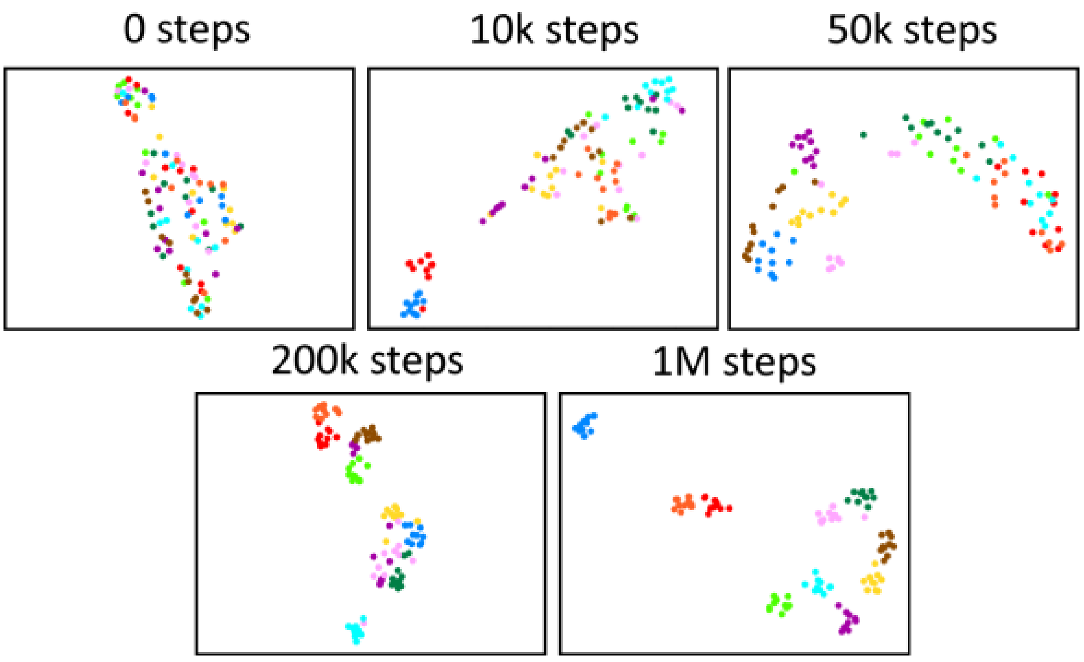


Figure 9: UMAP projections of utterance embeddings extrcacted from [20] from randomly selected batches from the train set at different iterations of the model. Utterances from the same speaker are represented by a dot of the same color. The clustering is entirely done by the model as there are no registered labels to UMAP.

The model seems to produce an efficient clustering and the UMAP projections are perfectly separating utterances from the test set of each of the three datasets, with large inter-cluster distances and small intra-cluster variance. In Figure 10, the test set used for this evaluation is the combination of the test sets of LibriSpeech, VoxCeleb1 and VoxCeleb2. Speakers annotated with an F are female speakers while those with an M are male speakers. Regarding the Equal Error Rate (EER), it is a measure commonly used in biometric systems to evaluate the accuracy of the system. It is the value of the false positive rate equal to the true negative rate. This is done by varying the similarity threshold above which a user is recognized by the system. Unfortunately, the authors of SV2TTS don’t mention their method of evaluating the EER, which is problematic, as the EER is tricky to compute and highly depends on the number of enrollment utterances chosen. In addition, the authors do not mention either if the utterances they use (both for enrollment and test) are complete or partial utterances.

Due to the lack of information provided by the authors of SV2TTS and as there is not any reliable source available indicating how the EER is computed for such a system, the correctness or fairness of the EER measure is not guaranteed. However, it is an indicator that our speaker encoder is performing very well on the task of speaker verification and is able to generate meaningful embeddings.

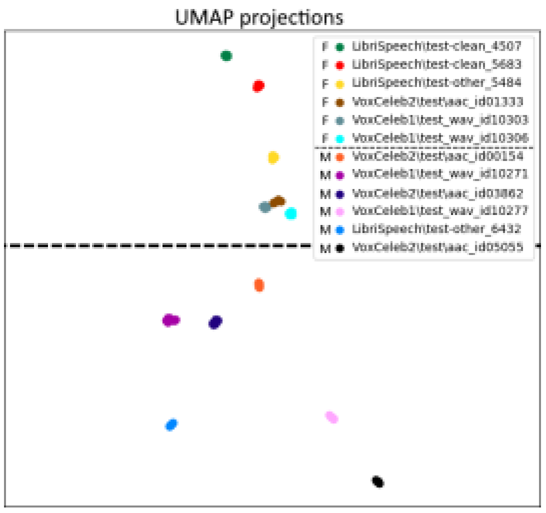


Figure 10: UMAP projections of 120 embeddings, 10 for each of the 12 speakers, extracted from [20]. Six male and six female speakers are selected at random from the test sets.

3.2 Experiments with the synthesizer

In SV2TTS, the authors consider two datasets for training both the synthesizer and the vocoder. These are LibriSpeech-Clean and VCTK 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 sampling of LibriSpeech. They find that a synthesizer trained on LibriSpeech generalizes better than on VCTK when it comes to similarity, but at the cost of speech naturalness. They assess this by training the synthesizer on one set, and testing it on the other. These results are in Table 2. Our initial model is trained on the dataset that would offer the best voice cloning similarity on unseen speakers, which is LibriSpeech.

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 2: Cross-dataset evaluation on naturalness and speaker similarity for unseen speakers. This table is extracted from [1].

Following the preprocessing recommendations of the authors, an Automatic Speech Recognition (ASR) model is used to force-align the LibriSpeech transcripts to text. The Montreal Forced Aligner [31] seems to perform well on this task. With the audio aligned to the text, utterances are split on silences longer than 0.4 seconds. 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. The final 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 11. Note how long silences already account for 64 hours (13.7%) of the dataset.

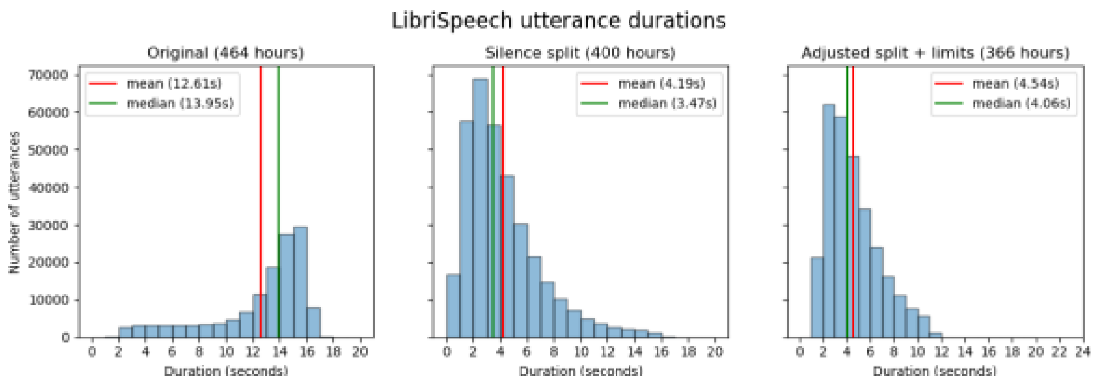


Figure 11: (left) Histogram of the duration of the utterances in LibriSpeech-Clean, (middle) after splitting on silences, (right) after constraining the length and readjusting the splits**.** This figure is extracted from [20]

Isolating the silences with force-aligning the text to the utterances essentially allows to create a profile of the noise for all utterances of the same speaker. A python implementation of the LogMMSE algorithm [39] is used for this task. LogMMSE aims to clean an audio speech segment by profiling the noise in the earliest few frames and updating this noise profile on non-speech frames continuously throughout the utterance. In fact, this additional preprocessing step seems to greatly help reducing the background noise of the synthesized spectrograms.

The authors in SV2TTS use the speaker embeddings to condition the synthesizer at training, while at [20] they suppose that utterance embeddings of the same target utterance make for a more natural choice instead. At inference time, utterance embeddings are also used. While the space of utterance and speaker embeddings is the same, speaker embeddings are not L2-normalized. However, in SV2TTS 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. In addition, 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 [1]. 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 spoken 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 (as seen in Table 3).

	Reference utterance duration				
	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 3: Impact of duration of the reference speech utterance. Evaluated on VCTK. This table is extracted from [1].

Concerning the default model, the synthesizer was trained for 278,000 steps, with a batch size of 144. The number of decoder outputs per step is set to 2, as in Tacotron 1. During training, the model is set in Ground Truth Aligned (GTA) mode, where the input to the pre-net is the previous frame of the ground truth spectrogram instead of the predicted one. With GTA, the pitch and prosody of the generated spectrogram is aligned with the ground truth, allowing for a shared context between the prediction and the ground truth as well as faster convergence. Without GTA, the synthesizer would generate different variations of the same utterance given a fixed text and embedding input (as is the case at inference time). Trying one output per decoder step, the model didn't converge or performed poorly [20]. The loss function is the L2 loss between the predicted and ground truth mel spectrograms. Although we don't have the privilege of having a MOS table as an evaluation metric, one can verify that the alignments generated by the attention module are correct in Figure 12, where the number of decoder steps (223) are matching the number of frames predicted (446) by the number of decoder outputs per step (2). Also, the predicted spectrogram is smoother than the ground truth, indicating an improvement in removing the noise.

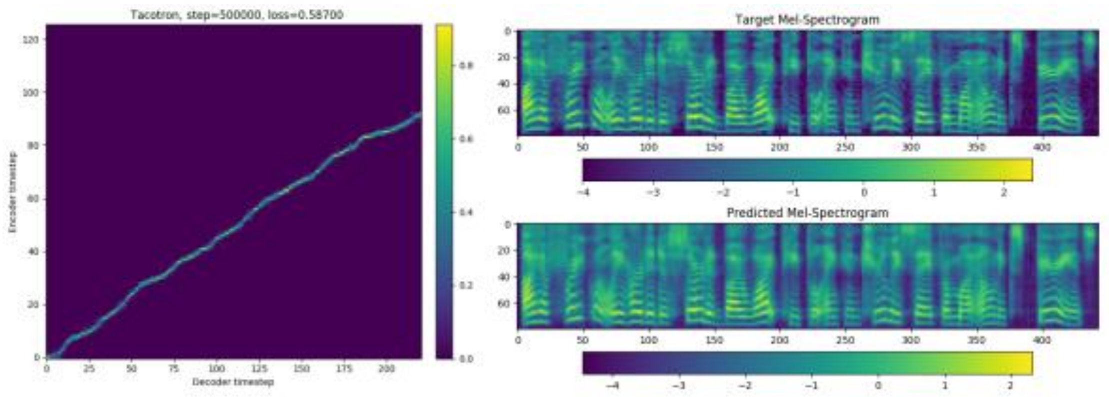
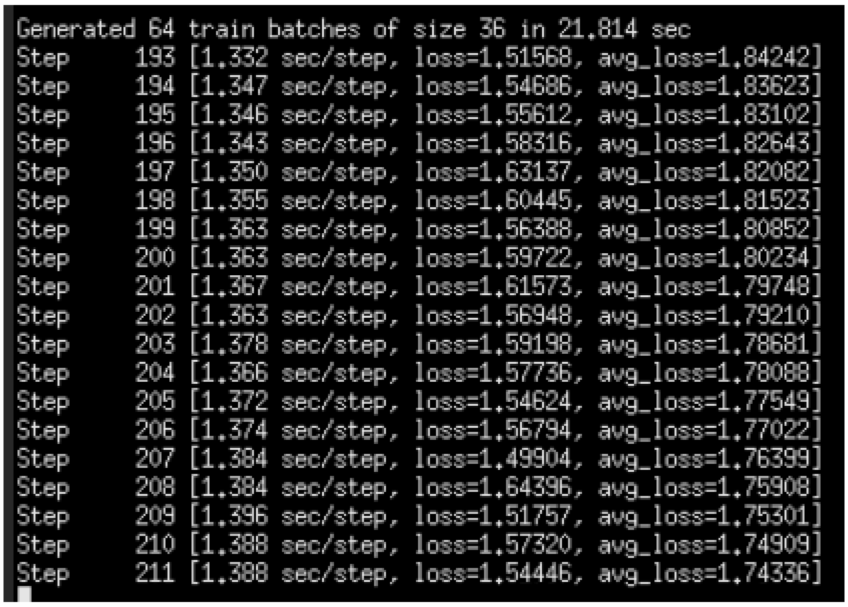


Figure 12: (left) An example, extracted from [20], of alignment between the encoder steps and the decoder steps. (right) Comparison between the GTA predicted spectrogram and the ground truth spectrogram.

Furthermore, using Griffin-Lim algorithm (21) as vocoder, it is evident that the speech generated by the synthesizer matches correctly the text, even in the presence of complex or fictitious words. However, the prosody is sometimes unnatural, with pauses at unexpected locations in the sentence, or the lack of pauses where they are expected, something that is more noticeable in speakers who talk slowly. The lack of punctuation in LibriSpeech is partially responsible for this, forcing the model to infer punctuation from the text alone, an issue that was highlighted by the authors as well. An other reason might be the limits imposed on the duration of utterances in the dataset (1.6s - 11.25s), as sentences that are too short will be stretched out with long pauses, and for those that are too long the voice will be rushed. This is addressed by manually inserting breaks to split the input text, in order to synthesize the spectrogram in multiple parts. This has the benefit of creating a batch of inputs rather than a long input, allowing for fast inference.

We start training a new synthesizer, based on LibriTTS dataset. Only one GPU is used (GeForce GTX 1080 ti). LibriTTS offers several advantages over LibriSpeech: The transcripts contain punctuation so the model will respond to it instead of ignoring it as it does currently. Audio has been split into smaller segments making alignments unnecessary and lastly, it has a higher sampling rate of 24 kHz instead of 16 kHz. We considered updating the hyperparameters so we can ultimately generate 24 kHz audio from it but in the end we decided keeping it at 16,000 and also proceeding with the default hyperparameters, as it speeds training and retains compatibility with the current vocoder.



```
Generated 64 train batches of size 36 in 21.814 sec
Step     193 [1.332 sec/step, loss=1.51568, avg_loss=1.84242]
Step     194 [1.347 sec/step, loss=1.54686, avg_loss=1.83623]
Step     195 [1.346 sec/step, loss=1.55612, avg_loss=1.83102]
Step     196 [1.343 sec/step, loss=1.58316, avg_loss=1.82643]
Step     197 [1.350 sec/step, loss=1.63137, avg_loss=1.82082]
Step     198 [1.355 sec/step, loss=1.60445, avg_loss=1.81523]
Step     199 [1.363 sec/step, loss=1.56388, avg_loss=1.80852]
Step     200 [1.363 sec/step, loss=1.59722, avg_loss=1.80234]
Step     201 [1.367 sec/step, loss=1.61573, avg_loss=1.79748]
Step     202 [1.363 sec/step, loss=1.56948, avg_loss=1.79210]
Step     203 [1.378 sec/step, loss=1.59198, avg_loss=1.78681]
Step     204 [1.366 sec/step, loss=1.57736, avg_loss=1.78088]
Step     205 [1.372 sec/step, loss=1.54624, avg_loss=1.77549]
Step     206 [1.374 sec/step, loss=1.56794, avg_loss=1.77022]
Step     207 [1.384 sec/step, loss=1.49904, avg_loss=1.76399]
Step     208 [1.384 sec/step, loss=1.64396, avg_loss=1.75908]
Step     209 [1.396 sec/step, loss=1.51757, avg_loss=1.75301]
Step     210 [1.388 sec/step, loss=1.57320, avg_loss=1.74909]
Step     211 [1.388 sec/step, loss=1.54446, avg_loss=1.74336]
```

Figure 13: Training a new synthesizer based on LibriTTS.

We train the synthesizer for 200000 steps, noticing that cloned voices sound nearly identical to the initial LibriSpeech model, with better performance for very short text inputs (1-5 words). It is still liable to have gaps, but they are not multiple seconds like in the LibriSpeech model. Some punctuation has an effect (periods and commas), but we didn't notice an effect with question marks. Overall, there is a slight improvement over the initial model and final loss is 0.58670, similar to the LibriSpeech model. Again, we can verify that the alignments generated by the attention module are correct, since the number of decoder steps (105) are matching the number of frames predicted (210) by the number of decoder outputs per step (2). (Figure 14 & Figure 15). Furthermore, the predicted spectrogram is smoother than the ground truth, a typical behaviour of the model predicting the mean in presence of noise.

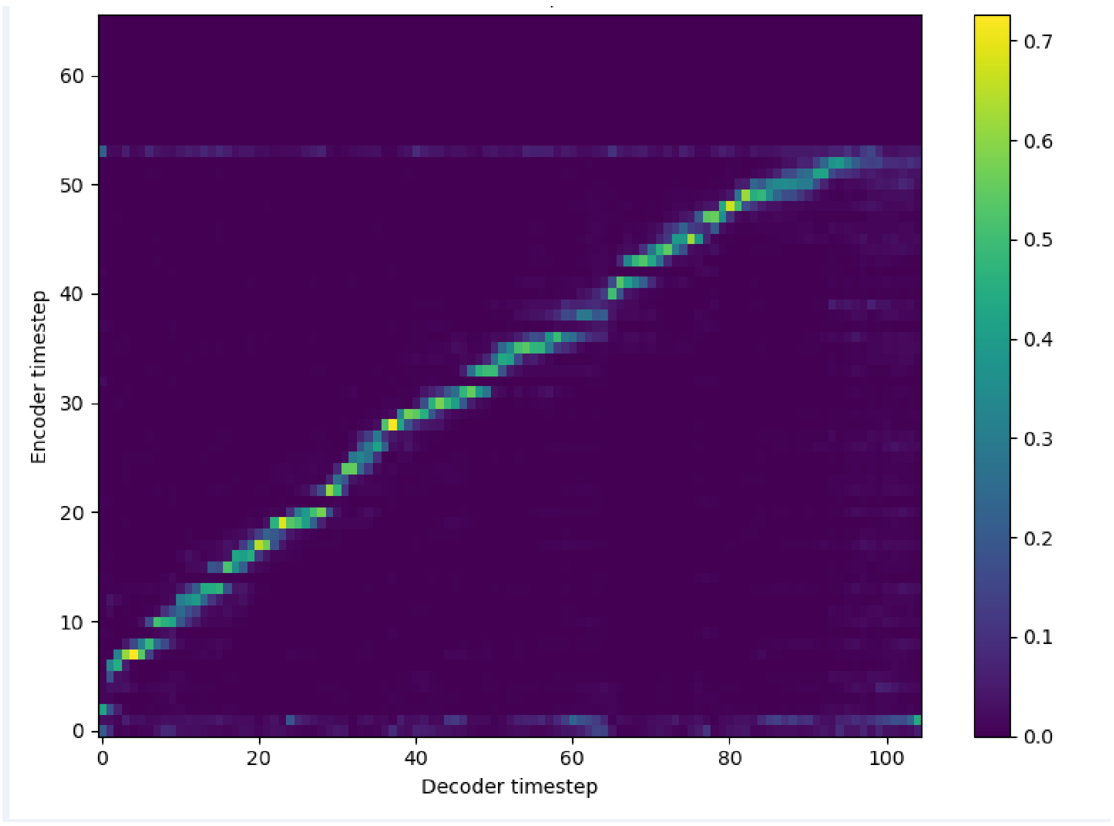


Figure 14: Alignment between the decoder steps and the encoder steps for the synthesizer model based on LibriTTS.

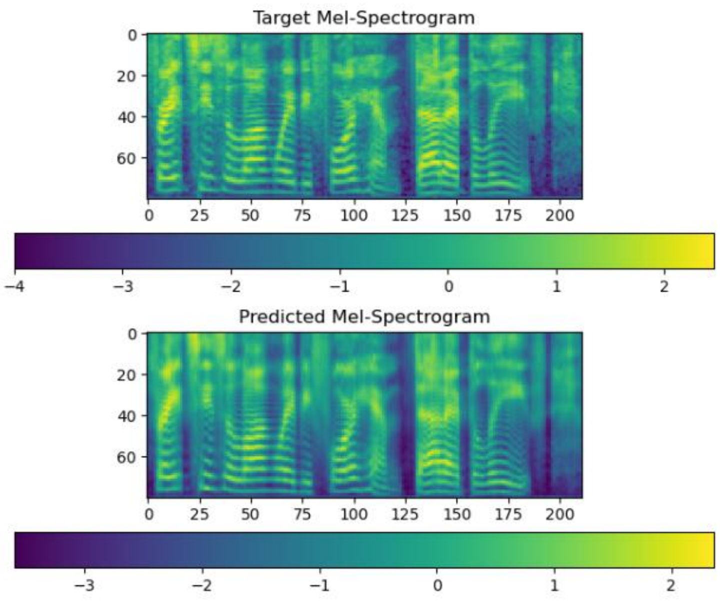


Figure 15: Comparison between the predicted mel-spectrogram and the target mel-spectrogram for the synthesizer model based on LibriTTS.

3.3 Experiments with the vocoder

When dealing with minimal-length utterances, the vocoder often runs below real-time. The inference speed is highly associated with the number of folds in batched sampling. In fact, the network operates nearly in constant time acknowledging the number of folds, with only a small upturn in time as the number of folds increases. It is simpler to talk about a threshold duration of speech above which the model runs in real time. On our setup, this threshold is of 12.5 seconds; meaning that for utterances that are shorter than this threshold, the model will run slower than real-time. Regarding the pruning aspect mentioned by the authors, they claim that a large sparse WaveRNN will perform better and faster than a smaller dense one. In [20], they state that the matrix multiply operation addmm for a sparse matrix and a dense vector only breaks even time-wise with the dense-only addmm for levels of sparsity above 91%. Below this value, using sparse tensors will actually reduce the forward pass speed. The authors report sparsity levels of 96.4% and 97.8% [18] while maintaining decent performances, while at [20] a sparsity level of 96.4% would lower the real-time threshold to 7.86 seconds, and a level of 97.8% to 4.44 seconds. This preliminary analysis indicates that pruning the vocoder would be beneficial to inference speed.

Concerning the initial model, we can say that it is successful in creating a TTS model that could clone most voices, but not some uncommon ones. Although the model was trained for 428000 steps, some artifacts and background noise were present, which is probably caused by the synthesizer. One disadvantage of the SV2TTS framework is the necessity to train models in sequential order. Once a new encoder is trained, the synthesizer must be retrained and so must the vocoder.

We start training a vocoder using the default hyperparameters with the following overrides:

- n_fft=2048
- hop_size=300
- win_size=1200
- sample_rate=24000
- speaker_embedding_size=768
- voc_upsample_factors=(5, 5, 12)

Using the LibriSpeech dataset, we train the vocoder for 301000 steps, observing that it is working without the synthesizer aligning. Loss is still high at 3.6495, however, the quality is improving quite a bit. There are a few examples of generated audio that sounds just as good if not better than the original. Based on generated examples while training, the output is better on male than female speakers. While some of the generated audio sounds excellent, there are quite a few that have artifacts (pops, high-pitched, static, etc). Overall, we can say that the quality was a little bit worse than that of the original model, as there is some trouble discriminating female from male voices efficiently.

Conclusion

The aim of this thesis was to study the framework of [1] and suggest improvements. We trained a new synthesizer model, as well as a vocoder. The results seem to be satisfying and the voice cloning ability of the framework is reasonably good but not on par with methods that make use of more reference speech time. We note some unnatural prosody, as the model is not able to completely isolate the speaker voice from the prosody of the reference audio. Furthermore, as the authors of SV2TTS concluded, most of the ability to clone voice lies in the training of the encoder. Finally, an additional limitation lies in the model's inability to transfer accents, appealing mainly to native English speakers. Given sufficient training data, this could be addressed by conditioning the synthesizer on independent speaker and accent embeddings.

Regarding future work, we hope to improve this framework beyond the scope of this thesis, and possibly implement some of the newer advances in the field. Training a high quality encoder should be a main focus in the future, based on multilingual data. Secondly, a conditioned synthesizer on different accents would be useful. Lastly, multi-GPU experiments must also be tested, in order to enhance training speed and possibly, quality of results.

References

[1] 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.

[2] Simon King. An introduction to statistical parametric speech synthesis. Sadhana, Indian Academy of Sciences, 36(5):837–852, 2011.

[3] Paul Taylor. Text-to-Speech Synthesis. Cambridge University Press, 2009.

[4] Masanori MORISE, Fumiya YOKOMORI, and Kenji OZAWA. World: A vocoder-based high-quality speech synthesis system for real-time applications. IEICE Transactions on Information and Systems, E99.D(7):1877–1884, 2016.

[5] Ranniery Maia, Masami Akamine, and Mark J.F. Gales. Complex cepstrum for statistical parametric speech synthesis. Speech Commun., 55(5):606–618, June 2013.

[6] H. Kawahara, I. Masuda-Katsuse, and A. Cheveigne. Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneousfrequency-based f0 extraction. Speech Commun., 27(3-4):187– 207, 1999.

[7] K Tokuda, T Kobayashi, and S Imai. Speech parameter generation from hmm using dynamic features. ICASSP, 1995.

[8] H Zen, K Tokuda, and T Kitamura. Reformulating the hmm as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. Computer Speech and Language, 21(1):153–173.

[9] Tomoki Toda and Keiichi Tokuda. Speech parameter generation algorithm considering global variance for hmm-based.

[10] Heiga Zen, Keiichi Tokuda, and Alan W. Black. Statistical parametric speech synthesis. Speech Communication, 51(11):1039–1064, 2009.

[11] A-V. I. Rosti and M. J. F. Gales. Generalized linear gaussian models. Technical Report CUED/F-INFENG/TR.420, University of Cambridge, Department of Engineering, 2001.

[12] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In Proceedings of the IEEE. IEEE, 1989.

[13] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. Neural Computation, 11(2), 1999.

[14] Carl Quillen. Kalman filter based speech synthesis. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2010, pages 4618–4621. IEEE, 2010.

[15] Vassilis Tsiaras, Ranniery Maia, Vassilis Diakoloukas, Yannis Stylianou, and Vassilis Digalakis. Linear dynamical models in speech synthesis. Technical report, Technical

University of Crete, School of Electronic Technical University of Crete and Toshiba Research Europe Limited, Cambridge Research Laboratory, Cambridge, UK, 2014.

[16] Vassilis Tsiaras, Ranniery Maia, Vassilis Diakoloukas, Yannis Stylianou, and Vassilis Digalakis. Towards a linear dynamical model based speech synthesizer. In Proceedings of the Interspeech. IEEE, 2015.

[17] Tacotron-2 implementation, tensorflow. https://github.com/Rayhane-mamah/Tacotron-2.

[18] 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.

[19] Real-Time Voice Cloning, https://github.com/CorentinJ/Real-Time-Voice-Cloning.

[20] Jemine Corentin, Automatic Multispeaker Voice Cloning, University of Liège, 2019.

[21] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. IEEE Transactions on Acoustics, Speech, and Signal Processing, 32(2):236– 243, April 1984. ISSN 0096-3518. doi: 10.1109/TASSP.1984.1164317.

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

[23] 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.

[24] Aaron 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.

[25] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer. End-to-end textdependent speaker verification. CoRR, abs/1509.08062, 2015. URL http://arxiv. org/abs/1509.08062.

[26] 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.

[27] 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.

[28] Tom Le Paine, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A. Hasegawa-Johnson, and Thomas S. Huang. Fast wavenet generation algorithm, 2016.

[29] Kyunghyun Cho, Bart van Merrienboer, C¸ aglar G¨ul¸cehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoderdecoder for statistical machine translation. CoRR, abs/1406.1078, 2014. URL http: //arxiv.org/abs/1406.1078.

[30] Sisamaki Irene, End-to-End Neural based Greek Text-to-Speech Synthesis, University of Crete, 2019.

[31] http://1https//montreal-forced-aligner.readthedocs.io/en/latest/.

[32] Sharan Narang, Gregory F. Diamos, Shubho Sengupta, and Erich Elsen. Exploring sparsity in recurrent neural networks. CoRR, abs/1704.05119, 2017. URL http: //arxiv.org/abs/1704.05119.

[33] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression, 2017.

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

[35] 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.

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

[37] 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.

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

[39] Y. Ephraim and D. Malah. Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. IEEE Transactions on Acoustics, Speech, and Signal Processing, 33(2):443–445, April 1985. ISSN 0096-3518. doi: 10.1109/TASSP. 1985.1164550.

[40] Arik, Sercan & Diamos, Gregory & Gibiansky, Andrew & Miller, John & Peng, Kainan & Ping, Wei & Raiman, Jonathan & Zhou, Yanqi. (2017). Deep Voice 2: Multi-Speaker Neural Text-to-Speech.

[41] Arik, Sercan & Diamos, Gregory & Gibiansky et al, Deep Voice 1, Real Time Neural Text to Speech, URL https://arxiv.org/abs/1702.07825.

[42] Wei Ping, Kainam Peng, and Jitong Chen. Clarinet: Parallel wave generation in end-to-end text-to-speech. 30 July 2018.

[43] Sotelo Jose, Char2Wav: End-to-End Speech Synthesis, http://josesotelo.com/speechsynthesis.

[44] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In Advances in Neural Information Processing Systems, volume 2015-January, pages 577– 585. Neural information processing systems foundation, 2015.

[45] WaveRNN, https://github.com/fatchord/WaveRNN