Transfer Learning in Speech Synthesis

Exploring Pretrained Weights Adaptation and Usage of Speaker Embeddings in Neural End-to-End Speech Synthesis

Georgios Roussos

Master's Thesis

UNIVERSITY OF EASTERN FINLAND

Philosophical Faculty

May 2020

ITÄ-SUOMEN YLIOPISTO – UNIVERSITY OF EASTERN FINLAND

Tiedekunta – Faculty Philosophical Faculty	**Osasto – School** School of Humanities
Tekijät – Author Georgios Roussos	
Työn nimi – Title Transfer Learning in Speech Synthesis: Exploring Pretrained Weights Adaptation and Usage of Speaker Embeddings in Neural End-to-End Speech Synthesis	

Pääaine – Main subject	**Työn laji – Level**		**Päivämäärä – Date**	**Sivumäärä – Number of pages**
MDP in Linguistic Sciences	Pro gradu -tutkielma	x	08.06.2020	43 pages + Appendix of 2 pages
	Sivuainetutkielma			
	Kandidaatin tutkielma			
	Aineopintojen tutkielma			

Tiivistelmä – Abstract

This paper is an attempt to work towards a unified neural network approach for the issue of speaker adaptation. In recent years, machine learning advances have afforded the field of Speech Synthesis a great leap towards solutions that are human-sounding and more flexible; however, more research remains to be done on speaker adaptation. The goal of this thesis project is to examine whether few-shot adaptation techniques are able to produce speech that resembles a target speaker, by either fine-tuning a previously trained neural network on a different speaker's voice, or using the target speaker's voice embeddings on a multi-speaker Text to Speech pipeline. Experiments with different attention mechanisms, as also different sound attributes are performed and show that, using transfer learning, one is able to achieve production of intelligible speech, when finetuning a pretrained model, on only 30 minutes of spoken speech from a speaker of a different gender; the same amount of data yields expectedly disappointing results, when the model is trained from scratch, while testing a TTS engine conditioned on neural speaker embeddings shows that speaker modeling only using vector representation is possible. The paper ends on conclusions, with thoughts and suggestions for further research.

Avainsanat – Keywords
speech synthesis, text to speech, neural networks, transfer learning, speaker adaptation, embeddings

ITÄ-SUOMEN YLIOPISTO – UNIVERSITY OF EASTERN FINLAND

Tiedekunta – Faculty Filosofinen tiedekunta	**Osasto – School** Humanistinen osasto
Tekijät – Author Georgios Roussos	
Työn nimi – Title Transfer Learning in Speech Synthesis: Exploring Pretrained Weights Adaptation and Usage of Speaker Embeddings in Neural End-to-End Speech Synthesis	

Pääaine – Main subject	**Työn laji – Level**		**Päivämäärä – Date**	**Sivumäärä – Number of pages**
MDP in Linguistic Sciences	Pro gradu -tutkielma	x	08.06.2020	43 sivua + 2 liitettä (2 s.)
	Sivuainetutkielma			
	Kandidaatin tutkielma			
	Aineopintojen tutkielma			

Tiivistelmä – Abstract

Tämä tutkielma pyrkii kohti keinotekoisiin hermoverkkoihin perustuvaa lähestymistapaa puhuja-adaptaatioon. Viime vuosina koneenoppimisen edistys on suuresti auttanut myös puhesynteesin kehitystä joustavissa ja entistä luonnollisemmalta kuulostavissa ratkaisuissa; lisää tutkimusta kuitenkin tarvitaan. Tutkielman tavoitteena on selvittää, pystyvätkö ns. few shot -adaptointimenetelmät tuottamaan kohdepuhujan kaltaista puhetta joko hienosäätämällä toisen puhujan tuotosten perusteella opetettua hermoverkkoa tai käyttämällä kohdepuhujan monipuhuja-TTS-järjestelmään upotettua ääntä. Erilaisten attentiomekanismien ja ääniominaisuuksien kokeilut osoittavat, että siirto-oppimisella saavutetaan ymmärrettävää puhetta, jos 30 minuutilla toisen sukupuolen puhujan puhetta esiopetettua mallia hienosäädetään; samalla datamäärällä saadaan odotetun huonoja tuloksia, jos mallin opetus aloitetaan tyhjästä. Säätelemällä tekstistä puheeksi -moottoria hermoverkkoon upotetuilla puhujaäänillä voidaan osoittaa, että pelkän vektorirepresentaation käyttö on mahdollista. Tutkielman päätelmissä pohditaan tarpeellisia jatkotutkimuksia.

Avainsanat – Keywords
puhesynteesi, TTS, hermoverkot, siirto-oppiminen, puhuja-adaptaatio, upotus

Foreword

Transitioning into Natural Language Processing has been one of the biggest challenges I have ever taken up on. Completing this thesis and acquiring a degree in the field of Language Technology comes with a relief and a reasonable sense of pride. I, of course, did not do this alone and there are a lot of people I must extend my gratitude to, starting with my supervisor, Stefan Werner: words cannot express how lucky I feel that you granted me the opportunity to write my thesis in the university of Eastern Finland, on an objective that is of intriguing interest to me. Thank you for being patient with me, while I tried to balance work and writing and for letting me work independently, along with all of your input and humor, which has been nothing short of valuable. A model supervisor and without your feedback, this thesis would not have been pretty. To my absolutely loving colleagues at Storytel: you are all the best and you inspire me. Thank you for listening to the numerous TTS voices I try to produce, especially the ones that sound like HAL97 and cannot say the Swedish version of “hello”. A big thank you to the open source community, which is always there to answer any questions I have and issues I may be facing when I try to make computers speak. To my mother and to my father; thank you both for providing me a big part of the financial backend needed during my education years and for saying “alright“ when I announced I decided I wanted to branch out of my original field of studies. To my sister and each and everyone of my friends: thank you for being there, yesterday and today. You believe in me when I do not and your friendship is one of the greatest gifts this life has given me. To my friends, Stina and Jenny; thank you for always being family to me, supporting me and encouraging me. To my friend Jakob: you have been an incredible mentor to me and a great friend; this thesis is for you, as much as it is for me. To my partner, Neil: thank you, Neil, for always trusting me and my abilities, as well as putting up with all of my strange ways and my self-shaming. I love you.

Finally, to myself, a pat in the back: Good job! Take the day off.

List of Abbreviations

CNN	Convolutional Neural Networks
CSV	Comma Separated Values
E2E	End To End
GAN	Generative Adversarial Network
GL	Griffin - Lim
NLP	Natural Language Processing
RNN	Recurrent Neural Networks
SEQ2SEQ	Sequence To Sequence
SSRN	Spectogram Super Resolution Network
STFT	Short Time Fourier Transform
TTS	Text To Speech
VAD	Voice Activity Detection
UEF	University of Eastern Finland
WAV	Waveform Audio File Format

Contents

List of Figures

List of Tables

1 Introduction

Speaker Adaptation is the process of adapting an existing speech synthesis neural network, which has been trained on a source speaker different than the target. This is usually executed as means to producing speech that resembles the voice of a speaker, for which access to available data is limited. By leveraging pre-trained weights (already trained neural networks), the task of training a new machine learning model from scratch may merely be elevated to only adjusting parameters of the pre-existing neural network. In the present day, with voice assistants being very readily available and services like content streaming in the form of audiobooks or podcasts emerging much faster than before, there exists a need for personalized solutions, an objective which is further magnified by accessibility purposes, such as helping people that have lost their voice, or individuals that have translation needs. Thus, speaker adaptation is proving to be a pertinent issue in the field of Natural Language Processing (NLP) and is one that has recently been met with more in-depth studies (Luong and Yamagishi, 2019; Tits et al., 2019; Jia et al., 2018).

Up until recently, the majority of research on speaker adaptation had been done on transfer learning, within the domain of finetuning pre-existing speech synthesis neural networks; however, the usual setting of the latter dictates a learning environment, where at least a small amount of data is available, which at times may prove impractical and a situation not possible to meet (Arik et al., 2018); to this end, speaker adaptation using neural speaker embeddings is, instead, proposed, which involves mapping characteristics of the voice of the speaker to vectors of real numbers (Cooper et al., 2019). With the aid of a speaker verification task, an objective which usually requires data of sub-par quality compared to text-to-speech initiatives, a vector representation of a speaker's vocal identity may be extracted, with the representation being robust and independent of target speaker data. In recent years, speech recognition is another domain which has seen great advances and is, thus, able to aid speech synthesis with neural networks that can provide with good quality voice embeddings (Chiu et al., 2018). These speaker embeddings can then be used to condition multispeaker TTS systems, without finetuning the entire model (Cooper et al., 2019) (Figure 1). This proves to be more challenging, when an unseen speaker that the multispeaker TTS has never been trained on, is involved, since the model is called to generalize and both similarity and quality greatly depend on the architecture of the model, since it dictates the number of trainable parameters and how easy it is to train the model.

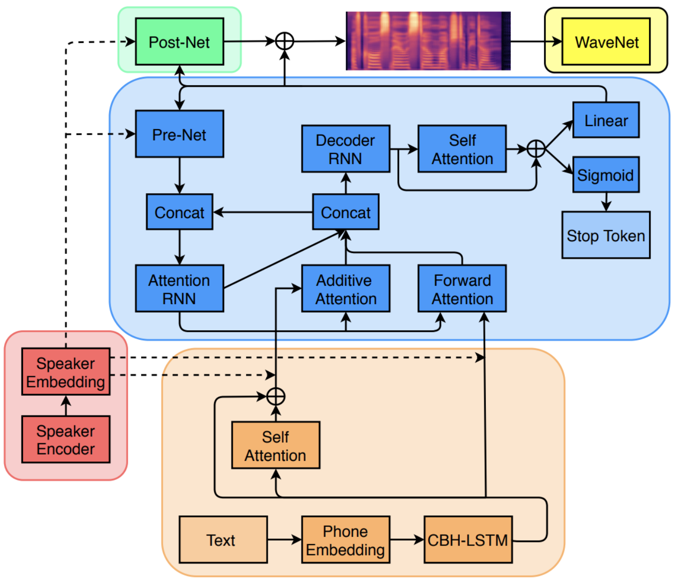

Figure 1: A proposed multi-speaker TTS system, which is conditioned on speaker embeddings (Cooper et al., 2019). The embeddings are concatenated to the speaker encoder outputs during inference time, before synthesizing the Mel spectrograms.

In this thesis, there is an attempt to study the process of speaker adaptation, both from a few-shot (finetuning a pre-trained model) and a zero-shot (neural speaker embeddings) perspective, in order to determine whether the latter can be considered as a good alternative when data for the target speaker is not available. For the objective of this paper, open source text to speech code is leveraged and built on, which is written by Mozilla and can be found on GitHub [1]. The pre-trained model used for the finetuning part of the experiment is trained on an open source dataset; the dataset used for the finetuning part of the experiment is publicly available and consists of speech, which is extracted off an open source dataset. Of specific focus is training neural networks that are industry oriented, that is, the data that is used is clean and of high quality. This serves an emerging need in the commercial field, which calls for adaptable TTS systems, which are fit for many applications.

The rest of the paper is structured as follows; the second section touches on related work, on the domains of speech synthesis by training a neural network from scratch, finetuning an existing model and relying solely on neural speaker embeddings. Chapter three introduces data collection and preparation (preprocessing). Following, in the fourth chapter, the architecture of a neural TTS system is further analysed. Chapter

[1]https://github.com/mozilla/TTS

five presents with the experimental setup followed to satisfy the needs of the paper, as well as training of the models, while chapter six includes the results acquired. A discussion of these may be in found in chapter seven, along with reasoning for specific experiment choices. Chapter eight concludes the paper on future research suggestions.

2 Related Work

2.1 Few shot: Finetuning existing models

The aim of speaker adaptation is defined as the objective of adapting voice models to arbitrary new speakers using a small amount of data (Cooper et al., 2019). This usually calls for a setting where a pre-trained speech synthesis model is available, or alternatively, sufficient amounts of data of the voice of a different speaker than the one the modeling is attempted on, are present. Where speaker recognition is involved, speaker adaptation enables the model to perform better on data of unseen speakers, while when speech synthesis is the aim, speaker adaptation enables a model to produce speech in the voice of new speakers. Tits et al. (2019), explore transfer learning with finetuning an existing TTS, when the available data for the target speaker is of low amount. They are particularly interested in discovering whether finetuning a pre-trained model is also a viable option when it comes to producing emotional speech, since most neural end to end (E2E from now on) TTS networks are able to learn both character and style representations (Pan and Yang, 2010). They choose DCTTS for their experiments, which is a system that models a sequence-to-sequence problem using an encoder-decoder structure, combined with an Attention Mechanism, but is based on a Convolutional Neural Network (CNN) architecture and not a RNN one. In this pipeline, a Text2Mel module is responsible for mapping character embeddings to the output of Mel Filter Banks that are applied to a mel spectrogram, which is used alongside a spectrogram Super Resolution Network (SSRN) and consists of a Text Encoder, an Audio Encoder, an Attention Mechanism and an Audio Decoder. These two modules are trained separately and a Griffin Lim vocoder takes care of producing the end spectrograms. An emotional speech dataset is used, which contains utterances from both male and female speakers, recorded in a contained and professional environment. The authors note the importance of good data pre-processing, which includes frequency sampling, silence trimming and removal of non verbal expressions. In the case of fine-tuning a pretrained model, the same sample rate must be used on the smaller, to-be used dataset. The silence trimming enables the model's attention mechanism to perform better. The authors want to see how the Spectrogram Super Resolution Network (SSRN from now on) may perform when generalizing the mapping to other speaking styles, since there is the possibility of the model overfitting to the speaking style of the initial speaker; they choose to train the entire Text2Mel module, in order to overcome rhythm problems in synthesized speech. They discover that speech synthesis produces unintelligible speech when the training parameters are randomly initialised, given that the amount of data is small; however, using a pre-trained TTS model and finetuning on the layers of the former, they are able to get the model to produce intelligible speech on the new voice. Additionally, they are able to get speech that is correctly perceived as emotional, after finetuning the pretrained model on the new dataset. As future research, they suggest training multispeaker TTS models, in order to acquire a better vector representation of speaker characteristics.

However, transfer learning does not only include the usage of pretrained speech synthesis models. Fang et al. (2019), attempt to tackle the issue of a TTS model needing huge amounts of high-quality data; to this end, they follow an approach which depends on the use of Google's Bidirectional Encoder Representations from Transformers (BERT) (Vig, 2019). They hypothesize that, by providing a TTS system with textual knowledge which is extracted by pre-trained neural networks using BERT (which has shown to perform extremely well, seen in Tenney et al. (2019), the TTS system's reliance on high-quality data may be reduced. The authors look into assisting a neural speech synthesis model training based on the Tacotron2 architecture (Shen et al., 2017) with BERT representations, by concatenating the representations extracted from both branches in each step. They notice better inference stopping and faster convergence during training. As future research, they mention the usage of a pre-trained auto-regressive predictive coding model.

2.2 Zero shot: Using neural speaker embeddings

Zero shot speaker adaptation has only recently begun to be researched into more extensively. In their recent work, Cooper et al. (2019), explore the usage of state-of-the art neural speaker embeddings, in a zero-shot speaker adaptation objective, similar to Louppe (2019), by leveraging multi-speaker speech synthesis, a speaker verification task and neural speaker embeddings. The hypothesis posits that, by conducting transfer learning by training the speaker embedding network separately, robust speaker representations can be obtained, which may then be used with E2E speech synthesis models, in order to adapt to a speaker's voice in zero-shot manner, thus avoiding fine-tuning of the entire model. Cooper's research is the first to investigate many different types of speaker embeddings, in order to find the best fitting ones, when it comes to modeling voices of unseen speakers. The authors make use of a modified version of Tacotron, which is modified to accept external speaker embeddings. Phoneme input is used, together with normalization and a self-attention block. A speaker embedding is extracted for each training utterance and is averaged per speaker. This is then projected to 64 dimensions using a dense layer, before being input anywhere in the model. The same process is used for unseen speakers. The VCTK is the dataset used, while the authors use different kinds of embeddings, in order to determine the best one. It is shown that Learnable Dictionary Encoding embeddings prove to work the best (specifically LDE-3), similarity wise; for seen speakers, scores are very close to vocoded speech. For unseen speakers, the scores are lower, as expected. For future research, the authors suggest exploring ways to reduce overfitting, by experimenting on speaker augmentation, and also working on dialect and speaking styles.

Louppe (2019), experiment on speaker adaptation, by proposing a three-stage pipeline, which involves a speaker encoder for extracting the speaker embedding of the speaker, in order to capture the voice characteristics, a synthesizer which produces the spectrograms and a vocoder, which turns the spectrograms to waveforms. A modified Tacotron architecture accepts, once again, external speaker embeddings, while the vocoder is based on WaveNet (Shen et al., 2017). The author notes that, due to time constraints, they cannot comment on new experiments, however, they mention good results in general, and present with some arbitrary results that show good adaptation metrics and some artifacts during the vocoding stage.

Luong (2019), presents with a unified speaker adaptation method, using transcribed and untranscribed speech, along with backpropagation. They posit that, by having a single fixed-length vector as output when representing speaker characteristics, a neural multi-speaker speech synthesis model can be conditioned, in order to produce speech using the characteristics it is fed. While this technique may also apply to unseen speakers when the transcribed text is not available, performance is sub-par, since the speaker component is a single bias vector. Luong proposes a novel speech synthesis model, which splits the acoustic model into a speaker-independent linguistic encoder and a speaker-adaptive acoustic decoder. The encoder is, then, trained, in order for it to be used whenever linguistic features are not available to model. The author conducts a study on whether scaling and bias may improve model performance and finds that, while they do, they also do not help if more data is available. To combat this, additional adaptable parameters are introduced, which allow for each layer to have its own scaling and bias. Further, the author proposes a new approach which uses different speaker components when modeling speaker characteristics, in place of assuming a single fixed-length vector. They discover that, in both supervised and unsupervised speaker adaptation, results are better when the entire acoustic decoder is finetuned, however, the performance of the adapted model varies greatly from speaker to speaker. As future research, they propose working on their approach, in order to improve it, by using General Adversarial Networks that, given a training set, can learn to generate new data with the same statistics as the training set (Hong et al., 2019).

3 Neural Speech Synthesis

3.1 Tacotron

With the progress in deep recurrent neural networks, speech synthesis has, since 2017, been seeing a tremendous development as well (Wang et al., 2017). With concatenative speech generation having been the go-to technique for developing TTS solutions for years and neural approaches fast replacing the former with the emergence of recurrent deep neural networks (Li et al., 2019), speech synthesis is beginning to be much more interesting to work on, which has in turn seen an increase in research on it. This comes after neural networks like Tacotron (Wang et al., 2017), Tacotron2 (Shen et al., 2017), WaveNet (van den Oord et al., 2016a) and WaveRNN (Kalchbrenner et al., 2018), have proven to be able to produce high-quality speech, which resembles that of a human and does away with concatenative speech synthesis hindrances, which are, most usually, lack of adaptation and the requirement that a large database be present for the task of unit selection (Hunt and Black, 1996). Tacotron consists of a sequence-to-sequence (seq2seq) model with attention, an encoder, an attention-based decoder, and a post-processing net; the architecture takes characters as input and, produces spectrogram frames, which are to be converted into waveforms (Figure

2). This can be achieved by either using algorithms like GL (Griffin-Lim) (Wang et al., 2017; Masuyama et al., 2019), or Generative Adversarial Networks (GAN) (Neekhara et al., 2019). GL and GAN provide a solution to the problem of vocoding, which for TTS systems translates to GL spectrogram representations discarding useful signal and phase information derived from the short-time Fourier Transform (STFT), and compressing the linearly-scaled frequency axis of the STFT magnitude spectrogram into a logarithmically-scaled one (Neekhara et al., 2019).

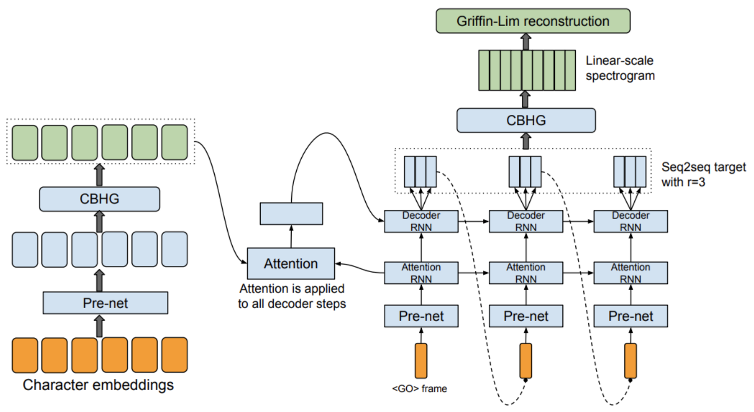

Figure 2: The Tacotron system architecture (Wang et al., 2017). It consists of a seq2seq model with attention, an encoder, an attention-based decoder, and a post-processing net.

Wang et al. (2017) mention that the encoder aims to extract robust sequential text representations, while the goal of the decoder is to produce the corresponding sequence at each step. The decoder target is predicted using a simple fully-connected output layer. Here, convergence speed is increased by predicting r frames. The post-processing network is, then, initialised, in order to convert the seq2seq target to waveform.

3.2 Tacotron2

Shen et al. (2017) build on Tacotron and propose Tacotron2, a unified architecture which consists of a recurrent seq2seq prediction network predicting the mel spectrogram frames for any input character sequence, and a modified version of WaveNet (van den Oord et al., 2016a) as the vocoder, which aims to produce waveforms, which are difficult to distinguish from human speech. The difference with Tacotron is that, the Tacotron2 model uses simpler building blocks, which consist of vanilla LSTM layers and convolutional layers in the encoder and decoder, instead of “CBHG” stacks and GRU recurrent layers (Wang et al., 2017) (Figure 3). As in Tacotron, mel spectrograms are computed through a STFT, using a 50 ms frame size, 12.5 ms frame hop and a Hann window function. The STFT magnitude is converted to the mel scale using an 80 channel mel filterbank and a log dynamic range compression. The encoder converts the input sequence into a fixed-length context vector for each decoder output step, while a location-sensitive attention mechanism accounts for all the previous decoder time steps; this provides the model with better consistency when producing mel spectrograms, which are predicted by the decoder part of the network, one frame at a time. A small pre-net layer preprocesses the prediction from the previous time step, and the output of it is concatenated to the attention context vector. The concatenated output is projected down to a scalar and passed through a sigmoid activation to predict the probability that the model should stop inferring (referred to as the stopnet). All the convolutional layers in the network are regularised using dropout, with a probability of 0.5, while on LSTM layers, zoneout (randomly preserving hidden activations of the network) is used with a probability of 0.1.

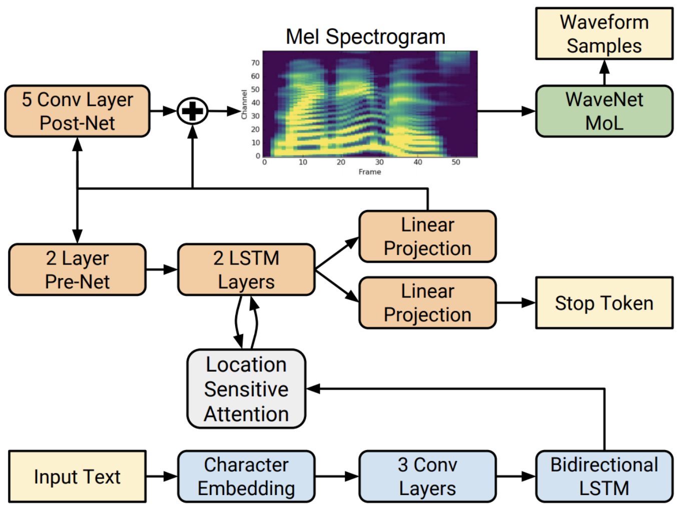

Figure 3: The Tacotron2 system architecture (Shen et al., 2017). The architecture is much simpler than Tacotron, while the model has more trainable parameters, allowing for better quality of speech synthesis.

3.3 WaveNet

With Tacotron and Tacotron2 being used to produce high-quality spectrograms, there arises the problem of generating high-quality waveforms, as well, since the GL algorithm both Tacotron and Tacotron2 may use, leaves a lot to be desired, in terms of sound fidelity. To this end, van den Oord et al. (2016a) propose WaveNet, which is an audio generative model based on the PixelCNN (van den Oord et al., 2016b) architecture and is, thus, able to generate raw speech signals based on the spectrograms synthesized by the Tacotron or Tacotron2 backends. WaveNet's versatility lies in its ability to produce waveforms for different voices, when conditioned on speaker identity, while it can also be used for neural music generation. The joint probability of a waveform **x** = *{x_1, . . . , x_T}* is factorised as a product of conditional probabilities as follows:

$$p(x) = \prod_{t=1}^{T} p(x_{\mathrm{t}}|x_1, ..., x_{\mathrm{t\text{-}1}}) \tag{1}$$

Here, each audio sample $\mathrm{x_t}$ is conditioned on the samples at all previous timesteps (van den Oord et al., 2016a; p. 2), which is also what Shen et al. (2017) leverage, when building their Tacotron2 implementation. Their modified version of WaveNet includes 30 dilated convolution layers, grouped into 3 dilation cycles, however, instead of predicting bucket with a softmax in a discrete nature, they use the PixelCNN (van den Oord et al., 2016b) architecture, as also Parallel WaveNet (van den Oord et al., 2017) and use a 10-component mixture of logistic distributions (MoL) to generate 16-bit samples at 24 kHz. The WaveNet stack output is passed through a ReLU activation followed by a linear projection, in order to predict parameters (mean, log scale, mixture weight) for each mixture component; the final loss is the negative log-likelihood of the ground truth sample.

3.4 WaveRNN

To combat inefficient vocoding times that usually occur when sequential vocoders are implemented in speech synthesis, Kalchbrenner et al. (2018) propose WaveRNN, which is a single-layer RNN, equipped with a dual softmax layer, that is able to match WaveNet in quality, while generating 24 kHz 16-bit audio four times faster than the former. Size reduction of the model is achieved by applying weight pruning; the technique yields the discovery that for a constant number of parameters, large sparse networks perform better than small dense networks and this relationship holds for sparsity levels beyond 96% (Kalchbrenner et al., 2018). The efficiency of the WaveRNN vocoder is further showcased with the authors claiming that high-fidelity audio synthesis can be achieved on a smartphone CPU in real-time. Since a sequential model learns the joint probability of data by producing conditional probabilities over each sample and then factorizing them, the sampling process is a serial one, which may also be slow during inference time. The efficiency may be improved, by decomposing the time required to produce each sample into computational time and overhead, for each one of the layers of the network, as:

$$T(u) = |u| \sum_{i=1}^{N} (c(op_{\text{i}} + (d(op_{\text{i}})) \tag{2}$$

Here, the value of **T(u)** can grow exponentially larger, depending on the audio bit-rate and frequency, if the size of the network is big, which in turn implies more layers, if the number of parameters included is large, or if the overhead **d(op$_\mathbf{i}$)** is high, because each operation is run individually (Kalchbrenner et al., 2018). WaveRNN is an attempt to reduce the time these operations require, but at the same time maintain the quality a WaveNet equipped system outputs. A model trained on sequences of 960 samples of 16-bit bit-rate and full back-propagation puts WaveRNN on the same level of quality which WaveNet is (Kalchbrenner et al., 2018).

3.5 Griffin - Lim Algorithm and Vocoder

Fast experiment cycles require algorithms that are efficient and quick to provide results; therefore a neural network vocoder is usually preserved for higher scale testing. To this end, the Griffin - Lim algorithm may be used, in order to provide real-time inference when testing the performance of a speech synthesis model. The algorithm was first proposed by Griffin and Lim (1984) and is an attempt at computing a signal from its modified STFT. It is a version of the double-projection algorithm originally suggested by Gerchberg and Saxton (1972) for solving the phase recovery problem in terms of the Fourier transform. The Gerchberg-Saxton works for a non-redundant system (the Fourier transform) by considering additional side-constraints to make the solution unique; the GLA algorithm on the other hand works for redundant systems without any side constraints, where the uniqueness of the solution comes via the redundancy (Perraudin et al., 2013; p.2). The Griffin - Lim algorithm allows for a convergence towards the estimated phase layer, by acquiring the sound signal, which has an STFT as close as possible to the modified STFT. The algorithm allows for reconstruction of a signal from a spectrogram, which a synthesizer constructs. The algorithm is iterative, so a number of iterations is set, depending on the size of the spectrogram (Figure 4). Inference is slower when the number of iterations is set higher; a number between 30 and 60 often proves sufficient.

Algorithm 1 Griffin-Lim algorithm (GLA)

```
**Fix** the initial phase $\angle c_0$
**Initialize** $c_0 = s \cdot e^{i\angle c_0}$
**Iterate** for $n = 1, 2, ...$
    $c_n = P_{C_1}(P_{C_2}(c_{n-1}))$
**Until convergence**
$x^* = \mathbf{G}^\dagger c_n$
```

Figure 4: Griffin - Lim pseudocode (Perraudin et al., 2013). The algorithm consists of many iterations and at times produces low-quality signals owing to the lack of prior knowledge of the target signal.

3.6 Goals and Hypotheses

On the basis of transfer learning and its usage of pretrained weights, the aim is to train a neural network on a very small amount of data and still be able to produce intelligible speech. To further demonstrate the success of transfer learning as means to finetuning, the output is compared to a model trained from scratch on the same data. While the general goal is within reach, especially seeing as previous work on transfer learning and speech synthesis has been made, exceptional results are not expected, due to the scope of the paper. As far as the model trained from scratch is concerned, it is anticipated that it will perform poorly, while, finally, for the testing of the TTS model that is based on embeddings, what is expected, is a baseline result that will not outperform the finetuned model. In short, the following hypotheses are expressed:

1. Hypothesis 1: Adapted models will perform better when finetuned on small amounts of data, when the language is the same and the alignments have been learnt, thus the weight initialisation will be easier and quicker. The hypothesis follows what Tits et al. (2019) discover, in terms of transfer learning when the training data amount is low.

2. Hypothesis 2: The trained from scratch model is not going to be able to produce intelligible speech. The assumption is based on the complex architecture both Tacotron and Tacotron2 have, so more data would be needed, when no zero-shot adaptation is involved.

3. Hypothesis 3: The network learnt on speaker embeddings will be able to retain the voice of the new speaker, without being able to model the prosody or the accent of the former. As section 2 recalls, usage of neural speaker embeddings implies an environment where learning data is not present.

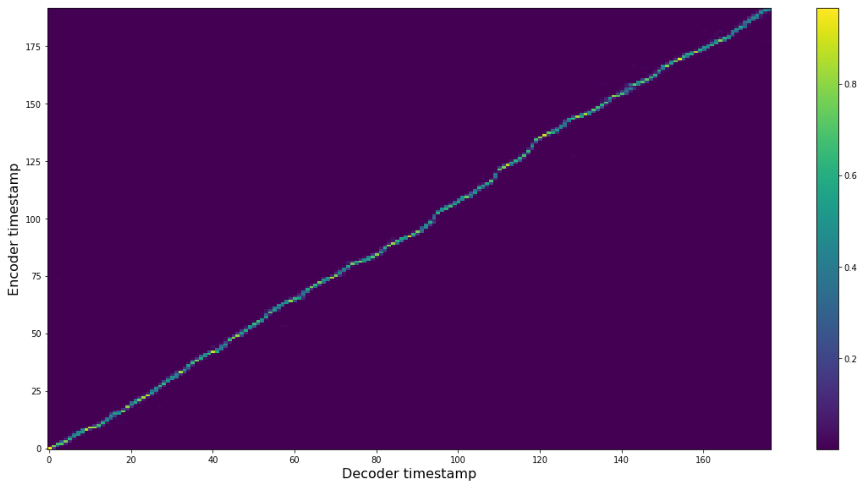

Figure 5: A neural network with a robust alignment. Here, diagonal aligning may be observed, which shows no trouble in inferring the text and stop token.

In the above picture (Figure 5), the model has learnt phoneme alignments, while also preserves robust attention, as evident by the smooth transition of the decoder to the next step and presents with a good stopnet, which is showcased by the fact that there is no yellow colour at the end of the diagonal line, which would signify that the model was having trouble knowing when to stop inferring. A good stopnet is critical, as it signifies the end of inference and, thus, the usage of computational resources. A stopnet that has not converged enough means longer inference times and repeating words at the end of sequences, since the model does not know where to stop.

4 Methodology

4.1 Data Preparation

There currently exist many open source speech datasets fit for speaker verification tasks; however, in the case of training a speech synthesis neural network, finding suitable candidates becomes a more challenging task; as section 1 recalls, when training networks for speaker verification purposes, there is a lesser need for data to be accurately transcribed, if at all, while the quality may also be sub-par. In the case of training a text to speech model, data must be as high of quality as possible, with accurate sound transcriptions and lack of noise. To this end, it is important that a quality check be carried out, especially when transfer learning is involved and the amount of data of target speaker is limited. The steps and tools for data acquisition and processing are outlined in this section.

4.2 Speech Dataset

For the experiments in this paper, an audiobook recording excerpt derived from LibriTTS is used (Zen et al., 2019, in order to extract a target speaker and their speaker identity, in regard to voice characteristics like accent, pitch and prosody. The LibriTTS speech corpus is a corpus which is derived from LibriSpeech, which is a dataset used for research in speech recognition. LibriTTS addresses properties in LibriSpeech that make it a less than ideal candidate for text to speech research, by ensuring all audio matches the transcription perfectly, with neighbouring sentences information being available and high resolution sound; the transcriptions are normalized and utterances with background information are removed. LibriTTS is a good example when attempting to construct multispeaker datasets for speech synthesis objectives, since it combines many attributes that make it attractive for text to speech research.

The source and target language is English. The audiobook excerpt is well recorded, with no audible high noise levels in the sound and is of approximately 30 minutes length. The narrator is consistent in reading out what the book reads, enunciates well and, in combination with the ebook text, a one to one (1:1) mapping between the audio and its transcription is obtained. For the purpose of the finetuning experiment, these 30 minutes of recording are used, in order to make it harder on the model to transfer and comply with the setting in reference, that is, a low amount of data. That accounts to approximately 200 sentences, with a duration that usually varies between from 7 to 10 seconds and a word length of 3-15 words, accounting for each sentence; on character level, this length translates to approximately 80-90 characters, at most. This amount of data is significantly less than what an average TTS model calls for, in order to perform adequately, which accounts to approximately 24 hours, in order for it to avoid word skipping, repeating of words in harder cases and mel-spectrogram speed generation

(Ren et al., 2019). Using processing software, only the first 30 minutes of the audiobook recording are thus selected, along with its corresponding ebook transcription.

The pipeline used for transfer learning automatically splits the data in training, validation and test sets, so no manual splitting is carried out manually. The quality of this specific dataset serves the initial purpose, which, as section 1 notes, is training speech synthesis models, that are industry oriented.

4.3 The VoxCeleb Dataset

For the part of any experiment where neural speaker embeddings are leveraged, a multispeaker TTS model is to be trained; this enables a TTS model to gain a rich representation of various speaker identities and generalize better when presented with an unseen speaker. As a result, a dataset of many utterances from many different speakers is required. To this end, an implementation that is trained on the VoxCeleb dataset (Nagrani et al., 2017) is evaluated. The corpus consists of hundreds of thousands utterance and hours of spoken speech and is derived from the original audio and text materials of various celebrities recordings (Panayotov et al., 2015), which have been used for training and evaluating automatic speech recognition systems. The dataset presents both with male and female voices, is gender balanced and and includes both video and audio. Neural E2E speaker verification and identification systems trained on the VoxCeleb dataset, are shown to achieve a high performance score higher than 4.0, in regard to naturalness. For the purposes of the experiments in the paper, evaluation is based on an already trained TTS neural network that works with neural speaker embeddings, since training a new model presents with long waiting times and is outside the time domain of this project.

4.4 Sentence - Level Segmentation

In order to train a robust speech synthesis model, it is necessary that the audio be segmented in sentences, have the correct transcriptions and the distribution of word length be even across all of the dataset. This renders the task of learning the phoneme alignments easier for the model, while it ensures less errors, since sequences with a lot of words usually use a lot of GPU memory, bottle-necking the training session

(Pascual et al., 2019). While the LibriTTS dataset comes segmented in sentences, that is usually not the case with audiobooks, which mostly present with bigger sound chunks. Manually segmenting audiobooks proves to be time consuming, thus there exist two options:

1. Take advantage of Voice Activity Detection (VAD), in order to split the audio on sentence breaks, using a CNN. The process is described further below.
2. Pass the audio through the Google Speech To Text API (Chiu et al., 2018), in order to obtain word timestamps. Cross-referencing the results with the ebook transcription is then possible and, using an edit distance algorithm, construct a mapping between audiobook and ebook words; Sentence - level segmentation may then be achieved, using the Python programming language.

The first option is to leverage VAD. To this end, inaSpeechSegmenter (Doukhan et al., 2018) may be used, in order to process the audio files. The library provides with numerous options, such as gender, speech, music or silence detection and "has been designed in order to perform large-scale gender equality studies based on men and women speech-time percentage estimation". Using the library, a Comma Separated Values (CSV) file is obtained, which presents with the timestamps of the sound file, split on silence. Using Python, one can only keep the narrator's utterances, doing away with music, silence and chapter information, which is spoken by a speaker of a different gender. After a quality check, mapping the audio with the corresponding text from the ebook is performed. This is rendered possible, since the quality check is only performed for 3 hours of data and the quality of the library used is sufficient, so there are not a lot of mistakes. In addition to this, using the library, a big amount of trailing silences may also be avoided, which make it harder for the model to align. In the case of larger amounts of data, one may proceed with Speaker Recognition methods instead, albeit the timestamps the STT API provides are not entirely accurate (off by a few milliseconds), so the edit distance would have also not worked for all of the sentences and an even smaller number of the former would comprise the dataset. However, this proves to be a viable solution when the VAD method underperforms and wrongly classifies genders, or mid-sentence silences. This generally is affected by the voice narrator, their narrating skill and the duration of the audio. In general, it should be noted that the process of creating datasets out of audiobook recordings cannot be entirely automated and some manual correction and quality checking is always welcome and, to a certain extent, required.

4.5 Preprocessing

The LibriTTS corpus has had many normalization techniques applied to it; for that reason, there is not any need to apply anything further. For the dataset constructed using the audiobook recording excerpt and its ebook counterpart, checks that it complies with the following are performed:

1. Normalize each number and currency to its natural language counterpart.
2. Normalize each abbreviation to its natural language counterpart (Mr. to Mister, Ms. to Miss etc.).
3. Upper case every sentence beginning.
4. Remove instances where code-switching occurs.
5. Phrase break punctuation that is not signified with the semi colon sign, is replaced with one.
6. Remove instances where the narrator imitates book characters, as that makes it harder for the model to learn alignments.

The following sound information is used:

1. 22050Hz sampling rate.
2. Mono channel.
3. 16-bit.
4. Waveform audio format (WAV).

Finally, any sentence longer than ten (10) second is split to smaller segments, as longer sequences prove to be harder for the network to model during evaluation time and, thus, learn alignments for.

For the above, the SoX library (2012) is used, in order to batch process the sentence files. A CSV file is constructed, that includes the number of the sentence, the original transcription of the sentence and its normalized counterpart, in the style of the LJSpeech dataset (Ito, 2017), which is the chosen format used while training the TTS model. These are some of the considerations to be made when making constructing a speech dataset for any speech recognition or speech synthesis task. Most open source TTS pipelines that are based on deep neural network architectures, require that the format of the dataset follow that of aligned sound-text pairs, and that the alignment be included in a CSV or text (TXT) file (Iakushkin et al., 2018). The sound must be monophonic, with a sample rate starting from 16kHz and a bit-rate of 16bit. Even though 16kHz may prove enough for Speaker Recognition, however, Speech Synthesis requires sound of better quality for more optimal results (Arik et al., 2017).

4.6 Tacotron and Tacotron2: A seq2seq approach to neural speech synthesis

For the experiment cycles of this paper, Tacotron and Tacotron2 architectures are leveraged; thus, the problem of neural speech synthesis is treated as a seq2seq issue, using seq2seq learning, which is first seen on Sutskever et al. (2014); seq2seq converts one sequence to another, using RNN. Both Tacotron and Tacotron2 are generative models, take raw character as input, and output spectrograms. Both models may be trained using <text,audio> pairs. A traditional seq2seq system consists of an encoder, which encodes a sequence of symbols into a fixed length vector representation, while the decoder decodes the vector representation into another sequence of symbols (Sutskever et al., 2014). Since Tacotron and Tacotron2 do not require forced phoneme alignments, they both can be trained from scratch, by using clean transcripts and the matching audio. This comes in contrast to concatenative speech synthesis methods, which most usually require that each sentence in the training corpus be accompanied by files indicating start and end boundaries on sentence, word, and phoneme level, as also phonemic transcriptions of all words, including suprasegmentals, so the unit selection can combine the best parts of phonemes (Khan and Chitode, 2016. This lack of alignment requirement renders Neural Speech Synthesis a much more enticing alternative, dictates, however, an absolute need that the corpus to be used be manually checked and corrected before training, so the model is able to learn robust phoneme alignments.

Silences must also be trimmed, as they are impediment to the former.

4.7 Mozilla TTS

The Common Voice Project is an initiative led by Mozilla, in the goals of enabling users to leverage the power of machine learning in the fields of Speech Recognition and Speech Synthesis, by making access to large datasets easier (Ardila et al., 2019). Mozilla TTS is a part of this initiative and is an engine that was first released in 2018. The first implementations were based on Tacotron, while Tacotron2 support followed later. Being an open source project anyone can contribute to, Mozilla TTS has been seeing a lot of traffic and quick adoption, which has rendered it a competitive option when it comes to training neural networks for speech synthesis objectives. Aside from Tacotron and Tacotron2 support, Mozilla TTS also includes a speaker encoder side project, which is useful when it comes to computing speaker embeddings for speakers, as also pretrained models, free to use. It also provides with tools fit for dataset analysis, which may be used to plot utterance distributions and decide on whether certain utterances may be broken or not. The engine supports multispeaker TTS learning, prosody modeling using Global Style Tokens (Wang et al., 2018) and some normalization techniques that may be used in the case of small datasets. The user is free to choose an attention mechanism from Forward, implemented in the style of Zhang et al. (2018), Graves attention, or a bidirectional auto encoder.

Since the engine is lightweight, adaptable and trivial to use, Mozilla TTS is used for the purposes of this paper, when attempting transfer learning and training models from scratch. An implementation of multispeaker TTS from scratch is out of the time frame of this paper, thus Louppe's (2019) implementation is evaluated, in order to conclude whether voice embeddings are able to model an unseen speaker's voice. Finally, the Griffin - Lim vocoder is the choice of vocoder, in order to keep up with the quick experiment cycles.

4.8 Hyperparameters

Once the resources and datasets are organized, the training phase starts. In single speaker training environments, the Mozilla TTS offers pre-processing scripts which make data handling easier. In general, the engine expects the dataset in the format of a CSV file that contains the name of each sentence, the sentence and its normalized counterpart, all separated by the pipe symbol. On the basis that the dataset's sample rate is 22050 Hz, the following set of hyperparameters is proposed:

1. Size of the mel spectrogram frame: 80, in order to retain as many speech features as possible.
2. Number of STFT frequency levels: 1025. This size of the linear spectrogram frame fits in memory of the GPU and is not wide enough to provide with poor time resolution.
3. Sample rate: 22050Hz, which is a popular sample rate.
4. STFT window length (in ms): 1024.
5. STFT window hop-length in ms: 256.
6. A pre-emphasis level of 0.98, in order to reduce spectrogram noise and make the speech more structured.
7. A normalization range of -100 dB.
8. A reference level of 20 dB.
9. A power value of 1.5, which corresponds to wav signals sharpening after the Griffin - Lim algorithm inverts the spectrogram, to make the sound more structured.
10. A number of 60 Griffin - Lim iterations, in order to have both quick an inference time and acceptable quality.
11. A minimum frequency level of 50.0 for the mel spectrogram, since the speaker is male.
12. Trimming of silences, to make it easier on the model to learn phoneme alignments.
13. A trim threshold of 60 dB, which serves trimming background noise and silences.

These attributes serve the purposes of quick extreme experiment cycles, without loss of audio quality, or memory problems. In addition to these, a batch size of 64 is initially set and enable gradual training. Gradual training proves to be helpful when a lot of steps are involved in the training of the neural network, as observed in Wang et al. (2017). The implementation of gradual training in training the TTS network keeps the batch size of 64 until the 5000th step, when it drops it to 32 and then, finally, to 16, on the 290000th step. The same applies to decoder frames prediction, which starts from 7 and gradually decreases to 1, as the model starts to converge and is more confident in its alignments. A learning rate of 0.000001 is implemented, combined with a forward, location sensitive attention mechanism. Attention is a vital component used when training a speech synthesis model; it is important to use an attention mechanism which serves the diagonal nature of the problem, where the output aligns perfectly with the text in a monotonic style. Location sensitive attention enables the network to perform better, by allowing it to look into the previous alignment vectors and learn diagonal attention more easily (Chorowski et al., 2015). A good learning rate is crucial to the performance of the network, as having too small of a rate may result in never reaching a convergence level, and a big learning rate runs the risk of overfitting the model to the training data. To remedy the LSTM's tendency to forget alignments when inferring long sequences, a limit of 150 characters is set, when it comes to maximum sequence length. Instead of raw characters, phonemes are used, as it helps with the pronunciation. This also makes easier to adapt text-to-speech pipelines to most languages, without worrying about language specific characters.

4.9 Training Environment

All models and experiments are trained and performed on a Compute Engine instance on the Google Cloud platform, which is equipped with 30GB of RAM, 8 vCPU cores, one NVIDIA K80 GPU, and runs on Ubuntu 18.04 LTS. Where local experiments and data pre-processing are involved, a MacBook Pro 2019 with an 2,4 GHz Quad-Core Intel Core i5 and 16 GB of RAM is leveraged.

4.10 Evaluation Method

The evaluation of a TTS system can be as trivial as listening to the output samples. To this end, a combination of evaluating the output and Mean Opinion Score (Kim et al., 2013) is deemed sufficient, which offers a scale of 1, which is bad, to 5, which is excellent. In addition, the performance of the trained neural networks is further examined, in terms of decoder loss, postnet loss, alignments during evaluation and testing phases, gradient normalization stepping and epoch times. In general, a diagonal alignment during evaluation and inference times, signifies that the model has learnt robust alignments and knows when to stop inferring. These are the evaluations implemented, as regards the finetuned model and the pipeline that uses neural speaker embeddings, since the model trained from scratch does not produce acceptable results.

5 Results

5.1 Transfer Learning using Pretrained Weights

Starting with transfer learning, a pretrained model, which has been trained on the Tacotron2 architecture, using a Forward Attention mechanism for 400.000 steps and batch normalization from that point, until it reaches 670.000 steps, is used. The model has been trained on a single-speaker dataset, with utterances recorded from a female speaker. Finetuning this model on the small dataset, intelligible speech is obtained in less than 8 hours. The model has adapted to the target speaker and is able to retain the prosody of them, while the alignments remain robust, albeit less so than what observe before finetuning the model. With that said, the network is able to produce short, unseen sentences that do not lack varying prosody and are generally of an acceptable quality, especially considering that the vocoder of choice if Griffin - Lim based. In short, the finetuned model provides with the scores found in Table 1, during evaluation and train time.

The alignment scores refer to how well the model has learnt representations of phonemes and is also a pseudo-metric when it comes to audio quality measurement. Tacotron and Tacotron2 models start forming meaningful alignments approximately after 30000

Score	**Evaluation**	**Train**
Alignment	0.69	0.58
Loss (Decoder)	0.298	0.15
Lost (Postnet)	0.228	0.135
Stop Loss	0.050	0.280

Table 1: Results obtained when transfer learning.

steps of training (Wang et al., 2017). The decoder loss values refer to how well the model can cope with sequences, when it has to compress the information it learns into a fixed length vector, while the postnet loss values regard the part of the network where a residual is applied to the mel spectrogram, for better reconstruction (Shen et al., 2018). Finally, the stop loss values explain how well the model knows when to stop performing inference, when it reaches the end of the sequence.

An alignment score of 0.69 during evaluation time, signifies that the model performs well when it comes to hyperparameter tuning and that it obviously benefits from using all the compatible layers from the pretrained weights it is provided with; since all weights are trained on the same language, all the layers of the pretrained weights are used. As seen in the scores in Table 1, the finetuned model goes as high in alignments as a score of approximately 0.70. However, as seen on Figure 6, there are parts of the sequence it does not decode well (seen on the picture as blurry spots), so the performance is diminished, in the form of pronunciation, or sound glitches. This is observed during test time as well and, more specifically, in the alignments of the test sentences (see Figure 9, Figure 10, Figure 11 and Figure 12). In general, while the network seemingly performs adequately, the alignments present with artifacts and parts where the model is confused.i

All loss scores, both during evaluation and train times show that the model may benefit from longer training times, or more data. Loss scores seen in Table 1, leave much to be desired. In general, the culprit to these values seems to be the fact that the dataset is very limited. More specifically, as Figure 7 shows, the decoder loss value slowly keeps increasing with each epoch, when the exact opposite should be happening. The model seems to not be able to generalize to unseen data.

It is interesting that, while all values seem to be very close during evaluation and train times, which would otherwise signify good hyperparameter tuning and general performance, the stop loss value seems to spike during train time (see Table 1), which is assumed to be more of a telltale sign that more data is needed. During test time, it is verified in one of the tests we put the model through, that it does not know when to stop inference; otherwise, the sequences that are produced do not exhibit the same behaviour. In general, the loss scores leave much to be desires, however the pretrained weights seem to be rectifying any issue the network may be having during training time, which is promising.

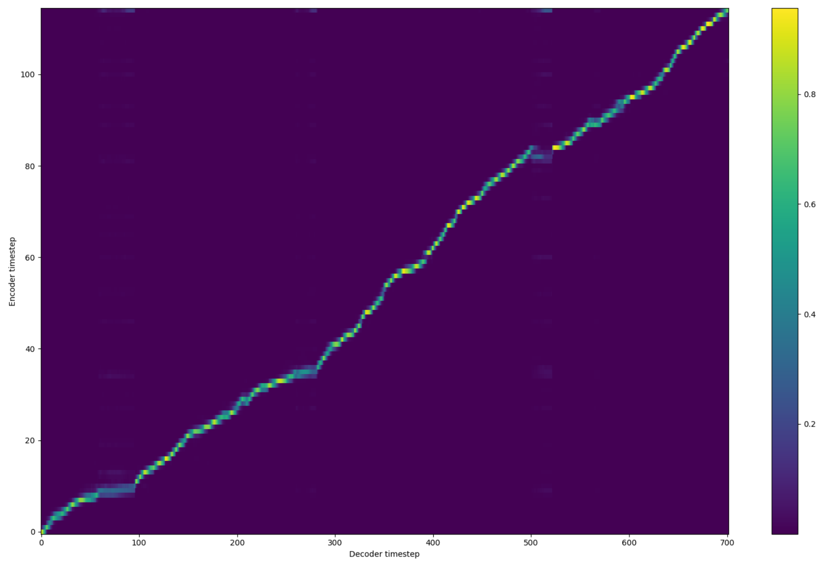

Figure 6: Alignments during evaluation time. The blurred parts show that the model has trouble in predicting the ground truth spectrogram, attributing to either noise or bad phoneme alignments learning.

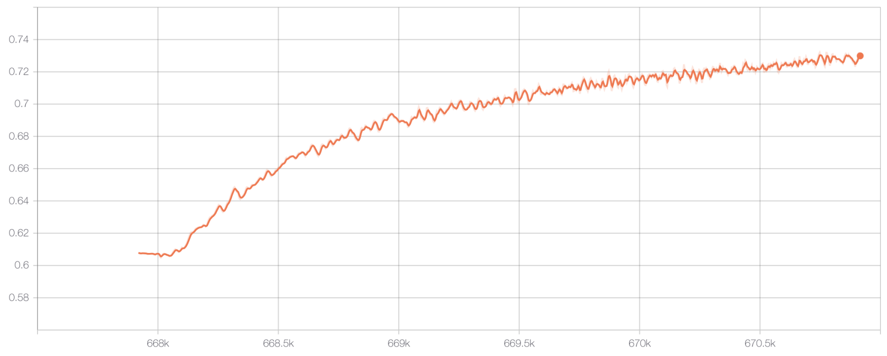

Figure 7: Decoder loss during evaluation time; the increase is attributed to the small amounts of data Tacotron2 is provided with.

As expected and shown on Figure 8, spectrogram prediction is satisfying, attributing to the pretrained weights the network is provided with. Although the loss values are high, the network is still able to produce spectrogram representations, that are improved using the L2 loss function. During test time, Figures 9, 10, 11 and 12 consistently show with good spectrogram predictions, in terms of sound quality; however, the test sentences in Figure 10 and 11 are the ones where the model seems to fall short in its stopnet predictions, and it is noticed that there are repetitions of the end of the sequence.

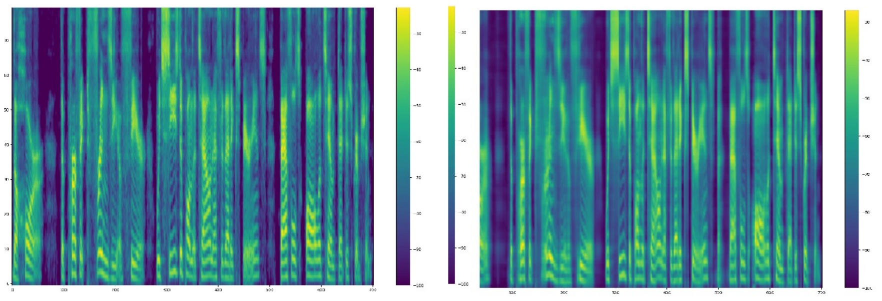

Figure 8: Ground truth and prediction of evaluation sentence. L2 loss function is applied during evaluation time.

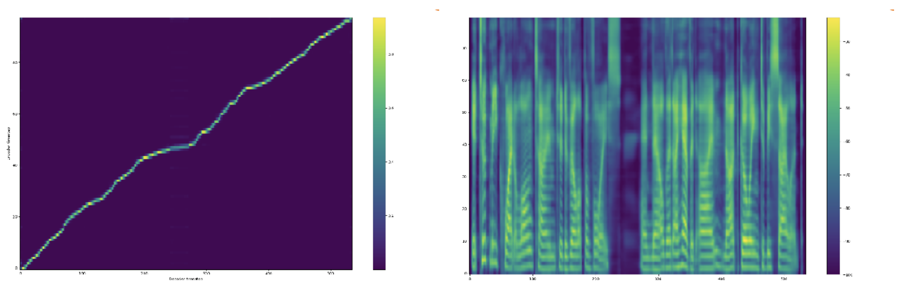

Figure 9: Sentence 1 alignments and spectrogram prediction.

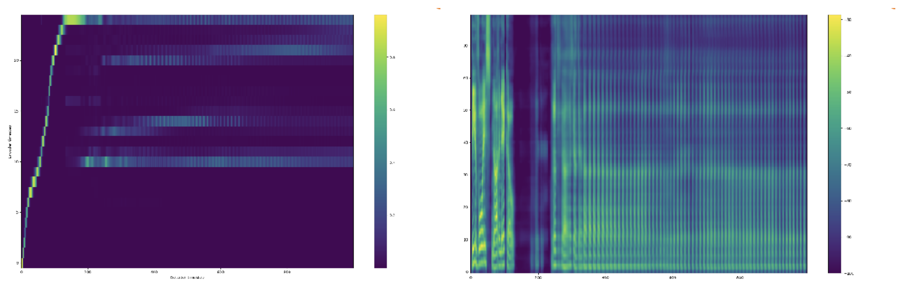

Figure 10: Sentence 2 alignments and spectrogram prediction. The network is having trouble predicting the correct spectrogram and knowing when to stop inferring.

5.2 Training a new model on 30 minutes of speech

For the model trained from scratch on the same dataset used to obtain the results observed in section 5.1, results are expectedly not good; specifically, it is evident that

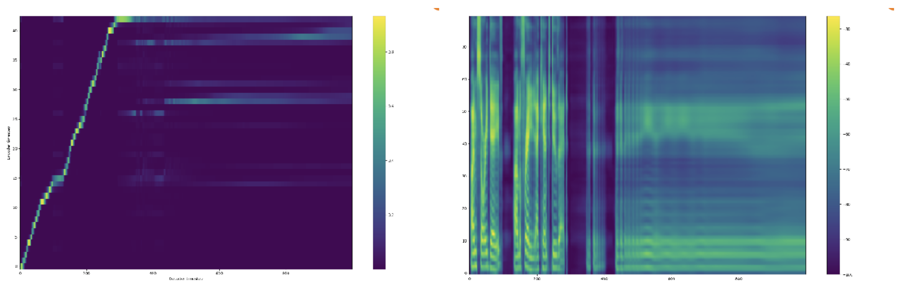
Figure 11: Sentence 3 alignments and spectrogram prediction.

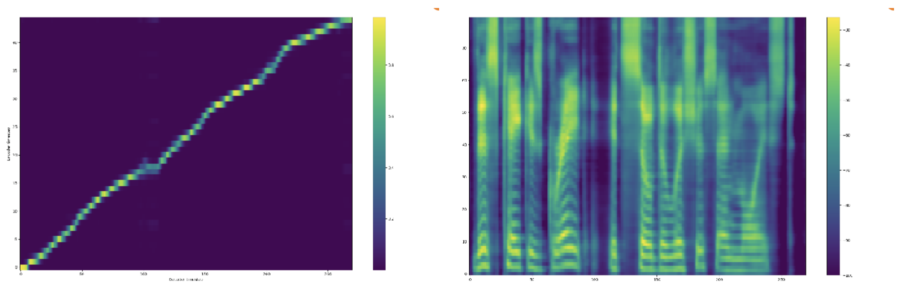
Figure 12: Sentence 4 alignments and spectrogram prediction.

the model is not able to learn any kind of phoneme alignments. There is no speech produced and the ground truth prediction during evaluation time never picks up on the ground truth mel spectrograms. For the sake of fairness, the model is allowed to train for two days on the same setup that is used for the experiments results obtained in section 5.1 (on a model that was trained for 8 hours), in order to allow for more time. In short, this model scores as follows:

Score	**Evaluation**	**Train**
Alignment	9.466e-3	9.300e-3
Loss (Decoder)	2.0	1.80
Lost (Postnet)	2.30	2.20
Stop Loss	0.280	0.250

Table 2: Results obtained when training a new model.

The obtained results show low performance, in terms of alignments and no improvements on any part of the network. More specifically, as Table 2 and Figure 13 exhibit, the loss curve plateaus somewhere around the value of 2, which is an extremely high number and proves that the amount of data provided (30 minutes), is simply not enough to train a seq2seq model from scratch. As mentioned above, a diagonal alignment in the timestep frame signifies the model's ability to form alignments and, as seen on Figure 14, the timestep of this specific network purely consists of artifacts, which explains the very low alignment scores, also (the number approximately translates to -2.66). This same behaviour can be observed in Figure 15, where the spectrogram prediction during evaluation time is, essentially, noise and the network cannot predict the ground truth spectrogram. For these reasons, we are not able to conduct any meaningful tests in the form of synthesis, since the model only produces noise during training, evaluation and test time. It can also be expected that the model would not perform any better using Tacotron2 as the architecture, seeing as Tacotron is smaller and more efficient to train, as it presents with less trainable parameters (Wang et al., 2017).

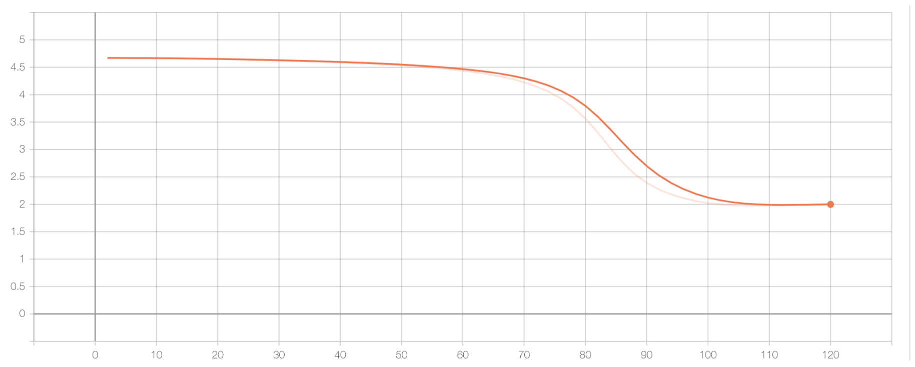

Figure 13: Decoder loss when training the model from scratch; the high rates and the plateuing make it clear that the data is not enough.

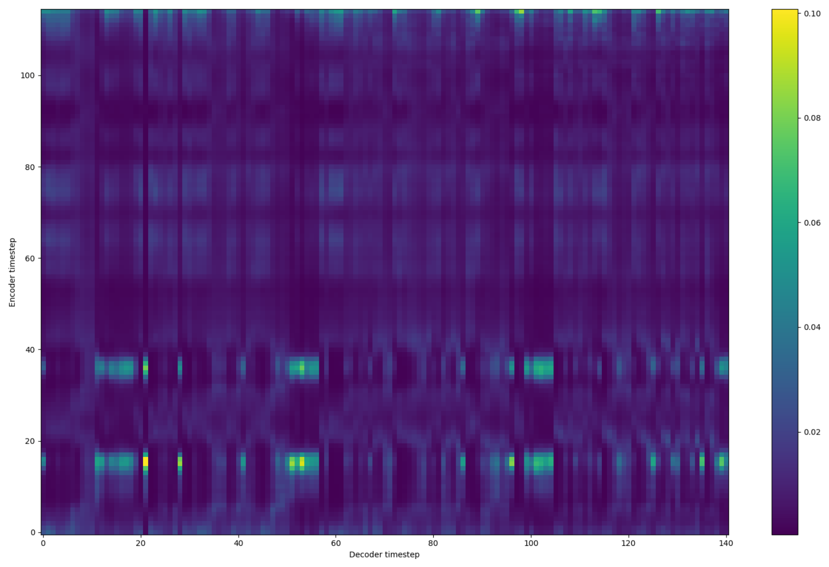

Figure 14: Alignments of model during evaluation time, which showcase an inability to learn phoneme representations.

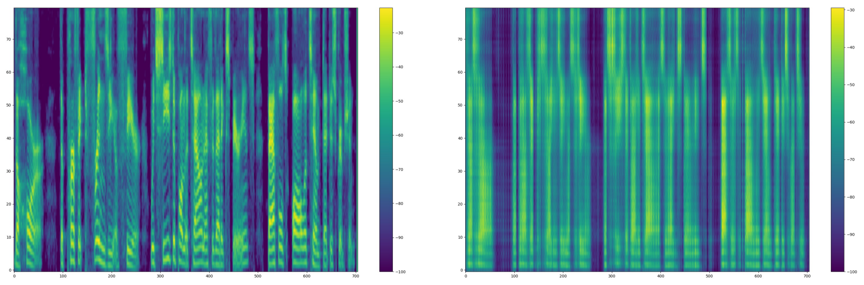

Figure 15: Ground truth and prediction of model during evaluation time; here, the predicted spectrogram is noise.

5.3 Speaker Embeddings

For the evaluation of the pipeline that synthesizes speech using neural speaker embeddings, a command line interface is used, which provides no front-end part; therefore, no visualisation is possible and no screenshots or pictures of how the model performs are provided; instead, listening to the samples obtained is the chosen way of evaluation, after passing the recording of the source speaker and the text to be synthesized. The pipeline, then, computes the voice features of the speaker of the utterance, takes the text as input and conditions the decoder output on the features it extracted from the audio utterance. In general, we find that the performance is satisfying and the output speech resembles that of the source speaker. A discussion of the results can be found further down below.

6 Discussion

6.1 Hypothesis Outcomes

Hypotheses 1 and 2 stated that the finetuned model would perform quite well, given that the pretrained weights were trained for a long time and that the model that were to be trained from scratch would perform very poorly. Both hypotheses are confirmed; from tables 1 and 2, it is observed that the model which was finetuned on pretrained weighs performs consistently much higher than the model that was trained from scratch, and the conditioning of Louppe's (2019) TTS pipeline on the neural speaker embeddings computed from the dataset. Hypothesis 3 states that the TTS engine that is conditioned on neural speaker embeddings will return acceptable results, although not of better quality than the one obtained with the finetuned model. Once again, the hypothesis is confirmed.

6.2 Individual Models

As section 5 recalls, the finetuned model presents with the best performance and results off the three experiments presented in this paper. This does not come as a surprise, since the pretrained weights used serve as a robust backdrop, after having been trained for 670.000 steps on a relatively large dataset of 24 hours, on the same language. It is interesting that, while the alignments the model presents with might look erratic at times (refer to Figure 10 and 11), the model still manages to produce speech of acceptable quality; the loss values and the presence of under-fitting (refer to Figure 7) make it clear that, even though finetuning may, in general, yield better than expected results, a large and balanced dataset is always needed for the network's performance to not be impeded; in the experiments, there is the attempt to leverage the least amount of data, so to simulate an environment where the presence of the former is lacking, but pretrained weights exist. This serves the objective that for different needs, different datasets exist, however the lack of specific data types should not render a task of training a neural network to a different end impossible; for example, a model trained on a dataset that is industry oriented, should serve as a good starting point for a network intended to be used for personal needs. This is showcased by the fact that the original dataset the pretrained weights were trained on, together with the dataset of choice, all comprise of audiobook recordings and the test sentences chosen, are general news headlines extracts.

The model that was trained from scratch does not achieve any accuracy, which is to be expected, considering the amount of data it was fed was incredibly small and would never be enough. In general, it is accepted that a good, balanced dataset with a duration of approximately 24 hours is needed, in order for a model to be able to generalize and achieve robust alignments. The term balanced refers to audio recordings that are, in general, up to 100 characters long, are clean and well transcribed. The efficiency and number of parameters neural network architectures like Tacotron and Tacotron2 offer should, also, not support the assumption that a small amount of data would render a model to perform well. Additionally, the results that are obtained here correlate to what Chung et al. (2018) highlight, which is that, when baseline Tacotron is trained on approximately 24 hours minutes of speech, the network fails to produce intelligible speech. In lieu of these scores, there have been proposed different, semi-supervised training frameworks, in order to improve the data effieciency of Tacotron, by introducing textual and acoustic knowledge in the form of vector representations, that has been gained by training other neural networks on different amounts of text (2018). In these experiments, Tacotron is able to produce intelligible speech, when trained on 30 minutes of speech; however, the time limitations of this specific experiment do not allow such exploration.

As far as evaluating the pipeline that produces speech conditioned on neural speaker embeddings, in the listening tests, a small difference in the speech produced on the neural speaker embeddings computed, can clearly be heard. This has many explanations; first, the authors state that the pretrained models provided have been trained on the VoxCeleb dataset only, and nothing else. This of course sets a limit as to how accurate of embeddings the speaker encoder can compute and how accurate of mel spectrograms the synthesizer can produce. In general, the concept of producing mel spectrograms on speaker embeddings is supported by the notion that all people have different voices, however all of them share some base characteristics and some voices will be more similar to others. It is also quite expected that a single-speaker model is able to perform better on that specific voice and, finally, it is safe to assume that, the fact that the architecture of the author's pipeline has been trained on speech of frequency of 16kHZ, plays a role in quality. All in all, the TTS pipeline seems to perform quite adequately and that similarity between source audio and produced output is there. It is observed that there are sound artifacts between utterances, especially when punctuation is concerned, where the model produces some static noise. In taking the punctuation out, the model produces speech that is very quick and lacks sentiment and prosody. With that said, it should be noted that the speaker encoder of this particular neural network performs very well in capturing the source speaker style in the form of speaker embeddings, seeing as it has been trained on large amounts of data, for a long time; thus, it also affects the quality of the synthesized voice, as presented by Jia et al. (2018). In general, training such neural networks is often an easier task, considering that no text transcriptions are needed, so sound of any kind can be used. Under this prism, these networks may be trained independently of the synthesizer and the data (and quality thereof) the latter requires. An example of a speaker encoder network and its performance is showcased in Figure 16, when the same speaker encoder network that is evaluated in Louppe's (2019) pipeline, is used in combination with a different

Python library, and it has learnt to distinguish between genders, thus is able to form different clusters, based on the gender of the speaker[2].

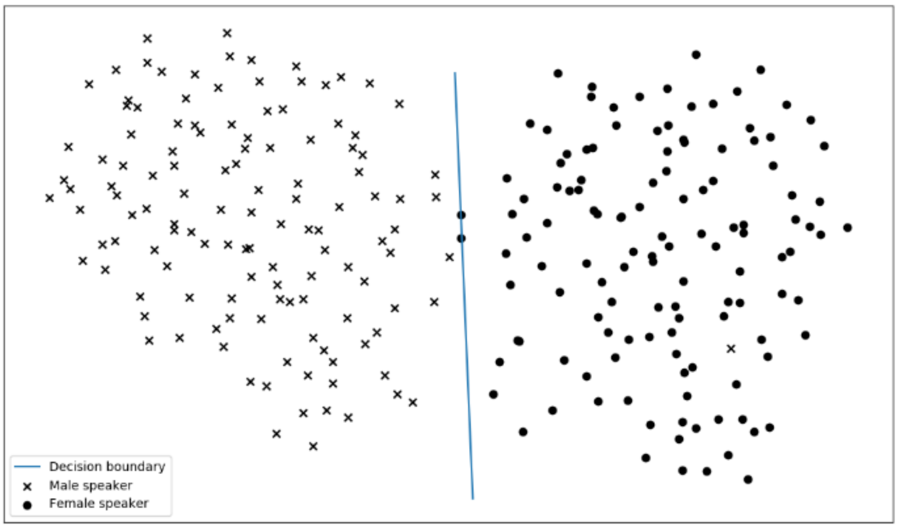

Figure 16: Visualization of a clustering algorithm, using a speaker encoder network. The model is able to distinguish between genders and can form different clusters for male and female voices.

6.3 Overall

The trend that is observed is high loss values, in the two training experiments that are run. Although there is a very big difference in scores and performance between the two models, with the model trained from scratch not performing at all, the scores further solidify the view that a good neural network architecture will not yield any good results if the amount of data is small or of low-quality. The neural speaker embeddings' performance leads us to think that, although machine learning has reached a point where zero-shot neural speech synthesis may be achieved, there still are a lot of factors at play, such as the number of parameters of the network and the construction of the TTS engine in terms of speaker encoder quality and training data. As section 2 recalls, other researchers have considered different types of neural speaker embeddings when modelling speech, in order to rank them and make use of the best one. The types of embeddings the neural speech synthesis engine uses, in order to perform speaker adaptation using embeddings are not known; it should also be noted that the pipeline is, by today's standards, not up to date with current technology. The initial experiment

[2]Picture taken from Rezemblyzer: https://github.com/resemble-ai/Resemblyzer

scope included training a multispeaker TTS on Tacotron2 from scratch, in order to test its performance, but such activity would take a long time, and it was possible to implement it.

The models trained, both depend on Tacotron2 architecture and not Tacotron. The reason for this, because of the fact that the pretrained weights the author is provided with, are also trained on Tacotron2, so they are not compatible with Tacotron setups. It would have been interesting to see if Tacotron might perform better in smaller data, since the trainable parameters of the network are also more limited.

6.4 Considerations when evaluating

It is wise to consider a number of things when evaluating neural speech synthesis. First, one should be aware of the fact that the choice of neural vocoder affects the output waveforms, but should not be confused with the performance of the synthesizer, which produces the mel spectrograms. In all of the tests, the Griffin Lim algorithm is used, since there is not any time to train a neural vocoder, howeverthis is taken into account, when testing the output audio. Even though vocoding should be a part of the evaluation, it should usually come last, since it is a part of the whole pipeline that can be finetuned on its own and so the primary concerns should include naturalness of speech and prosody. To this end, when Mean Opinion Scoring is performed, the sentences are synthesized and the evaluators are asked that they judge whether the voice resembles that of the reference speaker, rather than the sound quality of the clips, which is to be expectedly diminished because of the usage of Griffin - Lim instead of a neural vocoder. In short, four different sentences (found in the appendix) are distributed, and the average scores obtained for each sentence are as follows:

Test Sentence	**MOS: Trained Model**	**MOS: Embeddings**
1	3.75	3.25
2	3.50	2.25
3	3.0	3.0
4	3.25	2.50

Table 3: Mean Opinion Scores acquired on the test sentences. On a scale from 1 to 5, 1 is bad and 5 is excellent, almost wholly resembling of the source speaker.

In table 3, a general score of 3 and above is observed, that does not exceed 4, which is to be expected, since the data is small and so the model seems to be limited in performance; however, in the scale of MOS, with 3 being classified as a "fair" scoring, it does seem that the model is able to perform adequately, preserving prosody and resembling the source speaker. In general, all evaluators have graded the same. These scores posit that, with longer training times and more data, a speech synthesis neural network should be able to achieve an even better result, a performance which may further be improved using a neural vocoder. In terms of speaker similarity as regards the speaker embeddings pipeline, the evaluators are asked to grade on voice similarity and naturalness of utterances. The lower scores are attributed to the less natural speech the engine produced and the static noise that was included in the output samples.

In terms of performance, the model should also be robust when it comes to reading out longer sentences, since LSTMs are known to have trouble when handling long sequences (Nugaliyadde et al., 2019). A solution to that would be to include long sequences during training, although that would increase training times and would, most definitely, put more strain on the GPU, as different experiments have showcased. Another solution includes truncated back propagation through time, where the gradient can be estimated from a subset of the last time steps. It allows all sequences to be provided as input and execute the forward pass, but only the last tens or hundreds of time steps would be used to estimate the gradients and used in weight updates (Tallec and Ollivier, 2017).

Another point of discussion is the nature of data and how well the model can generalize. Even though it is obvious that the goal of a neural network is for it to be able to perform on out-of-domain data, it remains a fact that a model learns what a model is fed; so, if the dataset consists of webpage content readings, this model will obviously not be the best candidate for audiobook reading. This should not take away that such datasets and pretrained weights would not serve as good starting points, however it is a point of interesting discussions to be made; the finetuned model uses pretrained weights that have been trained on a dataset which consists of audiobook recordings, so good performance is achieved, in terms of prosody. To determine whether a neural network trained on different content may serve as a good candidate in audiobooks reading, a smaller experiment is run; the goal here is to train a model on a professionally recorded dataset and have it output speech that comes from text off ebooks. The Swedish Corpus for Speech Synthesis that is found on the National Bank of Norway is used, which includes 7 hours of utterances recorded from a native speaker from Stockholm, sampled at 44kHZ (Nordisk Språkteknologi Holdings, 2002) and consisting of stereo channel, including a laryngograph recording. The corpus is noted to be phonemically optimized, aimed at covering as many diphone combinations as possible, with a goal to include prosody variation. Since the utterances found in it are short, the scope of the experiment is limited to the one of childrens' books, which are known to include shorter sentences. Tacotron2 is used for training, since it has more trainable parameters than Tacotron and offers for better alignments. In general, it is observed that the model is not able to align well and the scores obtained are poor when trying to synthesize speech found on childrens' ebooks, further solidifying the take on how prosody matters and that specific domains call for specific types of data.

6.5 Data criticality: what makes a good TTS dataset?

The conclusion that is drawn, as regards the quality of the output synthesis, is that the quality of the data used for training is responsible for a big part of the performance of the system. These traits correspond to many attributes, such as signal to noise ratio, the pitch of the speaker, the average length of the sentences and the variation of different phonemic combinations. As a general rule, any dataset fit for training machine learning algorithms must be cleaned and organized before it is used on the model. Although the mantra of "more data, more representation" sounds very alluring and, to a certain extent, very reassuring that a model will perform better, when it comes to neural speech synthesis, it is much more preferable to have half the amount of correctly segmented and transcribed recorded speech, than double the latter, with wrong transcriptions and cut speech chunks.

In independent experiments that are run, the objective is to train Tacotron2 with data that include some wrong transcriptions, abrupt sentence endings and aspiration sounds at the ends of phrases, where there should have been a silence, or padding added. It seems that, especially Tacotron2, is able to form much more robust alignments using just 6 (six) hours of speech, if the dataset is mostly clean and assuming that is has the correct transcriptions and a relatively normal, Gaussian-like sentence distribution, while it had a harder time converging on double the amount of hours and a lot of transcription mistakes. The discovery that Tacotron2 produces perfectly intelligible speech on 6 hours of training data is very promising, as it paves the way for additional experiments and partly gives solution to the problem of acquiring clean, transcribed speech data, which is known to be very costly and time consuming to label. Further-more, it is evident that, the dataset must have a high signal-to-noise ratio, devoid of as much background noise as possible. While different authors note on audiobooks being an unfit source of data for text to speech purposes, the author maintains that they are indeed some of the best options, considering that they mostly are recorded in professional settings, which include properties such as a high bit-rate and recordings made in anechoic rooms, on professional equipment. The fact that Tacotron2 is able to learn phoneme alignments itself, only given pairs in the form of <text,audio>, means that more resources may be devoted to good recording quality and annotation, instead.

6.6 The issue of underwhelming speech: prosody modelling

Another trend that is prevalent when neural speech synthesis is leveraged, is the way neural networks tackle speech prosody. In general, as section 2 recalls, Tacotron and Tacotron2 learn, by default, character representations during training and do not retain any prosody characteristics. It is no secret that neural speech synthesis is suggested as an alternative, when the goal is to produce a TTS engine that provides with more natural speech that is more human sounding. In the alternatives of neural TTS, namely concatenative speech synthesis and parametric speech synthesis, one of the requirements is that a large speech database be in a place, which contains as many graphemes and their phoneme counterparts as possible, in order for the model to generalize well and perform an accurate unit selection, especially in out of domain words. That is not to say that a neural network does not need a large phonemic representation, since a neural network essentially learns what it is fed during training time; however, the main disadvantage of the unit selection method, is that the dataset used consists of the speaker speaking in a suppressed tone, with as few vocal variations as possible (Khan and Chitode, 2016). The above leads to a system that is fairly even in tone, thus makes it less apparent when different units are concatenated to form a phoneme. This means that a concatenative speech synthesis system is to be fairly monotonic in speech; however it makes these solutions much less flexible and costly to produce, since a new database is needed when speech must be added.

Both Tacotron and Tacotron2 are able to capture vocal characteristics when trained on data that is expressive and rich in emotional range, however are limited by the approach of learning character embeddings as input. This in turn means the prosody cannot be controlled during inference time and neural networks are, mostly, unpredictable in what they output. Prosody modeling using Global Style Tokens (Wang et al., 2018) can remedy that, by adding an extra embedding layer to the network's architecture, which is responsible for capturing prosody characteristics, while training. These attributes are modeled and included in the hidden layer and during inference time, may be concatenated to the encoder decoder outputs, to control the style of the output speech. The global style tokens are text predicted and do not require explicit labels during training and can be used to control speech synthesis in many levels, such as speed of talking, and style which is independent of the text content. Wang et al. (2018) use the tokens to make Tacotron2 to replicate the speaking style of a single audio clip across an entire long-form text corpus; Stanton et al. (2018) highlight that the tokens are also able to remove background noise that is there, when the dataset the model is trained on is noisy. While Global Style Tokens are usually used to make speech synthesis more expressive, the last part means that global style tokens may also be used in combination with datasets formed from found data, crawled for the internet, when low-resource languages are in question and the data acquired is noisy. In this paper, there are no experiments with Global Style Tokens present, but varying prosody in the output clips is noted, attributing it to a dataset rich in n-gram distributions only seen a handful of times, thus the model was able to model the voice characteristics as well.

6.7 High quality synthesis using a CPU: GAN vocoders

Generating high fidelity audio waveforms is a task that happens with the usage of an additional neural network, which takes in the spectrograms the synthesizer has produced and turns them into sound. As section 3 recalls, these vocoders can be based on architectures like the ones WaveNet and WaveRNN propose (Shen et al., 2017; Kalchbrenner et al., 2018). However, it is also known that these networks are computationally expensive and require access to GPU during inference time. This also calls for increased computation times which may prove inefficient. To this end, Kumar et al. (2019), explore the usage of Generative Adversarial Networks as vocoders, in order to achieve high quality, real time speech synthesis, by training such a network for raw audio generation purposes, without additional distillation. The architecture of the network is that of a typical GAN, that is it consists of a generator, which is a fully convolutional network, that takes a mel spectrogram as input and three discriminators, which operate on different sound scales. Weight normalization ensures that the sound quality does not diminish. The authors find that the model is lightweight, easy to train and produces sound waves of reasonable quality, but that it can also generalize to unseen speakers, when trained on a multispeaker dataset.

Experimenting with GAN techniques for high quality audio synthesis, Yamamoto et al. (2019) propose ParallelWaveGAN, which uses a non-autoreggresive WaveNet network, which "is trained by jointly optimizing multi-resolution spectrogram and adversarial loss functions, which can effectively capture the time-frequency distribution of the realistic speech waveform" (Yamamoto et al., 2019). The network is compact, only has 1.44 million trainable parameters and can generate high quality speech in real time, using a CPU. The novelty lies in the fact that the architecture can be trained without any teacher-student framework, which follows the idea of training two networks, and then the second network tries to replicate the outputs of the first, bigger network (Guo et al., 2018). The absence of this architecture leads to faster training and convergence times. When coupled with a TTS acoustic model, ParallelWaveGAN achieves a MOS of 4.16. The model consists of 30 dilated residual convolutional blocks, with exponentially increasing three dilation cycles. Weight normalization is applied to all the layers, both for the generator and the discriminator. STFT spectral loss is computed, using the sum of three STFT losses, with a Hanning function applied before the FFT process. The evaluations show that the systems trained with STFT losses perform better than the counterparts which do not have STFT losses computed. The autoreggresive WaveNet system produces a high-frequency sound, as a result of it using a band limited mel spectrogram for local conditioning, while the systems with STFT loss were able to extract full band frequency information from the STFT loss itself. As future research, the authors propose improvements in the multi-resolution STFT auxiliary loss, which may lead to better capturing speech characteristics and emotion.

7 Conclusions and Future Work

This paper addresses the task of speaker adaptation, using pretrained weights and neural speaker embeddings. There are several areas that the experiments could have been improved in, if the time frame of the paper were to be bigger, or the reach of the paper were to be further. Firstly, it is noted that the amount of time provided was not enough to add improvements to the models, such as training a neural vocoder, or trying different sets of hyperparameters to achieve better performance scores and alleviate the problem of feeding the neural network with not a lot of data. It is clear from the results that, in situations where small amounts of data may be used, available pre-trained weights prove to be extremely helpful and can be a good starting point in providing with robust alignments; however the high loss score that is present in all of the experiments conducted, which involve training, highlight the importance of data and the role they play in a model's performance. To this end, in section 5.3, evaluation of the performance of an already constructed neural speech synthesis pipeline is done, which conditions the speech on neural speaker embeddings. The initial plan was to train a similar pipeline by ourselves, with different sets of data and evaluate the performance of it and how well it might be able to generalize to unseen speakers. Such objective could, unfortunately, not be completed.

Suggestions for future work are aplenty and generally center around speaker adaptation using neural speaker embeddings. The notion of zero-shot speech synthesis is intriguing and one that should further be explored, in lines of evaluating different types of embeddings and ways to inject them in the model. Training a large, multispeaker TTS model with good speaker representation should, in theory, yield good results when attempting voice cloning, using an unseen speaker's voice embeddings.

Speech vocoding remains a computationally expensive task and requires an additional neural network that turns the mel spectrograms the synthesizer predicts to sound quality waveforms. Generative Adversarial Networks may help in bridging the gap between high quality audio generation and computation times.

Global Style Tokens are promising in the sense that they may allow a speech synthesis model more control over speaking style and multispeaker TTS systems trained on large, expressive, multispeaker datasets would greatly benefit from that. Architectures like Tacotron and Tacotron2 have made it possible that expressive speech can be produced, although more work remains to be done on prosody modeling.

8 References

Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F. M., and Weber, G. (2019). Common voice: A massively-multilingual speech corpus.

Arik, S., Diamos, G., Gibiansky, A., Miller, J., Peng, K., Ping, W., Raiman, J., and Zhou, Y. (2017). Deep voice 2: Multi-speaker neural text-to-speech.

Arik, S. O., Chen, J., Peng, K., Ping, W., and Zhou, Y. (2018). Neural voice cloning with a few samples.

Barras, B. (2012). Sox : Sound exchange.

Chiu, C.-C., Sainath, T., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R. J., Rao, K., Gonina, K., Jaitly, N., Li, B., Chorowski, J., and Bacchiani, M. (2018). State-of-the-art speech recognition with sequence-to-sequence models.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition.

Chung, Y.-A., Wang, Y., Hsu, W.-N., Zhang, Y., and Skerry-Ryan, R. (2018). Semi-supervised training for improving data efficiency in end-to-end speech synthesis.

Cooper, E., Lai, C.-I., Yasuda, Y., Fang, F., Wang, X., Chen, N., and Yamagishi, J. (2019). Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings.

Doukhan, D., Lechapt, E., Evrard, M., and Carrive, J. (2018). Ina's mirex 2018 music and speech detection system. In *Music Information Retrieval Evaluation eXchange (MIREX 2018)*.

Fang, W., Chung, Y.-A., and Glass, J. (2019). Towards transfer learning for end-to-end speech synthesis from deep pre-trained language models.

Gerchberg, R. W. (1972). A practical algorithm for the determination of phase from image and diffraction plane pictures.

Griffin, D. and Jae Lim (1984). Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243.

Guo, T., Xu, C., He, S., Shi, B., Xu, C., and Tao, D. (2018). Robust student network learning.

Hong, Y., Hwang, U., Yoo, J., and Yoon, S. (2019). How generative adversarial networks and their variants work. *ACM Computing Surveys*, 52(1):1–43.

Hunt, A. J. and Black, A. W. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 373–376 vol. 1.

Iakushkin, O., Fedoseev, G., Shaleva, A., and Sedova, O. (2018). Building corpora of transcribed speech from open access sources. In *Proceedings of the VIII International Conference "Distributed Computing and Grid-technologies in Science and Education" (GRID 2018)*, volume 2267 of *CEUR workshop proceedings*, pages 475–479, Germany. RWTH Aahen University.

Ito, K. (2017). The lj speech dataset. `https://keithito.com/LJ-Speech-Dataset/`.

Jia, Y., Zhang, Y., Weiss, R. J., Wang, Q., Shen, J., Ren, F., Chen, Z., Nguyen, P., Pang, R., Moreno, I. L., and Wu, Y. (2018). Transfer learning from speaker verification to multispeaker text-to-speech synthesis.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., van den Oord, A., Dieleman, S., and Kavukcuoglu, K. (2018). Efficient neural audio synthesis.

Khan, R. A. and Chitode, J. S. (2016). Concatenative speech synthesis: A review. *International Journal of Computer Applications*, 136:1–6.

Kim, C.-c., Ha, S.-y., Roh, B.-h., Jeong, J.-p., and Choi, J.-y. (2013). Measurement method for mean opinion score in actual home environments. pages 16–19.

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brebisson, A., Bengio, Y., and Courville, A. (2019). Melgan: Generative adversarial networks for conditional waveform synthesis.

Li, N., Liu, S., Liu, Y., Zhao, S., and Liu, M. (2019). Neural speech synthesis with transformer network. In *AAAI*.

Louppe, G. (2019). Master thesis : Automatic multispeaker voice cloning.

Luong, H.-T. and Yamagishi, J. (2019). A unified speaker adaptation method for speech synthesis using transcribed and untranscribed speech with backpropagation.

Masuyama, Y., Yatabe, K., Koizumi, Y., Oikawa, Y., and Harada, N. (2019). Deep griffin-lim iteration.

Nagrani, A., Chung, J. S., and Zisserman, A. (2017). Voxceleb: a large-scale speaker identification dataset.

Neekhara, P., Donahue, C., Puckette, M., Dubnov, S., and McAuley, J. (2019). Expediting tts synthesis with adversarial vocoding.

Nordisk Språkteknologi Holdings, A. (2002). Nst swedish speech synthesis (44 khz).

Nugaliyadde, A., Wong, K. W., Sohel, F., and Xie, H. (2019). Language modeling through long term memory network.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). 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.

Pascual, S., Serrà, J., and Bonafonte, A. (2019). Exploring efficient neural architectures for linguistic–acoustic mapping in text-to-speech. *Applied Sciences*, 9:3391.

Perraudin, N., Balazs, P., and Søndergaard, P. L. (2013). A fast griffin-lim algorithm. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4.

Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. (2019). Fastspeech: Fast, robust and controllable text to speech.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., Saurous, R. A., Agiomvrgiannakis, Y., and Wu, Y. (2018). Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R., Saurous, R. A., Agiomyrgiannakis, Y., and Wu, Y. (2017). Natural tts synthesis by conditioning wavenet on mel spectrogram predictions.

Stanton, D., Wang, Y., and Skerry-Ryan, R. (2018). Predicting expressive speaking style from text in end-to-end speech synthesis.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks.

Tallec, C. and Ollivier, Y. (2017). Unbiasing truncated backpropagation through time.

Tenney, I., Das, D., and Pavlick, E. (2019). Bert rediscovers the classical nlp pipeline.

Tits, N., Haddad, K. E., and Dutoit, T. (2019). Exploring transfer learning for low resource emotional tts.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016b). Conditional image generation with pixelcnn decoders.

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. (2017). Parallel wavenet: Fast high-fidelity speech synthesis.

Vig, J. (2019). A multiscale visualization of attention in the transformer model.

Wang, D. and Chen, J. (2017). Supervised speech separation based on deep learning: An overview.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. A. (2017). Tacotron: Towards end-to-end speech synthesis.

Wang, Y., Stanton, D., Zhang, Y., Skerry-Ryan, R., Battenberg, E., Shor, J., Xiao, Y., Ren, F., Jia, Y., and Saurous, R. A. (2018). Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis.

Yamamoto, R., Song, E., and Kim, J.-M. (2019). Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram.

Zen, H., Clark, R., Weiss, R. J., Dang, V., Jia, Y., Wu, Y., Zhang, Y., and Chen, Z. (2019). Libritts: A corpus derived from librispeech for text-to-speech. In *Interspeech*.

Zhang, J.-X., Ling, Z.-H., and Dai, L.-R. (2018). Forward attention in sequence- to-sequence acoustic modeling for speech synthesis. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

A Mozilla TTS github repository

https://github.com/mozilla/TTS

B Samples used for Mean Opinion Scoring

https://drive.google.com/open?id=12mOaSNbKlMtJO8XXFRHS274Fn8cqAYj2