Fakultät I
Institut für Sprache und Kommunikation
Fachgebiet Audiokommunikation

Expressive End-To-End Speech Synthesis by Varying Structural and Variational Capacity

Adam Froghyar

Primary Supervisor:	Prof. Dr. Stefan Weinzierl
Secondary Supervisor:	Dr. Athanasios Lykartsis

[redacted]

A thesis submitted to Technische Universität Berlin in fulfilment of the requirements for the degree Master of Science in Audio Communication and Technology.

Technische Universität Berlin
Fakultät I - Geistes- und Bildungswissenschaften
Institut für Sprache und Kommunikation
Berlin, 2021-09-13

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt gegenüber der Fakultät I der Technischen Universität Berlin, dass die vorliegende, dieser Erklärung angefügte Arbeit selbstständig und nur unter Zuhilfenahme der im Literaturverzeichnis genannten Quellen und Hilfsmittel angefertigt wurde. Alle Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, sind kenntlich gemacht. Ich reiche die Arbeit erstmals als Prüfungsleistung ein. Ich versichere, dass diese Arbeit oder wesentliche Teile dieser Arbeit nicht bereits dem Leistungserwerb in einer anderen Lehrveranstaltung zugrunde lagen.

Titel der schriftlichen Arbeit

Expressive End-To-End Speech Synthesis by Varying Structural and Variational Capacity

Verfasser

Adam Froghyar, [redacted]

Betreuende Dozenten

Prof. Dr. Stefan Weinzierl,
Dr. Athanasios Lykartsis

Mit meiner Unterschrift bestätige ich, dass ich über fachübliche Zitierregeln unterrichtet worden bin und verstanden habe. Die im betroffenen Fachgebiet üblichen Zitiervorschriften sind eingehalten worden. Eine Überprüfung der Arbeit auf Plagiate mithilfe elektronischer Hilfsmittel darf vorgenommen werden.

[redacted]

Ort, Datum

A

Abstract

Recent developments in neural networks have introduced a novel state-of-the-art technology in human speech synthesis which dominates today's consumer as well as research side of text-to-speech systems. With increasing efforts put into all aspects of the pipeline, the past few years have seen the successful implementation of prosody control and transfer into end-to-end speech synthesis systems with prominent results.

This work gives an introduction into the theory of expressive speech synthesis with focus on a recent, specific synthesis method, and reports on the practical implementation and evaluation of this method within an open source software framework. Exploring the prosodic augmentation of a state-of-the-art neural speech synthesis system, this work extends discussion and evaluation on two variables influencing the augmentation specific to this method: the structural and variational capacities that control the amount of expressiveness of the synthetic speech. Through replicating and modifying two subjective evaluation tests introduced in the original method, this thesis outlines a direct and critical assessment of the implemented method based on the prosodic transfer characteristics and overall naturalness of synthesised speech samples.

Offering the implemented method as open source code, this work aims to be a contribution to the vast ecosystem of open source text-to-speech software to enable researchers to experiment with and further develop expressive speech synthesis technologies.

Zusammenfassung

Die jüngsten Entwicklungen auf dem Gebiet der neuronalen Netze haben eine neuartige, hochmoderne Technologie für die menschliche Sprachsynthese eingeführt, die heute sowohl die Verbraucher- als auch die Forschungsseite von Text-to-Speech-Systemen dominiert. Mit zunehmenden Bemühungen um alle Aspekte der Pipeline wurde in den letzten Jahren die Prosodiekontrolle und -übertragung in End-to-End-Sprachsynthesesysteme mit herausragenden Ergebnissen implementiert.

Diese Arbeit gibt eine Einführung in die Theorie der expressiven Sprachsynthese mit Fokus auf einer neueren, spezifischen Synthesemethode und berichtet über die praktische Implementierung und Evaluierung dieser Methode innerhalb eines Open-Source-Software-Frameworks. Diese Arbeit untersucht die prosodische Ergänzung eines hochmodernen neuronalen Sprachsynthesesystems und erweitert die Diskussion und Evaluierung von zwei Variablen, die die Erweiterung spezifisch für diese Methode beeinflussen: die strukturellen und variationalen Kapazitäten. Durch die Replikation und Modifikation von zwei subjektiven Hörversuchen, die in der ursprünglichen Methode vorgestellt wurden, stellt diese Arbeit eine direkte und kritische Bewertung der implementierten Methode auf der Grundlage der prosodischen Übertragungseigenschaften und der allgemeinen Natürlichkeit der synthetisierten Sprachproben dar.

Durch die Bereitstellung der implementierten Methode als Open-Source-Code soll diese Arbeit einen Beitrag zu dem umfangreichen Ökosystem von Open-Source Text-to-Speech-Software leisten, das es Forschern ermöglicht, zu experimentieren und aussagekräftige Sprachsynthesetechnologien zu entwickeln.

Acknowledgements

I would like to thank the entire Audio Communication group at TU Berlin for the inspiring learning and working environment and especially Dr. Athanasios Lykartsis who has supported me along the way of this thesis project despite the fact that he was no longer working at the university and has taken off his personal time to actively support and assist on this work. I am thankful for the opportunities and the knowledge this faculty has provided me.

Furthermore, I would like to thank the entire team at why do birds GmbH for sponsoring and supporting this project. They have provided the technical backbone to the software development, in addition to the constant trust and support in my work. Special thanks to Antoine Nivet for his patience and guidance along the development cycle.

This work would not have been possible without the great open source community around Coqui TTS, who have not only provided the base implementation for the new model implemented in this work, but also showed constant support and gave valuable feedback to questions along the way. Special thanks to Julian Weber and Eren Gölge for their helpful discussions and feedback, to Edresson Casanova for the pre-trained vocoder model and code reviews and to Nicolas Müller for the dataset segmentation and the many helpful discussions, meetings and code debugging sessions.

I would like to especially acknowledge Eric Battenberg from Google, the main author of the Capacitron method, whose invaluable insight into the model mechanisms and training processes made this work a reality. Our meetings and email exchanges were an essential part of the process and I am very thankful for his time.

Furthermore, I would like to thank my friends and family for supporting me along the road of finishing this project. Their patience and loving support made me appreciate the development of this Thesis even when the road was rocky. Special thanks to Carla for always shining a light.

Table of Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence 3

API Application Programmable Interface 42

DL Deep Learning 3

DNN Deep Neural Network 27

GPU Graphics Processing Unity 23, 26, 27, 35

i.i.d. Independent and Identically Distributed 15

LHS Left Hand Side 33

ML Machine Learning 3, 26–28, 30, 59

MLP Multi Layer Perceptron 32

MOS Mean Opinion Score 9, 39–42

NLP Natural Language Processing 30

NN Neural Network 13

RHS Right Hand Side 33, 68, 69

STT Speech-To-Text 25, 59

TTS Text-To-Speech XV, 1, 4, 9, 10, 13, 17, 18, 20, 21, 23, 24, 26, 27, 29, 30, 32, 35–38, 40, 41, 55, 57, 59, 60, 62

VAE Variational Autoencoder 14, 17–19, 21, 40, 59

VRAM Video Random Access Memory 23, 26

Contributors

This thesis is the candidate's own original work except for commonly understood and accepted ideas or where explicit reference to the work of other people, published or otherwise, is made. The thesis is formatted as a monograph comprising seven chapters.

The candidate was responsible for every step involved in designing and carrying out all experiments mentioned in this thesis, as well as analysing collected data and preparing visualisations in this work. Athanasios Lykartsis, the secondary supervisor of this thesis, provided guidance on the experimental setups, data analysis, the interpretation of the results and the structure of this Master's Thesis.

The company why do birds GmbH in Berlin provided computing hardware, office space and resources for the model trainings detailed in Chapter 4. The Berlin based Coqui GmbH provided the open source base Text-To-Speech implementation into which the method discussed in this work was implemented.

1 Introduction

The two foundational tools of communication between humans are written and verbal communication. We have developed constructs of languages as instruments to carry out these two different ways of sharing information between us. Next to human-to-human interactions, our society is more and more reliant on human-to-machine interaction. With the arrival of personal computers, we developed new types of languages, techniques and routines for *human-machine interaction.* The 1992 book from the New York based Association for Computing Machinery defines human-computer interaction as

> *„... a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them."*[11, p. 5]

The major tool behind these human-machine languages, however is still the fundamental instrument of writing. Verbal communication with a machine is one of the most researched topics in the human-computer interaction discipline, with focus on the two major areas of speech recognition and speech synthesis [12]. The main goal of these technologies is to achieve the level of communication between two humans in the context of a human-machine scenario. Speech synthesis in the context of this work refers to the TTS technology whereby the machine/computer is able to convert given text into audible speech [12]. TTS solutions have been present since many years primarily for helping the visually disabled [13], however development of these technologies have gained immense attention since the arrival of personal assistants in smart phones and other devices.

This Master's Thesis is a report and a contribution to current TTS research and engineering topics about closing the gap in the level of communication between human-to-human and human-to-machine interaction.

In order to define what is meant under the levels of communication, a framework for assessing synthesised speech is required. Aida–Zade, Ardil and Sharifova in [14] introduce two parameters, the *naturalness of sounding* and the *intelligibility of speech* as the two criteria for a good speech synthesizer. While the *naturalness of sounding* can loosely depend on how many generated sounds are close to natural human speech, *intelligibility* can be defined as the easiness of artificial speech understanding. Existing and developing technologies aim to improve on both of these characteristics.

Beginning in the mid 20th century, the three major generations of speech synthesis systems were the parametric, concatenative, and unit-selection approaches [14]. Parametric synthesis uses a range of source-filter models to estimate human speech through the resonant frequencies of the vocal tract called formants. Concatenative synthesis, on the other hand produces speech by concatenating small, prerecorded units of speech to construct an utterance [13], while unit-selection synthesis expands on this idea with longer segments and the implementation of a cost function to regulate the model. With the introduction of statistical parametric speech synthesis using Hidden Markov Models, the early 2000s saw the application of a new state-of-the-art technique which defined the ground for further speech synthesis research. While increasingly satisfying one of the above mentioned criteria - intelligibility -, the audio produced by these systems often sounds muffled and unnatural compared to human speech [4].

What defines the *naturalness of sounding*? Speech synthesis is an *underdetermined* problem, meaning the same text input has an infinite number of reasonable spoken realisations [8]. A system aiming to realistically reproduce human speech must implicitly or explicitly impute many factors that are not given in a simple text input [6] so that the intrinsic meaning expressed by *how* the utterance was spoken out can be coupled with *what* the input text contained. Factors referring to *how* an utterance is spoken out include the intonation, stress, rhythm and style of speech - collectively referred to as *prosody*. The prosodically determined structure of an utterance - timing, amplitude and fundamental frequency - are in line with the elementary attributes of sound itself and thereby establish a varying dimension of information to be processed by the listener [15]. This high-level dimension of meaning is what makes the determined problem of turning text into intelligible speech an underdetermined challenge whereby the basic text input to audio output scenario is augmented with a consideration for this meta-level information not encoded in text.

When it comes to building controllable and custom prosody into the synthesis of speech utterances, the systems detailed above are disabled by limitations: parametric speech synthesis offers high customisation ability stemming from the number of parameters in the signal flow, however, the results sound unnatural. The principle of concatenation of speech segments, on the other hand seems more natural but allow the control of few parameters [16]. In order to achieve close to human-like *naturalness of sounding*, these clear limitations present strong motivation to implore different modelling techniques.

Recent developments in the DL subset of ML technologies have enabled a new frontier of computer synthesised speech. ML enabled researchers and engineers to think and work differently about the problem: now, with the emphasis shifting away from the traditional systems detailed above, the focus lays on deep neural networks which feed on great amounts of input data with which the machine learns a statistical representation of the way the input speaker speaks and can reproduce speech utterances previously unseen during training. With increasing investments in AI research and engineering [17], these technologies have become the forefront of speech synthesis systems and the active research by companies such as Google [1] enabled some breakthrough results in the *naturalness of sounding* criteria [4]. A major difference to older technologies is the fact that these newer systems work in an end-to-end fashion, meaning that a single text input to the system yields a synthesised audio output. We refer collectively to these as TTS systems.

An important limitation of these new systems, however is the global limitation of all machine learning algorithms: data quality and availability [18]. The definition for data accuracy and quality in the context of reproducing speech is an under-researched topic in spite of the fact that some of these models utilising multi-speaker compatibility use up to 320 hours of recorded speech material [8]. Availability for such large datasets is sparse, with the exception of a few open source and free to use speech corpuses [2] that are not always applicable to the specific case.

Older, signal processing and concatenation-based speech synthesis systems managed to achieve satisfactory intelligibility, however with the arrival of new, data-based DL approaches the question arises: Can machines speak to us in an emotionally conveying way?

This Thesis details the work of assessing existing open source research of expressive speech synthesis and of implementing a previously not publicly available state-of-the-art expressive speech synthesis method[3] into an open source code framework. Additionally, this work also reports on the acquisition and cleaning of the same dataset used in the original method. In order to evaluate the quality of the implementation, this work also outlines the execution and results of a custom made remote listening test. To expand on the existing synthesis method, the thesis and the corresponding listening test have been adjusted to consider a previously not-so-discussed aspect of the implementation, namely the influence of two specific parameters on the subjective evaluation of the model performance.

[1]Tacotron: An end-to-end speech synthesis system by Google, last visited: 04.10.2020
google.github.io/tacotron/

[2]LJSpeech Dataset, last visited: 04.10.2020
keithito.com/LJ-Speech-Dataset/

[3]Capacitron: Effective Use of Variational Embedding Capacity in Expressive end-to-end Speech Synthesis, last visited: 04.10.2020
google.github.io/tacotron/publications/capacitron

The main research questions can be formulated as follows:

- Can this implementation be evaluated for A) its prosodic transfer characteristics and B) its naturalness and intelligibility via a listening test?
- How does varying structural and variational capacity impact the prosodic transfer characteristics of the Capacitron method?

Beyond scientific research, the direct implication of this project is the openly available implementation of a previously unavailable expressive end-to-end speech synthesis system. The open source availability of such an expressive system for synthesised speech applications would offer an increased amount of information transferred through TTS applications today ranging from communication aids for the physically or visually impaired to on-demand mass consumer products where the customer directly converses with a device. Human-computer interfaces, conversational assistants as well as content narration are all touchpoints where deterministic prosody would improve the user experience. Moreover, there is increasing interest in the creation of recognisable custom voices for brands and organisations where prosody is one of the major features of differentiation. The open source nature of the development cycle and the implementation code will enable future collaborators and researchers to use and improve the system, further contributing to the expansive world of open source TTS software.

1.1 Thesis Structure

Chapter 1 introduced the main ideas and motivation behind this work and defined the main research questions for the thesis. Chapter 2 offers a brief summary of the major milestones of expressive TTS synthesis, including the methods relevant for this work. Chapter 3 then presents the reader with the main theoretical framework behind the Capacitron method - including the derivation and definition of the governing loss function for the deep learning model -, and defines the structural and variational capacities investigated in this work. Chapter 4 offers the technical report on the development process, including the review on the acquisition and implementation of the dataset used for the machine learning models. Then, Chapter 5 describes the design, execution, and results of the subjective evaluation conducted for this work. This chapter also includes the discussion of the results, including the presentation of processed data from the model trainings to compare the subjective test results with the objective scalar values of certain model parameters. While Chapter 6 briefly describes the open source contributions of this Master's Thesis, Chapter 7 concludes the report with a summary of the performed work and includes the open questions and aspects left for further discussions about this research topic.

2 Expressive Text to Speech Synthesis

This chapter briefly introduces the current state-of-the-art of the two major pillars of the proposed thesis: TTS and expressive TTS via prosody transfer.

2.1 Text To Speech

Modern models of text-to-speech conversion by computers are complex pipelines [19]. The two fundamental processes of *text analysis* and *speech synthesis* defined by [19] entail that such system can often have

- a text front-end extracting various linguistic features,
- a duration model,
- an acoustic feature prediction model and
- a complex signal-processing-based vocoder [3].

With each section requiring broad expertise, laborious design and separate training where errors may compound, such designs require substantial engineering efforts [3]. In the following, the three major developments in the past four years are detailed to introduce the evolution of the current state-of-the-art technology.

2.1.1 WaveNet

The current wave of DL-based TTS systems can be strongly defined by the 2016 release of the *WaveNet* neural network from the Google subsidiary Deep Mind research team [1]. WaveNet is a generative, deep neural network model operating directly on the raw audio waveform. Given waveform $\mathbf{x} = \{x_1, ..., x_T\}$, its joint probability is factorised as a product of conditional probabilities as

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t|x_1,...,x_{t-1}), \tag{2.1}$$

where each audio sample x_t is conditioned on the samples at all previous time steps. In order to efficiently model this probabilistic relationship, the authors applied a dilated causal convolution depicted in Figure 2.1 to increase the network's receptive field by orders of magnitude, without greatly increasing computational cost [1].

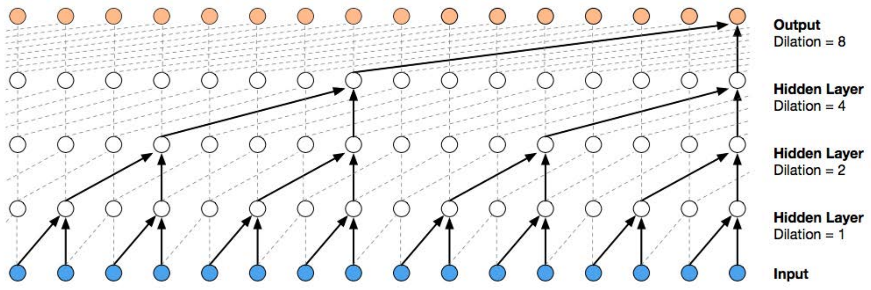


Figure 2.1: Visualisation of a stack of dilated causal convolutional layers[1]

At training time, the input sequences are real wave-forms recorded from human speakers, while after training, the network is sampled to generate synthetic utterances [2]. Using this model, the authors conducted mean opinion score (MOS) tests where the subjects were asked to rate the naturalness of the stimulus in a five-point scale score (1: Bad, 2: Poor, 3: Fair, 4: Good, 5: Excellent). The results of this test - depicted in Figure 2.2 - concluded that the technology produced significantly more natural speech utterances than previous technologies and thereby became the new state-of-the-art.

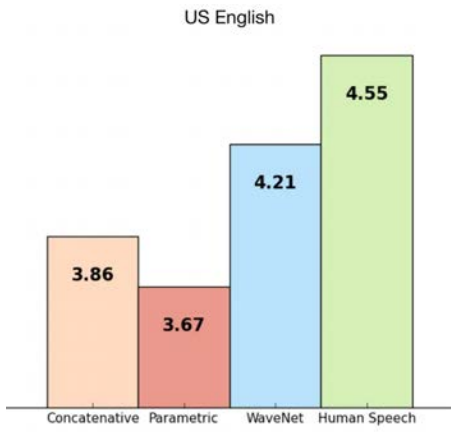


Figure 2.2: Results of the WaveNet MOS test [2]

The two major limitations of this technology are the computing costs and that it still requires conditioning on linguistic features from an existing TTS front-end.

Firstly, due to its sample-level autoregressive nature and the fact that raw audio data is typically very high-dimensional (e.g. 24000 samples per second for audio sampled at 24kHz), the original WaveNet technology could not be implemented in any real time synthesis scenario [20].

Secondly, within the pipeline for a TTS system detailed above, WaveNet only replaces the vocoder and the acoustic model [3]. It is still not an over-arching solution that would ideally only require a pair of <text, audio> for training, since it still requires linguistic features, predicted log fundamental frequency (F0), and phoneme duration as inputs [4].

Since the inception of the WaveNet technology in 2016 and 2017, multiple new studies have been released [21] [20] that deal with speeding up the audio sample generation. Using WaveNet-like designs, there are several systems that have reached real-time or faster than real-time synthesis speeds with similar output audio quality [22] [23]. A derivative of these vocoders is to date in production as Google's leading TTS voice offering [1].

2.1.2 Tacotron 1

Aiming for a solution that would require the least complexity in the training and inference pipeline, the 2017 proposal from the Google research team introduced the first steps towards and end-to-end TTS solution, named Tacotron [3]. The aim of this research was to define an integrated end-to-end TTS system that can be trained on <text, audio> pairs with minimal human annotation [3]. There are multiple advantages to such a design:

- it alleviates the need for laborious feature engineering, which may involve heuristics and brittle design choices,
- it more easily allows for rich conditioning on various attributes, such as speaker or language, or high-level features like sentiment,
- a single model is more robust than a multi-stage model [3].

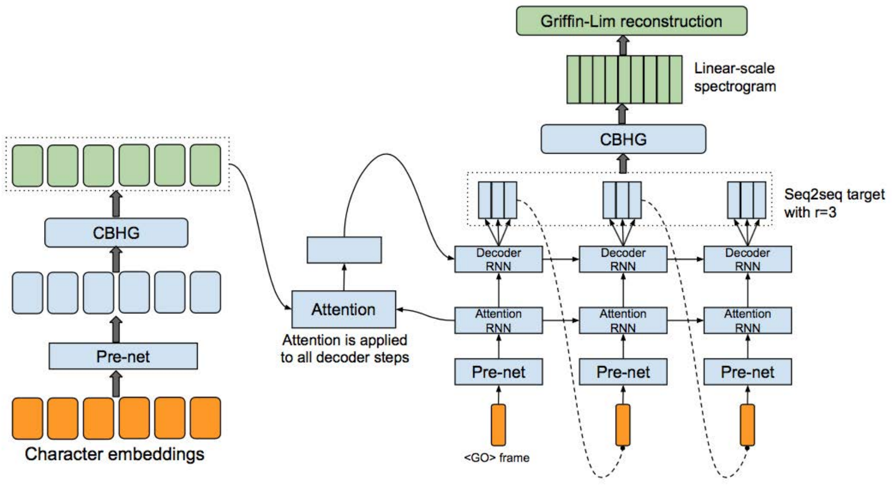


Figure 2.3: Tacotron 1 model architecture [3]

Figure 2.3 demonstrates this end-to-end fashion, where in inference time, the model takes characters as input and outputs the corresponding raw linear-scale spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesise speech [3].

[1]Google Cloud text-to-speech Service, last visited: 04.10.2020
cloud.google.com/text-to-speech

The vocoder algorithm is not conditioned on the input data and therefore often produced noisy results [3]. While the output audio from Tacotron 1 has not reached the MOS score of that of WaveNet's, the successful design of such a system paved the way for further research into end-to-end TTS architectures.

2.1.3 Tacotron 2

The 2018 landmark paper from the same team introduced Tacotron 2, a truly end-to-end TTS system where the last part of the pipeline (among other changes to the sequence-to-sequence model) of converting a mel-scale spectrogram into raw audio has been designed using a trainable WaveNet vocoder. This architecture is now a unified, entirely neural approach to speech synthesis that combines the best of the previous approaches: a sequence-to-sequence Tacotron-style model that generates mel spectrograms, followed by a modified WaveNet vocoder [4]. Figure 2.4 shows the entirely trainable system design.

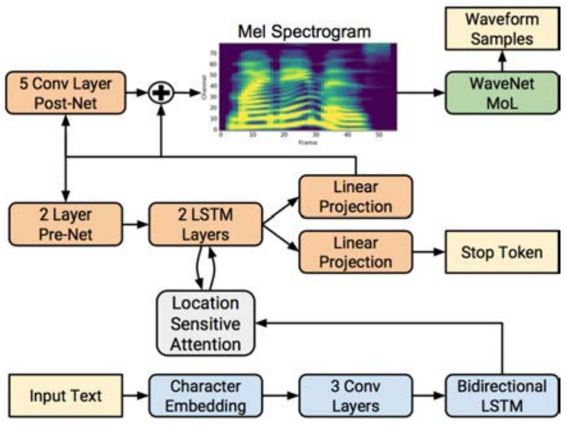
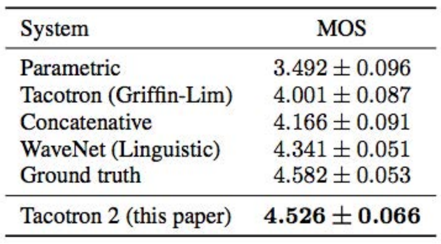


Figure 2.4: Tacotron 2 model architecture [4]

The MOS scores illustrated in Figure 2.5 show that the new state-of-the art in end-to-end TTS synthesis has been achieved.



System	MOS
Parametric	3.492 ± 0.096
Tacotron (Griffin-Lim)	4.001 ± 0.087
Concatenative	4.166 ± 0.091
WaveNet (Linguistic)	4.341 ± 0.051
Ground truth	4.582 ± 0.053
Tacotron 2 (this paper)	$\mathbf{4.526 \pm 0.066}$

Figure 2.5: Tacotron 2 MOS [4]

2.1.4 HiFi-GAN

The acoustic model used for this work to transform synthesised spectrograms into audio files is the state-of-the-art High Fidelity Generative Adversarial Network (HiFi-GAN) vocoder [24]. This modern model from late 2020 achieves both very efficient and high-fidelity speech synthesis without being reliant on an autoregressive architecture.

> *„As speech audio consists of sinusoidal signals with various periods, modelling the periodic patterns matters to generate realistic speech audio. Therefore, we propose a discriminator which consists of small sub-discriminators, each of which obtains only a specific periodic parts of raw waveforms. This architecture is the very ground of our model successfully synthesizing realistic speech audio. As we extract different parts of audio for the discriminator, we also design a module that places multiple residual blocks each of which observes patterns of various lengths in parallel, and apply it to the generator."* [24, p. 5]

While there are many different models for spectrogram conversion, the choice for this method has been made based on

- its availability in the used open source framework;
- its state-of-the-art naturalness and intelligibility score on MOS tests;
- its synthesis speed: human-quality speech audio is synthesised at speed of 3.7 MHz on a single Nvidia V100 GPU [24].

2.1.5 Evolution of TTS

The introduction of modern TTS technologies thus far has been short and has only concentrated on the landmark methods and models relevant for this work. Neural TTS technologies have been thriving since 2016 and there is a large amount of research released focusing on the different aspects of the pipeline, with the quality of synthesised speech steadily improving [5]. A comprehensive 2021 survey from the Microsoft TTS research team summarised the many different methods and models released since 2016, shown in Figure 2.6. While this work is not aiming to contribute to the list of methods with a new approach, it is, however aiming to contribute to the vast ecosystem of open source TTS technologies by realising the first publicly available implementation of the Capacitron model.

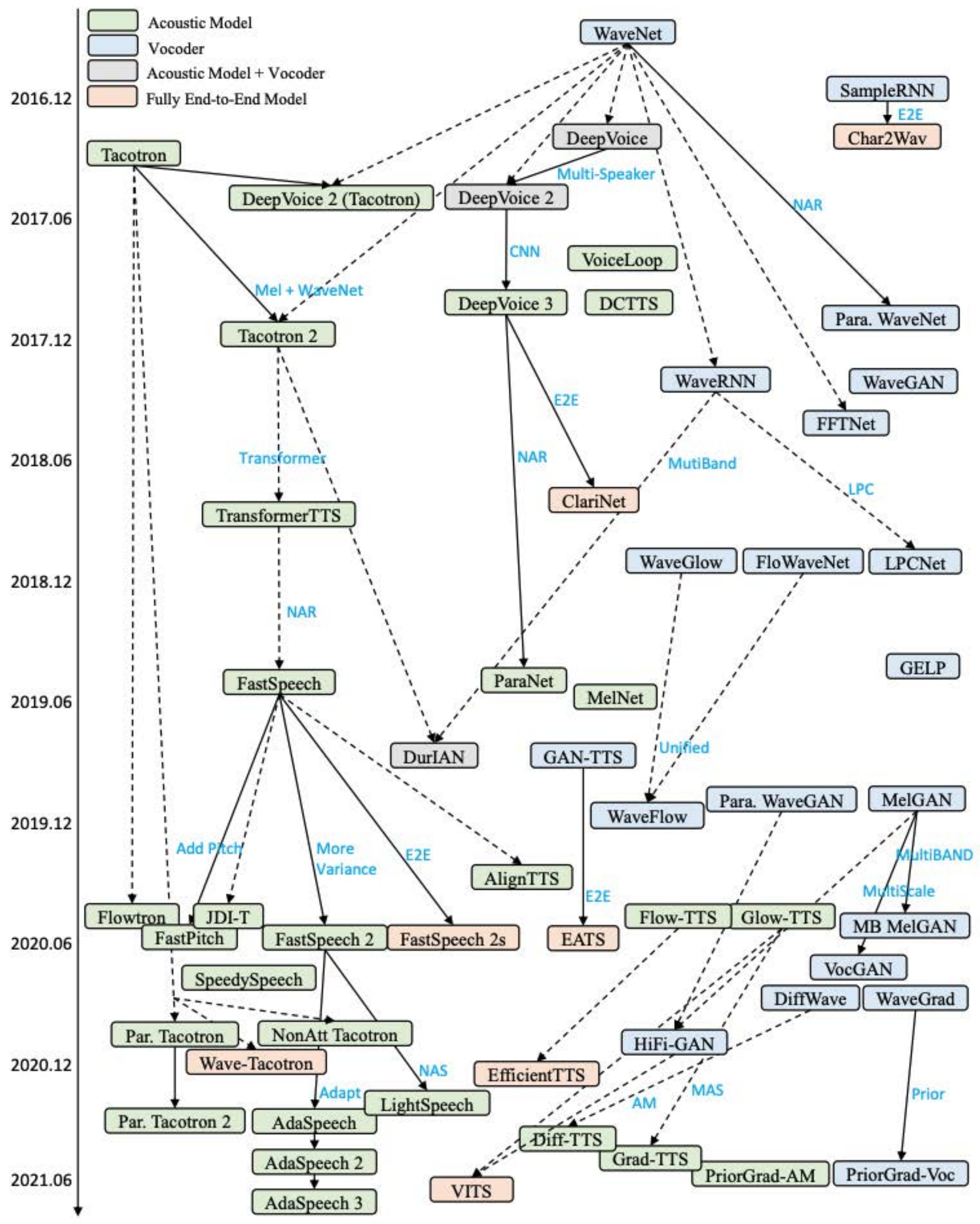


Figure 2.6: Evolution of neural TTS models [5]

2.2 Prosody Transfer and the Reference Encoder

2.2.1 Learning the Latent Embedding Space of Prosody

The current generation of prosody control implementations in the Tacotron system detailed above was established by the 2018 paper from the Google research team that presented an extension to the base speech synthesis architecture that learns a latent embedding space of prosody, derived from a reference acoustic representation containing the desired prosody [6]. Similar to Tacotron 1, this paper first introduced the general ideas of prosody transfer in an end-to-end TTS system, produced good results and laid the foundations for further research. Figure 2.7 shows the architecture of the augmented Tacotron system, where the autoregressive decoder is conditioned on the result of the reference encoder, transcript encoder and speaker embedding via an attention module [6].

Using this deterministic reference encoder to project the reference speech into a learned embedding space, this model enables prosody transfer between two speakers ("say it like this"), but does not work for transfer between unrelated sentences nor does it preserve the pitch range of the target speaker.

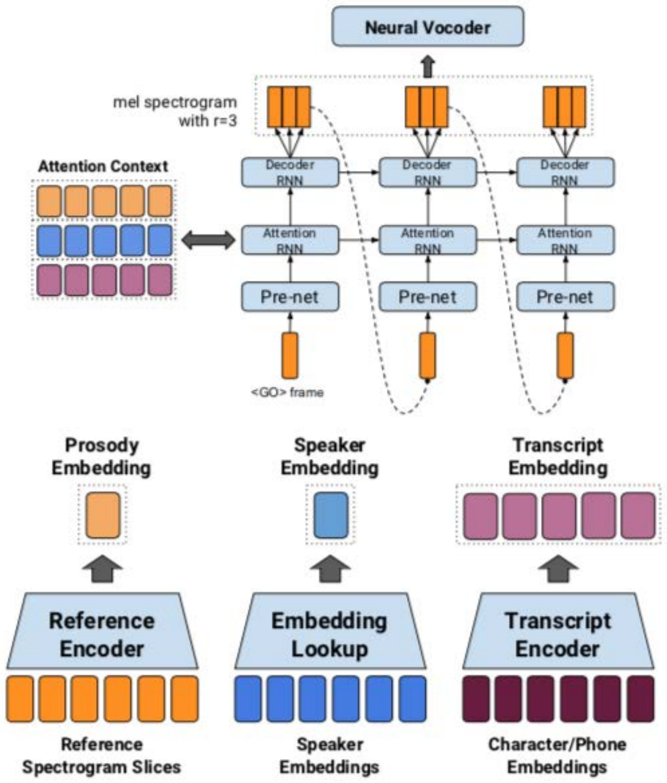


Figure 2.7: Tacotron architecture for prosody control [6]

3 Capacitron

This chapter introduces the theoretical background for the *Capacitron* model, namely the variational autoencoder neural network architecture as well as the concept of structural and variational capacity, the two variables directly controlling the dimensionality and extent of expressiveness. Then, bringing together these two concepts, the end of this chapter introduces the governing equation for this thesis, the loss function used in the neural network architecture as well as the definition for the two types of capacities investigated in this work.

3.1 The Probabilistic Model - Variational Autoencoders

In his 1867 work, *"Handbuch der physiologischen Optik"*, Hermann von Helmholtz defines a term of perceptual psychology, the *unconscious inference* (*unbewusster Schuss*) to explain various conscious, perceptual and cognitive phenomena by postulating inference-like processes that operate over unconscious representational states [25]. As Helmholtz puts it,

> *„the inferences that the perceptual system engage in ... are in general not conscious, but rather unconscious. In their outcomes they are like inferences insofar as we from the observed effect on our senses arrive at an idea of the cause of this effect. This is so even though we always in fact only have direct access to events at the nerves, that is, we sense the effects, never the external objects."* [26, p. 430]

Following this framework, we can view the human perceptual system as a statistical inference engine whose function is to infer the probable causes of sensory input [27]. When it comes to modelling such a system, however, a necessary condition for such a device is the ability to learn how to perform these inferences without requiring a teacher to label each sensory input vector with its underlying causes due to the high dimensionality of the solution space. In the case of the human perception and production of speech, the underdetermined nature of expressive speech offers an obvious candidate for the modelling of this system using a Helmholtz architecture.

In other terms, such representational learning models aim to *„identify and disentangle the underlying causal factors of the data, so that it becomes easier to understand the data, to classify it, or to perform other unsupervised tasks"* [28, p. 1]. For audio data (which due to modern NN techniques is often transformed into image data), this often means that we are interested in uncovering the “global structure” that captures the content of an image and its “style” [29].

3.1.1 Limitations of Autoencoders

Heuristic (non-variational) autoencoders explicitly compress high dimensional data into a smaller, latent dimension **z** using an encoder architecture, in our case parameterised by a neural network. This compressed data is then decoded using a neural decoder network that learns to reconstruct the latent, compressed vector into the original input during training. The term *reconstruction loss* used in this work originates from looking at the accumulated loss between the reconstructed output and the true input. A typical schema for a basic autoencoder model is shown in Figure 3.1.

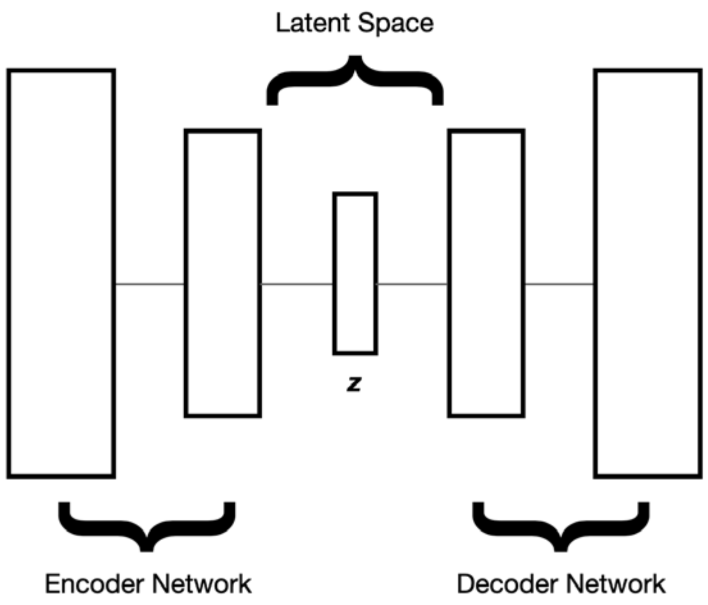


Figure 3.1: Schema of a typical autoencoder network architecture

A major limitation of autoencoders, however is that even with non-linear activation functions along the network connections, the latent variable **z** produced by the network is still a *fixed vector*. This means that during inference, a certain input to the model will produce the same output every time the model is initialised with that input. In terms of expressive TTS applications, this is an undesirable effect, since synthesising the same sentence multiple times will produce the same prosody in the output audio.

This type of set-up does not satisfy the requirement of naturalness stated earlier, since this one-to-one mapping from text to audio is not in line with the underdetermined, one-to-many nature of expressive speech, whereby some text has ultimately many different ways to be uttered [30].

3.1.2 Variational Extension to Autoencoders

The fundamental basis of the Helmholtz architecture is based on a *recognition model* used to infer a probability distribution over the underlying causes from the sensory input, and a separate learned *generative model* that is used to train the recognition model [27]. Modern implementations in machine learning interpret this recognition-generation coupling using a Bayesian probabilistic model, presented in the 2014 paper as the *variational autoencoder* architecture from Kingma and Welling [31].

Instead of mapping an input to a fixed vector, the input data is compressed down onto a *probability distribution.* In practice, this means using a parameterised neural network to output values for the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of a Gaussian normal distribution $q(\mathbf{z}|\mathbf{x})$. The fixed vector bottleneck from the basic autoencoder is now replaced by two separate vectors that we use to create a distribution. In order to capture the latent embedding $\mathbf{z}$ from this architecture, we can sample from this parameterised distribution after the input has been fed through the encoder. A schema for a simple VAE model is shown in Figure 3.2

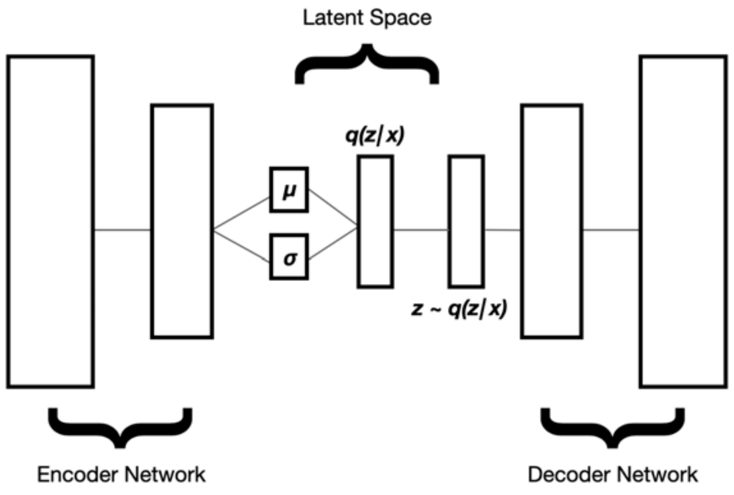


Figure 3.2: Schema of a typical variational autoencoder network architecture

Given some dataset $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ consisting of N i.i.d. samples of some continuous or discreet variable $\mathbf{x}$, we assume the data are generated by some random process, involving an unobserved random variable $\mathbf{z}$. This generative process originating from the latent space can be split into two subprocesses:

1. a value $\mathbf{z}^{(i)}$ is generated from some *prior distribution* $p(\mathbf{z})$,
2. a value $\mathbf{x}^{(i)}$ is generated from some conditional distribution, the *likelihood* $p(\mathbf{x}|\mathbf{z})$.

Given the prior belief that the probability density function of the latent variable is $p(\mathbf{z})$ and that the observations $\mathbf{x}$ have likelihood $p(\mathbf{x}|\mathbf{z})$, the true posterior distribution based on Bayes' Theorem is defined as

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})}{p(\mathbf{x})} \cdot p(\mathbf{z}), \tag{3.1}$$

with the evidence $p(\mathbf{x})$ defined as the integral of the *marginal likelihood*

$$p(\mathbf{x}) = \int p(\mathbf{z})p(\mathbf{x}|\mathbf{z})dz. \tag{3.2}$$

In order to calculate the integral introduced in Equation 3.2, we would have to marginalise over all configurations of latent variables which would require exponential time. By parameterising $p(\mathbf{x})$ and $p(\mathbf{z}|\mathbf{x})$ with neural networks with non-linear layers and parameters ϕ, this integral becomes *intractable*. We have

$$\begin{aligned} p_\phi(\mathbf{z}|\mathbf{x}) &= \frac{p_\phi(\mathbf{x}|\mathbf{z})}{p_\phi(\mathbf{x})} \cdot p_\phi(\mathbf{z}), \\ p_\phi(\mathbf{x}) &= \int p_\phi(\mathbf{z})p_\phi(\mathbf{x}|\mathbf{z})dz, \end{aligned} \tag{3.3}$$

the intractable true posterior density and true marginal likelihood respectively, parameterised by the neural network parameters ϕ. For these cases of intractability, [31] introduces a recognition model, an *approximate posterior* $q_\theta(\mathbf{z}|\mathbf{x})$ distribution to model the *true posterior* $p_\phi(\mathbf{z}|\mathbf{x})$. Here, the parameters θ are not computed from some closed-form expectation, instead the authors introduce a method for learning the recognition model parameters θ jointly with the generative model parameters ϕ.

The unobserved variables $\mathbf{z}$ have an interpretation of a latent representation, or *code* - this is why the recognition model $q_\theta(\mathbf{z}|\mathbf{x})$ can be referred to as a *probabilistic encoder*. Given a data point $\mathbf{x}$, the recognition model produces a distribution (e.g. a Gaussian) over the possible values of the code $\mathbf{z}$, from which the data point $\mathbf{x}$ could have been generated. In turn, we can now refer to $p_\phi(\mathbf{x}|\mathbf{z})$ as the *probabilistic decoder*, which, given a code $\mathbf{z}$, produces a distribution over the possible corresponding values of $\mathbf{x}$.

3.1.3 Variational Bound

The marginal likelihood of the data $p_\phi(\mathbf{X})$ is composed of a sum over the marginal likelihoods of individual data points

$$\log p_\phi(\mathbf{x}^{(1)},...,\mathbf{x}^{(N)}) = \sum_{i=1}^{N} \log p_\phi(\mathbf{x}_i), \tag{3.4}$$

which can be written as:

$$\log p_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}) = D_{KL}(q_\theta(\mathbf{z}|\mathbf{x}^{(i)}) \| p_\phi(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}; \mathbf{x}^{(i)}), \tag{3.5}$$

where the KL-divergence term on the RHS is defined as

$$D_{\mathrm{KL}}(q\|p) = \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})} \mathrm{d}\mathbf{z}, \tag{3.6}$$

and is non-negative $D_{\mathrm{KL}}(q\|p) \geq 0$. Since this measure of discrepancy between the true and approximate posteriors is non-negative and only zero if the distributions are equal, we define the second term on the RHS of Equation 3.5 to be the *variational lower bound* on the marginal likelihood of the data point $\mathbf{x}^{(i)}$, $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}; \mathbf{x}^{(i)})$. To put the lower bound nature of this equation into perspective, we can rewrite Equation 3.5 as

$$\log p_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_\theta(\mathbf{z}|\mathbf{x})}[-\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{z})], \tag{3.7}$$

from which we have the explicit definition for the lower bound as

$$\mathcal{L}_{ELBO}(\boldsymbol{\phi}, \boldsymbol{\theta}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[-\log p_{\boldsymbol{\phi}}(\mathbf{x}^{(i)}|\mathbf{z})] + D_{KL}(q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)}) \| p_{\boldsymbol{\phi}}(\mathbf{z})), \tag{3.8}$$

where the first part of the RHS is the KL-divergence term between the approximate posterior (probabilistic encoder) and the prior distributions and the second part is the expected reconstruction error of the probabilistic decoder. The authors of [32] define Equation 3.8 as the negative Evidence Lower BOund (ELBO) of the marginal likelihood of the data, where the KL term acts as a regulariser on the expected reconstruction error of the data. In variational models, the ELBO functions as the total loss for the model, where the objective is to optimise Equation 3.8 w.r.t. the generative parameters ϕ and variational parameters θ.

3.1.4 Reparameterisation Trick

As shown in Figure 3.2, the VAE model entails a sampling operation to produce the latent embedding for further processing by the decoder. A practical problem with this operation is that running backpropagation or gradient updates through a sampling node is not possible due to the fact that we would want to compute gradients on a node encompassing a random sampling operation.

To be able to run the gradients through the entire network and train everything end-to-end, [31] introduces the *reparameterisation trick*. We look at the sampling operation of the latent vector as a sum $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \epsilon$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are learnable constants of the model and ϵ is a fixed, stochastic sampling operation from a standard normal distribution $\epsilon \sim \mathcal{N}(0,1)$. This way, we are only interested in updating gradients by backpropagation on two learnable parameters and the source of randomness in the model is reduced to a simple sampling operation from a fixed distribution without any trainable weights. Figure 3.3 shows the sampling operation from the approximate posterior with and without the the reparameterisation trick.

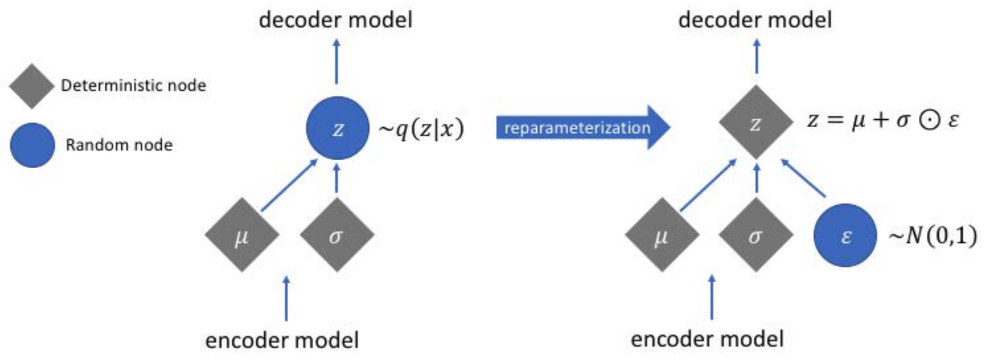


Figure 3.3: Sampling operation from the approximate posterior with and without the reparameterisation trick [7]

3.2 Tacotron 1 and Capacitron VAE

In this Section, the VAE architecture detailed above is connected into the general frame of TTS, with the parallels between Equation 3.8 and the basic Tacotron 1 decoder presented. Additionally, the connection between the KL-term and the mutual information between the data and the latent representation is explained, to give the reader the framework of how the Capacitron VAE architecture works to encode prosodic information. At the end of this Section, these pieces of information are compiled to define the governing loss function for the Capacitron method.

3.2.1 Likelihood and the Tacotron Decoder

Existing sequence-to-sequence TTS models start by defining the *teacher forced* reconstruction loss,

$$L\left(\mathbf{x}, \mathbf{y}_{\mathrm{T}}\right) \equiv -\log p\left(\mathbf{x} \mid \mathbf{y}_{\mathrm{T}}\right)=\left\|f_{\theta}\left(\mathbf{y}_{\mathrm{T}}\right)-\mathbf{x}\right\|_{1}+K, \tag{3.9}$$

where $\mathbf{x}$ is an audio spectrogram, $\mathbf{y}_T$ is the input text, K is a normalisation constant and $f_\theta(\cdot)$ is a deterministic function that maps the text inputs to spectrogram predictions, in our case modelled by a neural network. Teacher forcing implies that $f_\theta(\cdot)$ is dependent on $\mathbf{x}_{<t}$ when predicting spectrogram frame $\mathbf{x}_t$ [8].

Heuristic (non-variational) end-to-end approaches to prosody and style transfer augment Equation 3.9 with a deterministic reference encoder, $g_e(\mathbf{x})$

$$L^{\prime}\left(\mathbf{x}, \mathbf{y}_{\mathrm{T}}\right) \equiv -\log p\left(\mathbf{x} \mid \mathbf{y}_{\mathrm{T}}, g_e(\mathbf{x})\right)=\left\|f_{\theta}\left(\mathbf{y}_{\mathrm{T}}, g_e(\mathbf{x})\right)-\mathbf{x}\right\|_{1}+K, \tag{3.10}$$

where the model's general decoder takes into account a *reference embedding* produced by the reference encoder, paramaterised by a separate neural network.

To make the connection to probabilistic models, we recognise that the absolute value difference between the true and predicted spectrograms - known as the *L1 loss* - functions as a stand in for the negative log likelihood minimisation process detailed in Section 3.1.3. When looking at the model optimisation from a probabilistic perspective, we aim to perform maximum likelihood estimation on our input data $\mathbf{x}$, given the transcript $\mathbf{y}_T$. With this, we have made the connection between the VAE and TTS architectures in terms of the LHS of Equation 3.15. We recognise, that the variational reconstruction loss defined above is simply the loss of a standard Tacotron decoder extended with a probabilistic input.

3.2.2 Mutual Information and the KL Term

Recent work [33] [34] [35] has shown that the KL term in Equation 3.8 provides an *upper bound* on the mutual information between the data $\boldsymbol{x}$ and the latent embedding $\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x})$ [8]. Firstly, we define the KL term averaged over the data distribution:

$$\begin{aligned} R^{\mathrm{AVG}} &\equiv E_{\mathbf{x} \sim p_D(\mathbf{x})}\left[D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))\right], \\ R &\equiv D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})), \end{aligned} \tag{3.11}$$

where R is the KL term in Equation 3.8.

Next, we define the *representational mutual information* as

$$
\begin{aligned}
I_q(\mathbf{X};\mathbf{Z}) &\equiv \iint p_D(\mathbf{x}) q(\mathbf{z}|\mathbf{x}) \log \frac{q(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} d\mathbf{x} d\mathbf{z}, \\
I_q(\mathbf{X};\mathbf{Z}) &\equiv E_{\mathbf{x}\sim p_D(\mathbf{x})} \left[D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x}) \| q(\mathbf{z})) \right],
\end{aligned}
\tag{3.12}
$$

where $q(\mathbf{z}) \equiv E_{\mathbf{x}\sim p_D}(\mathbf{x}) q(\mathbf{z}|\mathbf{x})$ is the *aggregated posterior.* Putting together Equation 3.11 and 3.12, we have

$$
R^{\mathrm{AVG}} = I_q(\mathbf{X};\mathbf{Z}) + D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z})), \tag{3.13}
$$

where $D_{\mathrm{KL}}(q(\mathbf{z}) \| p(\mathbf{z}))$ is the *aggregate KL*, a measure for the mismatch between the aggregated posterior and the prior. From the non-negativity of the KL divergence and from Equation 3.13 it follows:

$$
I_q(\mathbf{X};\mathbf{Z}) \leq R^{\mathrm{AVG}}. \tag{3.14}
$$

This derivation is expanded in Appendix B.

Controlling the KL Term

A desirable feature of VAEs is to be able to control the KL-term defined in Section 3.1.3. By manipulating the KL-term, one can control the upper bound on the mutual information between the data $\mathbf{x}$ and the latent embedding $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$ [35]. In terms of modelling the latent space of the input audio's prosody, the term's manipulation translates to the ability of specifically targeting how much of this mutual information we want our model to incorporate into itself, giving us tight control over the prosodic information content *capacity.*

Controlling this KL term has been attempted by various approaches. The *beta-VAE* method [36] defines a varying weight on the KL-term, β, while other methods [35] [37] penalise the KL-term's deviation from a specific target value. Bringing together these two approaches, the Capacitron method aims to find a way to smoothly optimise for a *specific bound on the embedding capacity*, by adapting a Lagrange multiplier-based optimisation approach from [38].

3.2.3 Model Loss

The main model loss for the paper is bringing together these concepts from above, defining a modified ELBO based on Equation 3.8:

$$\min_{\theta} \max_{\beta \geq 0} \left\{ E_{\mathbf{z} \sim q_\theta(\mathbf{z}|\mathbf{x})} \left[-\log p_\theta \left(\mathbf{x} | \mathbf{z}, \mathbf{y}_\mathrm{T} \right) \right] + \beta \left(D_\mathrm{KL} \left(q_\theta(\mathbf{z}|\mathbf{x}.\mathbf{y}_\mathrm{T}) \| p(\mathbf{z}) \right) - C \right) \right\}, \tag{3.15}$$

where θ are the model parameters.

The first term of Equation 3.15 is the expected reconstruction loss of the generative model, where we use the basic Tacotron L1 decoder loss as a stand-in for the negative log likelihood of a generated spectrogram $\boldsymbol{x}$ given latent embedding $\boldsymbol{z}$ and text $\boldsymbol{y_T}$.

The second term of Equation 3.15 is the augmented KL-term calculation between the approximate posterior and the prior, including the automatically tuned Lagrange multiplier-like constant β and the capacity limit C. We constrain β to be non-negative by passing an unconstrained parameter through a softplus non-linearity, which makes the capacity constraint a limit rather than a target [8]. To gain intuition about how this double optimisation process is working, Appendix A offers a descriptive summary.

In previous works [39] [32], the approximate variational posterior has the form $q(\boldsymbol{z}|\boldsymbol{x})$. which matches the form of the true posterior for a simple generative model $p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$. However, for the conditional generative model used in TTS, $p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{y}_\mathrm{T})p(\boldsymbol{z})$, it is missing conditional dependencies present in the true posterior $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}_\mathrm{T})$ [8].

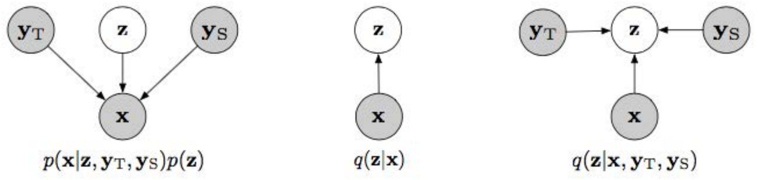

$\mathbf{y}_\mathrm{T}$
$\mathbf{z}$
$\mathbf{y}_\mathrm{S}$
$\mathbf{x}$
$p(\mathbf{x}|\mathbf{z}, \mathbf{y}_\mathrm{T}, \mathbf{y}_\mathrm{S})p(\mathbf{z})$
$q(\mathbf{z}|\mathbf{x})$
$q(\mathbf{z}|\mathbf{x}, \mathbf{y}_\mathrm{T}, \mathbf{y}_\mathrm{S})$

Figure 3.4: Adding conditional dependencies to the variational posterior [8]

Figure 3.4 shows the visual representation of this concept. Shaded nodes indicate observed variables. The true generative model is shown on the left. In the center, the variational posterior missing conditional dependencies present in the true posterior is shown. Lastly, on the right, the variational posterior that matches the form of the true posterior is shown. In practice, this means that we inject extra information about the text into the variational encoder architecture, to account for an extra conditional input to this module.

3.3 Capacity

In these heuristic models, the choice of architecture for the reference encoder defines the transfer characteristics of the model, which offers a trade-off between transfer *precision* and *generality*. Setting the *variational capacity* of the reference encoder model high would prioritise precision and are better suited for prosody transfer to similar text, while lower capacity embeddings prioritise generality and are better suited for text-agnostic style transfer [8]. In terms of the Capacitron method, we can distinguish between two types of capacities in the model. One of the pillars of this work is to investigate the relationship between these two capacities further than in the original method, to see how these two variables influence certain aspects of model evaluations. In the following, these two types of capacities are defined, while their influence on the subjective evaluation of the models is discussed more in Chapter 5.

Variational Capacity

Also referred to as information capacity, the variational capacity is defined by the constant C in Equation 3.15. This scalar value is the one directly influencing the bound on the mutual information, by limiting the KL term to a desired value.

Structural Capacity

Simply put, the structural capacity of the VAE architecture can be easily adjusted by changing the dimensionality of the latent embedding $\boldsymbol{z}$. Setting this dimensionality forces the reference encoder to translate the latent space onto a fixed tensor, whose size can be manipulated to control how big the network's variational embedding should be.

3.4 Practical Use of the VAE Architecture

The architecture detailed above gives us 3 different possibilities to utilise the VAE structure in terms of TTS synthesis inference.

3.4.1 Posterior Sampling

By sampling from the posterior distribution, we can use an arbitrary sample to create a reference embedding from this sample that we feed to the model's decoder together with the text to synthesise. This gives us controllable *prosody transfer* from a reference utterance onto a synthesised one. We have the choice to synthesise the same text as in the reference (same text prosody transfer) or infer a text prompt different to the one spoken in the reference (inter text style transfer).

3.4.2 Prior Sampling

By sampling from the prior distribution, we sample from the learned prosody space of the model. Since the model has optimised the approximate posterior and prior to be in a certain divergence (limited by the variational capacity) to each other, the deterministic decoder has learned how to interpret samples from a standard Gaussian. Therefore, when we randomly sample a latent embedding from this distribution, it will give us realistic and true prosody, learned from the input data's latent space. This means that by not providing a reference to the Capacitron model, it will yield different prosody every time the synthesis is run on a prompt. This source of randomness and expressiveness is a great contribution of this method, since vanilla Tacotron systems function in a deterministic way, whereby a certain text prompt will always yield the same synthesised utterance.

4 Development Method

This Chapter offers the full technical report on the development process, including the hardware description; the review on the acquisition and implementation of the dataset used for the models; the programming timeline as well as individual module descriptions and finally the practical accounts of the model trainings.

4.1 Hardware

The main hardware used for the development and training of the models is a custom-built PC sponsored by *why do birds GmbH*. Through rigorous research about the importance of individual components, the computer has been built with the aim of training models for high fidelity TTS synthesis. Synthesising speech with high frequency content means being able to fit the training data at higher sampling rates into the computer's GPU, where most of the parallelised tensor operations are taking place. Moreover, a higher capacity GPU also enables training faster because we can fit a higher batch size into one training iteration step. For this reason, a larger than average VRAM capacity for the GPU was chosen. Specifications for the most important components are listed in Table 4.1.

Hardware	**Type**	**Specification**
CPU	AMD Threadripper	3.5 GHz, 12 Core
GPU	Nvidia Titan RTX	24 GB VRAM
Memory	Kingston DDR4	64 GB
Storage	Samsung NVME SSD	1 TB

Table 4.1: Development and Training Hardware Specifications

4.2 Dataset

4.2.1 Acquisition

The dataset used for the thesis project is a collection of high fidelity audio-books provided by *The Voice Factory* and *Lessac Technologies, Inc* for the 2013 *Blizzard* speech synthesis challenge [40]. The dataset includes approximately 300 hours of chapter-sized mp3 files as well as approximately 19 hours of non-compressed wav files from a single speaker, Catherine Byers. The dataset contains various audio books from many different genres, which makes it a great resource for expressive TTS research [8] [6] [41] due to the richness of prosody in the audio files and its open license for research purposes.

4.2.2 Processing

Modern TTS model training processes work with short, utterance-length segmented audio and corresponding text, where normally a sentence or a part of a sentence is counted as one datapoint in the structure. The original *Blizzard 2013* dataset's high fidelity audio samples were only split into chapters, so a segmentation task needed to be carried out in order to split chapter-length audio files into shorter snippets. For this task, the open source segmentation data from Nicolas Müller from the Fraunhofer Institute [1] has been used, which splits the audio files of the chapters into short segments based on openly available transcripts of the books. Using the *Montreal Forced Aligner* [2], the segmentation split the sentences on longer pauses and removed leading and trailing silences.

During the development process, many different trained models were producing faulty speech, so as a cautionary measure about the accuracy of the segmented audio snippets, a subjective evaluation of randomly sampled datapoints has been carried out. During this process, 50 audio files have been randomly selected and subjectively compared with the corresponding transcript. This process has been repeated 3 times. Results from this subjective sampling process is summarised in Table 4.2.

Iteration	**Error Rate**
1	0%
2	8%
3	10%

Table 4.2: Percentage of misaligned or incorrect audio-transcript data pairs from 3 iterations of sampling 50 random utterances from the data

[1] github.com/mueller91/tts_alignments
[2] montreal-forced-aligner.readthedocs.io/en/latest/

The results from this test prompted further processing and cleaning of the data to get rid of misaligned and incorrect utterance-transcript pairs. For this purpose, all segmented audio utterances were run through an open-source, pre-trained STT model[3] to transcribe all audio files to text. This text afterwards has been compared with the transcript from the existing dataset and a similarity score has been assigned to each utterance. In order to define the threshold similarity score - between the ground truth and synthesised text - under which datapoints would be discarded, another subjective individual listening test of randomly selected datapoints was carried out. This step resulted in discarding 10% of the original segmented dataset, resulting in 93,074 training utterances totalling to 85 hours of high fidelity audio. Figure 4.1 shows the distribution of the dataset transcript lengths, while Figure 4.2 shows the mean audio duration compared to the text length.

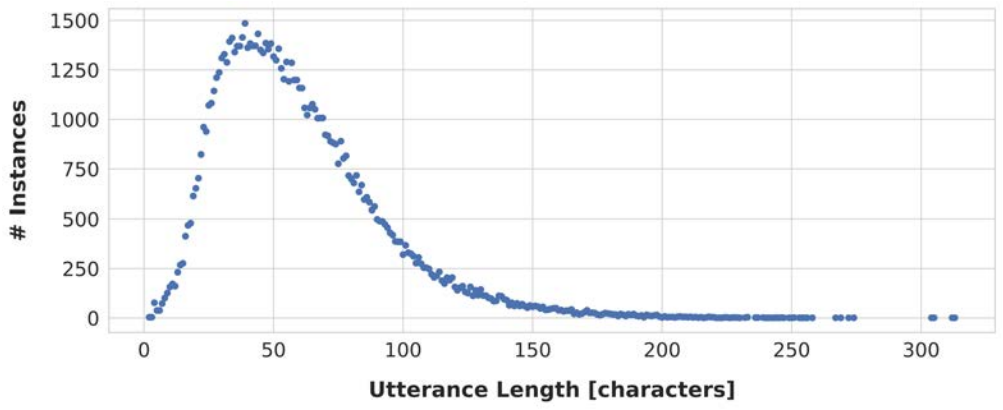


Figure 4.1: Distribution of utterance lengths in the final, untrimmed dataset

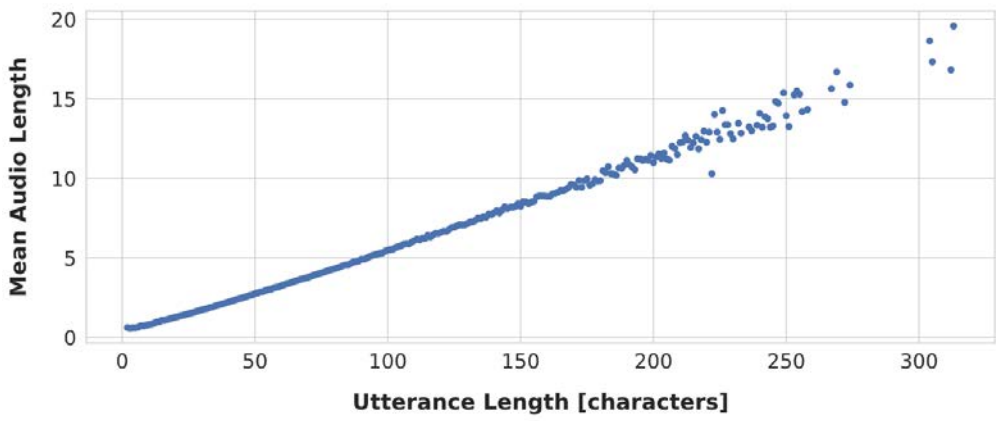


Figure 4.2: Mean audio duration in seconds vs. Text length

[3]github.com/coqui-ai/STT

In order to be able to maximise the GPU's VRAM capacity, the final training data has also been trimmed using the `max-seq-len` flag in the data-processing pipeline to exclude long utterances. Using `110` for this setting essentially excludes all utterances with transcript character lengths higher than 110 - translating to an average audio length of around 5 seconds. This trimming procedure speeds up the training process because by zero-padding shorter utterances inside the batches of data including these longer utterances, there is valuable VRAM capacity consumed that would result in decreasing the batch size. By minimising the utterance length, less space is wasted on long, zero-padded instances which in turn enables the use of a higher batch size, therefore quicker and more efficient training. The shorter training samples also do not degrade inference synthesis quality, because the autoregressive attention mechanism is able to infer longer utterances than what has been seen during training [41]. This operation is also in line with the dataset processing used in [8]. After trimming, the effective dataset contained 83,366 utterances, totalling to 71 hours at 24kHz sampling rate. Figure 4.3 shows the distribution of the text lengths from the final trimmed dataset. The distribution resembles a normally distributed dataset which is a desirable attribute for TTS synthesis [42].

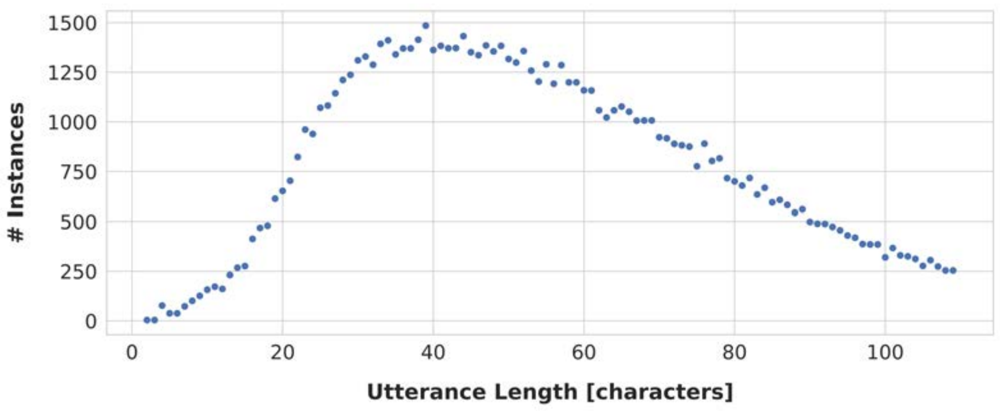


Figure 4.3: Distribution of utterance lengths in the final, trimmed dataset

4.3 Programming

This sections details the software development process and methodology, including the description of multiple attempts of implementing the *Capacitron* model in different environments.

4.3.1 Environment and Tools

Similarly to other fields in ML, there is a vibrant open source community around TTS synthesis, with many researchers releasing code for their work and other students, industry members or enthusiasts implementing methods without officially released code. In order to write the first open implementation of the *Capacitron* model, a fundamental base TTS code repository needed to be found that includes the overall structure of the model, upon which changes and new modules could be built. Specifically, repositories implementing the base reference encoder architecture detailed in [6] and [43] were searched for.

At the time of development of this project, the two major ML frameworks used for GPU accelerated deep learning are *Tensorflow* [4] and *PyTorch* [5]. Both are open source libraries utilising a powerful frontend python programming environment and a C++/CUDA backend for computation. While Tensorflow version 1 utilises a static computational graph (*define and run*) to compute the gradients of the DNN nodes, PyTorch uses a dynamic graph (*define by run*) for this process resulting in a development process more similar to vanilla python code and development practices.

Based on limited previous experience in the *Tensorflow 1* framework as well as the subjective evaluation of samples produced by the different repositories, the decision has been taken to start the development based off of an open-source *Tensorflow 1* repository [6] of the *Global Style Tokens* [43] implementation, which includes a modified version of the fundamental autoregressive TTS Tacotron 1 model [3], including the principal reference encoder.

Regarding implementation details and questions about the model, contact has been made with the main author of the paper from Google, Eric Battenberg [7]. Eric was kind enough to answer emails and schedule calls to go through the theoretical background and specific implementation details about the model. During one of these calls, he also shed light on an essential implementation detail that wasn't fully described in the paper and what turned out to be a very important feature of the model that was omitted from previous open source implementations of the reference encoder architecture. This aspect of the model is detailed in the next section.

Due to increasing difficulties with the programming environment, inconsistencies in the base repository as well as the lack of support and maintenance for the codebase, 10 weeks into working with the mentioned *Tensorflow 1* repository, the decision has been made to pivot to a new implementation written in *PyTorch* [8]. This repository is developed and maintained by the former Mozilla AI team who recently established a new open source software company, Coqui AI [9]. The code base has a vibrant open source community around it with many university researchers, commercial engineers and hobbyists among the developers, some of whom directly and indirectly contributed to this project with discussions and code reviews.

[4]Tensorflow.org/
[5]pytorch.org/
[6]github.com/syang1993/gst-tacotron
[7]github.com/ebattenberg
[8]github.com/coqui-ai/TTS
[9]coqui.ai

The section below offers a comprehensive report on the chronological development methodology for both attempts.

4.3.2 Methodology

Having decided on the initial code-base to implement the Capacitron method into, contact has been made with the main author of the paper, Eric Battenberg from Google to clarify questions and uncertainties about the implementation plans. Eric was kind enough to go through the main theoretical principles that form the basis of the model for more intuition about the system. Extending on some of the slightly vague notions in the paper, he has explained certain modules' contribution to the method in detail. These sessions were very helpful in understanding the specific blocks that needed to be coded to make the method work.

Before starting to code, a road-map of milestones have been defined, in order to implement smaller parts of the model using agile development methodologies. The way these milestones have been defined was by comparing the available code and the proposed method to find refined features, building blocks and the logical infrastructure that were not present in the base implementation. These milestones are briefly detailed in the next section with the final PyTorch implementation attached in Appendix D.1.

Having implemented most of the modules into the Tensorflow 1 infrastructure mentioned above, the first attempts at training the model were launched. These early attempts mainly served the purpose of finding run time bugs in the newly implemented modules' code. This turned out to be a lengthier process than expected because of the static nature of Tensorflow 1. In order to check the required shapes of initialised tensors and modules, the shapes of previous outputs needed to be propagated all the way to the entry point of the code to evaluate these in the static graph. This motive was also generally the case for debugging tensors and logic in the code after the models started to train without them breaking down due to shape mismatches or other static bugs. The static nature of the computation graph made it difficult and timely to experiment in the code and a significant amount time has been spent on trying to examine shapes, logic and general behaviour.

Once these first difficulties have been tackled, the model was finally training without breaking down due to static errors in the code. The next step in examining the behaviour of the model was through the *Tensorboard* interface, a visualisation toolkit for ML model trainings, where the developer can define parameters and plots to be printed onto a visual console. This tool enables developers to see how different variables in the model evolve over time and provides a good visual representation of the performance of the model. An early print of the console is show in Figure 4.4.

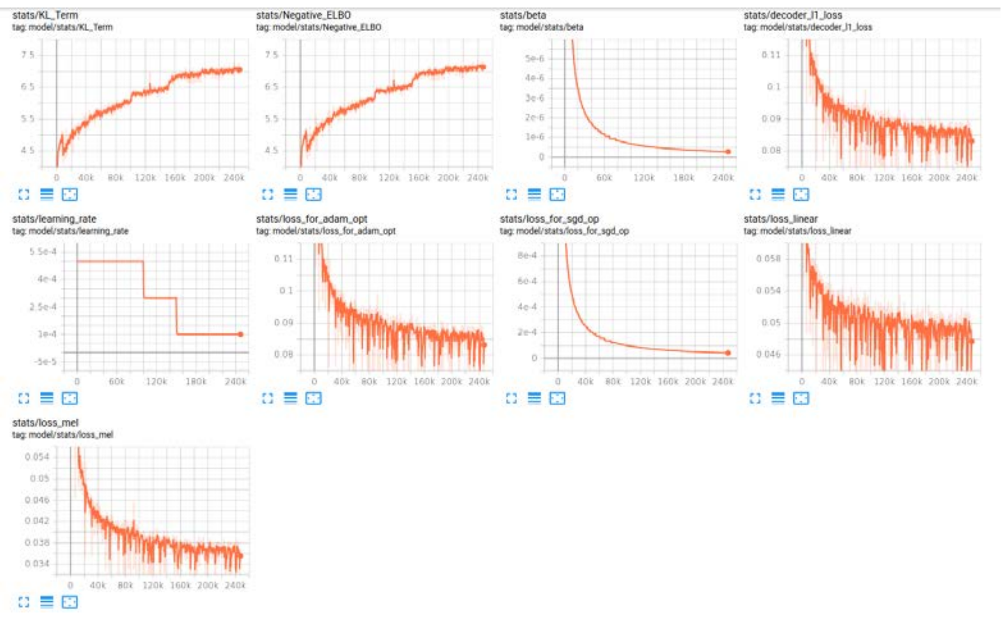


Figure 4.4: Tensorboard print of an earlier model training

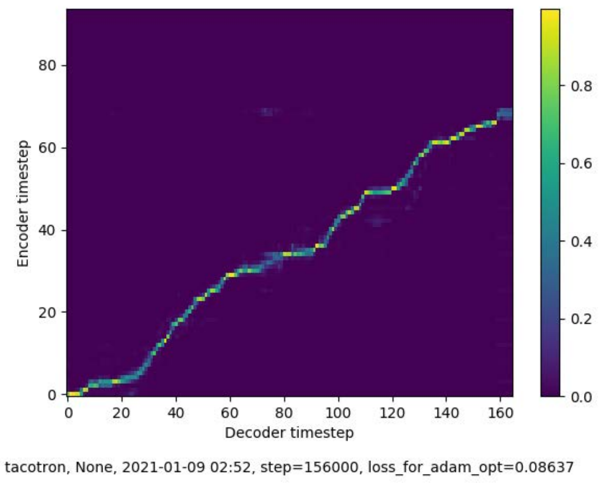


Figure 4.5: Tensorboard alignment plot print of an earlier model training

Another useful tool for examining TTS model performance is through the *alignment plots*. Alignment plots show the performance of a sequence-to-sequence encoder-decoder network, in which the encoder learns to encode a variable-length sequence into a fixed-length vector representation and the decoder learns to decode a given fixed-length vector representation back into a variable-length sequence [44].

„A potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus" [45, p. 1] - this limitation is aimed to be solved by *attention*, a separate model that searches for a set of positions in a source sentence where the most relevant information is concentrated to account for this information (so-called *context vectors*) when the model is predicting new units in the output sequence [45]. While comprehensively discussing the topic of attention mechanisms is beyond the scope of this work, it needs to be mentioned that the attention mechanism is an essential part of modern NLP and TTS systems and the alignment plots are a standard way of examining a model's performance. An alignment plot from an early training is seen in Figure 4.5 where the nonlinear lines show signs of a not-so-well performing (attention) model. A perfect alignment plot between the encoder and the decoder time-steps would be a diagonal line that would demonstrate a one-to-one mapping between encoder input and decoder output.

Upon examining the variable values and alignment plots on the Tensorboard visualisations, it appeared that some essential parameters were significantly away from the ideal values. These models naturally also produced very poor speech - the Tensorboard also offers the playback of synthesised samples from certain model checkpoints. These samples were also constantly monitored to try to evaluate whether the model is doing what was intended. A factor that made the method experiments difficult and slow was that such TTS models need a significant time to train. While the Capacitron paper defined 300,000 steps with `batch_size=256` for the fully trained models, an essential piece of information was provided by Nicolas Müller from the Munich Institute of Technology: after 20,000 steps, one should already hear and see if the model is producing the intended speech. Naturally, synthesis and model stability significantly increases with more training, however for the sake of experimentation, trial models were only trained until 20,000 steps to save development time. On the above described on-site hardware, this translated to around 8 hours for a single model to train to a point where it can be remotely decided if the model failed or not.

Due to the black-box nature of ML and especially TTS - where large models with millions of parameters and many layers of connections are aiming to produce speech -, the debugging of the models was a slow and tedious process where often no improvements were made for weeks. To ask for help and guidance, the Tensorboard values have been sent to the original author of the paper. Eric replied back with helpful comments and even commented on some pseudo-code that has been sent to him for sanity-checks. During one of the discussions he has mentioned an important detail that wasn't explicitly stated in the paper, however was essential to get the model working. This part of the model is detailed in the next section as *Convolutional masking.*

After many failed attempts to get the model working, the decision has been made to abandon the Tensorflow 1 implementation. The reasons for this decision were as follows:

- slow and inefficient debugging and experimentation,
- poor implementation of the base model,
- lack of up-to-date libraries and technologies,
- lack of a more modern attention mechanism.

4.3.3 Implementation Modules

Most of these modules' corresponding code implementation is found in Appendix D.1. When not stated, the implementation is found as an attachment to this work and in the open source repository [10].

Step dependent learning rate decay function

A training feature not present in the Coqui AI code-base was a step-dependent learning rate decay function, where the engineer could hard-code learning rate values and the corresponding training step change thresholds. This general training utility tool has been implemented to replicate the decaying learning rate behaviour introduced in [8].

β initialisation

The Lagrange-like multiplier constant needed to be initialised in the Capacitron main class shown in Appendix D.1.1 to equal to 1 at the beginning of training, after it has passed through a `softplus()` nonlinear function.

Conditional text input

While the original authors of the Capacitron method experimented on models with or without a conditional text dependency in the variational posterior, this work automatically assumes the use of this conditional text input, since it greatly improved the synthesised samples from these models. The module - shown in D.1.3 - takes a variable length tensor of characters and maps them onto a fixed size text embedding that is later concatenated to the output of the reference encoder.

[10]github.com/a-froghyar/Capacitron

GMM attention

An important part of the model was the use of the appropriate attention mechanism as mentioned in [8]. While earlier model training attempts tried to use a different attention module, it turned out that the Gaussian Mixture Model attention defined in [46] was essential to get the model working. The Coqui TTS repository used for the base TTS infrastructure had this module already built-in.

Reference encoder variations

While the Coqui TTS repository already included the reference encoder architecture defined in Global Style Token Tacotron model [43], some variations to the module's code needed to be applied to make it applicable for the Capacitron method. The biggest change here was to substitute the `GRU()` layer with a `LSTM()` layer and accordingly change the output fetching routine to account for the different outputs of the `LSTM()` layer. The code for this module can be found in Appendix D.1.2.

Post reference encoder MLP

To produce the parameters of the diagonal Gaussian which we sample from to produce a reference embedding during training, the output of the reference encoder (and the vector describing the text) is passed through an MLP with 128 `tanh` hidden units. The module is shown in Appendix D.1.3.

Prior and posterior distributions

Both simple distributions have been modelled with the simple `MultivariateNormal` distribution object from PyTorch. While the prior is a fixed distribution with `mean=0` and `standard_deviation=1`, the posterior's parameters are learned parameters, as shown in Appendix D.1.1.

Loss function and KL divergence

In order to prepare the proper values for the loss optimisation, a major change to the standard Tacotron 1 loss function needed to be implemented. As described in Section 3.2 and in Equation 3.15, the standard $l1$ decoder loss functions as a stand-in for the variational negative log likelihood term show in Equation 3.9. An important detail here is that since we're marginalising over all the latents sampled from the posterior distribution to get the expected value of the negative log likelihood, the stand-in $l1$ loss is only valid if the absolute value calculation also includes a summing operation.

This way, the decoder loss on the LHS of Equation 3.15 will be in the appropriate scale of values for the KL Divergence term on the RHS of the same equation in order to optimise the prosody embedding space to a certain structural capacity limit. The code for this module is shown in Appendix D.2.

Double optimisation

A major part of the implementation centred around the double optimisation routine detailed in [8]. The loss function defined in Equation 3.15 was to be optimised by two separate processes. The main `ADAM` optimiser is used for the all model parameters excluding the single scalar parameter β, while a separate `SGD` optimiser is used only for β. The way this double optimisation was carried out was by splitting the loss function into two separate assignments which each only included the parameters that needed to be minimised, utilising PyTorch's `.detach()` method to detach parameters from the automatic differentiation engine. The implementation of this routine is found in Appendix D.2.

Gradient clipping

Similar to the previous implementation aspect, the repository's handling of gradient clipping also needed to be adjusted. Since gradient clipping was normally applied to all parameters in the code-base, an explicit splitting of parameters to be clipped was defined. This meant that on all but the parameter β were to be gradient clipping applied.

Convolutional masking

One of the decisive aspects of the implementation is a not explicitly mentioned feature of the convolutional network inside the reference encoder. Vanilla convolutional networks normally take an input and compress it down to some desired output using different values for the `kernel_size` and `stride` parameters that define the receptive field and the sliding of the convolutional filters. These basic networks work with fixed input sizes - usually static sized images - on which they perform the convolution steps to reduce and compress the data.

A unique feature of working with speech data, is that the audio being fed to the networks has varying length. Withing a training batch, these varying length single samples are ordered in descending order so that similar length examples are following each other, however the longest sample in the batch is always defining the ultimate size of the input tensor withing a specific training step. Other, shorter samples are naturally zero-padded to match the size of the longest sample, so that the input tensor has a uniform shape.

Inside the `ReferenceEncoder` module, this input tensor with the zero-padded instances is passed through a stack of 6 convolutional layers, with 3x3 filters, 2x2 stride and batch normalisation. The 6 layers have 32, 32, 64, 64, 128 and 128 filters respectively.

In order to demonstrate why this operation needs extra attention, Figure 4.7 shows a simple 1D example of an input tensor with `input_length=5`. The convolution module from PyTorch enables specifying additional zero padding on different axes of the input tensor, in this case, a zero padding of 2 is shown on the `width` dimension. This is to ensure that the receptive field of the convolutional filter properly includes the information on the edges of the input tensor. With `filter_width=3` and `stride=2`, we see that the filter takes 3 input values to convolve these values into a single output value. The filter then slides 2 values along the width axis to include the next triplet of values and so on. This singular convolution operation now compressed down the input tensor from `input_length=5` to `output_length=4`.

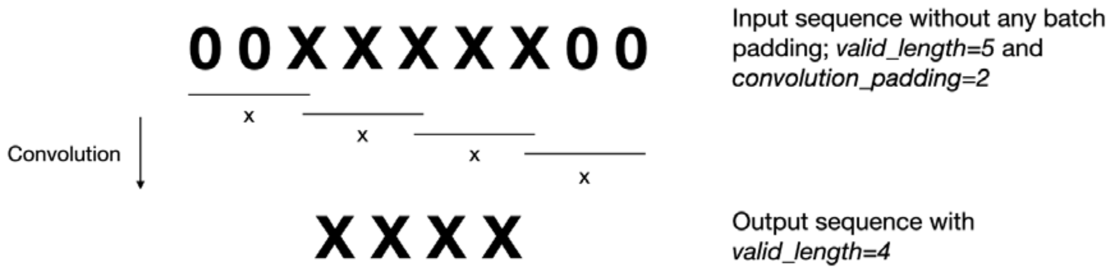

0 0 X X X X X 0 0
Input sequence without any batch padding; *valid_length=5* and *convolution_padding=2*
x
x
x
x
Convolution
X X X X
Output sequence with *valid_length=4*

Figure 4.6: 1D convolution example with `kernel_size=3`, `stride=2`, `padding=2` without zero-padding from data loader batching

Now, the previous case clearly didn't need to be treated any differently than normal convolution operations - PyTorch has all the built-in helper functions and methods to account for the data compression. In our special case of working with variable length audio, however, Figure 4.7 shows where more consideration for this operation is needed. In this case, the same `valid_length` example is in a different batch, where it is no longer the longest sample, so there is already some zero padding applied to the end of the signal. In this specific case, the input is zero padded already from the data-feeder network by two zero values. With the same convolution operation as described above, the output of this convolution is no longer length 4, but 5. The last filter is applying convolution on zero values, however because of the bias, these terms are not going to be zero after the convolution step. Even if they are small, in our case of a stack of 6 such convolutions, the amount of invalid information compounded essentially makes this convolutional net unable to properly process the input data and to produce meaningful outputs during test time. The solution for this problem is to calculate the `valid_length` of each instance in the batch after a single convolution pass and to mask off all invalid values before feeding this output back into the next convolution. With this convolutional masking, the variable length input audio is properly downsampled into a valid, compressed representation.

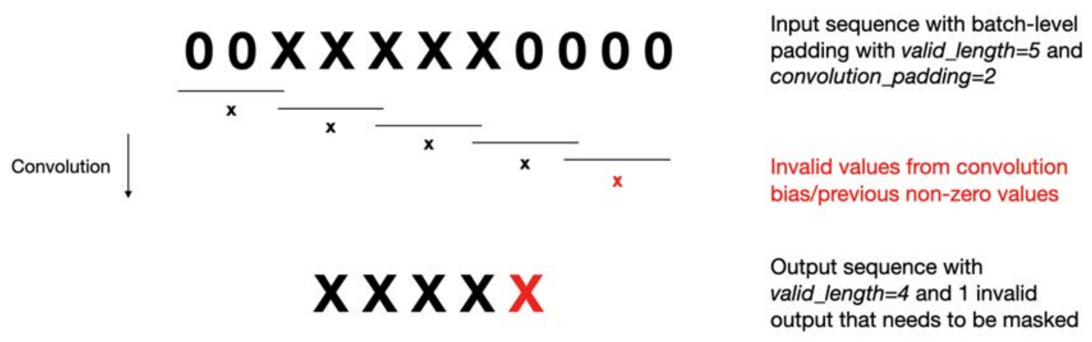

0 0 X X X X X 0 0 0 0
Input sequence with batch-level padding with valid_length=5 and convolution_padding=2
x
x
x
x
x
Convolution
Invalid values from convolution bias/previous non-zero values
X X X X X
Output sequence with valid_length=4 and 1 invalid output that needs to be masked

Figure 4.7: 1D convolution example with `kernel_size=3`, `stride=2`, `padding=2` with zero padding from data loader batching

This implementation detail turned out be an essential aspect of the Capacitron method - models trained without this convolutional masking completely failed to learn the latent space of the input data and produced unintelligible audio during test time. While the papers using this reference encoder architecture do not explicitly mention this trick, the main author of the Capacitron method shared the importance of this feature during one of our sessions.

4.4 Model Trainings

This section details the specific models trained for the model evaluations detailed in the next chapter. As mentioned in the previous sections, the training of such TTS models take a long time. The models from [8] have been trained 300,000 steps with `batch_size=256`. Through iterative experimentation and optimisation, the locally available hardware described in section 4.1 managed to start the training process with the same `batch_size`, with the GPU's memory capacity averaging to 98-99% during training. However, with this set-up, a single training step averaged to ~1,85s, meaning that a single fully fledged model training as described above would take around 6,5 days. In total, 8 models were planned for the listening evaluation, which would total to around 52 days (excluding around 8 days spent on training the acoustic model detailed below) of non-stop GPU time. Due to time constraints and simply the fact that running the hardware on full capacity for such a long time is very expensive in terms of consumed electricity resources, the decision has been made to split the model training processes into two, with 6 models training only for 100,000 steps and 2 models training for 300,000 steps. The reasoning behind how the reduction of training steps has been carried out is detailed below. This decision then halved the amount of time spent on training the spectrogram models, totalling the GPU time to around 26 days. The additional training time for the acoustic (vocoder) model is detailed below.

Splitting the model trainings as well as the listening evaluation tests into two, the first phase of trainings focused on the varied capacity models for assessing the influence of varying structural and variational capacities on the prosody transfer performance of the models. Afterwards, the second phase of trainings focused on models for the evaluation of naturalness and quality. If not stated otherwise, all other model and training parameters have been kept constant through all model trainings detailed below.

4.4.1 Varied Capacity Models

As described in Chapter 3 and Equation 3.15, the two fundamental ways of controlling how the latent embedding is influencing the trained TTS models is by varying the structural (embedding, E_{Dim}) and variational (informational, C) capacities. While [8] explored many different values for these two variables, upon listening to example samples of their trained models[11] as well as consulting the results section in the original paper, the following values for the two capacities have been chosen to be investigated in this work: $C = [50,150,300]$ and $E_{\text{Dim}} = [64,128]$. These parameters then yield a grid of 6 models for training shown in Table 4.3

Model #	**C Value**	**E_{Dim} Value**
1	50	64
2	150	64
3	300	64
4	50	128
5	150	128
6	300	128

Table 4.3: Embedding and Variational Capacity Values for the Varied Capacity Model Trainings

Through continuous sample examination by the candidate and Dr Athanasios Lykartsis, the prosodic content of the samples as well as the transfer capacities of these models have been proven to be good enough to stop training at 100,000 steps. Since these models were only going to be used to evaluate their prosody transfer characteristics and would not be evaluated for their naturalness and quality, the decision to stop training at 100,000 steps has been taken to save time and resources.

[11] google.github.io/tacotron/publications/capacitron

4.4.2 Naturalness and Quality Models

In order to assess the naturalness and quality of the implemented Capacitron method, two models have been chosen to be trained to the full 300,000 steps in accordance with [8]. These models would then be used for a separate evaluation round, where there would be no mixing of samples between short- and long-trained models.

Model #	C **Value**	E_{Dim} **Value**
1	150	128
2	300	128

Table 4.4: Embedding and Variational Capacity Values for the Naturalness and Quality Model Trainings

Cloud Computing

In order to speed up the development process and to be able to train multiple models in a parallel way, a request to the Audio Communication faculty has been made to rent a cloud computing instance. Since the budget allocated for this project by the faculty restricted the scope of instance type choices, a recommendation from the Coqui TTS community has been taken up to rent an EC2 G4[12] instance from the cloud computing service provider Amazon Web Services. This instance has been used to train Model # 2 in Table 4.4, with a `batch_size=150` and 300,000 training steps.

4.4.3 Acoustic Model Training

In order to train the acoustic model that converts the synthesised spectrogram into an audio file, many different types of vocoders have been considered. While the choice of vocoder does not influence the prosodic content of the synthesised utterances, it is an essential part in making the synthetic voice appear more realistic and higher fidelity to the listener. Thanks to the Coqui AI open source TTS community, there are many different vocoders available in their codebase that are ready to use and are in line with the main infrastructure of their library. As previously mentioned in Chapter 2, the choice for the HiFi-GAN vocoder has been made based on its fast inference time and state-of-the-art quality characteristics.

[12]aws.amazon.com/ec2/instance-types/g4/, last visited: 04.10.2020

The author of [47], Edresson Casanova[13] is a PhD student at the University of São Paulo in Brazil who is one of the mein contributors of the Coqui AI TTS library. Edresson was kind enough to share an already pre-trained (350,000 steps in total based on a different dataset) HiFi-GAN vocoder model, that could be used as a base model for further training and therefore saving time. Then, the method of model *fine-tuning* could be started, whereby an already pre-trained model's parameters could be reloaded and further trained using a new dataset and new hyper parameters. Such "transfer learning" is not uncommon for vocoder models, since these models usually need many hundreds of thousands of steps to yield high fidelity and quality audio. An interesting aspect of this fine-tuning process is the fact that the pre-trained model received has been trained on a completely different dataset with a lower sample-rate and different audio parameters. Despite of this, continuing and fine-tuning of the model was started with an additional 250,000 steps trained this time with the dataset used for this project and its corresponding audio parameters.

Following the training process detailed in [47], after the first part of fine-tuning the model, the training was interrupted. Using an already existing script in the Coqui TTS library, one of the fully trained spectrogram models has been used to extract synthesised and teacher-forced spectrograms to essentially create a new dataset for the vocoder training. While previously the ground truth samples from the database have been used for the fine-tuning training, [24] shows that extracting teacher-forced mel-scale spectrograms from an already fully trained spectrogram model improves the quality of the synthesised samples. With this newly extracted dataset, the training was resumed and continued for another 330,000 steps, totalling 930,000 steps at `batch_size=110` for the HiFi-GAN-FT (fine tuned HiFi-GAN [47]) vocoder used for all listening evaluation tests.

[13]github.com/Edresson

5 Model Evaluation

This chapter details the evaluation methods used for the trained models. It includes the description of the custom-modified online testing tool as well as the description of the participants and the results of the listening test. Additionally, data from the training variables and scalar results are presented. At the end of the chapter, a section discussing the results offers a report on the performance of the implemented models and reports critical aspects for improvement.

5.1 Method

5.1.1 Listening Experiment Design

The aim of this work was to design two subjective tests to evaluate trained models based on the proposed evaluation method in [8]. These included the following methods:

1. evaluation of transfer characteristics in form of an *AXY-Discrimination* test,
2. evaluation of speech naturalness and intelligibility in form of a *MOS* test.

AXY Discrimination Evaluation

Firstly introduced in [6], the AXY Discrimination test proposes a new subjective evaluation method for prosody transfer applications, described by the authors as *anchored prosody side-by-side*:

> *„A human rater is presented with three stimuli: a reference speech sample (A), and two competing samples (X and Y) to evaluate. The rater is asked to rate whether the prosody of X or Y is closer to that of the reference on a 7-point scale. The scale ranges from "X is much closer" to "Both are about the same distance" to "Y is much closer", and can naturally be mapped on the integers from -3 to 3. Prior to collecting any ratings, we provide the raters with 4 examples of prosodic attributes to evaluate (intonation, stress, speaking rate, and pauses), and explicitly instruct the raters to ignore audio quality or pronunciation differences."* [6, p. 11]

As described in Section 3.4, there are three ways one can utilise the VAE architecture to infer from the models - same text prosody transfer (STT), inter text style transfer (ITT) and prior sampling (Prior). The task explored in this part of the listening tests was competing 6 models with each other in two parts - STT and ITT. While the authors of the original Capacitron method explored many different models' transfer capabilities compared to a non-prosodic baseline, they didn't compete different capacity models with each other to explore how participants would rate the transfer characteristics between Capacitron models with different structural and variational capacities. The aim of this comparison is to answer the research question stated in this work, namely to find out to what extent does a limited variation in the embedding and variational capacities influence the transfer characteristics of the implemented model. The 6 models listed in Table 4.3 are competed against each other in a way that participants must decide which synthesised utterance between two samples inferred from different models is closer to a ground truth reference. For the six models, this gives us $\binom{6}{2} = 15$ competing tests. Since we're exploring STT and ITT synthesis types, this yields 30 tests in total presented to participants in a random order.

Regarding the inter-text transfer tasks, the prompt to be synthesised was generated randomly [1] and the length of each reference-synthesis text pair was chosen to be similar. The decision to restrict the synthesis prompts' lengths to be similar to the reference was for the sake of attempting to capture an "even" transfer of prosody and because the transfer characteristics often failed and produced undesirable artefacts with certain models. This topic is further discussed in Section 5.3 below.

Mean Opinion Score (MOS) Evaluation

The MOS subjective evaluation method is generally regarded as the most reliable and definitive way of assessing speech quality and naturalness [10]. Formerly known as the absolute category rating, the MOS test became the standard evaluation method for TTS research, providing a baseline scalar value for synthesis models to compare each other with. In a standard MOS test, a participant is asked to rate audio files using a discrete 1-5 scale presented in Table 5.1. While the original description of each level of the scale shown to participants applies to the distortion found in audio signals, MOS tests used for the evaluation of synthetic speech change this information to a description of the speech quality. The descriptions used for the evaluation tests conducted in this work are also presented in Figure 5.1.

[1]randomwordgenerator.com/sentence.php

Rating	**Quality**	**Audio Distortion**	**Speech Description**
5	Excellent	Imperceptible	Completely natural speech
4	Good	Just perceptible, but not annoying	Mostly natural speech
3	Fair	Perceptible and slightly annoying	Equally natural and unnatural speech
2	Poor	Annoying, but not objectionable	Mostly unnatural speech
1	Bad	Annoying and objectionable	Completely unnatural speech

Table 5.1: MOS scores, original audio distortion hints from [10] and the description hints used in this work

In order to capture a valid MOS evaluation of audio files, [10] defines the following requirements:

- there are enough listening subjects of sufficient diversity to deliver statistically significant results;
- experiments are conducted in a controlled environment with specific acoustic characteristics and equipment;
- every subject receives the same instructions and stimuli.

In line with other TTS MOS evaluation methods [8] and the requirements listed above, the original evaluation tests have been slightly altered to fit into the framework of this thesis. The discreet 5-point scale has been switched to a discreet 100 point scale to be in line with the remote testing solution described below, which does not alter the statistical results of the tests [10]. The following sections detail how the other aspects for the validity of the tests were fulfilled with focus on the remote testing solution and participant diversity. A screenshot of the final experiment including the altercations detailed above is shown in Appendix C.

To investigate how the models with different variational capacity limits and their corresponding synthesis types compare with each other, the MOS tests have been split into two parts, whereby the first part compares *constant type* synthesis samples, and the second part explores *constant model* samples. In the *constant type* tests, the ground truth sample is compared with 2 samples of the same type (STT, ITT or Prior) from the two models with different C values. This is repeated twice for each type with different ground truth samples, yielding 6 tests in total. In the *constant model* tests, the ground truth sample is compared with 3 samples (STT, ITT and Prior) from a single model with a fixed C. This is repeated twice for each model, yielding 4 tests in total. The final MOS test round consists of the same 10 tests for each participant, who are presented with each test in a random order. The reason behind this split is to investigate participants' subjective evaluation of the utterances in different contexts.

By contesting same type synthesised utterances from different models, participants are encouraged to rate the differences between different capacity models to gain insight how the capacity limit influences naturalness. By contesting different synthesis types from the same models, participants can rate which synthesis type performs better on a naturalness scale, shedding light on the performance of the different inference types irrespective of the capacity limit.

An important distinction from the evaluation method described in [8], is that in this experiment, the participants were not given speech audio samples corresponding to the different values on the MOS-scale as audio hints. The candidate was not able to find any specific set of five synthetic speech samples that were accepted by the wider research community as standard references for the five-point MOS-scale. This referencing naturally influences the listener, since without any anchors, participants have different expectations. The effect of this decision is further discussed in Section 5.3 below.

5.1.2 Remote Testing Solution

Due to the COVID-19 global pandemic, the evaluation of the models in the form of a subjective listening test needed to be carried out remotely. Fellow students from the Audio Technology Faculty at TU Berlin have sent out listening tests during this time and through one of these tests, the *webMUSHRA* (MUltiple Stimuli with Hidden Reference and Anchor) browser based experiment software was discovered [9]. Utilising the web audio API [2], the open source tool enables researchers to conduct listening experiments over the Internet, as so called web-based experiments, compliant with the ITU-R Recommendation BS.1534 (MUSHRA) [9].

The webMUSHRA environment provided a working solution for the infrastructure of the listening test, meaning it had built-in audio transfer event control, front-end navigation and result saving capabilities. Still, for one of the two methods described above, a new test needed to be implemented, because no existing test type in the code-base would have sufficed to carry out the AXY discrimination test.

Newly Implemented AXY Pages

While the standard MUSHRA test was applicable for the MOS evaluation, in order to recreate the evaluation template from [8] shown in Appendix C.1, a new front-end layout and a slightly altered audio playback mechanism needed to be implemented.

A new instructions page show in Appendix C.4 was implemented to inform the participants about how the test works and what to pay attention to when rating. Afterwards, the front-end for the actual test prompts was made as shown in Appendix C.5. A simple change to the audio back-end was made to prevent the samples from looping.

[2]webaudio.github.io/web-audio-api/

5.2 Participants

In order to gather participants for the two subjective evaluation tests, the candidate has reached out to his personal and professional network through social media. Moreover, the university faculty has also sent out two emails about the tests to the student and staff faculty members. In addition, the Coqui AI team has also shared the link for the listening tests on their social media channels. When reaching out to these networks, it was important to not only gather audio-savvy listeners with better trained ears, but to try to cover a wide range of listeners who are not working or studying in the realm of audio. While there was no data collected about this, the personal responses from participants confirmed that the types of listener experiences varied. However, since the majority of the channels who gathered participants for these tests are normally visited by individuals interested in audio, it is assumed that listeners approached the tasks in a critical way with more precise listening capabilities than individuals with average listening ability.

In order to get a basic demographic overview of the studies' participants, after each completed test, the participants were asked about their age and gender. The accumulated responses to these questions are shown in Figures 5.1 and 5.2 for both test runs. While a clear majority of participants were males in the age group 25-31, the studies were also completed by a wide range of age groups and around a quarter of each run's were not male. The candidate and Dr Lykartsis deemed the data diverse enough not to prolong the listening tests to gather more responses from other demographic groups. While the AXY evaluation gathered 40 responses in total, the MOS evaluation saw 43 participants completing the tests.

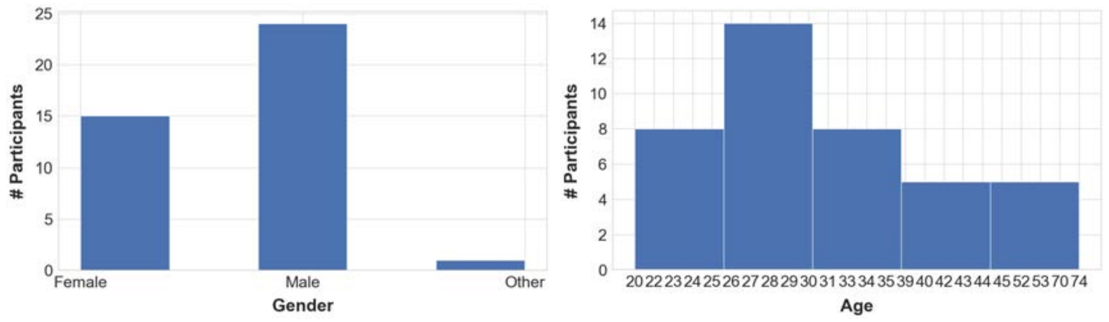


Figure 5.1: Demographic data for the AXY tests with 40 participants in total

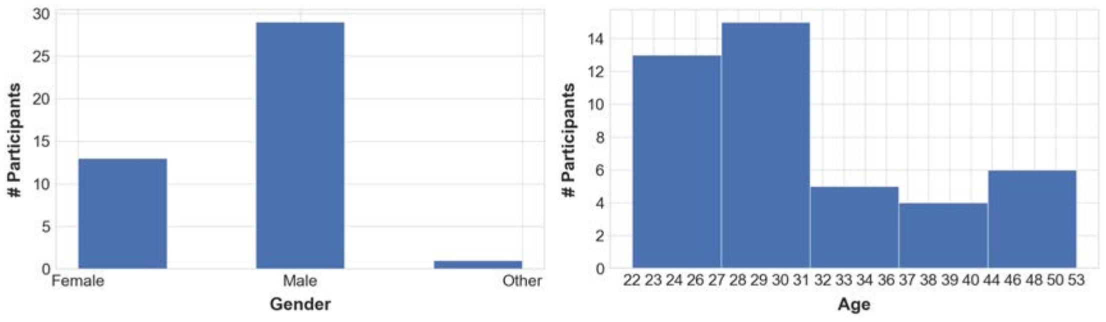


Figure 5.2: Demographic data for the MOS with 43 participants in total

5.3 Results and Discussion

Firstly, this section details the results of the objective and the subjective evaluation of the models trained for the prosody transfer comparison tests. In line with [8], the scalar values for the model loss is explored, followed by figures plotting the results of the subjective listening evaluation. Then, results for the subjective evaluation of models trained for the quality and naturalness tests are presented. The raw results for the subjective listening tests in a table format is provided in Appendix E. While an interpretation of the plots is provided first, the critical discussion of the results of individual categories is offered following each figure.

5.3.1 AXY Objective Evaluation

Figure 5.3 shows the scalar value of Equation 3.15, the main loss function of the Capacitron method after 100,000 training steps for the 6 separate models. This Figure demonstrates how varying the structural and variational capacities translates to the model loss. The reconstruction loss is the only objective evaluation result presented in this work, based on the introduced metric in [8]. The values give a representation about how good the model was able to reconstruct the target spectrogram. The lower the loss, the closer the synthesised spectrogram was to the ground truth.

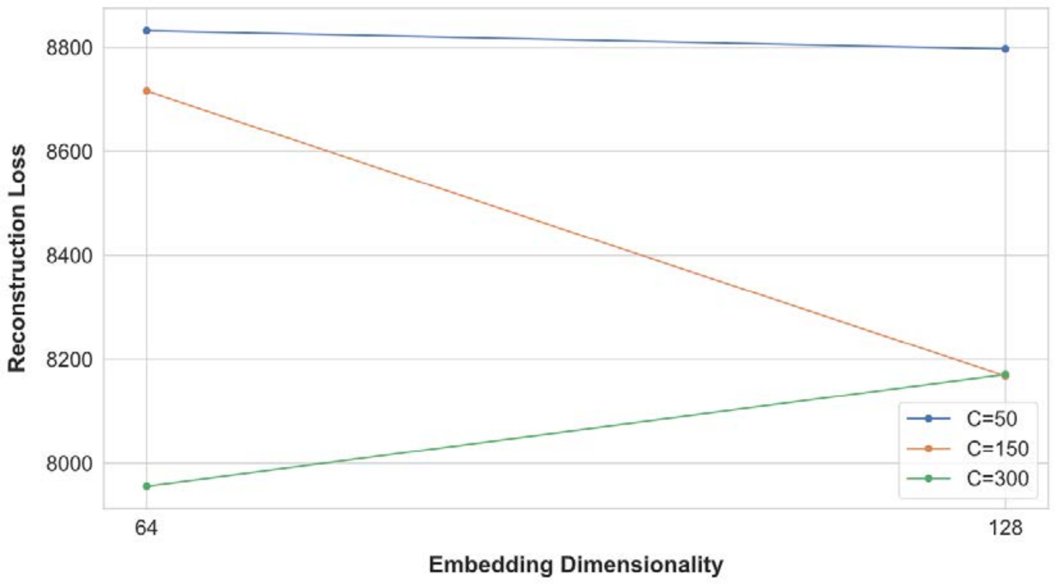


Figure 5.3: Reconstruction loss vs. structural capacity (embedding dimensionality)

This work only studies 6 combinations of models with different capacity combinations (compared to the 15 variational models explored in [8]), however two of these models with $C = 300$ have not been previously discussed. Generally confirming the results from the original method, increasing the variational capacity C improves the reconstruction error, whereby almost each model plotted with different colours have lower error values as C increases. Increasing structural capacity from $E_{\text{Dim}} = 64$ to $E_{\text{Dim}} = 128$, however yielded slightly different results than shown in previous work. The authors of [8] found that after a certain embedding dimensionality, the reconstruction error of models with values $C = [10,50,150]$ flattens out, meaning that they observed that the loss value does not improve anymore for $E_{\text{Dim}} > 128$.

While the models with the lowest variational capacity limit C did not improve dramatically by increasing structural capacity (confirming previous results), the reconstruction error greatly improved by allowing the prosody embeddings to have a higher dimensionality for the model with $C = 150$. On the contrary, a greater embedding dimensionality actually slightly worsened the reconstruction error for the model with $C = 300$. This result sheds light on the effects of these two values on the pure reconstruction nature of the models, whereby a greater variational capacity limit translates to better reconstruction up until a point where the embedding dimensionality is still small enough to balance out this increased variational capacity limit.

Examining purely the objective reconstruction of samples, a larger variational capacity limit is used to target a larger upper bound on the representational mutual information between the data and the latent space using the objective defined in Equation 3.15. Now, this upper bound is designed to limit how much of the latent space is actually embedded into the final model, and by sufficiently increasing this limit we wish to maximise the influence of this embedding. The calculation for the KL divergence term in Equation 3.15 is averaged over the entire dataset and by setting a large variational capacity limit, the neural network model responsible for modelling the prosody reference space is encouraged to encode samples with prominent prosody content form this dataset. Depending on the *range* of prosodic information in the dataset, the encoder may process samples with very high prosodic content, which - in case of a very high limit on the KL term - may increase the representational mutual information between the data and the latent space to an extent that the encoder architecture not only learns how to encode such information, but also encourages the embedding of this information from samples that would necessarily not contain that much prosodic information. This point is further discussed below, based on remarks from the subjective evaluation results.

5.3.2 AXY Subjective Evaluation

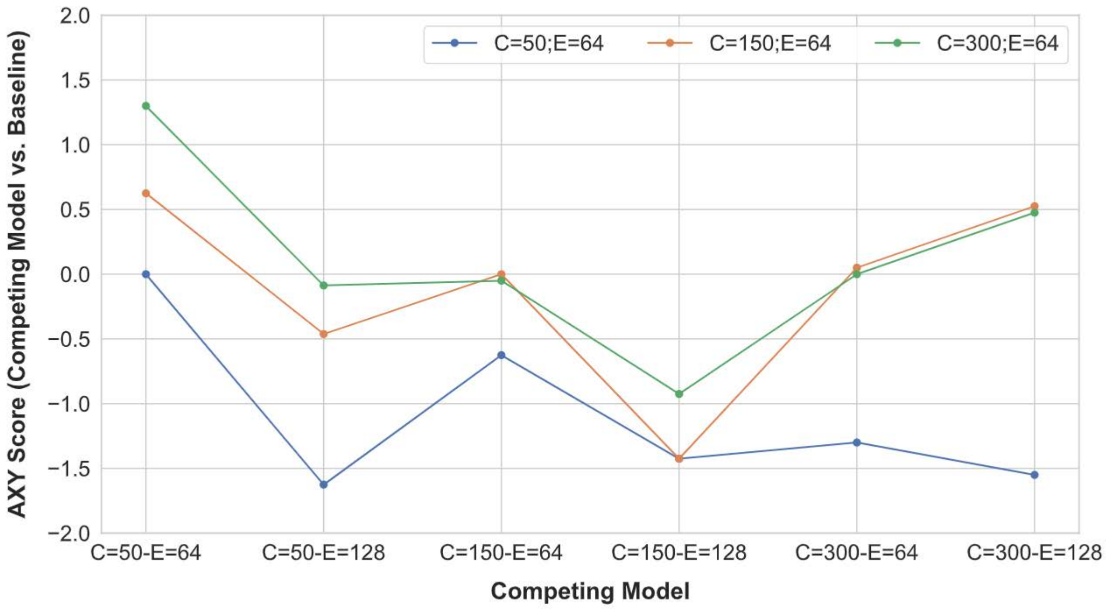
In order to visualise the effects of different structural and variational capacities, the results of the listening tests are presented in the following way:

- holding the structural capacity E_{Dim} constant, AXY scores for three models with different variational capacity C are plotted against each other;
- holding the variational capacity C constant, AXY scores for the two models with different structural capacity E_{Dim} are plotted against each other.

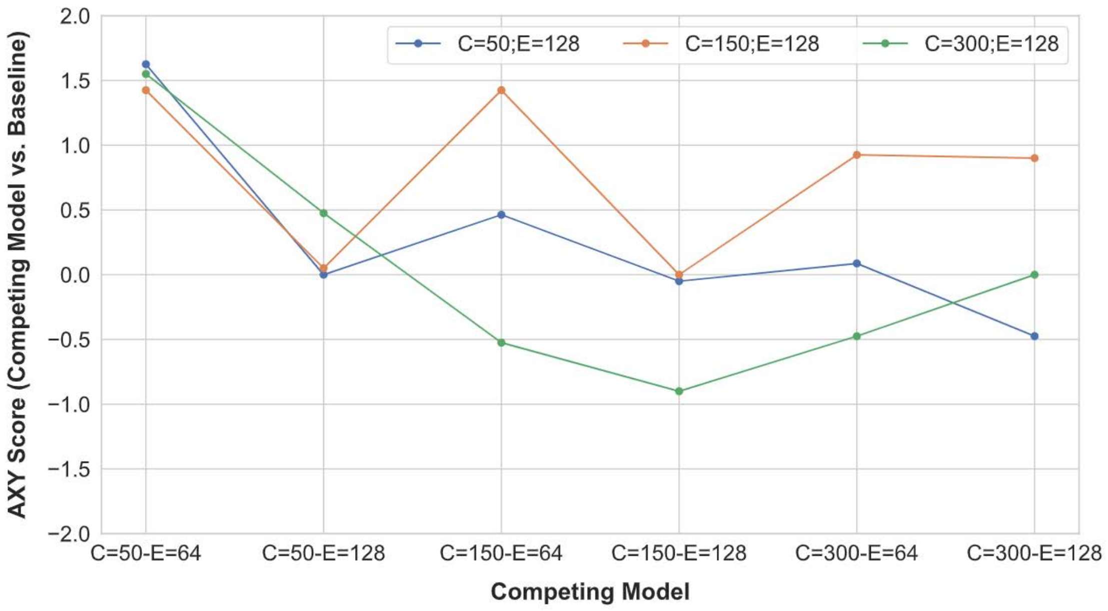
In the first case of holding E_{Dim} constant, we get two plots, since this work has explored two different values `[64, 128]` for this variable. By plotting these two plots next to each other, we can appreciate how changing only the structural capacity influences the perceived prosody transfer capability. In the second case of holding C constant, we get three plots for the three explored values `[50, 150, 300]`. Similarly, by having these plots next to each other, we get a clearer picture of how changing the variational capacity influences the perceived prosody transfer. This routine of data representation for the STT models is then repeated again for the ITT results below.

Figures 5.4-5.7 show the plotted AXY scores of *baseline* models versus the corresponding *competing* models. The competitors are the 6 models defined in Table 4.3. On each plot, a single baseline model (the coloured points and lines corresponding to the models described in the legend) is always competing against 5 other models. The score for the same baseline-competing model has been set to `0.0`. The subjective scores plotted on the Y-axis correspond to the mean taken of each participants' response to that specific competition of models. Positive values mean that the participants found the *baseline* model closer to the reference on the scale shown in Figure C.5, while negative values mean the they found the *competing* model's prosody to be perceptually closer to the reference. The reader is advised to study the plots in a way that the negative scores correspond to better prosody transfer characteristics perceived for the models shown on the X-axis against the colour-coded models shown in the legend and vice versa.

Same Text Prosody Transfer (STT)



(a) $E_{\text{Dim}} = 64$



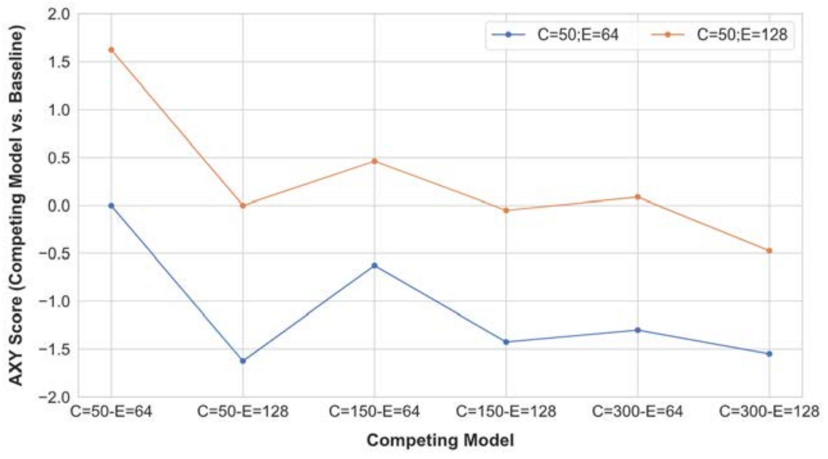
(b) $E_{\text{Dim}} = 128$

Figure 5.4: AXY-STT scores for baseline models with constant structural capacity E_{Dim}

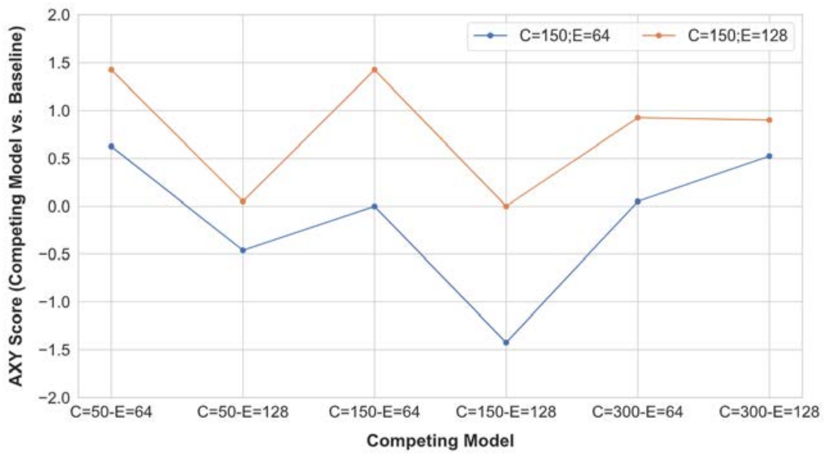
Figure 5.4 shows results of the competition between baseline models with constant structural capacity and all other models for the same text prosody transfer tasks. The coloured lines on Figure 5.4a show models with different variational capacities and constant structural capacity $E_{\text{Dim}} = 64$. Note how these baseline models have lost almost all competitions, since most of the points are in the negative Y-axis range. Most of the models clearly didn't have enough structural and/or variational capacity to be able to encode enough of the latent space for the synthesised utterances to be close to the reference. On the other hand, the competitors shown on the X-axis almost always won the competitions. Note how the model with $C = 50$ lost every competition, and by increasing C, the subjective evaluation values are getting larger. The $C = 150$ model won three times, once against the worst model mentioned before and just about against the $C = 300$ models. A significant result from this subplot is the very first point on the green curve. The $C = 300; E_{\text{Dim}} = 64$ model performed considerably better than the $C = 50; E_{\text{Dim}} = 64$ competitor which hints at the relationship of balance between the two capacity values. The significant increase in variational capacity makes up for the lack of structural capacity in the same text prosody transfer tasks.

Moving over to Figure 5.4b, all model parameters have been fixed as on the previous plot except for increasing the structural capacity to $E_{\text{Dim}} = 128$. It is very apparent how now a majority of the competitions have been won by the baseline models, since most of the points on the curves are in the positive Y-axis range. Significant results from the $C = 50$ and $C = 150$ models indicate that increasing the structural capacity was crucial in the perceived evaluation of successful prosody transfer. The only model that has lost multiple competitions is the $C = 300$ model, however this only happened against the models with sufficiently big variational capacity $C = 150$.

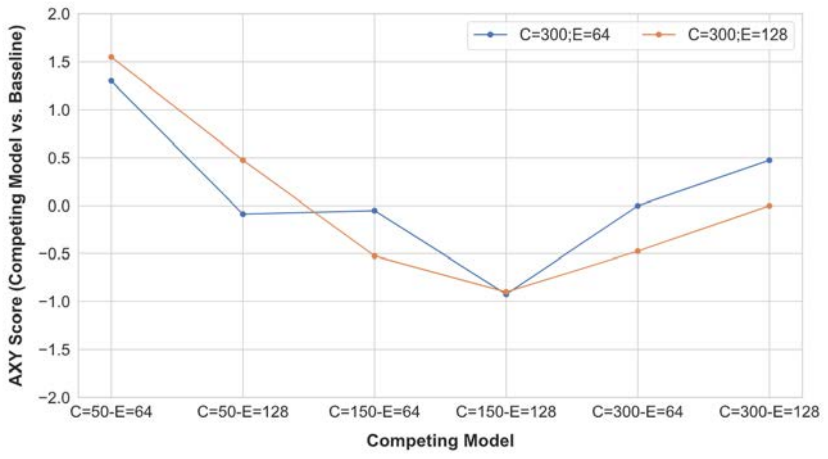
On both figures it is apparent, that the competition between the $C = 300; E_{\text{Dim}} = 64$ and $C = 300; E_{\text{Dim}} = 128$ models was won by the model with lower structural capacity. Referring back to the points described in the previous section, a possible explanation for this is that by restricting the structural capacity to a smaller value while the model is encoding samples with a very high limit on the mutual information between the data and the latent space, the capacities of this model are balanced out and thereby yield better prosody transfer.



(a) $C = 50$



(b) $C = 150$



(c) $C = 300$

Figure 5.5: AXY-STT scores for baseline models with constant variational capacity C

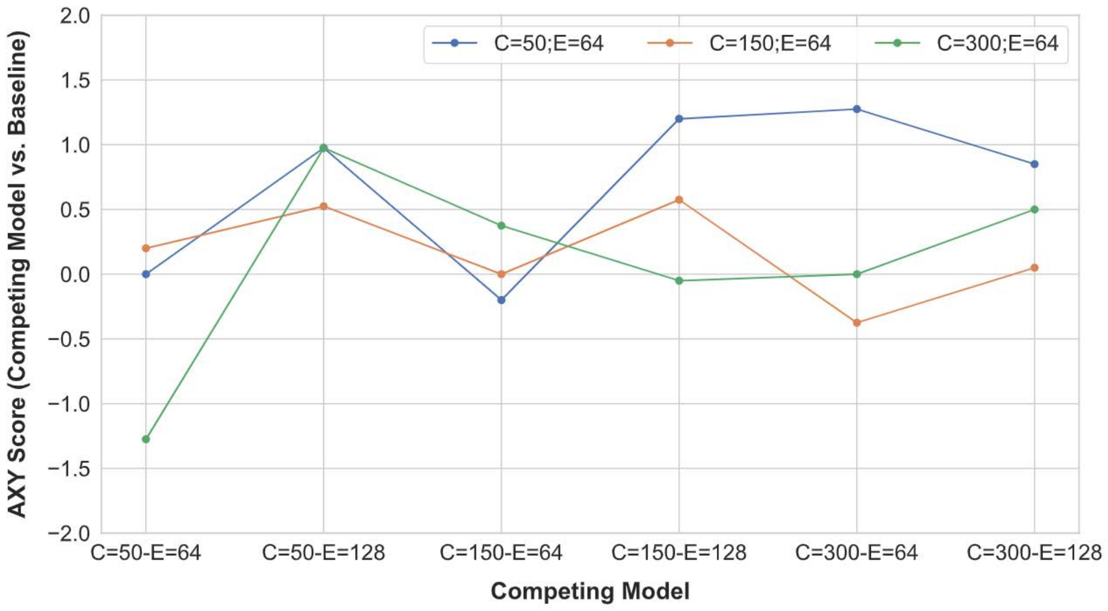
Figure 5.5 shows results of the competition between baseline models with constant variational capacity and all other models for the same text prosody transfer tasks. It is clear, that in almost all cases, the models with higher structural capacity outperformed the ones with lower E_{Dim} values. Figures 5.5a and 5.5b emphasise the need for a sufficiently high enough E_{Dim} for successful same-text prosody transfer.

Exploring the arguments detailed above from a different perspective, Figure 5.5 shows again, how high variational capacity is only desirable against models who do not have a sufficiently high enough capacity of either structural or variational. When models are trained with a balance between these two variables (be it by balancing out the large variational capacity with low structural capacity), their prosody transfer capabilities outperform those whose variational capacity is too high.

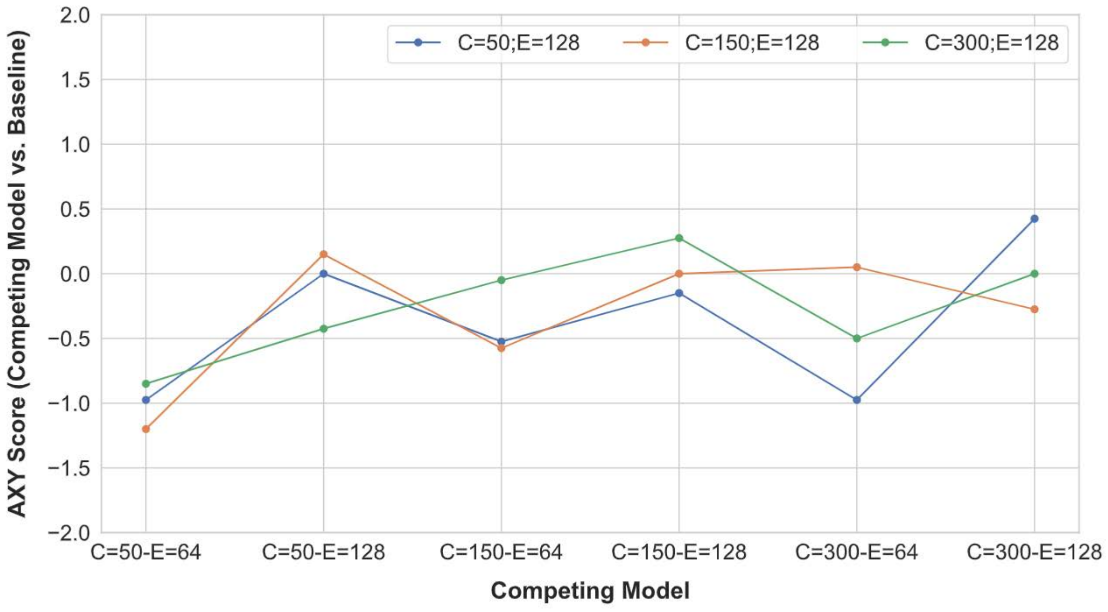
Based on these results it might suffice to claim that it is possible to balance out the high variational capacity model with lower structural capacity, however the practical implication of inferring these models conflicts this claim. During the preparation of the subjective evaluation tests, multiple rounds of synthesis needed to be carried out to be able to cherry pick samples that could be used in the subjective evaluation tests. While synthesis robustness and stability have many influencing factors (further discussed in Subsection 5.3.3 below), models with $E_{\text{Dim}} = 64$ produced significantly less clean and usable samples as models with $E_{\text{Dim}} = 128$. This observation highlights the fact that a sufficiently high enough structural capacity is required for both prosody transfer *and* synthesis stability.

On the other hand, inferring models with $C = 300$ often resulted in unnaturally intense prosody, which led to undesirable audio artefacts and sometimes incomprehensible speech. As mentioned in the previous section, by allowing the KL term in Equation 3.15 to grow to a very high limit, the model is prone to produce unnaturally intense prosody that in turn actually leads to poorer performance of exact prosody transfer. Depending on the prosodic content in the dataset as well as the intended use, a too large C value compromises the specific task of same-text prosody transfer. A balanced value for C - which is firstly defined by the sufficiently large chosen value for E_{Dim} as well as the perceived prosodic content in the used dataset - is essential for good performance in this task.

Inter Text Style Transfer (ITT)



(a) $E_{\mathrm{Dim}} = 64$



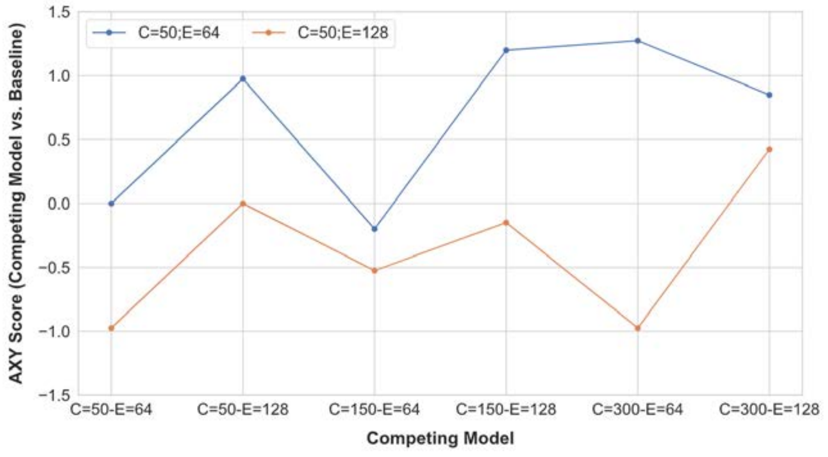
(b) $E_{\mathrm{Dim}} = 128$

Figure 5.6: AXY-ITT scores for baseline models with constant structural capacity E_{Dim}

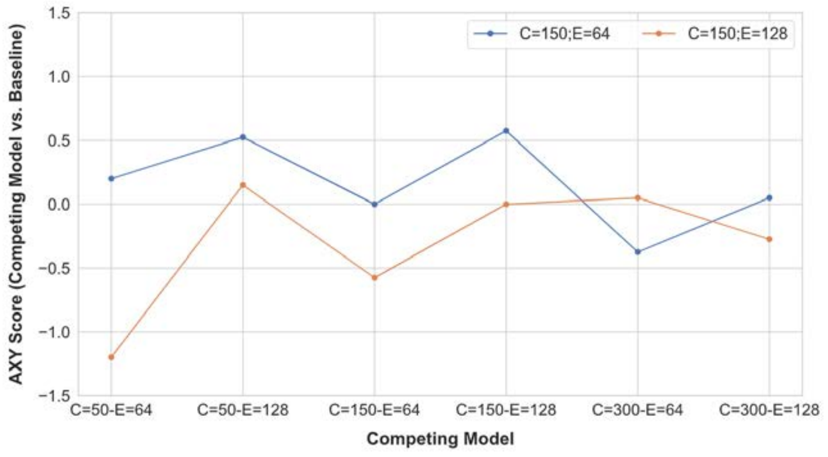
Figure 5.6 shows results of the competition between baseline models with constant structural capacity and all other models for the inter text style transfer tasks. In general, models with smaller structural capacity outperformed the ones with higher E_{Dim}, since all values on Figure 5.6a are significantly higher on the Y-axis than on Figure 5.6b. This observation is in-line with the *generality* aspect of low capacity models described in [8] - low capacity models are better suited for text-agnostic style transfer. While the effect of structural capacity is clear from these figures, it is not as straightforward to draw conclusions from the effects of different C values. However, it is clear that even between the low structural capacity models, the $C = 50; E_{\text{Dim}} = 64$ model significantly outperformed models with higher structural and variational capacity.

Figure 5.7 below further emphasises the fact the lower structural capacity models in general performed better in this task. As seen before, the only model that notably and consistently performed better than all others is the one with the lowest structural and variational capacity. Notably, the $C = 300; E_{\text{Dim}} = 64$ in Figure 5.7c greatly outperformed the $C = 50; E_{\text{Dim}} = 128$ model, which hints at the possible fact that a small enough structural capacity was more relevant for successful style transfer for this specific example.

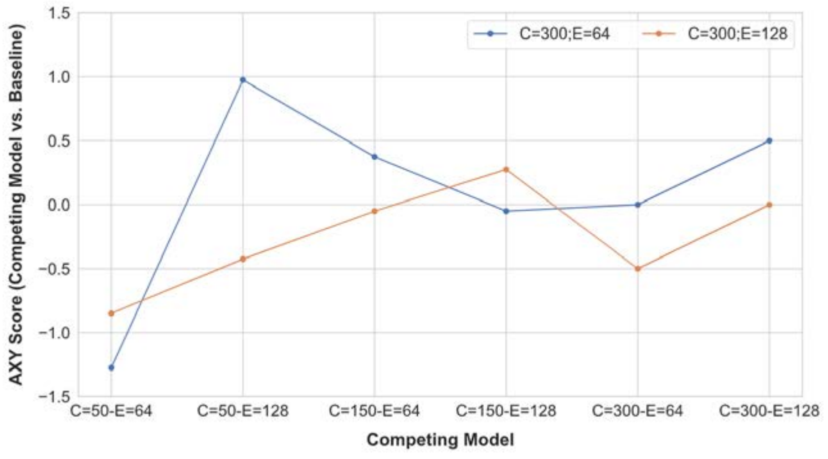
While the results described above show that lower structural capacity models performed better in the ITT task, the same practical limitation described in the previous section of inferring models with not high enough structural capacity still stands: cherry-picking good enough samples for the evaluation tests was a difficult task because of the artefacts produced by models with $E_{\text{Dim}} = 64$.



(a) $C = 50$



(b) $C = 150$



(c) $C = 300$

Figure 5.7: AXY-ITT scores for baseline models with constant variational capacity C

An important point to mention here is that the subjective evaluation of style transfer between a reference and a synthesised utterance with arbitrary text is a difficult task. The perception of prosody in general is very much dependent on the text that is being uttered, hence it is not possible to evaluate exact *prosody transfer* between utterances with arbitrary text. For this reason, we regard this task as the evaluation of *style transfer* from a reference utterance to custom text. Within this framework, the results show a successful extent of style transfer for a limited text-agnostic domain: during model inference, it became clear that the implementation struggles with synthesising an utterance whose length differs from the reference. Especially for high capacity models - where the latent embedding encourages the decoder to precisely reconstruct the prosody of the reference -, the speed of the synthesised utterance was very dependent on the speed of the reference and this often resulted in unnaturally quick or slow synthesised speech.

Further criticism of the evaluation method has also become apparent upon analysing the results. While the ITT results plotted above compare the models themselves, the synthesised utterances with arbitrary text and the corresponding references have been different for every single competition - a more meaningful way of comparing these models would have been keeping the synthesised text prompts and prosody references for this task constant. This way, the participants would have compared the same text prompt synthesised with the same prosody reference. Ideally, designing the task like this would have kept these variables with a strong influence on the rating constant, however this task already included 15 individual tasks, so it would've quickly lead to listener fatigue on the participants' side to listen to the same utterances this many times. Moreover, as described above, the synthesis of utterances was not always straightforward, due to the artefacts produced by certain models. If these artefacts occurred for a single model, the reference utterance or the text prompt would have to be changed for all other models too. Moreover, it may also be difficult to generalise the performance of the models based on a single utterance using a single reference - as also discussed in this work, models often yield inconsistent synthesis for different text and reference inputs.

These aspects pose a challenging task for the evaluation of inter text style transfer generally for TTS - perhaps a new method for the subjective evaluation of competing models is needed to tackle the balance between keeping inference variables consistent, to generalise the performance better and to avoid listener fatigue.

5.3.3 MOS Subjective Evaluation

Constant Model

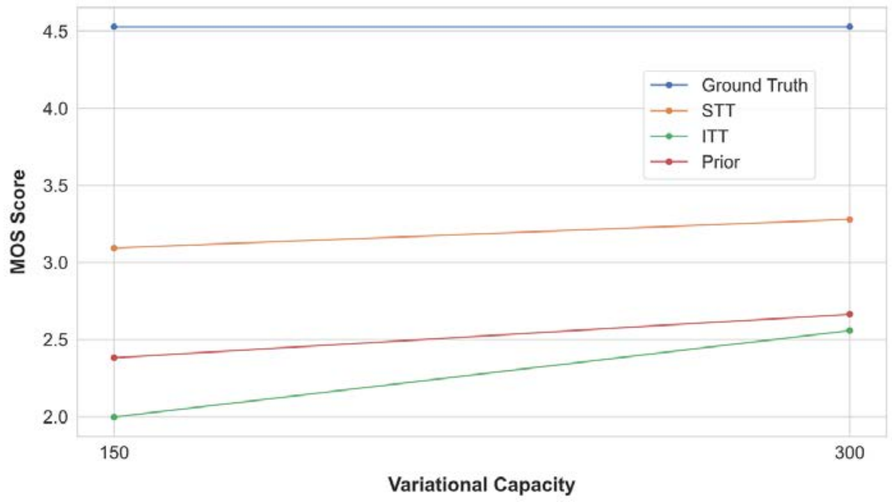


Figure 5.8: MOS scores for comparing constant models

Constant Synthesis Type

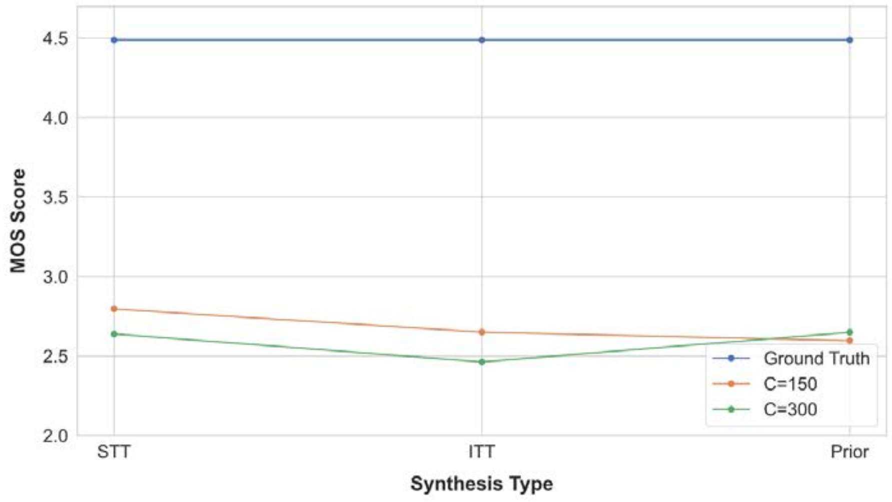


Figure 5.9: MOS scores for comparing constant synthesis types

Figure 5.8 above shows mean opinion scores for the two trained models with different capacity limits, for a task where participants had to rate three different utterance types synthesised using a single model together with the ground truth. It is clear, that the utterances synthesised using the same text as the prosody reference performed the best. Sampling from the prior of the model to produce realistic prosody performed better than samples from the ITT routine. While the samples were cherry-picked, ITT samples often included undesirable artefacts due to text mismatches. In this task, the model with $C = 300$ performed better for all synthesis types. Figure 5.9 shows the MOS scores for the three different synthesis types for the two models with different variational capacity values, where participants were presented with two utterances from two separate models using a constant synthesis type. Contrary to the AXY tests' results, the model with $C = 150$ slightly outperformed the model with $C = 300$. Similarly, the STT synthesis type outperformed the two other types.

The difference between results for the same models shown in the two figures above demonstrate the fact that the quality and naturalness of the samples are very utterance dependent - the same models produced worse results for different utterances. While it can be argued that by presenting participants in these two setups influences their perceived measure of naturalness, the significant difference between the individual types shows that the model produced unstable output. Furthermore, the MOS scores are significantly worse than in [8] and there are multiple reasons for it. Firstly, the vocoder trained for generating audio files from the synthesised spectrograms sometimes produced metallic artefacts, even after 800,000 training steps. Other methods evaluating MOS scores for TTS systems often train vocoders for millions of steps - this would have required significantly more time and resources to complete for this work. Secondly, the attention mechanism used for this project often yielded awkward results so using better modules such as location-sensitive attention [41] would significantly improve the samples.

As mentioned in Section 5.1.1, the participants in this study were not presented with reference anchors for the five levels of speech quality, despite the fact that the original method's experimental design described in [8] included five specific synthesised speech samples corresponding to the quality levels described in Table 5.1. Showcasing participants with such opinion anchors could have been beneficial for the ratings, however no specific samples were found to be able to accurately represent the five levels.

The reason for the underperforming prior samples can be an effect of multiple things. Firstly, as described in the previous section, models with a large C value often encoded too much of the latent space, so that during inference, the samples contained unnaturally intense prosody. This aspect naturally translates to sampling from the prior - while the model is capable of decoding a random latent embedding into realistic prosody, this decoding sometimes led to unnatural synthetic speech. Secondly, while the standard normal distribution is a powerful tool, it also might be too simple to model the intrinsic prosody space. By using more powerful distributions that have more flexibility and can scale better to high dimensional latent spaces, one could decrease the aggregate KL term in Equation 3.13 to have a tighter bound on the mutual information term which would lead to better prior samples. During the development of this project, attempts have been made to substitute the prior and posterior distributions with more powerful modules such as autoregressive and inverse autoregressive flows [48], however, as of writing this report, no working implementation has been successfully realised yet.

A short summary about the outcomes of the model evaluation results is presented.

- The objective evaluation metric of the models improve by increasing both structural and variational capacities up until a certain point where more variational capacity no longer improves the reconstruction error.
- The subjective evaluation results of the STT task show that higher structural capacity always improves prosody transfer, while a too high variational capacity limit worsens the perceived transfer:
  - the $C = 150; E_{\text{Dim}} = 128$ model outperformed all other models with comparably more reliable synthesis stability.
- The subjective evaluation results of the ITT task demonstrate that generally lower capacity models are more suitable for text agnostic style transfer:
  - the $C = 50; E_{\text{Dim}} = 64$ model outperformed all other models but synthesis stability is significantly worse and comparably less reliable due to the too small structural capacity.
- The evaluation of the MOS tests showed that while the scores were on average not as high as in [8], the STT synthesis type always outperformed all other inference types.
- A modified AXY subjective evaluation method has been presented to compete Capacitron models with each other, with the aim of investigating the influence of structural and variational capacities on the perceived prosody transfer for both same-text prosody transfer and inter-text style transfer tasks.
- A novel graphical representation of the gathered data has been provided to effectively draw conclusions of the results.
- The objective and subjective evaluation results of the models' prosody and style transfer capabilities fittingly confirm the results presented in the original method and thereby assert the successful implementation of the Capacitron method.

6 Open Source Contributions

A major objective of this thesis was the release of a previously not publicly available TTS method as open source code. As of writing, the model has not been merged into the Coqui AI TTS library yet but there is an open Pull Request[1] to merge the model into the main library for use by the wider community.

During the development process and discussions with the Coqui AI team and contributors, some members have already successfully trained models using the implementation of this thesis. Moreover, Edressen Casanova has contributed[2] to the model by reorganising the modules in a way that the Capacitron VAE architecture could be used in a modular way, paving completely new ways to explore prosody transfer, including its implementation into the more powerful Tacotron 2 architecture.

With the Coqui AI TTS library gaining significant attention (currently 2.5k stars on GitHub, previously 22k stars and over 500k downloads under Mozilla TTS and STT libraries[3]), the implemented model will be available for a wide audience including other students, researchers, commercial bodies and more.

In addition to the code, the public release of this thesis as well as the public discussions[4] about implementation problems is hoped to serve as a helping tool for other novice engineers aiming to contribute to open source TTS and ML technologies.

Releasing the custom-made AXY testing solution page is also planned in order to enable researchers to conduct reference-prosody evaluation tests remotely within the webMUSHRA framework.

[1] github.com/coqui-ai/TTS/pull/510
[2] github.com/a-froghyar/Capacitron/pull/6
[3] coqui.ai/
[4] github.com/coqui-ai/TTS/discussions/455

7 Conclusion

The effort put into modern TTS technologies has been steadily rising since the introduction of two landmark methods in 2016. The possibilities that deep neural network architectures offer today brought significant commercial and academic interest to the synthetic voice topic in recent years. With the general quality of speech synthesis constantly increasing, a natural area of interest has been the focus of many researchers: controllable and realistic synthetic prosody. Controllable expressive speech synthesis is considered to be one of the ultimate achievements to strive for with modern TTS technologies and recent research methods have demonstrated state-of-the-art performance and results. The research teams of the American internet and software corporate Google have released multiple scientific methods that discuss new ways of achieving such a controllable speech synthesizer. While the scholarly articles describing these methods are available with open-access, software companies rarely release code implementations of their described methods, with usually only a few impressive sound samples shared online for interested parties to listen to.

This Master's Thesis investigated a specific method from Google that has achieved state-of-the-art quality of prosody transfer between a reference utterance and synthetic speech. Following a description of the motivation and theory behind the Capacitron method, this work offered a detailed technical report on the realisation of this method in code within an open-source software framework. Expanding on a specific aspect of the original architecture, this work extended discussion on the capacity of the latent embeddings that influence the models' prosody transfer characteristics. In order to assess the implemented method, this work detailed the design, conception and analysis of a custom-made subjective evaluation routine. This routine partly expanded the subjective evaluation introduced in the original method to gain more insight into how different components of the latent embedding capacity influence the subjective evaluation of prosody transfer.

The technical development of this project has taken up the majority of time and resources allocated for this work. The implementation of a new machine learning TTS model into an existing open source infrastructure is a timely process, whereby continuous experimentation and training needed to be carried out to monitor bugs and general performance of the model. The translation of a method described in a scientific article into executable code was a difficult and timely task due to the black-box nature of machine learning models as well as the fact that some specific details of the model were not fully described in the original article. Since a single model needed at least eight hours to train to be able to decide if it is working or not, development was slow. However, with the assistance of an active open source community as well as helpful discussions with the original author of the article, the implementation could be successfully realised and the planning of the subjective evaluation of the implementation could be started.

The first objective of this Thesis was to expand the discussion on the notion of the two types of prosody embedding capacities, and their effect on the model's prosody transfer characteristics. While the original work has briefly discussed the effects of the structural and variational capacities on the model performances, it did not extend this notion to the subjective evaluation of these models. This work extends the discussion on the structural and variational capacities by designing subjective evaluation tests where different models with different capacity values competed against each other. While the number of models is limited due to time and resource constraints, this work still investigated the competition between six models with different capacity values. The second objective of this work was the evaluation of the implemented models on the industry-standard MOS scale, to gain insight into how the implementation performs in terms of speech naturalness and quality. Lastly, the final objective of this Master's Thesis was to release the code implementation of a previously not publicly available state-of-the-art expressive speech synthesis method within an open-source framework.

The most successful part of this project in terms of the subjective results is the prosody transfer from a reference utterance onto synthetic speech using the same text (STT). Within this frame, valuable insights have been gained into how the balance between the structural and variational capacities influence the transfer capabilities of the models. For both synthesis stability and perceived transfer, a sufficiently high enough structural capacity is essential. Models with higher structural capacity significantly outperformed others in terms of the subjective evaluation, and the collection of samples from the models was also much easier, since the synthesised utterances showed significantly more stability and less artefacts than the ones inferred from models with lower structural capacity. In terms of the variational capacities of the models, a conclusion from this work is that the value needs to be set in line with the prosodic content of the dataset. Too high values for the variational capacity often over-intensified the prosodic content of the synthesised utterances, which led to these models losing the prosody similarity contest against models with smaller variational capacity. Further research and experimentation needs to be carried out here, whereby one could estimate the average prosodic content capacity of the dataset by e.g. analysing the fundamental frequency and amplitude contours of individual utterances. This way, it could be estimated what the maximum value for the variational capacity could be without risking encoding too much of the latent space that results in unnatural prosody in inference time. While balancing out high variational capacity with low structural capacity often helped these models perform better in the competitions, it is practically not a solution due to increased amount of artefacts and instabilities during inference for models with not sufficiently high enough structural capacity.

Regarding the evaluation of the models for the inter text style transfer (ITT) tasks, the Thesis confirmed that lower capacity models are better suited for text-agnostic style transfer. However, drawing a clear conclusion from this part of the project is a difficult feat, because of the nature of the underlying task: the subjective evaluation of style transfer between utterances with different text is not straightforward and greatly text and utterance dependent. A universal evaluation system for text-agnostic style transfer needs to be proposed to be able to reliable evaluate and compete models with different capacities. This work used different synthesis text and reference audio for each competition in this task because using a single reference and the same text prompt for each competition was not possible due to inference inconsistencies and artefacts. However, even in the case of managing to use the same text and same reference, listener fatigue needs to be taken into consideration. Additionally, the implementation also struggled with length-agnostic transfer especially for high variational capacity models.

The subjective evaluation of the naturalness and quality of the models realised through this work performed worse than in the original method. TTS is a very resource intensive subject, and the notion of "the more the better" certainly applies to model trainings, especially when it comes to the vocoder model converting spectrograms to audio. While more training would definitely improve results, one of the major factors affecting model stability is the attention mechanism. While an extended discussion on the attention module is out of the scope of this work, a more powerful attention mechanism would certainly improve synthesis stability. Attempts have been made to use a more improved attention module from the Coqui TTS library, however there were unforeseen bugs originating from that implementation and fixing those bugs was outside of the time and resource scope of this project. Future work needs to be put into improving the attention mechanism to improve general stability and thereby the quality of the synthesised utterances. Further topics of interest about this specific model include the substitution of the prior and posterior distributions with more flexible distributions that can generalise to and map the latent space more effectively.

Despite the limitations and shortcomings of the implemented model compared to the released audio samples from Google, the successful realisation of the Capacitron method within an open-source framework is a welcome addition to the publicly available libraries of TTS technologies.

Bibliography

[1] OORD, Aaron van d. ; DIELEMAN, Sander ; ZEN, Heiga ; SIMONYAN, Karen ; VINYALS, Oriol ; GRAVES, Alex ; KALCHBRENNER, Nal ; SENIOR, Andrew ; KAVUKCUOGLU, Koray: *WaveNet: A Generative Model for Raw Audio.* 2016

[2] *WaveNet: A generative model for raw audio blog post.* `deepmind.com/blog/article/wavenet-generative-model-raw-audio`, . – Accessed: 2021-08-35

[3] WANG, Yuxuan ; SKERRY-RYAN, RJ ; STANTON, Daisy ; WU, Yonghui ; WEISS, Ron J. ; JAITLY, Navdeep ; YANG, Zongheng ; XIAO, Ying ; CHEN, Zhifeng ; BENGIO, Samy ; LE, Quoc ; AGIOMYRGIANNAKIS, Yannis ; CLARK, Rob ; SAUROUS, Rif A.: *Tacotron: Towards End-to-End Speech Synthesis.* 2017

[4] SHEN, Jonathan ; PANG, Ruoming ; WEISS, Ron J. ; SCHUSTER, Mike ; JAITLY, Navdeep ; YANG, Zongheng ; CHEN, Zhifeng ; ZHANG, Yu ; WANG, Yuxuan ; SKERRY-RYAN, RJ ; SAUROUS, Rif A. ; AGIOMYRGIANNAKIS, Yannis ; WU, Yonghui: *Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions.* 2018

[5] TAN, Xu ; QIN, Tao ; SOONG, Frank ; LIU, Tie-Yan: *A Survey on Neural Speech Synthesis.* 2021

[6] SKERRY-RYAN, RJ ; BATTENBERG, Eric ; XIAO, Ying ; WANG, Yuxuan ; STANTON, Daisy ; SHOR, Joel ; WEISS, Ron J. ; CLARK, Rob ; SAUROUS, Rif A.: *Towards End-to-End Prosody Transfer for Expressive Speech Synthesis with Tacotron.* 2018

[7] KIERCZAK, Marcin: *National Bioinformatics Infrastructure Sweden - Autoencoders in R - Neural Nets and Deep Learning.* `https://nbisweden.github.io/workshop-neural-nets-and-deep-learning/session_rAutoencoders/lecture_autoencoders_r/lecture_autoencoders.html#1`, . – Accessed: 2021-08-15

[8] BATTENBERG, Eric ; MARIOORYAD, Soroosh ; STANTON, Daisy ; SKERRY-RYAN, RJ ; SHANNON, Matt ; KAO, David ; BAGBY, Tom: *Effective Use of Variational Embedding Capacity in Expressive End-to-End Speech Synthesis.* 2019

[9] SCHOEFFLER, M. et a.: *webMUSHRA - A Comprehensive Framework for Web-based Listening Tests.* 2018

[10] PROTASIO RIBEIRO, Flavio ; FLORENCIO, Dinei ; ZHANG, Cha ; SELTZER, Mike: CROWDMOS: An Approach for Crowdsourcing Mean Opinion Score Studies. In: *ICASSP*, IEEE, May 2011

[11] Hewett, Thomas T. ; Baecker, Ronald ; Card, Stuart ; Carey, Tom ; Gasen, Jean ; Mantei, Marilyn ; Perlman, Gary ; Strong, Gary ; Verplank, William: ACM SIGCHI Curricula for Human-Computer Interaction. New York, NY, USA : Association for Computing Machinery, 1992. – Forschungsbericht. – ISBN 0897914740

[12] Mohasi, Lehlohonolo ; Mashao, Daniel: Text-to-Speech Technology in Human-Computer Interaction, 2006

[13] Rashad, Magdi ; El-Bakry, Hazem ; Il, Islam ; Mastorakis, Nikos: An overview of text-to-speech synthesis techniques. In: *International Conference on Communications and Information Technology - Proceedings* (2010), 07, S. 84–89

[14] K.R. Aida–Zade, C. A. ; Sharifova, A.M.: *The Main Principles of Text-to-Speech Synthesis System.* 2013

[15] Cutler, Anne ; Dahan, Delphine ; Donselaar, Wilma: Prosody in the Comprehension of Spoken Language: A Literature Review. In: *Language and speech* 40 (Pt 2) (1997), 04, S. 141–201. `http://dx.doi.org/10.1177/002383099704000203`. – DOI 10.1177/002383099704000203

[16] Tits, Noé ; Haddad, Kevin E. ; Dutoit, Thierry: *The Theory behind Controllable Expressive Speech Synthesis: a Cross-disciplinary Approach.* 2019

[17] Bughin, Jacques ; Hazan, Eric ; Ramaswamy, Sree ; Chui, Michael ; Allas, Tera ; Dahlstro, Peter ; Henk, Nicolaus ; Trencn, Monica: *Artificial Intelligence: The next digital frontier?* 2017

[18] Sessions, Valerie ; Valtorta, Marco: The Effects of Data Quality on Machine Learning Algorithms., 2006, S. 485–498

[19] Taylor, Paul: *Text-to-Speech Synthesis.* 1st. USA : Cambridge University Press, 2009. – ISBN 0521899273

[20] Oord, Aaron van d. ; Li, Yazhe ; Babuschkin, Igor ; Simonyan, Karen ; Vinyals, Oriol ; Kavukcuoglu, Koray ; Driessche, George van d. ; Lockhart, Edward ; Cobo, Luis C. ; Stimberg, Florian ; Casagrande, Norman ; Grewe, Dominik ; Noury, Seb ; Dieleman, Sander ; Elsen, Erich ; Kalchbrenner, Nal ; Zen, Heiga ; Graves, Alex ; King, Helen ; Walters, Tom ; Belov, Dan ; Hassabis, Demis: *Parallel WaveNet: Fast High-Fidelity Speech Synthesis.* 2017

[21] Kalchbrenner, Nal ; Elsen, Erich ; Simonyan, Karen ; Noury, Seb ; Casagrande, Norman ; Lockhart, Edward ; Stimberg, Florian ; Oord, Aaron van d. ; Dieleman, Sander ; Kavukcuoglu, Koray: *Efficient Neural Audio Synthesis.* 2018

[22] Valin, Jean-Marc ; Skoglund, Jan: LPCNet: Improving Neural Speech Synthesis Through Linear Prediction, 2019

[23] Tian, Qiao ; Zhang, Zewang ; Lu, Heng ; Chen, Ling-Hui ; Liu, Shan: *FeatherWave: An efficient high-fidelity neural vocoder with multi-band linear prediction.* 2020

[24] KONG, Jungil ; KIM, Jaehyeon ; BAE, Jaekyoung: *HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis.* 2020

[25] LUDWIG, Kirk ; MUNROE, Wade: *Unconscious Inference Theories of Cognitive Achievement.* 01 2019

[26] HELMHOLTZ, Hermann von: *Handbuch der physiologischen Optik.* Leipzig : Voss, 1867. – 430 S.

[27] DAYAN, Peter ; HINTON, Geoffrey E. ; NEAL, Radford M. ; ZEMEL, Richard S.: The Helmholtz Machine. In: *Neural Comput.* 7 (1995), September, Nr. 5, 889–904. `http://dx.doi.org/10.1162/neco.1995.7.5.889`. – DOI 10.1162/neco.1995.7.5.889. – ISSN 0899–7667

[28] MA, Xuezhe ; KONG, Xiang ; ZHANG, Shanghang ; HOVY, Eduard: *Decoupling Global and Local Representations via Invertible Generative Flows.* 2021

[29] CHEN, Xi ; KINGMA, Diederik P. ; SALIMANS, Tim ; DUAN, Yan ; DHARIWAL, Prafulla ; SCHULMAN, John ; SUTSKEVER, Ilya ; ABBEEL, Pieter: *Variational Lossy Autoencoder.* 2017

[30] ELIAS, Isaac ; ZEN, Heiga ; SHEN, Jonathan ; ZHANG, Yu ; JIA, Ye ; WEISS, Ron ; WU, Yonghui: *Parallel Tacotron: Non-Autoregressive and Controllable TTS.* 2020

[31] KINGMA, Diederik P. ; WELLING, Max: *Auto-Encoding Variational Bayes.* 2014

[32] HSU, Wei-Ning ; ZHANG, Yu ; WEISS, Ron J. ; ZEN, Heiga ; WU, Yonghui ; WANG, Yuxuan ; CAO, Yuan ; JIA, Ye ; CHEN, Zhifeng ; SHEN, Jonathan ; NGUYEN, Patrick ; PANG, Ruoming: *Hierarchical Generative Modeling for Controllable Speech Synthesis.* 2018

[33] HOFFMAN, Matthew D. ; JOHNSON, Matthew J.: *ELBO Surgery: Yet Another Way to Carve up the Variational Evidence Lower Bound.* 2016

[34] MAKHZANI, Alireza ; SHLENS, Jonathon ; JAITLY, Navdeep ; GOODFELLOW, Ian ; FREY, Brendan: *Adversarial Autoencoders.* 2016

[35] ALEMI, Alexander A. ; POOLE, Ben ; FISCHER, Ian ; DILLON, Joshua V. ; SAUROUS, Rif A. ; MURPHY, Kevin: *Fixing a Broken ELBO.* 2018

[36] HIGGINS, Irina ; MATTHEY, Loic ; PAL, Arka ; BURGESS, Christopher ; GLOROT, Xavier ; BOTVINICK, Matthew ; MOHAMED, Shakir ; LERCHNER, Alexander: *beta-vae: Learning basic visual concepts with a constrained variational framework.* 2017

[37] BURGESS, Christopher P. ; HIGGINS, Irina ; PAL, Arka ; MATTHEY, Loic ; WATTERS, Nick ; DESJARDINS, Guillaume ; LERCHNER, Alexander: *Understanding disentangling in β-VAE.* 2018

[38] REZENDE, Danilo J. ; VIOLA, Fabio: *Taming VAEs.* 2018

[39] ZHANG, Ya-Jie ; PAN, Shifeng ; HE, Lei ; LING, Zhen-Hua: *Learning latent representations for style control and transfer in end-to-end speech synthesis.* 2019

[40] King, Simon ; Karaiskos, Vasilis: *The Blizzard Challenge 2013.* https://www.cstr.ed.ac.uk/projects/blizzard/data.html, 2013

[41] Battenberg, Eric ; Skerry-Ryan, RJ ; Mariooryad, Soroosh ; Stanton, Daisy ; Kao, David ; Shannon, Matt ; Bagby, Tom: *Location-Relative Attention Mechanisms For Robust Long-Form Speech Synthesis.* 2020

[42] Ito, Keith ; Johnson, Linda: *The LJ Speech Dataset.* https://keithito.com/LJ-Speech-Dataset/, 2017

[43] Wang, Yuxuan ; Stanton, Daisy ; Zhang, Yu ; Skerry-Ryan, RJ ; Battenberg, Eric ; Shor, Joel ; Xiao, Ying ; Ren, Fei ; Jia, Ye ; Saurous, Rif A.: *Style Tokens: Unsupervised Style Modeling, Control and Transfer in End-to-End Speech Synthesis.* 2018

[44] Cho, Kyunghyun ; Merrienboer, Bart van ; Gulcehre, Caglar ; Bahdanau, Dzmitry ; Bougares, Fethi ; Schwenk, Holger ; Bengio, Yoshua: *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* 2014

[45] Bahdanau, Dzmitry ; Cho, Kyunghyun ; Bengio, Yoshua: *Neural Machine Translation by Jointly Learning to Align and Translate.* 2016

[46] Mnih, Volodymyr ; Heess, Nicolas ; Graves, Alex ; Kavukcuoglu, Koray: *Recurrent Models of Visual Attention.* 2014

[47] Casanova, Edresson ; Shulby, Christopher ; Gölge, Eren ; Müller, Nicolas M. ; Oliveira, Frederico S. ; Junior, Arnaldo C. ; Silva Soares, Anderson da ; Aluisio, Sandra M. ; Ponti, Moacir A.: *SC-GlowTTS: an Efficient Zero-Shot Multi-Speaker Text-To-Speech Model.* 2021

[48] Kingma, Diederik P. ; Salimans, Tim ; Jozefowicz, Rafal ; Chen, Xi ; Sutskever, Ilya ; Welling, Max: *Improving Variational Inference with Inverse Autoregressive Flow.* 2017

Appendices

A Double Optimisation Intuition

Intuition for the double optimisation problem of equation 3.15.

Fixed set of model parameters $\boldsymbol{\theta}$ and $D_{\mathrm{KL}} > C$:

- $D_{\mathrm{KL}} > C$ implies that the term on the right is positive, so the maximiser w.r.t. β is pushing β (only β) up.
- This larger β gets fixed, so the other optimiser for the whole model now sees a positive, large number on the RHS.
- Now, with this β fixed, this minimising optimisation needs to drive D_{KL} down, in order to make the term on the RHS smaller.

Fixed set of model parameters $\boldsymbol{\theta}$ and $D_{\mathrm{KL}} < C$:

- $D_{\mathrm{KL}} < C$ implies that the term on the right is negative, so the maximiser w.r.t. β is pushing β down.
- This small β gets fixed, so the main model ADAM optimiser sees a small negative number on the RHS.
- In order to make this even smaller, the optimiser is driving down the whole equation, however it is seeing the whole equation as one, including the decoder term.
- By minimising the decoder loss, the KL term increases: the decoder loss is an L1 loss between synthesised spectrogram and input spectrogram. The synthesised spectrogram is conditioned on the output of the reference encoder, whose output is actually a sample from the distribution $\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x})$). The more the model parameters $\boldsymbol{\theta}$ are trained, the better outputs the approximate posterior will output, because the model encourages the encoder network to encode useful information into this sampled vector from the distribution. Better outputs from approximate posterior means higher mutual information between the input data's distribution and the approximated latent distribution $I(\boldsymbol{X}, \boldsymbol{Z})$. Appendix B shows, that R^{AVG} (which is just the KL term in Equation 3.15 averaged over the data) is actually a higher bound on the mutual information $I(\boldsymbol{X}, \boldsymbol{Z})$.

The better the model gets (i.e. the smaller the decoder loss), the higher the mutual information between the real posterior distribution and the approximate posterior distribution will be. This in turn means that the upper bound on the mutual Information increases, which means that the D_{KL} term in equation 3.15 increases.

A negative RHS in equation 3.15 just encourages the model to learn more about the input data and this in turn actually increases the KL term up until a capacity C, whereby this positive-negative relationship oscillates the KL term around the capacity limit.

If D_{KL} is increasing because the model is learning more and better, it means that it will increase to a point where $D_{\mathrm{KL}} > C$, and this in turn then starts the whole process described above again.

B Derivations

Derivations from [8] and [35] of the bounding nature of the KL term on mutual information.

Definitions

$$
\begin{aligned}
R &\equiv \int q(\mathbf{z}|\mathbf{x}) \log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} d\mathbf{z}, && \text{(KL term)}\\
R^{\mathrm{AV}G} &\equiv \iint p_D(\mathbf{x}) q(\mathbf{z}|\mathbf{x}) \log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} d\mathbf{x} d\mathbf{z}, && \text{(Average KL term)}\\
I_q(\mathbf{X};\mathbf{Z}) &\equiv \iint p_D(\mathbf{x}) q(\mathbf{z}|\mathbf{x}) \log \frac{q(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} d\mathbf{x} d\mathbf{z} && \text{(Representational mutual information)}\\
q(\mathbf{z}) &\equiv \int p_D(\mathbf{x}) q(\mathbf{z}|\mathbf{x}) d\mathbf{x} && \text{(Aggregated posterior)}
\end{aligned}
$$

KL non-negativity

$$
\begin{aligned}
&\int q(x) \log \frac{q(x)}{p(x)} dx \geq 0\\
\implies &\int q(x) \log q(x) \geq \int q(x) \log p(x) dx
\end{aligned}
$$

Mutual information is upper bounded by the average KL

We have

$$
\begin{aligned}
I_q(\mathbf{X};\mathbf{Z}) &\equiv \iint p_D(\mathbf{x})q(\mathbf{z}|\mathbf{x})\log\frac{q(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})}d\mathbf{x}d\mathbf{z} \\
&= \iint p_D(\mathbf{x})q(\mathbf{z}|\mathbf{x})\log q(\mathbf{z}|\mathbf{x})d\mathbf{x}d\mathbf{z} - \iint p_D(\mathbf{x})q(\mathbf{z}|\mathbf{x})\log q(\mathbf{z})d\mathbf{x}d\mathbf{z} \\
&= \iint p_D(\mathbf{x})q(\mathbf{z}|\mathbf{x})\log q(\mathbf{z}|\mathbf{x})d\mathbf{x}d\mathbf{z} - \int q(\mathbf{z})\log q(\mathbf{z})d\mathbf{z} \\
&\leq \iint p_D(\mathbf{x})q(\mathbf{z}|\mathbf{x})\log q(\mathbf{z}|\mathbf{x})d\mathbf{x}d\mathbf{z} - \int q(\mathbf{z})\log p(\mathbf{z})d\mathbf{z} \\
&= \iint p_D(\mathbf{x})q(\mathbf{z}|\mathbf{x})\log q(\mathbf{z}|\mathbf{x})d\mathbf{x}d\mathbf{z} - \iint p_D(\mathbf{x})q(\mathbf{z}|\mathbf{x})\log p(\mathbf{z})d\mathbf{x}d\mathbf{z} \\
&= \iint p_D(\mathbf{x})q(\mathbf{z}|\mathbf{x})\log\frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})}d\mathbf{x}d\mathbf{z} \\
&\equiv R^{\text{AVG}} \\
\Longrightarrow & I_q(\mathbf{X};\mathbf{Z}) \leq R^{\text{AVG}},
\end{aligned}
$$

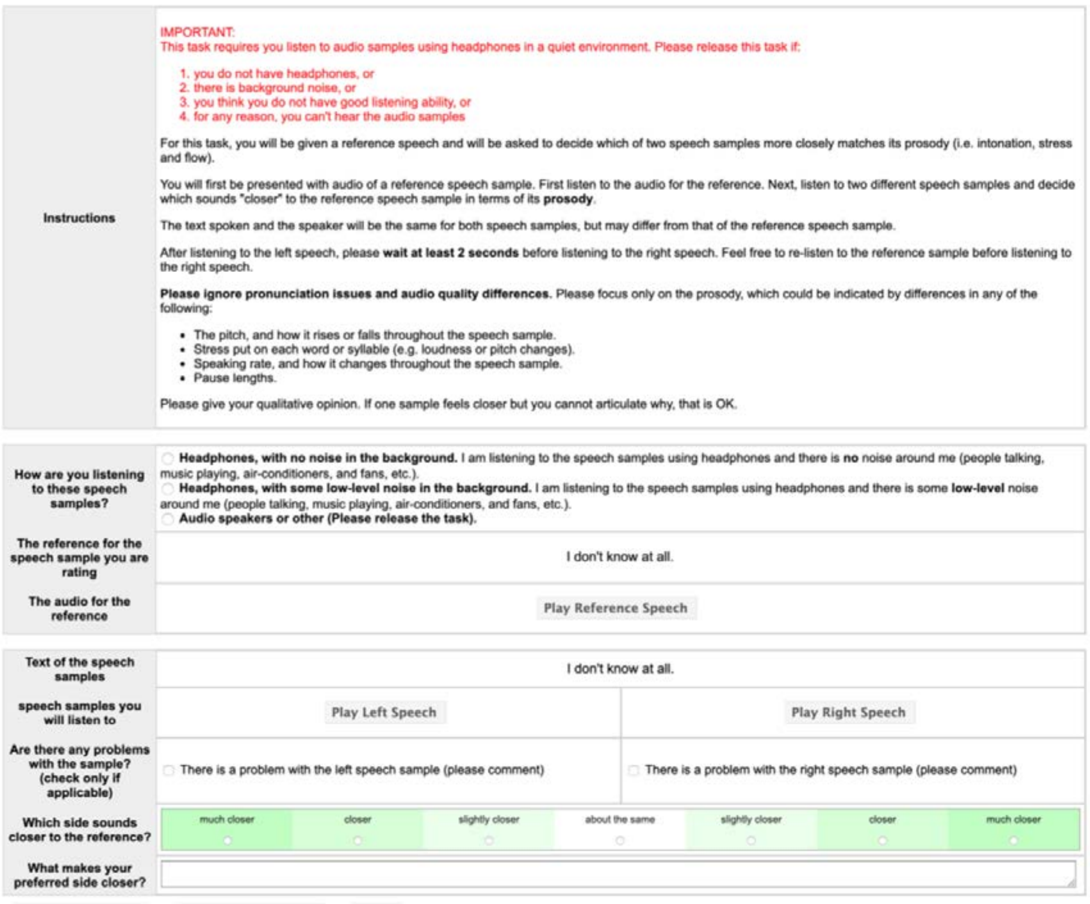
where the inequality follows from the KL non-negativity.

The difference between the average KL and the mutual information is the aggregate KL:

$$
\begin{aligned}
R^{\text{AVG}} - I_q(\mathbf{X};\mathbf{Z}) &= \iint p_D(\mathbf{x})q(\mathbf{z}|\mathbf{x})\log\frac{q(\mathbf{z})}{p(\mathbf{z})}d\mathbf{x}d\mathbf{z} \\
&= \int q(\mathbf{z})\log\frac{q(\mathbf{z})}{p(\mathbf{z})}d\mathbf{z} \\
&= D_{\text{KL}}(q(\mathbf{z})\|p(\mathbf{z})) \quad \text{(AggregateKL)}
\end{aligned}
$$

C Listening Tests

C.1 Evaluation Templates from the Capacitron Method



Instructions

IMPORTANT:
This task requires you listen to audio samples using headphones in a quiet environment. Please release this task if:

1. you do not have headphones, or
2. there is background noise, or
3. you think you do not have good listening ability, or
4. for any reason, you can't hear the audio samples

For this task, you will be given a reference speech and will be asked to decide which of two speech samples more closely matches its prosody (i.e. intonation, stress and flow).

You will first be presented with audio of a reference speech sample. First listen to the audio for the reference. Next, listen to two different speech samples and decide which sounds "closer" to the reference speech sample in terms of its **prosody**.

The text spoken and the speaker will be the same for both speech samples, but may differ from that of the reference speech sample.

After listening to the left speech, please **wait at least 2 seconds** before listening to the right speech. Feel free to re-listen to the reference sample before listening to the right speech.

Please ignore pronunciation issues and audio quality differences. Please focus only on the prosody, which could be indicated by differences in any of the following:

- The pitch, and how it rises or falls throughout the speech sample.
- Stress put on each word or syllable (e.g. loudness or pitch changes).
- Speaking rate, and how it changes throughout the speech sample.
- Pause lengths.

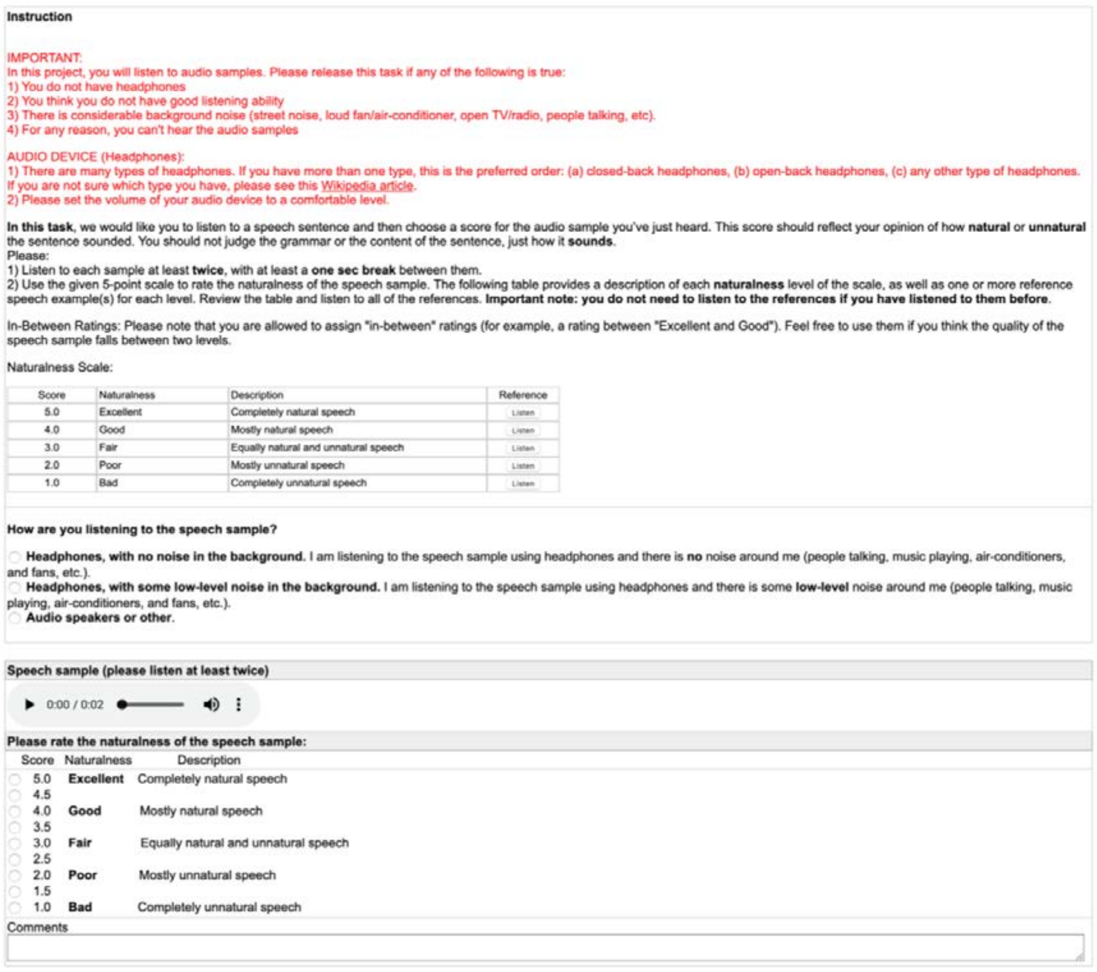
Please give your qualitative opinion. If one sample feels closer but you cannot articulate why, that is OK.

How are you listening to these speech samples?	○ **Headphones, with no noise in the background.** I am listening to the speech samples using headphones and there is **no** noise around me (people talking, music playing, air-conditioners, and fans, etc.). ○ **Headphones, with some low-level noise in the background.** I am listening to the speech samples using headphones and there is some **low-level** noise around me (people talking, music playing, air-conditioners, and fans, etc.). ○ **Audio speakers or other (Please release the task).**
The reference for the speech sample you are rating	I don't know at all.
The audio for the reference	Play Reference Speech

Text of the speech samples	I don't know at all.	
speech samples you will listen to	Play Left Speech	Play Right Speech
Are there any problems with the sample? (check only if applicable)	☐ There is a problem with the left speech sample (please comment)	☐ There is a problem with the right speech sample (please comment)

Which side sounds closer to the reference?	much closer ○	closer ○	slightly closer ○	about the same ○	slightly closer ○	closer ○	much closer ○
What makes your preferred side closer?							

Figure C.1: Evaluation templates from the original Capacitron paper for the AXY reference tests [8]. These templates were used as blueprints for the custom made online listening evaluation used in this work.



Instruction

IMPORTANT:
In this project, you will listen to audio samples. Please release this task if any of the following is true:
1) You do not have headphones
2) You think you do not have good listening ability
3) There is considerable background noise (street noise, loud fan/air-conditioner, open TV/radio, people talking, etc).
4) For any reason, you can't hear the audio samples

AUDIO DEVICE (Headphones):
1) There are many types of headphones. If you have more than one type, this is the preferred order: (a) closed-back headphones, (b) open-back headphones, (c) any other type of headphones. If you are not sure which type you have, please see this Wikipedia article.
2) Please set the volume of your audio device to a comfortable level.

In this task, we would like you to listen to a speech sentence and then choose a score for the audio sample you've just heard. This score should reflect your opinion of how **natural** or **unnatural** the sentence sounded. You should not judge the grammar or the content of the sentence, just how it **sounds**.
Please:
1) Listen to each sample at least **twice**, with at least a **one sec break** between them.
2) Use the given 5-point scale to rate the naturalness of the speech sample. The following table provides a description of each **naturalness** level of the scale, as well as one or more reference speech example(s) for each level. Review the table and listen to all of the references. **Important note: you do not need to listen to the references if you have listened to them before.**

In-Between Ratings: Please note that you are allowed to assign "in-between" ratings (for example, a rating between "Excellent and Good"). Feel free to use them if you think the quality of the speech sample falls between two levels.

Naturalness Scale:

Score	Naturalness	Description	Reference
5.0	Excellent	Completely natural speech	Listen
4.0	Good	Mostly natural speech	Listen
3.0	Fair	Equally natural and unnatural speech	Listen
2.0	Poor	Mostly unnatural speech	Listen
1.0	Bad	Completely unnatural speech	Listen

How are you listening to the speech sample?

- ○ **Headphones, with no noise in the background.** I am listening to the speech sample using headphones and there is **no** noise around me (people talking, music playing, air-conditioners, and fans, etc.).
- ○ **Headphones, with some low-level noise in the background.** I am listening to the speech sample using headphones and there is some **low-level** noise around me (people talking, music playing, air-conditioners, and fans, etc.).
- ○ **Audio speakers or other.**

Speech sample (please listen at least twice)

▶ 0:00 / 0:02

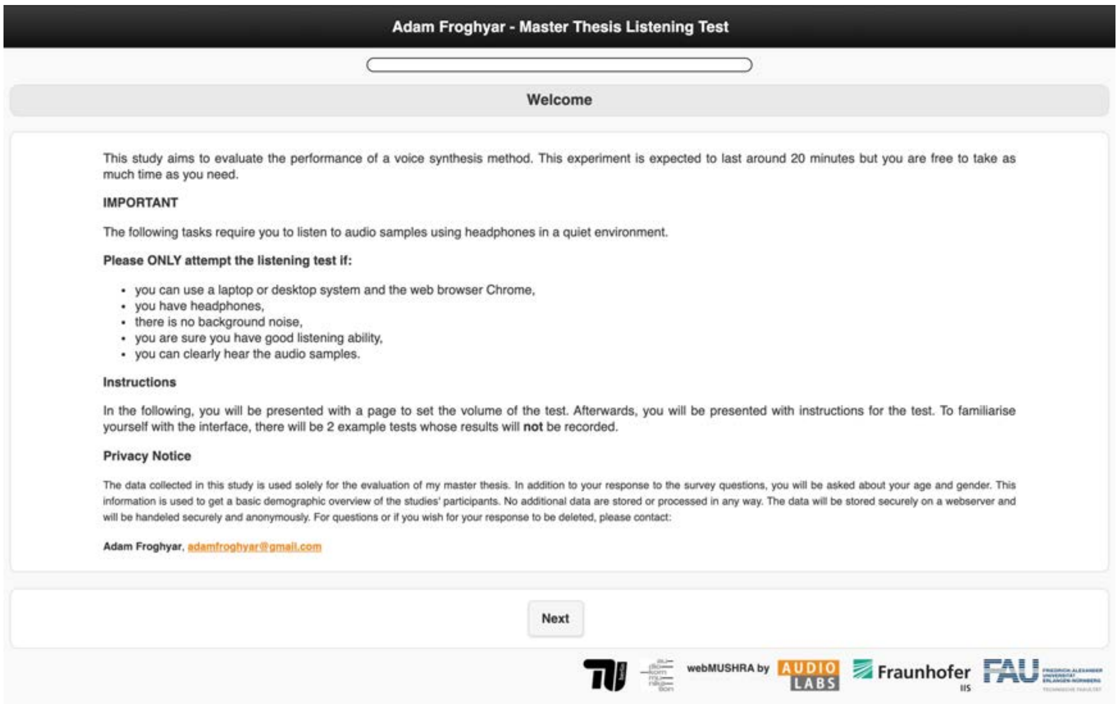
Please rate the naturalness of the speech sample:

	Score	Naturalness	Description
○	5.0	**Excellent**	Completely natural speech
○	4.5		
○	4.0	**Good**	Mostly natural speech
○	3.5		
○	3.0	**Fair**	Equally natural and unnatural speech
○	2.5		
○	2.0	**Poor**	Mostly unnatural speech
○	1.5		
○	1.0	**Bad**	Completely unnatural speech

Comments

Figure C.2: Evaluation templates from the original Capacitron paper for the MOS naturalness tests [8].

C.2 Recreated Evaluation Template Screenshots from the custom-made webMUSHRA Interface


Adam Froghyar - Master Thesis Listening Test

Welcome

This study aims to evaluate the performance of a voice synthesis method. This experiment is expected to last around 20 minutes but you are free to take as much time as you need.

IMPORTANT

The following tasks require you to listen to audio samples using headphones in a quiet environment.

Please ONLY attempt the listening test if:

- you can use a laptop or desktop system and the web browser Chrome,
- you have headphones,
- there is no background noise,
- you are sure you have good listening ability,
- you can clearly hear the audio samples.

Instructions

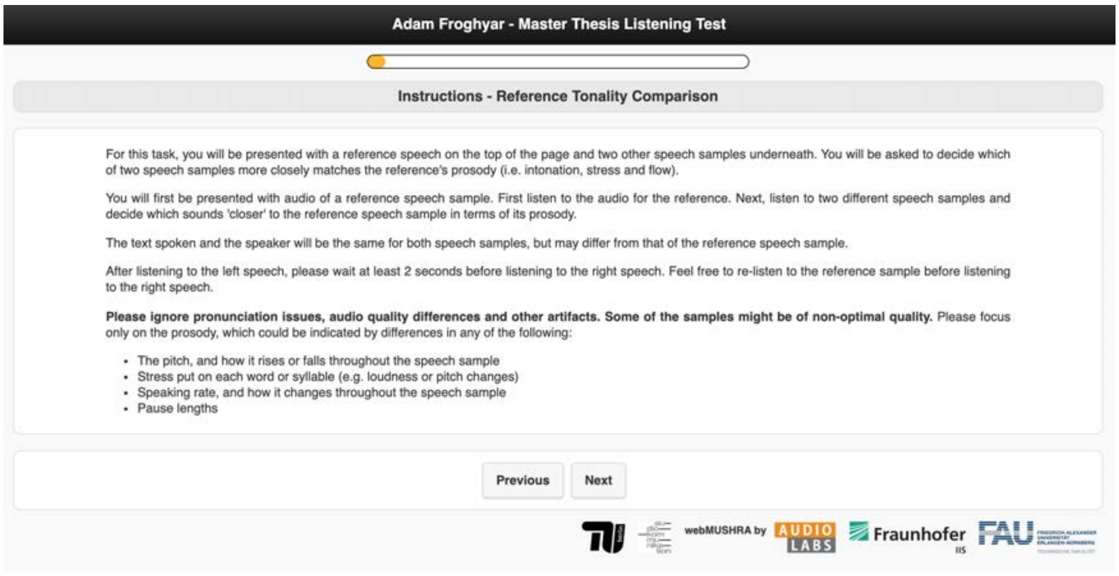
In the following, you will be presented with a page to set the volume of the test. Afterwards, you will be presented with instructions for the test. To familiarise yourself with the interface, there will be 2 example tests whose results will **not** be recorded.

Privacy Notice

The data collected in this study is used solely for the evaluation of my master thesis. In addition to your response to the survey questions, you will be asked about your age and gender. This information is used to get a basic demographic overview of the studies' participants. No additional data are stored or processed in any way. The data will be stored securely on a webserver and will be handeled securely and anonymously. For questions or if you wish for your response to be deleted, please contact:

Adam Froghyar, adamfroghyar@gmail.com

Figure C.3: Screenshot of the welcome page of the custom made listening test using the webMUSHRA [9] framework.


Adam Froghyar - Master Thesis Listening Test

Instructions - Reference Tonality Comparison

For this task, you will be presented with a reference speech on the top of the page and two other speech samples underneath. You will be asked to decide which of two speech samples more closely matches the reference's prosody (i.e. intonation, stress and flow).

You will first be presented with audio of a reference speech sample. First listen to the audio for the reference. Next, listen to two different speech samples and decide which sounds 'closer' to the reference speech sample in terms of its prosody.

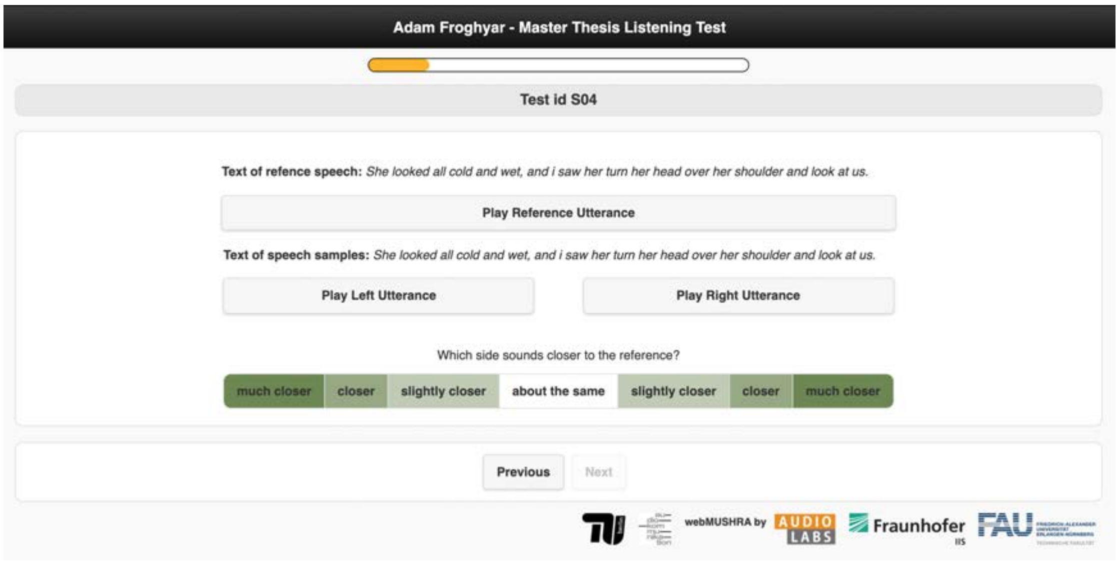
The text spoken and the speaker will be the same for both speech samples, but may differ from that of the reference speech sample.

After listening to the left speech, please wait at least 2 seconds before listening to the right speech. Feel free to re-listen to the reference sample before listening to the right speech.

Please ignore pronunciation issues, audio quality differences and other artifacts. Some of the samples might be of non-optimal quality. Please focus only on the prosody, which could be indicated by differences in any of the following:

- The pitch, and how it rises or falls throughout the speech sample
- Stress put on each word or syllable (e.g. loudness or pitch changes)
- Speaking rate, and how it changes throughout the speech sample
- Pause lengths

Figure C.4: Screenshot of the AXY Instructions Page presented to the listeners in the first round of listening tests.


Adam Froghyar - Master Thesis Listening Test

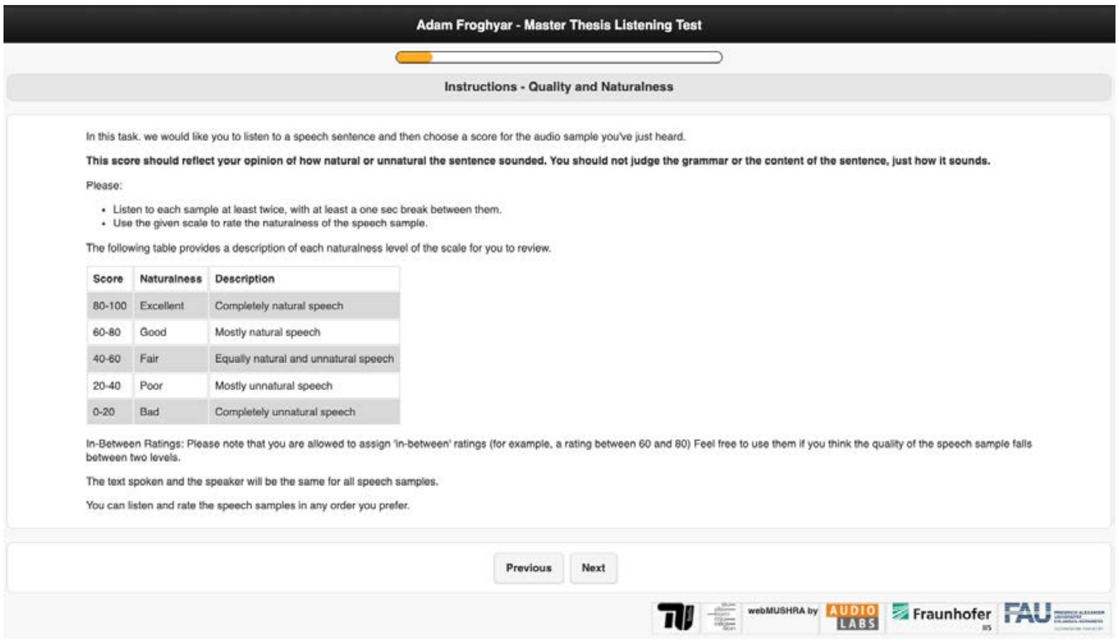
Test id S04

Text of refence speech: *She looked all cold and wet, and i saw her turn her head over her shoulder and look at us.*

Text of speech samples: *She looked all cold and wet, and i saw her turn her head over her shoulder and look at us.*

Which side sounds closer to the reference?

Figure C.5: Screenshot of one of the AXY Test Pages presented to participants.


Adam Froghyar - Master Thesis Listening Test

Instructions - Quality and Naturalness

In this task, we would like you to listen to a speech sentence and then choose a score for the audio sample you've just heard.

This score should reflect your opinion of how natural or unnatural the sentence sounded. You should not judge the grammar or the content of the sentence, just how it sounds.

Please:

- Listen to each sample at least twice, with at least a one sec break between them.
- Use the given scale to rate the naturalness of the speech sample.

The following table provides a description of each naturalness level of the scale for you to review.

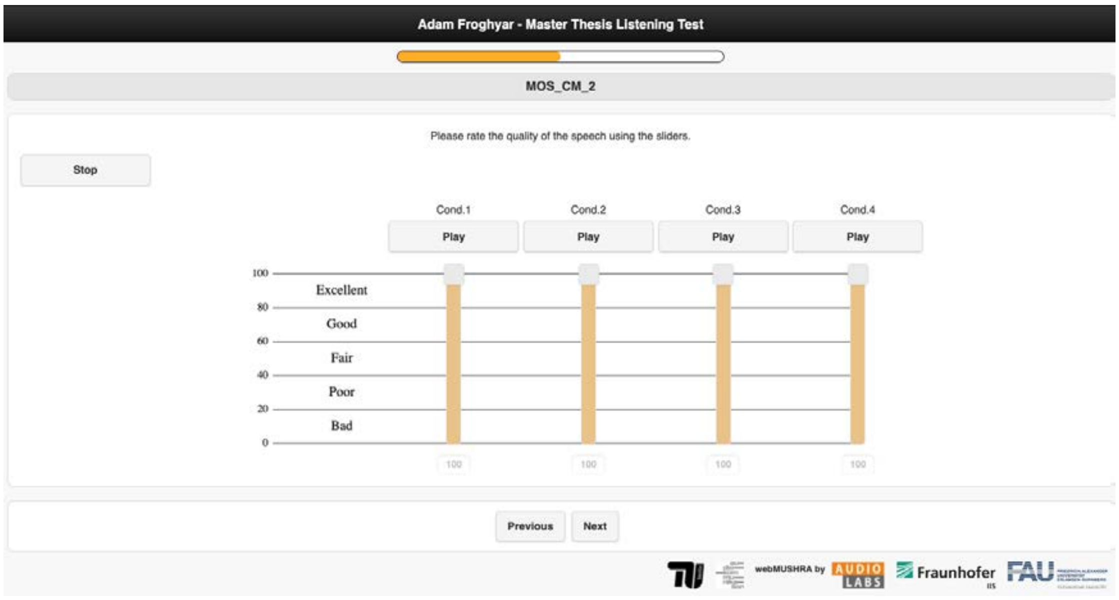
Score	Naturalness	Description
80-100	Excellent	Completely natural speech
60-80	Good	Mostly natural speech
40-60	Fair	Equally natural and unnatural speech
20-40	Poor	Mostly unnatural speech
0-20	Bad	Completely unnatural speech

In-Between Ratings: Please note that you are allowed to assign 'in-between' ratings (for example, a rating between 60 and 80) Feel free to use them if you think the quality of the speech sample falls between two levels.

The text spoken and the speaker will be the same for all speech samples.

You can listen and rate the speech samples in any order you prefer.

Figure C.6: Screenshot of the MOS Instructions Page presented to the listeners in the second round of listening tests.



Please rate the quality of the speech using the sliders.

Figure C.7: Screenshot of one of the MOS Test Pages presented to participants.

D Code

D.1 Capacitron Layer

D.1.1 Main Class

```python
class CapacitronVAE(nn.Module):
    def __init__(self,
                 num_mel,
                 capacitron_embedding_dim,
                 encoder_output_dim=256,
                 reference_encoder_out_dim=128,
                 speaker_embedding_dim=None,
                 text_summary_embedding_dim=None):
        super().__init__()
        self.prior_distribution = MVN(torch.zeros(capacitron_embedding_dim), torch.eye(capacitron_embedding_dim))
        self.approximate_posterior_distribution = None
        self.encoder = ReferenceEncoder(num_mel, out_dim=reference_encoder_out_dim)

        self.beta = torch.nn.Parameter(torch.log(torch.exp(torch.Tensor([1.0])) - 1), requires_grad=True)
        mlp_input_dimension = reference_encoder_out_dim

        if text_summary_embedding_dim is not None:
            self.text_summary_net = TextSummary(text_summary_embedding_dim, encoder_output_dim=encoder_output_dim)
            mlp_input_dimension += text_summary_embedding_dim
        if speaker_embedding_dim is not None:
            mlp_input_dimension += speaker_embedding_dim
        self.post_encoder_mlp = PostEncoderMLP(mlp_input_dimension, capacitron_embedding_dim)
    def forward(self, reference_mel_info=None, text_info=None, speaker_embedding=None):
        # Use reference
        if reference_mel_info is not None:
            reference_mels = reference_mel_info[0] # [batch_size, num_frames, num_mels]
            mel_lengths = reference_mel_info[1] # [batch_size]
            enc_out = self.encoder(reference_mels, mel_lengths)

            # concat speaker_embedding and/or text summary embedding
            if text_info is not None:
                text_inputs = text_info[0] # [batch_size, num_characters, num_embedding]
                input_lengths = text_info[1]
                text_summary_out = self.text_summary_net(text_inputs, input_lengths).to(reference_mels.device)
                enc_out = torch.cat([enc_out, text_summary_out], dim=-1)
            if speaker_embedding is not None:
                enc_out = torch.cat([enc_out, speaker_embedding], dim=-1)

            # Feed the output of the ref encoder and information about text/speaker into
            # an MLP to produce the parameteres for the approximate poterior distributions
            mu, sigma = self.post_encoder_mlp(enc_out)
            # convert to cpu because prior_distribution was created on cpu
            mu = mu.cpu()
            sigma = sigma.cpu()

            # Sample from the posterior: z ~ q(z|x)
            self.approximate_posterior_distribution = MVN(mu, torch.diag_embed(sigma))
            VAE_embedding = self.approximate_posterior_distribution.rsample()
        # Infer from the model, bypasses encoding
        else:
            # Sample from the prior: z ~ p(z)
            VAE_embedding = self.prior_distribution.sample().unsqueeze(0)

        # reshape to [batch_size, 1, capacitron_embedding_dim]
        return VAE_embedding.unsqueeze(1), self.approximate_posterior_distribution, self.prior_distribution, self.beta
```

D.1.2 Reference Encoder

```python
class ReferenceEncoder(nn.Module):
    """NN module creating a fixed size prosody embedding from a spectrogram.
    inputs: mel spectrograms [batch_size, num_spec_frames, num_mel]
    outputs: [batch_size, embedding_dim]
    """

    def __init__(self, num_mel, out_dim):

        super().__init__()
        self.num_mel = num_mel
        filters = [1] + [32, 32, 64, 64, 128, 128]
        num_layers = len(filters) - 1
        convs = [
            nn.Conv2d(
                in_channels=filters[i],
                out_channels=filters[i + 1],
                kernel_size=(3, 3),
                stride=(2, 2),
                padding=(2, 2)) for i in range(num_layers)
        ]
        self.convs = nn.ModuleList(convs)
        self.training = False
        self.bns = nn.ModuleList([
            nn.BatchNorm2d(num_features=filter_size)
            for filter_size in filters[1:]
        ])

        post_conv_height = self.calculate_post_conv_height(
            num_mel, 3, 2, 2, num_layers)
        self.recurrence = nn.LSTM(
            input_size=filters[-1] * post_conv_height,
            hidden_size=out_dim,
            batch_first=True,
            bidirectional=False)

    def forward(self, inputs, input_lengths):
        batch_size = inputs.size(0)
        x = inputs.view(batch_size, 1, -1, self.num_mel) # [batch_size,
        ↪  num_channels==1, num_frames, num_mel]
        valid_lengths = input_lengths.float() # [batch_size]
        for conv, bn in zip(self.convs, self.bns):
            x = conv(x)
            x = bn(x)
            x = F.relu(x)

            # Create the post conv width mask based on the valid lengths of
            ↪  the output of the convolution.
            # The valid lengths for the output of a convolution on varying
            ↪  length inputs is
            # ceil(input_length/stride) + 1 for stride=3 and padding=2
            # For example (kernel_size=3, stride=2, padding=2):
            # 0 0 x x x x x 0 0 -> Input = 5, 0 is zero padding, x is valid
            ↪  values coming from padding=2 in conv2d
            # ______
            #   x ______
            #       x ______
            #           x  _____
            #               x
            # x x x x -> Output valid length = 4
            # Since every example in te batch is zero padded and therefore
            ↪  have separate valid_lengths,
            # we need to mask off all the values AFTER the valid length for
            ↪  each example in the batch.
            # Otherwise, the convolutions create noise and a lot of not real
            ↪  information
            valid_lengths = (valid_lengths/2).float()
            valid_lengths = torch.ceil(valid_lengths).to(dtype=torch.int64)
            ↪  + 1 # 2 is stride -- size: [batch_size]
            post_conv_max_width = x.size(2)

            mask = torch.arange(post_conv_max_width)
                        .to(inputs.device)
                        .expand(len(valid_lengths), post_conv_max_width) <
                        ↪  valid_lengths.unsqueeze(1)
            mask = mask.expand(1, 1, -1, -1).transpose(2, 0).transpose(-1,
            ↪  2) # [batch_size, 1, post_conv_max_width, 1]
            x = x*mask

        x = x.transpose(1, 2)
        # x: 4D tensor [batch_size, post_conv_width,
        #               num_channels==128, post_conv_height]

        post_conv_width = x.size(1)
        x = x.contiguous().view(batch_size, post_conv_width, -1)
        # x: 3D tensor [batch_size, post_conv_width,
        #               num_channels*post_conv_height]

        # Routine for fetching the last valid output of a dynamic LSTM with
        ↪  varying input lengths and padding
        post_conv_input_lengths = valid_lengths
        packed_seqs = nn.utils.rnn.pack_padded_sequence(x,
        ↪  post_conv_input_lengths.tolist(), batch_first=True,
        ↪  enforce_sorted=False) # dynamic rnn sequence padding
        self.recurrence.flatten_parameters()
        _, (ht, _) = self.recurrence(packed_seqs)
        last_output = ht[-1]

        return last_output.to(inputs.device) # [B, 128]

    @staticmethod
    def calculate_post_conv_height(height, kernel_size, stride, pad,
                                   n_convs):
        """Height of spec after n convolutions with fixed
        ↪  kernel/stride/pad."""
        for _ in range(n_convs):
            height = (height - kernel_size + 2 * pad) // stride + 1
        return height
```

D.1.3 Utility Classes

```python
class TextSummary(nn.Module):
    def __init__(self, embedding_dim, encoder_output_dim):
        super().__init__()
        self.lstm = nn.LSTM(encoder_output_dim, # text embedding dimension from the text encoder
                            embedding_dim, # fixed length output summary the lstm creates from the input
                            batch_first=True,
                            bidirectional=False)

    def forward(self, inputs, input_lengths):
        # Routine for fetching the last valid output of a dynamic LSTM with varying input lengths and padding
        packed_seqs = nn.utils.rnn.pack_padded_sequence(inputs, input_lengths.tolist(), batch_first=True, enforce_sorted=False) # dynamic rnn sequence padding
        self.lstm.flatten_parameters()
        _, (ht, _) = self.lstm(packed_seqs)
        last_output = ht[-1]
        return last_output

class PostEncoderMLP(nn.Module):
    def __init__(self, input_size, hidden_size):
        super().__init__()
        self.hidden_size = hidden_size
        modules = [
            nn.Linear(input_size, hidden_size), # Hidden Layer
            nn.Tanh(),
            nn.Linear(hidden_size, hidden_size * 2)] # Output layer twice the size for mean and variance
        self.net = nn.Sequential(*modules)
        self.softplus = nn.Softplus()

    def forward(self, _input):
        mlp_output = self.net(_input)
        # The mean parameter is unconstrained
        mu = mlp_output[:, :self.hidden_size]
        # The standard deviation must be positive. Parameterise with a softplus
        sigma = self.softplus(mlp_output[:, self.hidden_size:])
        return mu, sigma
```

D.2 Double Optimisation

```
class CapacitronLoss(torch.nn.Module):

    def __init__(self, c, stopnet_pos_weight=10, ga_sigma=0.4):
        super(CapacitronLoss, self).__init__()
        self.stopnet_pos_weight = stopnet_pos_weight
        self.decoder_alpha = c.decoder_loss_alpha
        self.postnet_alpha = c.postnet_loss_alpha
        self.config = c

        # Main Capacitron loss
        self.criterion = nn.L1Loss(reduction='sum')
        self.postnet_criterion = nn.L1Loss()
        # stopnet loss
        self.criterion_st = BCELossMasked(
            pos_weight=torch.tensor(stopnet_pos_weight)) if c.stopnet else
            ↪  None


    def forward(self, postnet_output, decoder_output, mel_input,
    ↪  linear_input,
                stopnet_output, stopnet_target, output_lens,
                ↪  decoder_b_output,
                alignments, alignment_lens, alignments_backwards,
                ↪  input_lens,
                capacity, posterior_distribution, prior_distribution, beta):

        # decoder outputs linear or mel spectrograms for Tacotron
        postnet_target = linear_input if self.config.model.lower() in
        ↪  ["tacotron"] else mel_input

        return_dict = {}

        # decoder and postnet losses
        if self.decoder_alpha > 0:
            decoder_loss = self.criterion(decoder_output, mel_input) /
            ↪  decoder_output.size(0)
        if self.postnet_alpha > 0:
            postnet_loss = self.postnet_criterion(postnet_output,
            ↪  postnet_target) #/ postnet_output.size(0)

        # KL divergence term between the posterior and the prior
        kl_term =
        ↪  torch.mean(torch.distributions.kl_divergence(posterior_distribution,
        ↪  prior_distribution))

```

```python
        # VAE Loss: We use the l1 decoder loss as a stand-in for the negative log likelihood,
        # summed over all dimensions and normalised by the batch size
        negative_log_likelihood = self.decoder_alpha * decoder_loss

        # pass beta through softplus
        beta = torch.nn.functional.softplus(beta)

        # This is the term going to the main ADAM optimiser, we detach beta because
        # beta is optimised by a separate, SGD optimiser below
        reconstruction_loss = negative_log_likelihood + beta.detach() * (kl_term - capacity)

        # This is the term to purely optimise beta and to pass into the SGD
        beta_loss = torch.negative(beta) * (kl_term - capacity).detach()

        return_dict['decoder_loss'] = decoder_loss
        return_dict['postnet_loss'] = postnet_loss
        return_dict['reconstruction_loss'] = reconstruction_loss
        return_dict['beta_loss'] = beta_loss
        return_dict['kl_term'] = kl_term
        return_dict['capacitron_beta'] = beta

        # Expand loss
        loss = reconstruction_loss + self.postnet_alpha * postnet_loss

        # Stopnet loss
        stop_loss = self.criterion_st(
            stopnet_output, stopnet_target,
            output_lens) if self.config.stopnet else torch.zeros(1)
        if not self.config.separate_stopnet and self.config.stopnet:
            loss += stop_loss

        return_dict['stopnet_loss'] = stop_loss

        return_dict['loss'] = loss

        # check if any loss is NaN
        for key, loss in return_dict.items():
            if torch.isnan(loss):
                raise RuntimeError(f" [!] NaN loss with {key}.")
        return return_dict
```

E Subjective Evaluation Raw Results

E.0.1 AXY Tests

Same Text Prosody Transfer (STT)

Baseline/Competing Model	***C*=50;*E*=64**	***C*=50;*E*=128**
C=50-E=64	-	1.625
C=50-E=128	-1.625	-
C=150-E=64	-0.625	0.4625
C=150-E=128	-1.425	-0.05
C=300-E=64	-1.3	0.087
C=300-E=128	-1.55	-0.475

(a) $C = 50$

Baseline/Competing Model	***C*=150;*E*=64**	***C*=150;*E*=128**
C=50-E=64	0.625	1.425
C=50-E=128	-0.4625	0.05
C=150-E=64	-	1.425
C=150-E=128	-1.425	-
C=300-E=64	0.05	0.925
C=300-E=128	0.525	0.9

(b) $C = 150$

Baseline/Competing Model	***C*=300;*E*=64**	***C*=300;*E*=128**
C=50-E=64	1.3	1.55
C=50-E=128	-0.087	0.475
C=150-E=64	-0.05	-0.525
C=150-E=128	-0.925	-0.9
C=300-E=64	-	-0.475
C=300-E=128	0.475	-

(c) $C = 300$

Table E.1: Mean scores for the STT-AXY-evaluation of models with different variational capacity C

Inter Text Style Transfer (ITT)

Baseline/Competing Model	C=50;E=64	C=50;E=128
C=50-E=64	-	-0.975
C=50-E=128	0.975	-
C=150-E=64	-0.2	-0.525
C=150-E=128	1.2	-0.15
C=300-E=64	1.275	-0.975
C=300-E=128	0.85	0.425

(a) $C = 50$

Baseline/Competing Model	C=150;E=64	C=150;E=128
C=50-E=64	0.2	-1.2
C=50-E=128	0.525	0.15
C=150-E=64	-	-0.575
C=150-E=128	0.575	-
C=300-E=64	-0.375	0.05
C=300-E=128	0.05	-0.275

(b) $C = 150$

Baseline/Competing Model	C=300;E=64	C=300;E=128
C=50-E=64	-1.275	-0.85
C=50-E=128	0.975	-0.425
C=150-E=64	0.375	-0.05
C=150-E=128	-0.05	0.275
C=300-E=64	-	-0.5
C=300-E=128	0.5	-

(c) $C = 300$

Table E.2: Mean scores for the ITT-AXY-evaluation of models with different variational capacity C

E.0.2 MOS Tests

Constant Synthesis Type

Synthesis Type	**Ground Truth**	***C*=150**	***C*=300**
STT	4.4198	3.0930	3.2791
ITT	4.4744	1.9977	2.5587
Prior	4.6907	2.3825	2.6645

Table E.3: MOS naturalness scores for constant synthesis type comparison tests between models with $C = 150$ and $C = 300$

Constant Model

Model	**Ground Truth**	**STT**	**ITT**	**Prior**
***C*=150**	4.5692	2.7965	2.6506	2.5971
***C*=300**	4.4064	2.6389	2.4634	2.6500

Table E.4: MOS naturalness scores for constant model type comparison tests between models with $C = 150$ and $C = 300$