KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Controllable Expressive Speech Synthesis

Tobias Cornille

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdoptie Artificiële intelligentie

Promotoren:

Prof. dr. Luc De Raedt
Dr. ir. Jessa Bekker

Assessoren:

Prof. dr. Angelika Kimmig
Dr. Nyi Nyi Htun

Begeleiders:

Dr. ir. Vincent Pagel
Dr. Fengna Wang

Academiejaar 2020 – 2021

Preface

Speech synthesis technology helps many people with and without disabilities around the world, and I believe that the technology still has a lot of untapped potential. Therefore, I am very grateful that I got the opportunity to contribute to this exciting research field during the last year of my computer science education. However, I could not have done this alone, so some words of thanks are in order.

First and foremost, I want to thank my daily supervisor Jessa Bekker. Her guidance and insights helped shape this work into what it is today. Many thanks for encouraging me when I inevitably hit some roadblocks, and for all the helpful feedback throughout the year!

Furthermore, I would like to thank Vincent Pagel and Fengna Wang (from Acapela Group) for kindly providing me with their dataset and tools, and for the insightful remarks and discussions. I also want to thank my supervisor, prof. De Raedt, for his feedback on the presentations, and Thomas Winters for sharing his ideas on multiple occasions.

Next, thanks are due to my friends and family. It has not been an easy year, with a relentless pandemic impacting everyone's life, so their support has meant a lot to me. Special thanks to Sebastian for his feedback and to Simon for proofreading this entire text.

Lastly, I would like to thank all educators and content creators that have inspired me and helped me to pursue a career in STEM. Their interesting videos, insightful blog posts, and great classes have given me the knowledge and skills to carry out this work.

The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

Contents

Abstract

Speech synthesis, the artificial production of speech, is indispensable as assistive technology. Speech-generating devices help people with speech impairments communicate, screen readers help people with visual and reading impairments understand, and virtual assistants help many people live their daily lives.

Recent neural-based text-to-speech models are able to produce natural speech from some input text. However, humans use speech to express more than what is written down. Thus, many techniques have also been developed to make the synthesized speech more expressive. These techniques model and control the prosody (intonation, stress, and rhythm) of the synthesized speech. However, the controllability of these techniques is still limited. On the one hand, many techniques only allow control over the global style of the speech and do not allow fine-grained adjustments. On the other hand, manipulating fine-grained prosody attributes in order to obtain a desired speaking style is difficult.

The aim of this thesis is thus to create a model that can produce speech in a certain speaking style, while also allowing local adjustments to the prosody of the generated speech. To that end, ConEx, a novel model for controllable expressive speech synthesis, is proposed. ConEx builds on the non-autoregressive architecture of FastSpeech, but adds components to control and model prosody. In particular, a reference encoder is used to learn global prosody embeddings, and a vector quantized variational autoencoder is used to create fine-grained prosody embeddings. Furthermore, an autoregressive prior model is trained over the fine-grained prosody embeddings. A new method to edit local prosody is proposed, which uses the predictions from this prior model.

The controllability of the proposed model is evaluated on two datasets. Experiments show that the model can indeed be used to control the global speaking style and change the local prosody of the synthesized speech. However, the kind of dataset that is used strongly influences the success of the control. Audio samples from the experiments are available on the demo page[1].

[1] `https://bit.ly/conex-samples`

Samenvatting

Spraaksynthese, of het kunstmatig genereren van spraak, is onmisbaar als vorm van assistieve technologie. Spraakcomputers helpen personen met een spraakbeperking, schermlezers helpen personen met een visuele beperking of leesstoornissen, en virtuele assistenten kunnen ons allemaal helpen.

Recente *text-to-speech*-modellen op basis van neurale netwerken slagen erin een tekst om te zetten naar natuurlijke spraak. Menselijke spraak kan echter meer dan woorden alleen uitdrukken. Daarom zijn er ook technieken ontwikkeld om meer expressieve spraak te produceren. Deze technieken modelleren en controleren de prosodie (intonatie, klemtoon, en ritme) van de gesynthetiseerde spraak. Het niveau van controle over de prosodie is echter beperkt bij deze technieken. Vele technieken kunnen enkel de globale spreekstijl aanpassen en laten dus geen lokale wijzigingen toe. Er bestaan wel technieken die de prosodie op een fijner niveau modelleren, maar daarbij is het dan weer moeilijk om een bepaalde spreekstijl te bekomen.

Het doel van deze thesis is dus om een model te ontwerpen dat spraak in een bepaalde stijl kan produceren en dat ook lokale prosodiewijzigingen toelaat. Het resultaat is ConEx, een nieuw model voor controleerbare expressieve spraaksynthese. ConEx bouwt verder op de niet-autoregressieve FastSpeech-architectuur en voegt componenten toe om prosodie te modelleren en controleren. Zo gebruikt ConEx een *reference encoder* om globale prosodie-embeddings te leren, en een *vector quantized variational autoencoder* om prosodie-embeddings op foneemniveau te leren. Verder wordt er ook een autoregressief *prior*model getraind op basis van de fijne prosodie-embeddings. Ten slotte, wordt er een methode voorgesteld die de voorspellingen van dit *prior*model gebruikt om prosodie lokaal aan te passen.

Deze thesis gebruikt twee verschillende datasets om te evalueren hoe controleerbaar het voorgestelde model is. Experimenten tonen aan dat het model in staat is om spraak in een bepaalde spreekstijl te genereren, en dat de prosodie lokaal gewijzigd kan worden. Welke dataset er gebruikt wordt, beïnvloedt echter sterk in welke mate beide aspecten controleerbaar zijn. Audiofragmenten van de experimenten zijn beschikbaar op de demo webpagina[1].

[1] `https://bit.ly/conex-samples`

List of Abbreviations and Symbols

Abbreviations

TTS	Text-to-speech
IPA	International Phonetic Alphabet
NLP	Natural language processing
dB	Decibel
seq2seq	Sequence-to-sequence
RNN	Recurrent neural network
GRU	Gated recurring unit
FFT	Feed-forward Transformer
CNN	Convolutional neural network
VAE	Variational autoencoder
VQ	Vector quantization
VQ-VAE	Vector quantized variational autoencoder
ReLU	Rectified linear unit
MAE	Mean absolute error
MSE	Mean squared error
MCD	Mel cepstral distortion
MFCC	Mel frequency cepstral coefficients
DTW	Dynamic time warping

Symbols

F_0	Fundamental frequency
$\boldsymbol{x}$	Observed variables
$\boldsymbol{z}$	Hidden variables
$\boldsymbol{e}$	Codebook embeddings
sg[·]	Stop gradient operator

Chapter 1

Introduction

1.1 Problem statement

Many great inventions have their origins in helping people with impairments. Pellegrino Turri invented the typewriter for a blind friend, so she could write more legibly. Vint Cerf helped create the email, as it allowed him and his wife, who both have hearing impairments, to communicate more easily (Kuang and Fabricant, 2019). Similarly, speech synthesis, which now powers the virtual assistants in our devices, has helped people with disabilities for decades.

Speech synthesis technology gives a voice to people who can not speak, such as people with severe speech impairments[1] or people with nonverbal autism. Moreover, speech synthesis is used in screen readers to help people with visual impairments and reading disorders. The technology has improved greatly over the years, with the latest neural network-based techniques generating highly natural speech.

Even with all the achieved progress, synthesized speech still does not match human speech in many situations. One major problem is that speech synthesis systems often generate dull, expressionless speech. Meanwhile, humans use expressive speech to convey much more than words. We can express emotions, communicate our stance on a topic, and even add additional meaning (e.g. irony) through our way of speaking. A goal of current speech synthesis research is thus to give speech synthesis systems the ability to produce more expressive speech. Furthermore, the expressive features of the synthesized speech should also be controllable, so that the right meaning and emotion can be expressed.

The key to expressive speech synthesis is to model and control the *prosody* of speech. Prosody refers to the collection of musical aspects of speech, including intonation, stress, and rhythm. Using prosody, speech can be produced with a desired speaking style in

[1] One of the most famous users of a speech-generating device was Stephen Hawking. He used the same standalone voice synthesizer from 1988 until 2014, when it needed to be replaced. Even then he wanted to keep the same voice, so Hawking's assistants had to track down an employee of the company that made the original synthesizer (which had gone out of business already) and emulate the original system (Medeiros, 2015).

order to convey the right emotion and meaning. Many techniques have been proposed to model prosody, but controlling the prosody remains an issue in many systems. One approach is to only model the global speaking style, but then it becomes impossible to make fine-grained edits (such as emphasizing a word). Another approach is to model and control prosody on a finer level. However, then it becomes difficult to manipulate all of the fine-grained prosody attributes to obtain a desired speaking style.

1.2 Contributions

The aim of this thesis is to create a speech synthesis system that enables control over the global speaking style of the generated speech, while also allowing local prosody adjustments. To this end, a number of contributions are made:

- ConEx, a novel model for controllable expressive speech, is proposed. This model extends FastSpeech (Ren et al., 2019), a state-of-the-art text-to-speech model. Specifically, components are added to model and control prosody.
- A prosody encoder is developed to model both the global speaking style and the prosody at a fine level. It consists of two parts. First, a reference encoder (Skerry-Ryan et al., 2018) is used to capture the global speaking style of speech utterances. Second, a vector quantized variational autoencoder (van den Oord et al., 2017) is used to learn discrete fine-grained prosody embeddings.
- A new method for making local edits to the prosody of the synthesized speech is introduced. This method allows a user to change the fine-grained prosody embeddings, and thus change the prosody locally, while still maintaining the naturalness of the output speech. The method uses the predictions of an autoregressive prior model trained over the fine-grained prosody embeddings (Sun et al., 2020).
- Experiments are carried out to evaluate the controllability of the proposed model. The results on two datasets show that the model can indeed be used to control the global speaking style and change the local prosody of the synthesized speech. Limitations related to the datasets used and the proposed local prosody editing technique are discussed.

1.3 Thesis outline

Chapter 2 gives an extensive overview of expressive speech synthesis. First, the problem of mapping text to speech is detailed. The next section describes techniques that break down the problem to make it easier to solve. Then, two early speech synthesis methods are described, namely concatenative speech synthesis and statistical parametric speech synthesis. Next, Tacotron and FastSpeech, two state-of-the-art neural text-to-speech methods, are detailed. Finally, different techniques to model and control prosody are described.

In the next chapter (chapter 3), ConEx, a novel model for controllable expressive speech synthesis, is proposed. First, the different components of the model are motivated. Next, the model is positioned in relation to some related works. Then ConEx is detailed, with a focus on the architecture, how the model is trained, how it can be used at inference, and what capabilities it offers for controlling prosody. Finally, the implementation of the model is briefly discussed.

Chapter 4 describes the different experiments carried out to validate the proposed model. The training set-up (including the datasets used) and methodology are first described. Then, the results of the initial experiment, which uses a simplified version of ConEx, are described. Afterwards, the results of the experiments evaluating the complete ConEx model are described. Both the level of control over the global speaking style and the ability to make local prosody edits are assessed in these experiments.

Chapter 2

Background

This chapter introduces the problem of speech synthesis. It describes two existing methods to make mapping text to speech easier. Next, it gives an overview of the different techniques that were developed to solve the text-to-speech mapping problem. Special attention is paid to current text-to-speech methods that use deep neural networks. Finally, this chapter gives an overview of methods to model and control prosody, which is the cornerstone of expressive speech synthesis.

2.1 The text-to-speech problem

The task of *speech synthesis* or *text-to-speech (TTS)* is to generate speech, in the form of an acoustic waveform, from some input text. To solve this problem, a computer program needs to know or learn the mapping between text on the one hand, and speech represented as a waveform on the other hand. At inference, when a certain text is input, the computer program uses that mapping to generate a speech utterance that corresponds to the input text.

Mapping text to speech is a one-to-many mapping problem, as a text fragment can be uttered in many different plausible ways. Intonation, stress, and rhythm can all vary independently of the text content. These attributes are collectively referred to as *prosody*. Speech synthesis systems need a way of modeling this prosody to solve the one-to-many mapping problem[1].

Moreover, by varying the prosody of an utterance, humans can convey information that is not included in the text. This information can, for example, express the emotion of the speaker, or can even change the meaning of the text drastically (e.g., in the case of irony). Controlling prosody is thus highly desirable and is the key to more expressive speech synthesis. Modeling and controlling prosody will be further detailed at the end of this chapter.

[1] *Allophones*, the different phones that represent the same phoneme (see further), are another source of variation when mapping text to speech, but this topic is out of scope of this work.

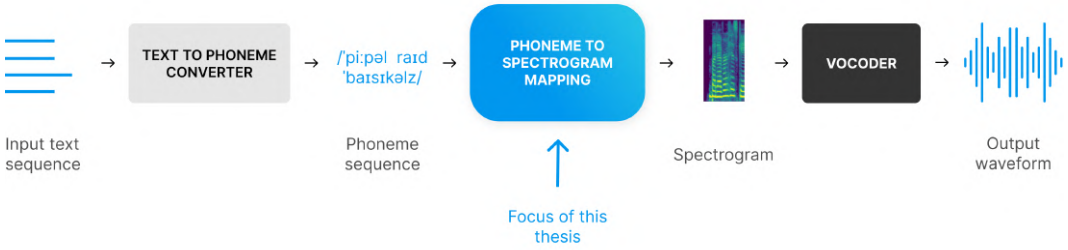


Figure 2.1: The components of a modern text-to-speech system.

2.2 Easing the text-to-speech mapping problem

Directly generating an acoustic waveform from text is challenging for a number of reasons. One issue is that text does not always indicate how a word should be pronounced. Another issue is that speech is represented digitally as tens of thousands samples per second. Generating that many samples accurately can be challenging for a text-to-speech system.

To solve these issues, text-to-speech systems typically consist of multiple different components. Figure 2.1 shows the different steps in a modern text-to-speech system. The text-to-phoneme conversion step can help with pronouncing text correctly, while using spectrograms as intermediate features can help with the issue of having to generate too many outputs per second. This section describes both solutions in more detail. The remainder of the thesis will focus on the phoneme-to-spectrogram mapping.

2.2.1 Text-to-phoneme conversion

How humans write down language varies widely around the world, and has an impact on how hard it is to map text to speech. Some languages like Chinese use *logograms*, characters that represent entire words. These give very little information about the pronunciation of the words they represent. Sound-based writing systems are closer to speech, and are used by many languages such as English, Arabic and Japanese. However, in some languages, like English, it is still hard to derive the correct pronunciation of a word solely from the way it is written.

To more easily map text to speech, it is therefore often helpful to first convert the input characters from standard alphabets to special characters from a phonetic alphabet. The characters in such an alphabet represent *phonemes* or *phones*[1] – segments of speech that represent distinct sounds. When using phonetic characters as the input to a speech synthesis system, the sounds of the characters are thus already determined, which makes generating speech easier.

[1] Phones and phonemes are actually somewhat different concepts. Phones and phonemes both represent the sounds speech is made up of, but a phone is only a phoneme when it can distinguish one word from another. For example, in Spanish, the “v” and “b” are different phones, but not different phonemes, as they can be used interchangeably. This thesis uses phonemes as the basic units of speech.

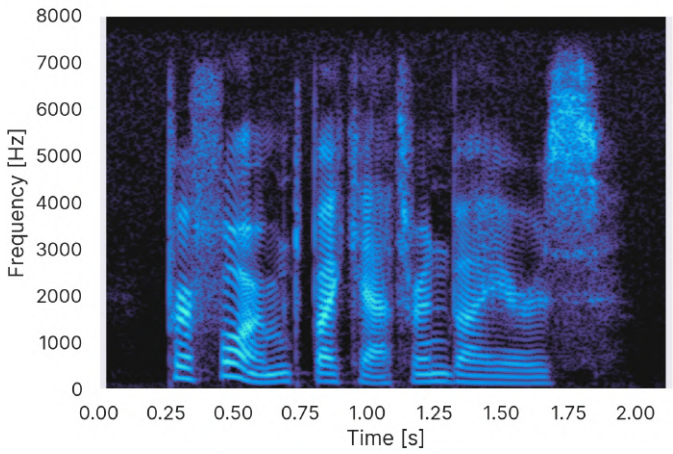


Figure 2.2: A mel spectrogram of the utterance "Performed by Catherine Byers".

Converting text to phoneme sequences can be achieved using phonetic dictionaries, rule-based algorithms, or more advanced machine learning methods. Dictionaries alone do not solve the problem, since real-world text often contains words that are not included in them (such as named entities or words with typos). Homographs can pose a problem as well. These are words that are pronounced differently in different contexts; e.g., "tear" in "I tear the paper" versus "a tear rolled down his face". Nonetheless, text to phoneme conversion is often used in text-to-speech systems.

It is important to note that it is still not trivial to map phonemes to actual waveforms, as phonemes do not determine the prosody of an utterance. For example, a phoneme like the /ʊ/ in /gʊd/ ("good"), can be uttered with a certain duration and at a certain pitch depending on the context (e.g. when it is part of a question). Techniques for modeling prosody are described at the end of this chapter.

2.2.2 Intermediate feature representation: the spectrogram

Speech is naturally an analog signal. Speech can be represented digitally by sampling the amplitude of the speech signal a certain number of times per second (=the *sampling rate*) and quantizing these samples (i.e., representing them as a number of bits). Typical sampling rates are 16, 22.05, or 44.1 kHz for voice recordings. This means that a text-to-speech system has to generate tens of thousands of output samples per second.

To deal with the complexity of generating that many samples, splitting the text-to-speech system in two separate steps is often well worth the effort. In the first step, a TTS system maps the input text/phonemes to a high-level intermediate feature representation of the output speech. Secondly, a separate system takes this representation and maps it to the samples that make up the final waveform.

A commonly used intermediate feature representation is the *mel spectrogram*. A *spectrogram* is a representation of an audio signal at lower time resolution than a waveform.

A spectrogram is a sequence of frames, each of which contains information on the different frequency components of a signal during a time step. For speech purposes, a mel spectrogram is often used. This is a spectrogram that uses the *mel scale* – a frequency scale that corresponds to human hearing (our hearing is not sensitive to all frequencies equally). Figure 2.2 shows a mel spectrogram.

A *vocoder* is a system that converts these mel spectrograms to a waveform, which is the final output of the speech synthesis process. mel spectrograms are coarser, more compressed speech representations than waveforms, so it is not trivial to perform this conversion. Algorithms such as the one introduced by Griffin and Lim (1984) are commonly used to approximate this conversion. More recent vocoders use more sophisticated learning techniques, such as deep neural networks. These can achieve better performance, as they are specifically trained for processing speech data.

Why the spectrogram? (optional reading)

The goal of an intermediate representation is to contain the most relevant speech information in a feature sequence that is as short as possible. This way, the TTS model has to generate far fewer output values. This section tries to explain why spectrograms are so widely used as an intermediate feature representation.

First, phonemes can be used to achieve a short feature sequence. The idea is to represent a group of waveform samples (a *frame*) by the phoneme that is being pronounced in those samples, as well as the prosody of that phoneme (*how* it is pronounced). The frames should be small enough so that most frames only contain one phoneme, but large enough, so that the frame sequence is short.

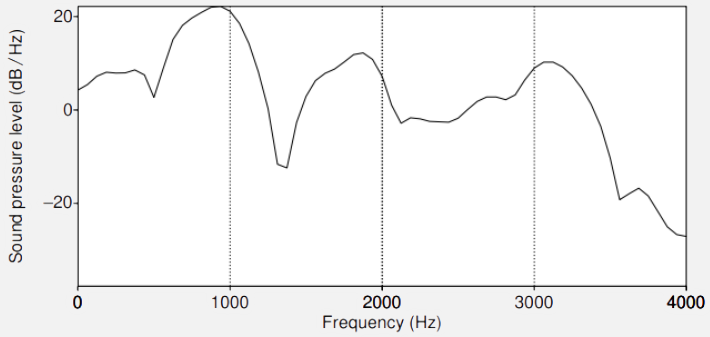


Figure 2.3: The spectrum of the vowel [ae]. Reprinted from Jurafsky and Martin (2009).

How can this phoneme-plus-prosody frame be represented so that a speech synthesis system can use it? The answer is found in spectral analysis. Fourier analysis shows that every wave can be decomposed as a sum of sinusoids of different frequencies. The representation of those different frequencies and their amplitudes is called the *spectrum* of a signal. It can be represented visually as a

graph where the x-axis shows the frequency, and the y-axis shows the amplitude of each frequency component (in dB). Figure 2.2.2 shows the spectrum of a frame containing a vowel. The spectrum of a frame can be efficiently computed using the Fast Fourier Transform algorithm.

The spectrum can now be used to represent a phoneme-plus-prosody frame. The reason for this is that each phoneme has a characteristic spectral fingerprint. By looking at the shape of the spectrum, the phoneme and the way the phoneme is pronounced can be detected. Thus, we can represent the different frames of a waveform as a series of coefficients, each containing the amplitude corresponding to a certain frequency band (Jurafsky and Martin, 2009).

The sequence of frames of a waveform can now be represented by a *spectrogram*. A spectrogram shows the spectrum (the frequencies and amplitudes) of each frame. A spectrogram's x-axis represents time in seconds, just like the waveform graph, but the y-axis now represents the frequency in Hertz. To show the amplitude of a frequency component of a frame, different colors are used.

2.3 Early speech synthesis methods

Many TTS methods have been developed over the years. Some approaches have tried to model the human vocal tract in detail, while others have focused more on manipulating waveform parameters to generate speech. A simple and successful technique, concatenative synthesis, is described first. Then the statistical parametric models are introduced, as they form the basis of modern techniques. Currently most research and applications make use of deep neural networks to synthesize speech. Because of their importance for this thesis, these modern methods are described extensively in the next section (2.4).

2.3.1 Concatenative synthesis

Concatenative synthesis is perhaps the easiest paradigm to understand. At its core, it tries to generate speech by stitching together prerecorded speech segments. The segments can be recordings of one phoneme, two phonemes (*diphone synthesis)*, or longer segments with no fixed length (*unit selection synthesis*). Sometimes, there might be different recordings corresponding to the same part of a text, to account for differences in prosody.

One of the main drawbacks of this paradigm is that it can cause unnatural sounding speech at the positions where two recordings are concatenated, although this can be partly overcome by smoothing techniques. Another downside is that this technique requires an extensive set of recordings of a single speaker to work, especially if recordings of entire words are used instead of recordings of phonemes. These recordings are also inflexible; if a different style of speech is desired, the whole recording process needs to be redone.

Despite these shortcomings, concatenative speech synthesis has seen great success, especially in applications within a specific domain, like weather reports or announcements in airports or train stations. Even more advanced applications like Siri still use concatenative approaches[1].

2.3.2 Statistical parametric speech synthesis

To overcome some of the limitations associated with concatenative speech synthesis, more flexible models were devised that try to learn the parameters used to synthesize speech using extensive amounts of speech data. These parameters include the frequency spectrum and duration associated with different phonemes. From the 1950s until the 1980s, researchers had already tried to generate speech using such features, resulting in a paradigm called *formant synthesis*. However, lacking machine learning techniques, they had to handcraft rules for these systems, which resulted in unnatural speech.

The emergence of *Hidden Markov Models*, which are generative sequential models, allowed speech parameters to be learned from data. The resulting models are more flexible than concatenative systems, as they are able to generate speech instead of using pre-recorded speech segments. This also allows different speech characteristics to be altered, without having to rerecord speech data. However, the generated speech is often still less natural than speech created using a concatenative approach.

2.4 Neural network-based TTS methods

In recent years, speech synthesis has been greatly improved by using deep neural networks. The core idea of neural network-based TTS is to train neural networks end-to-end on enormous amounts of raw data, i.e., on <text, audio> pairs. This way, handcrafted features requiring domain expertise are no longer necessary. With enough data, the neural networks learn the important features themselves. The resulting text-to-speech systems are less brittle, as they have fewer independent components where errors can compound.

The first neural-network based techniques for speech processing were introduced in the 1990s[2]. While showing promising results early on, a lack of computational resources meant these techniques could not compete with established techniques such as concatenative synthesis. The advent of graphics processing units (GPUs) and APIs like CUDA that made programming on GPUs more accessible, meant that deep neural networks could be trained much more efficiently. This resulted in a true deep learning revolution in the early 2010s, first in computer vision research, and later also in speech synthesis research.

[1] With the introduction of iOS 11 in 2017, Apple launched a new version of Siri that uses unit selection in combination with deep learning techniques (Capes et al., 2017).

[2] In fact, the first industrial application of deep learning was a system for speech recognition, developed by SRI International and launched in 1996. This shows that the field of speech processing has been at the frontier of deep learning for a long time.

One early breakthrough in neural network-based TTS was WaveNet (van den Oord et al., 2016), released by researchers at Google Deepmind in 2016. WaveNet is a deep neural network that generates raw waveforms and can be used in a TTS system, by using linguistic features as the input. Just one year after the WaveNet publication, Google decided to use WaveNet instead of a concatenative synthesis technique for the voice of Google Assistant, proving the practical use of the technique (van den Oord and Walters, 2017).

The last years have seen a staggering growth in the amount of research on neural network-based TTS, both from academia and from "Big Tech" firms. Adoption in real-world systems has also steadily increased. Next, two of the most important current speech synthesis models, Tacotron (Wang et al., 2017) and FastSpeech (Ren et al., 2019), are outlined and compared.

2.4.1 Tacotron

In an effort to move closer towards end-to-end speech synthesis, Google researchers Wang et al. (2017) proposed Tacotron, a model that takes raw text as input and outputs mel spectrograms[1]. While Tacotron does not generate raw waveforms, and thus is not a complete end-to-end TTS model, vocoders can be used to bridge the gap from mel spectrogram to waveform. These vocoders can even be trained jointly with Tacotron to create a true end-to-end speech synthesis system[2].

The architecture of Tacotron is based on the *sequence-to-sequence (seq2seq)* paradigm (Sutskever et al., 2014). A seq2seq model learns the mapping between an input sequence and an output sequence in two steps. First, the *encoder* creates a contextualized representation of each input element (i.e. also taking the preceding input elements into account). Then the *decoder*, transforms these representations into the output sequence, which can be of another type than the input sequence. Tacotron uses a sequence of characters as input and a sequence of mel spectrogram frames as output.

The encoder and decoder are neural networks. Typically, *recurrent neural networks* (RNNs) are used, as they can model data sequences of variable length. Recurrent neural networks have an internal memory state which they use to retain and use information across multiple time steps. In particular, Tacotron uses *gated recurrent units (GRUs)* (Cho et al., 2014) in its encoder and decoder layers. A GRU is a type of RNN that are better at modeling long sequences[1].

Even so, one input character typically still corresponds to many output mel spectrogram frames. Thus, the decoder should know which of the encoded inputs it should use to generate each output mel spectrogram frame. To achieve this, Tacotron uses an

[1] As discussed in section 2.2, using a mel spectrogram as the target of the TTS model reduces the number of output elements the model needs to generate.

[2] Tacotron 2, an improved version of Tacotron, uses a WaveNet-based vocoder to achieve more natural generated speech (Shen et al., 2018).

[1] Another popular type of recurrent neural network is the long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997). The GRU is a variant of the LSTM with fewer parameters, which makes it faster to train.

attention module (Bahdanau et al., 2014). This module learns the alignment between the encoded character sequence and the output sequence. A downside of this is that the learned alignment is sometimes imperfect, which can result in repeated or skipped words.

Another important issue in TTS models is the one-to-many mapping problem. One input text sequence corresponds to many plausible speech outputs, as the prosody is not always determined by the text. A TTS model thus has to determine the prosody of each phoneme, so that it results in a coherent speech output.

Tacotron eases this one-to-many mapping problem by conditioning an output mel spectrogram frame on the previously generated mel spectrogram frames, i.e., Tacotron has a *autoregressive* decoder. The previous frames can provide context to choose between the many possible ways of uttering a certain phoneme. The downside of using such a decoder, is that it can only output sequentially, and is thus very slow.

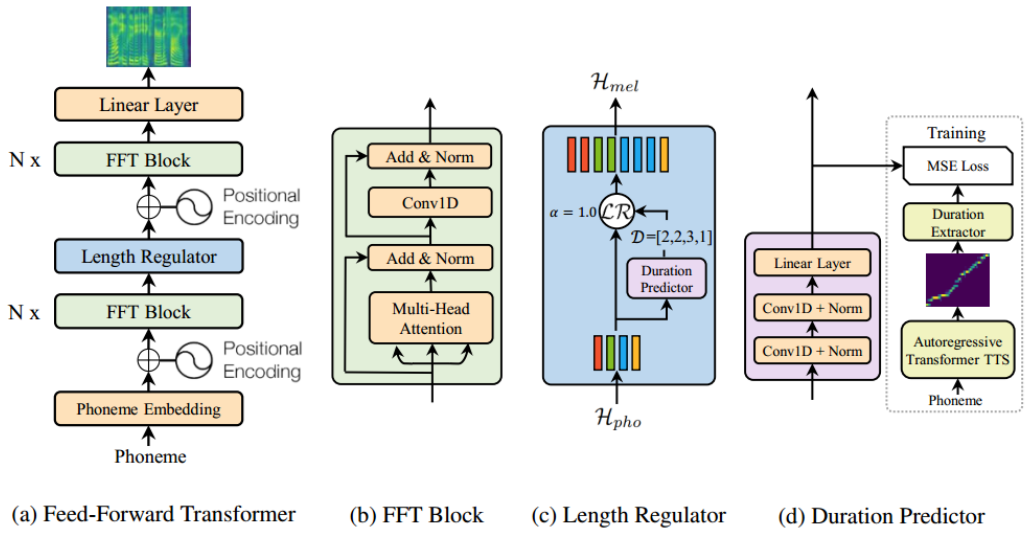
In conclusion, Tacotron is a seminal end-to-end TTS model, that generates mel spectrograms from characters. It is a seq2seq model with attention, consisting of a GRU-based encoder, and an autoregressive decoder. The results Wang et al. produced with this model, outperformed HMM-based techniques and set the bar for further neural network-based TTS methods.

2.4.2 FastSpeech

FastSpeech is an important TTS model developed by researchers at Microsoft (Ren et al., 2019) . It is used as the basis for the ConEx model, which will be proposed in the next chapter. The authors of FastSpeech build on the work of Tacotron, but propose a non-autoregressive architecture, which greatly improves the training and inference speed. Furthermore, FastSpeech uses hard alignment to overcome the issue of skipped or repeated words that can plague autoregressive models like Tacotron.

The fundamental building block powering FastSpeech is the Transformer architecture (Vaswani et al., 2017). This architecture is used for both the encoder and decoder layers, similar to the use of the GRUs in Tacotron. Because Transformers are feed-forward neural networks (i.e. without recurrent connections), FastSpeech works in a non-autoregressive manner. Outputs can thus be generated in parallel, as opposed to generating them one by one when using autoregressive models. This allows FastSpeech to perform end-to-end speech synthesis 38x faster than a similar autoregressive model.

As mentioned in the previous section, using an autoregressive decoder eases the one-to-many mapping problem. A non-autoregressive model is not conditioned on the previous output frames, and thus has difficulty in generating coherent speech. To solve this issue, FastSpeech uses *teacher-student knowledge distillation*. Essentially, the authors first trained an autoregressive (teacher) model and then trained their FastSpeech (student) model to output the same mel spectrograms. Unfortunately, this makes training the model complicated and slow, as two individual TTS models have to be trained.



(a) Feed-Forward Transformer (b) FFT Block (c) Length Regulator (d) Duration Predictor

Figure 2.4: The architecture of FastSpeech. Reprinted from Ren et al. (2019)

In 2019, Ren et al. introduced FastSpeech 2, an improved version of FastSpeech. One major change was the addition of pitch and energy information to ease the one-to-many mapping problem. The decoder uses this information to generate speech with a certain prosody. The model learns to predict the pitch and energy using the pitch and energy extracted from the ground-truth utterances as targets. With this additional information, it is no longer necessary to use knowledge distillation, thus greatly simplifying the training process.

The FastSpeech architecture

This section gives an overview of the FastSpeech architecture, as many of its components are used in ConEx, the model that will be proposed in this thesis. Each paragraph corresponds to a part of figure 2.4, which shows the complete FastSpeech architecture.

(a) Feed-forward Transformer The FastSpeech architecture, which the authors call the "feed-forward Transformer", is based on the seq2seq architecture and is thus composed of an encoder and a decoder. The encoder and decoder made up of multiple feed-forward Transformer (FFT) blocks. The encoder takes a phoneme sequence as its input, which makes mapping text to speech easier (see section 2.2.1). The decoder outputs mel spectrogram frames. FastSpeech has a length regulator between the encoder and the decoder to deal with the length mismatch between the input and output sequences.

(b) FFT block The feed-forward Transformer blocks are based on the Transformer blocks from Vaswani et al. (2017). An FFT block consists of two sublayers. The first sublayer is a multi-head self-attention mechanism, equal to the one in the original Transformer architecture. This self-attention enables the encodings to take into ac-

count the context. The second sublayer is a one-dimensional convolutional layer. Both sublayers have residual connections and are normalized.

(c) Length regulator After the encoding is done, the length regulator stretches the length of the encoded phonemes to match the length of the mel spectrogram output. This is necessary to fix the length mismatch that occurs because one input phoneme typically corresponds to many output mel spectrogram frames. At train time, the length regulator uses the ground-truth durations of the phonemes. At inference time, it uses the predicted phoneme durations output by the duration predictor.

Let $H_{pho} = [h_1, h_2, ..., h_n]$ denote the hidden sequence with length n, and let $D = [d_1, d_2, ..., d_n]$ denote the phoneme durations so that h_i has length d_i. The durations are expressed in the number of mel spectrogram frames (i.e., not in seconds). This means that $\sum_{i=1}^{n} d_i = m$ where m is the length of the complete output mel spectrogram. Furthermore, the length regulator also has a hyperparameter α that can be used to control the speech speed. Thus, the length regulator can be denoted as $LR(H_{pho}, D, \alpha) = H_{mel}$ To match the durations, the length regulator simply has to copy the hidden state h_i $\lfloor d_i \cdot \alpha \rceil$ times, where $\lfloor\rceil$ denotes rounding off. The length regulator thus outputs a vector

$$H_{mel} = [\underbrace{h_1, h_1, ..., h_1}_{\lfloor d_1 \cdot \alpha \rceil}, \underbrace{h_2, h_2, ..., h_2}_{\lfloor d_2 \cdot \alpha \rceil}, ..., \underbrace{h_n, h_n, ..., h_n}_{\lfloor d_n \cdot \alpha \rceil}]$$

(d) Duration predictor When generating speech, the duration of each phoneme is not known a priori. FastSpeech thus includes a module that predicts the durations D of the phoneme latent sequence H. In the original paper, Ren et al. use phoneme durations from the teacher model to train the duration predictor. The autoregressive teacher model learns multiple possible alignment as in Tacotron, so only the best alignment is used, which by and large avoids alignments that skip or repeat words. In FastSpeech 2, durations are extracted from the ground-truth speech utterances, using the Montreal forced aligner (McAuliffe et al., 2017).

The duration predictor architecture consists of two one-dimensional convolutional layers. The output is then projected to output a scalar (the duration) in the logarithmic domain. This is done to make the durations more Gaussian and thus easier to train.

2.5 Modeling and controlling prosody

Prosody refers to expressive aspects of speech, such as intonation, stress, and rhythm. Prosody manifests itself in speech aspects such as pitch, energy, and phoneme duration. Using prosody, speech can convey more than words; information such as (un)certainty, speaker identity (gender, age, and dialect among others), and affective information (emotions) can all be expressed. To make synthesized speech sound natural and convey the appropriate meaning, prosody needs to be modeled and controlled in speech synthesis systems.

The previous section already introduced the fact that for autoregressive models (such as Tacotron), determining the right prosody is easier, as these models can reuse prosody

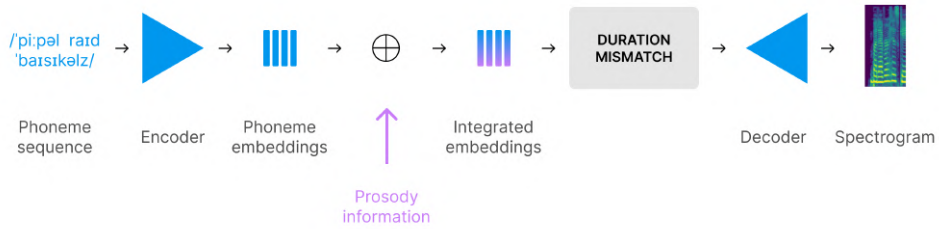


Figure 2.5: Prosody information in a seq2seq TTS model.

information from previously generated outputs. However, these models still have to determine the initial prosody. Furthermore, recent research has also led to successful non-autoregressive models (such as FastSpeech), for which the prosody needs to be determined for all output frames independently. Thus, dedicated components for modeling prosody are needed.

A first approach might be infuse the text with all additional prosodic information, i.e., to add prosody labels to the input. The effects of these additional labels would be learned by the text-to-speech model, and changing the prosody labels could enable control over the prosody at inference time. An example of a labeling scheme that could be used, is ToBI (Silverman et al., 1992). However, such labels are expensive to acquire, and there can be significant disagreement between human annotators.

Instead, prosody is typically extracted automatically from the ground-truth speech fragments and used to train the system to produce the correct prosody. At inference time (i.e., when generating speech), the prosody information can be generated by a dedicated component or taken from a different audio fragment (*style transfer*). The prosodic information can sometimes be controlled to obtain a certain speaking style or to make local adjustments to the prosody.

Concretely, prosody information is typically added after the encoder in modern seq2seq TTS models (shown in figure 2.5). This way, the decoder is conditioned on the phonemes and the prosody, and can learn to generate speech in the given prosody. To input prosodic information, a fitting representation needs to be found, and the level at which to represent prosody needs to be chosen. The next section describes the different levels prosody can be represented at.

Furthermore, there are two main ways of representing prosody as features for use in speech synthesis systems. The first is to use handcrafted prosodic features, based on the knowledge of domain experts. Section 2.5.2 details this approach and gives an example of a TTS model using this technique. Another option is to use neural-based representations, which are learned during the training of the TTS model. Section 2.5.3 describes four neural-based techniques for learning such representations, each time using a concrete architecture as an example. Finally, this chapter describes the limitations of techniques to model and control prosody.

2.5.1 Prosody level

Prosody can be represented at different levels:

- At the utterance level, the global speaking style can be represented. This can include aspects such as emotion, overall pitch, rhythm and energy of an utterance and the variation in pitch and energy. Techniques using global prosody representations include those introduced by Skerry-Ryan et al. (2018); Wang et al. (2018); Zhang et al. (2018); Hsu et al. (2018); Elias et al. (2020)
- At phoneme level, local prosodic aspects can be represented, such as the duration, pitch and loudness of a single phoneme. The fine-grained representations can be used to make local edits, such as emphasizing a certain phoneme. Fine-grained representations are used in Lee and Kim (2018); Ren et al. (2020); Sun et al. (2020); Elias et al. (2020), among others.
- Some researchers have proposed prosody representations at the syllable level, as they state that "syllables are considered as the primary carrier of important prosodic events like tone and stress" (Zhang et al., 2020).
- Finally, hierarchical approaches have seen increasing attention in research on expressive speech synthesis. In these systems fine-grained representations are typically conditioned on coarser representations. Hierarchical representations are proposed by Sun et al. (2020); Chien and Lee (2021), among others.

2.5.2 Handcrafted prosodic features

Prosody can be modeled using attributes that are designed by domain experts. Even though there is no definitive list of prosodic variables, pitch, loudness, and rhythm are seen as the most characteristic attributes (Jurafsky and Martin, 2009). Acoustically, these variables are closely related to *fundamental frequency* (F_0)[1], *energy*, and (phoneme) duration, respectively. These variables can be modeled explicitly in a TTS system to provide the necessary prosody information.

The advantage of using handcrafted prosodic features (such as pitch, loudness, and duration) is that their effects are clear, i.e., the variables are interpretable. However, a downside it is not easy to use these variables to create a desired prosodic effect. In order to obtain a desired prosody, the variables sometimes have be combined in complex ways and thus requires some domain expertise.

Handcrafted prosodic features have been used in the Tacotron architecture (Raitio et al., 2020)[2]. FastSpeech 2 also makes use of these variables. The optional section below details how these variables are used in the FastSpeech 2 architecture.

[1] When humans speak at a certain pitch, their *vocal folds* (or *vocal cords* informally) vibrate at a certain frequency. This frequency is called the fundamental frequency, abbreviated as F_0.

[2] Raitio et al. (2020) also model the *timbre* (or voice quality) in an utterance using the *spectral tilt* variable.

FastSpeech 2 (optional reading)

The original FastSpeech model (Ren et al., 2019) used knowledge distillation to ease the one-to-many problem, but this created a cumbersome training process. In the next version of the model, FastSpeech 2 (Ren et al., 2020), this knowledge distillation step is removed. Instead, FastSpeech 2 models two prosodic features – F_0 and energy – directly. In both FastSpeech versions, the phoneme duration is also modeled directly using the duration predictor.

Architecture FastSpeech 2 models pitch and loudness (or energy) in a similar way to phoneme duration. Each feature has its own predictor neural network, which consists of two one-dimensional convolutional layers. Both predictors take the phoneme embedding sequence as input. The pitch predictor outputs a pitch spectrogram frame per output frame, while the energy predictor outputs the energy level for each frame. The pitch and energy information is added to the phoneme embeddings after they have been stretched by the length regulator (see section 2.4.2), i.e. prosody information is used at frame-level[3].

Training To train the TTS model, the F_0 and energy values of the ground-truth speech utterances are used. These are extracted from the speech utterances in a preprocessing step. To train the pitch and energy predictors, the phoneme embedding sequence is input, and the loss between the predicted outputs and the ground-truth prosody values is calculated.

Inference When generating speech, F_0 and energy is predicted using the phoneme embeddings, and this information is used by the decoder to synthesize speech at the right pitch and with the right energy level.

Controllability The predicted F_0, energy, and durations can be tweaked at inference time to obtain an utterance with different prosody. There is extensive research on the effects of pitch, loudness, and duration on prosody and adjustments can thus be made to achieve various speaking styles. Nevertheless, tweaking these parameters to get the desired prosody is still difficult, as mentioned before. While Ren et al. did not show this kind of controllability in their paper, Łańcucki (2021) shows that pitch can be controlled well through an intuitive user interface. Łańcucki developed this interface for FastPitch, a TTS model similar to FastSpeech 2.

[3] FastPitch (Łańcucki, 2021), a model based on FastSpeech that was developed at the same time as FastSpeech 2, uses phoneme-level prosody information.

2.5.3 Neural prosody embeddings

Another approach to modeling prosody is to have a neural network learn a representation of prosody. Such a prosody encoder network takes a speech fragment as its input and outputs an embedding which represents the prosody in that fragment. The embeddings should be sufficiently small/compressed, so that they only represent general prosodic

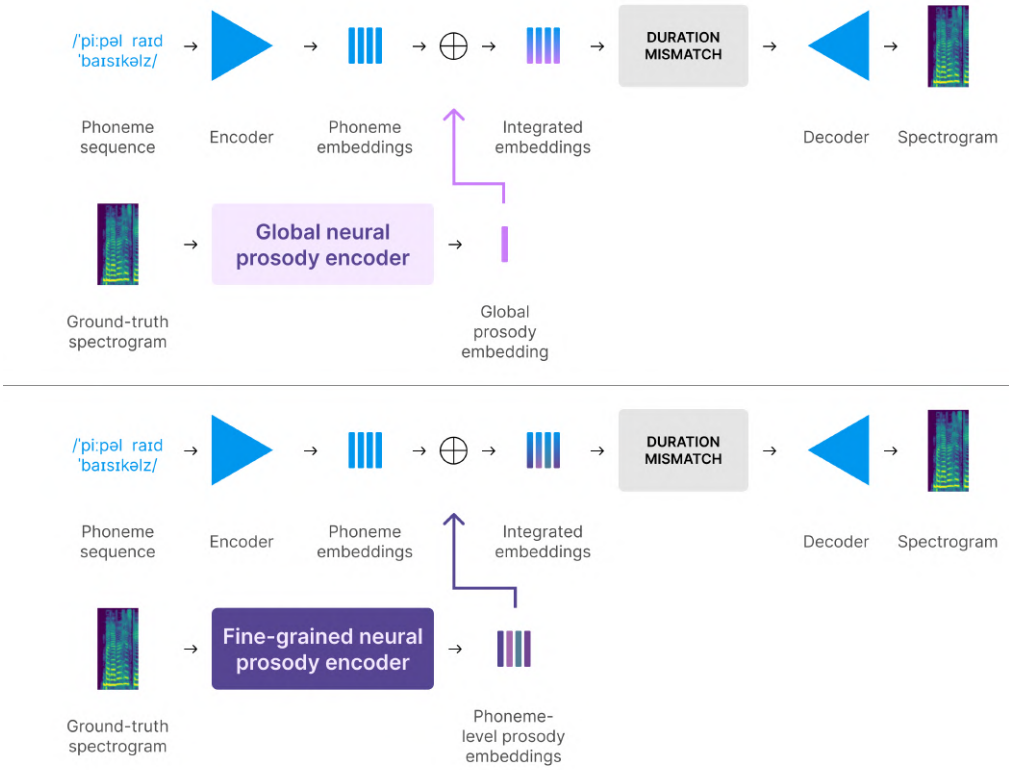


Figure 2.6: A TTS model with a prosody encoder that learns global prosody embeddings (top), or fine-grained phoneme-level prosody embeddings (bottom).

features which occur in many utterances. This representation or embedding can then be added to the phoneme embeddings created by the TTS encoder. The decoder can then use these phoneme and prosody embeddings to generate speech with a certain prosody. Figure 2.6 shows a TTS model architecture that uses a neural prosody encoder.

Mathematically, the goal is to discover the hidden or *latent* factors in speech that explain the variance due to prosody. Together, these latent factors can be seen as a high-dimensional space, with each dimension of this space representing a latent factor. A single prosody embedding is a point in this high-dimensional embedding space. If the embedding space is *smooth*, then similar embeddings are close to each other in the space. In the case of prosody embeddings, this means that points close to an embedding can also be used to generate speech with similar prosody. Figure 2.7 shows an example of a smooth embeddings space.

This section first discusses four neural-based prosody modeling techniques. The first two, the reference encoder and global style tokens, seek to capture a single embedding per utterance that captures prosodic information and other variance not explained by the text input. Next, variational autoencoders are discussed, which are used for learning smooth embeddings spaces. Finally, vector-quantized variational autoencoders are detailed. These learn discrete embeddings, as opposed to the continuous embeddings

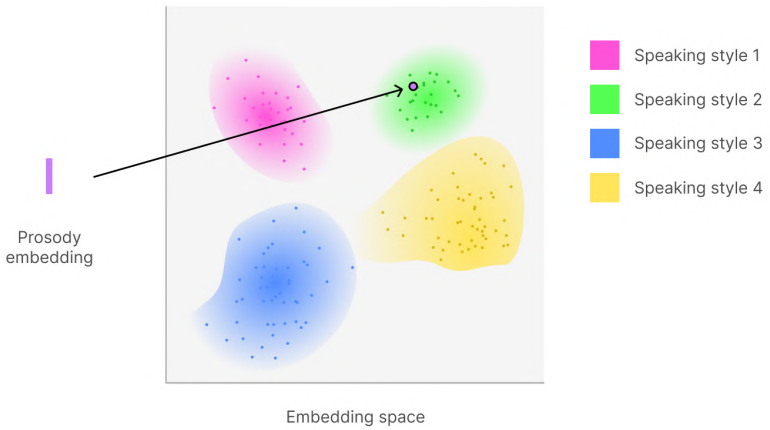


Figure 2.7: A smooth embedding space visualized in 2D. Different speaking styles can form clusters in this embedding space. A certain prosody embedding corresponds to a point in the embedding space.

of VAEs.

Reference encoder

The reference encoder is an early architecture used to learn global prosody embeddings (i.e., one embedding per utterance)[1]. It was introduced as an extension to Tacotron by Skerry-Ryan et al. (2018) of the Google Tacotron team. The reference encoder is used in the novel model that is presented in this work (see chapter 3) and is thus detailed in what follows.

The authors define prosody as "the variation in speech signals that remains after accounting for variation due to phonetics, speaker identity, and channel effects (i.e. the recording environment)." Because prosody is independent of phonetics in this formulation, the same prosody embedding can be used for different text inputs, thus enabling style transfer.

Architecture The reference encoder takes a reference spectrogram as its input and inputs a single fixed-length embedding. The reference encoder consists of six 2D convolutional layers to downsample the input spectrogram sequence and one GRU layer to compress the downsampled sequence to a single embedding. This embedding is concatenated with all phoneme embeddings, before they are passed to the attention module of Tacotron.

Training The Tacotron TTS model and the reference encoder are trained simultaneously using the same loss. During training, the reference encoder uses the ground-truth spectrogram frames as its input.

[1] A variant of the reference encoder which learns fine-grained prosody embeddings, was proposed by Lee and Kim (2018)

Inference At inference time, any reference audio fragment can be input to the reference encoder. The reference encoder will then generate an embedding that represents the prosody of this reference audio, and the decoder will generate speech with the prosody of the reference audio.

Controllability Prosody embeddings created with a reference encoder can be used for style transfer. The style of a reference audio fragment can be copied to another text, thus enabling a "say it like this" mechanism. However, the individual prosody embeddings can not be tweaked to change the prosody, as the embedding space can be sparse. In other words, changing a prosody embedding slightly does not result in slightly different prosody, instead, the prosody might become unusable. Additionally, the authors note that the learned embeddings do not fully succeed in disentangling prosody from the content of a reference audio fragment. This means that style transfer works best between utterances of similar length and structure (Wang and Skerry-Ryan, 2018).

Global style tokens

A team of Google researchers introduced *Global Style Tokens (GSTs)* to deal with the limitations of the reference encoder (Wang et al., 2018). In particular, global style tokens capture prosodic attributes, and are disentangled from actual audio content. This allows for better style transfer. Furthermore, the tokens can manipulated directly to change the global style. For example, scaling the tokens tunes the strength of their attributes and combining tokens can create complex styles. In the optional section that follows, the GST technique is further detailed.

Global style tokens in detail (optional reading)

Architecture A global style token layer is added on top of a reference encoder (as introduced in the previous section). The global style token layer consists of an attention module, which uses the generated reference embedding as a query vector. The values which are being queried, are a fixed number of global style tokens, and these are reused for all training sequences. The reference embedding thus determines which style embeddings will be combined and how each of them will be scaled to create the final style embedding. This embedding is added to all phoneme embeddings, before they are passed to the attention module of Tacotron.

Training The reference encoder again uses the ground-truth spectrogram frames to generate a reference embedding. This embedding is used by the attention module to take the weighted average of the style tokens most similar to the reference embedding. A fixed number of style tokens is used and kept in a bank. They are all randomly initialized.

Inference Similar to the style transfer with the reference encoder, it is again possible to input a reference audio fragment to generate a reference embedding. This embedding is used to get a weighted combination of the style tokens in the bank. Alternatively, any combination of style tokens in the bank can also be used directly as style embedding.

Controllability Style transfer can be performed to achieve a certain speaking style. Wang et al. demonstrate in their experiments that GSTs can enable style transfer even when the target utterance differs significantly from the utterance that was used to generate the style embedding. Furthermore, GSTs can be scaled and combined, which enables control over various prosodic attributes. One caveat is that one style token does not correspond to a single attribute. Thus, to interpret the exact effects of a certain style token, extensive testing needs to be done. Lastly, the authors show that different style tokens can be used for different parts of an utterance, thus enabling local style control.

Variational autoencoders

Autoencoders are neural networks that compress (encode) and reconstruct (decode) data. Internally, they learn a compact representation of the data that captures the most salient features of the data[1]. *Variational autoencoders (VAEs)* are autoencoders that learn smooth latent spaces, i.e. similar compressed features are close to each other in the learned latent space[2]. VAEs were introduced by Kingma and Welling (2014).

Variational autoencoders learn smooth embedding spaces by not learning point representations of the input data, but by learning distributions in the latent space. These distributions can then be sampled and the samples can be decoded to generate outputs. This means that not only specific points can be decoded to generate meaningful data, but also the surroundings, creating a smooth embedding space.

VAEs can be used to represent prosody at utterance level for global embeddings (Zhang et al., 2018; Hsu et al., 2018; Elias et al., 2020), at phoneme level for fine-grained prosody embeddings (Elias et al., 2020), or VAEs can be used in hierarchical models (Sun et al., 2020). The model of Zhang et al. (2018) for global prosody embeddings is detailed in the following section (optional reading).

A VAE for Tacotron 2 (optional reading)

Architecture Zhang et al. use the Tacotron 2 (Shen et al., 2018) TTS model and extend the reference encoder architecture (Skerry-Ryan et al., 2018) to create prosody embeddings. As such, the input spectrogram is first fed to the reference

[1] A reference encoder (section 2.5.3) can be seen as an autoencoder, with the reference encoder compressing the spectrogram data, and the Tacotron decoder using this data to recreate spectrograms.

[2] As mentioned before, the embedding space learned by a reference encoder is not smooth.

encoder to create a global embedding. This embedding is then passed through two fully-connected layers which predict the mean and standard deviation of the latent prosody distribution. A prosody embedding can then be sampled from this distribution and can be concatenated with the phoneme embeddings and can then be sent to the decoder.

Training The VAE is trained together with the Tacotron 2 model. It takes ground-truth frames as input and outputs the mean and standard deviation of the prosody embedding distribution. This distribution is then sampled to obtain a prosody embedding, which can be passed to the Tacotron decoder. When training VAEs, it sometimes occurs that the decoder largely ignores the VAE output, and the VAE thus does not learn the variation in the data. This issue is called *posterior collapse* (van den Oord et al., 2017). Special training schemes can be used to prevent it, although it still presents a major challenge when working with VAEs.

Inference At inference, similar to the reference encoder, any reference audio fragment can be input. Then we obtain a distribution for the prosody embedding, which can be sampled to obtain different (but similar) prosodic variations.

Controllability Style transfer can be achieved in a similar way to the process using the reference encoder. One difference is that the learned distribution needs to be sampled now, and thus multiple variations on the same style can be created. The smooth embedding space created by the VAE also enables the change of a sampled prosody embedding along its many dimensions. Changing these dimensions can change attributes such as pitch and speaking rate, among others. Not all dimensions lead to interpretable changes, however. Lastly, interpolation between different prosody embeddings is also possible. The authors found that their VAE technique outperformed the global style token technique (Wang et al., 2018), which offers similar control over the prosody.

Vector quantized variational autoencoders

van den Oord et al. (2017) introduced a kind of variational autoencoder for learning discrete representations, which they called the vector quantized variational autoencoder (VQ-VAE). The authors argue that discrete representations are a natural fit for many problems (such as language, which is inherently discrete) and show that discrete learned representations can perform as well as continuous representations. Furthermore, quantizing the embedding solves the posterior collapse issue that plagues normal VAEs.

VQ-VAEs can be used in expressive TTS to encode prosody. A VQ-VAE keeps a fixed number of different discrete embeddings in a *codebook*[1], which are used when quantiz-

[1]These discrete embeddings are thus similar to the collection of global style tokens (Wang et al., 2018). However, in Wang et al.'s approach, multiple GSTs are combined to create the final prosody representation. As such, these GST embeddings are not truly discrete, unlike the discrete representations learned by VQ-VAEs (Henter et al., 2018).

ing the inputs. When encoding prosody, these discrete embeddings represent different prosodic phenomena.

There are a number of reasons for why discrete embeddings might be a good way to represent prosody. First, discrete representations could make prosody easier to control, as the discrete embeddings could represent different prosodic phenomena that can be enabled or disabled. The effects of the different dimensions of continuous embeddings on the contrary, are hard to interpret. Furthermore, Zhang et al. (2020) remark that "[the] discrete representation of prosody is also found to be more suitable for describing human perception". Lastly, Sun et al. (2020) found that "using a quantized representation improves the naturalness over audio samples generated from the continuous latent space, while still ensuring reasonable diversity".

Sun et al. (2020) use a VQ-VAE to learn fine-grained phoneme-level prosody embeddings, Zhang et al. (2020) use it to learn syllable-level embeddings and Chien and Lee (2021) propose a hierarchical architecture based on the VQ-VAE. The approach of Sun et al. (2020) is detailed further, as the ConEx model, which will be introduced in next chapter, also uses a VQ-VAE for learning fine-grained prosody embeddings.

Architecture The architecture proposed by Sun et al. (2020) extends the Tacotron 2 architecture (Shen et al., 2018). To obtain fine-grained prosody embeddings (i.e., one embedding per phoneme), the input spectrogram first has to be aligned with the phoneme embedding sequence. This is done using an attention mechanism. The aligned inputs are encoded. These encodings are then quantized by replacing them with the closest embedding from the codebook. This process is called *vector quantization*. These quantized embeddings are then concatenated with the phoneme embeddings, after which they can be used by the decoder to generate a spectrogram.

Furthermore, a separate *autoregressive prior (AR)* network is used to predict the discrete prosody embeddings from the phoneme embeddings, as there is no spectrogram to run at inference time. This AR prior consists of a single-layer LSTM with a softmax activation function, to obtain a categorical distribution over the different discrete prosody embeddings.

Training The VQ-VAE and Tacotron 2 are trained jointly. Ground-truth spectrograms are used to create the fine-grained prosody embeddings, as explained above. The AR prior is trained afterwards, using the phoneme embeddings as input and the prosody embeddings generated from the ground-truth spectrograms as targets.

Inference When generating speech, the AR prior directly predicts the prosody embeddings from the phoneme embeddings created by the Tacotron encoder. Alternatively, a reference spectrogram can also be input to the VQ-VAE for style transfer.

Controllability Style transfer with fine-grained phoneme embeddings only makes sense using an input audio fragment that has the same content as the text of the speech to be synthesized. This is because the input audio fragment needs to be aligned with the

phoneme embeddings. However, controllability over the prosody can also be achieved using the AR prior. Individual prosody embeddings can be swapped out, and since the prior is autoregressive, the rest of the phonemes can be predicted using this swapped prosody embedding, which could lead to more consistent prosody. However, Sun et al. do not perform such experiments. Zhang et al. (2020) do evaluate the control by swapping their syllable-level prosody embeddings and find that it leads to interpretable changes in local prosody.

2.5.4 Limitations

There exist several limitations to current techniques for modeling and controlling prosody in speech synthesis systems. These are the focus of current research, including this work. This section gives an overview of the most important, and gives some proposed solutions.

Disentangling prosody from speaker and channel characteristics

Many prosodic characteristics such as overall pitch, timbre, and pitch range, are tied to how a specific person speaks. In expressive speech synthesis, these personal characteristics are of lesser interest, as they do not convey any extra meaning. Furthermore, there are also other channel characteristics (such as background noise and speech distortion) that can influence prosody embeddings.

To solve this issue, Skerry-Ryan et al. (2018) condition the decoder on the embedding obtained by the reference encoder, but also on a speaker embedding. However, they find that prosody and speaker representations are still entangled. For example, the prosody representation seems to encode pitch in an absolute manner. Williams and King (2019) proposed two autoencoders with different objective functions to disentangle speaker characteristics from style.

Interpreting prosody embeddings

When using neural prosody embeddings, it is hard to interpret what effect the different dimensions of the embedding space have on perceived prosody. Sometimes, a dimension has effect on multiple prosodic attributes. This was, for example, discovered in the global style tokens system (Wang et al., 2018), where a single token sometimes represented a mixture of attributes. Hsu et al. (2018) show that a VAE can be used to learn more independent prosody factors.

Even when the factors learn independent prosodic attributes, it is still necessary to manually evaluate the effect factor. This is the case because there is guarantee that the same factors will be learned when retraining the model. Furthermore, evaluating the effect can be a time-consuming process, depending on how many dimensions or style tokens are used. The evaluation is also subjective, as prosodic effects are hard to define precisely.

Obtaining a desired prosody

In order to obtain a desired prosody, two general approaches are available:

1. Generate speech in a certain global speaking style. This style can be obtained from a reference speech clip, or from a set of existing style embeddings (which have to be interpreted, see above). Small adjustments can only be made by generating multiple versions of the same speech. The generated speech with the desired prosody can then be selected, although there are no guarantees that the exact right prosody will be generated.
2. Generate speech using fine-grained prosody features/embeddings. This means the prosody has to be determined for each phoneme or syllable. Even when the prosodic features are interpretable, it can still be a challenge to manipulate them to obtain the desired prosody. For example, it can be hard to control these prosodic variables in order to change the style of the speech or obtain a certain prosodic phenomenon (such as stressing a word).

As mentioned before, hierarchical systems have been proposed (Sun et al., 2020; Chien and Lee, 2021) to model prosody at different levels. However, the authors of these papers have not shown how prosody can be controlled at different levels in an intuitive way. This thus remains an important challenge for expressive speech synthesis.

Evaluating prosody modeling and control Since prosody can be subjective, it is hard to define objective evaluation metrics. Often, research has to resort to listening tests to evaluate expressive speech synthesis systems. These listening tests are expensive and time-consuming to conduct. Skerry-Ryan et al. (2018) define objective metrics for evaluating prosody transfer.

Chapter 3

ConEx: Controllable Expressive Speech Synthesis

Modeling and controlling prosody is necessary in order to synthesize expressive speech. However, controlling prosody is still challenging in many current models. This work thus proposes ConEx, a novel controllable model for expressive speech synthesis. Using ConEx, the global speaking style of the synthesized speech can be controlled, and local adjustments to the prosody of the generated speech can be made. To achieve this, ConEx builds on the non-autoregressive FastSpeech architecture (Ren et al., 2019) and adds components to model and control prosody.

In particular, ConEx utilizes a reference encoder (Skerry-Ryan et al., 2018) for modeling the global speaking style, as well as a vector quantized variational autoencoder (van den Oord et al., 2017) for creating fine-grained prosody embeddings. Furthermore, an autoregressive prior model is trained over the fine-grained prosody embeddings. This model predicts the fine-grained prosody embeddings at inference time, based on the input phoneme sequence and the global speaking style.

This chapter first motivates the use of the different components in ConEx. Then, the relationship between ConEx and other related works is described. Next, the complete architecture of ConEx is detailed, focusing on the components that model and control prosody. Subsequently, the chapter details how to train ConEx, how to use the model at inference, and how the model can be used to control prosody. Finally, the implementation of ConEx is briefly described.

3.1 Motivation

Current expressive speech synthesis systems often use learned representations to model and control prosody. Prosody can either be represented on a global level (i.e., using one embedding per utterance), or on a local level (i.e., using one embedding per syllable or phoneme). Global prosody embeddings can be used to generate speech in a certain speaking style. Fine-grained prosody embeddings allow for local prosody control. ConEx aims to use both representations to allow control over both global and local prosody.

This section motivates the use of FastSpeech as the TTS architecture, as well as the use of a reference encoder and a VQ-VAE for modeling and controlling prosody.

Use of FastSpeech

Most expressive speech synthesis research has been carried out using autoregressive models such as Tacotron (Wang et al., 2017) (see section 2.5). However, non-autoregressive models such as FastSpeech (Ren et al., 2019) allow speech to be synthesized many times faster. Moreover, non-autoregressive models have a greater need for prosody modeling, as the one-to-many mapping problem cannot be eased by using information from the previously generated outputs. Thus, ConEx introduces techniques to model and control prosody to a non-autoregressive model. Concretely, the FastSpeech architecture is used, since it achieves strong results and has openly available implementations. Building on an existing TTS architecture also facilitates adjusting or extending the proposed architecture in the future.

Use of a VQ-VAE and reference encoder

Section 2.5.3 described the many neural-based techniques to learn prosody embeddings. To learn global prosody embeddings, ConEx uses a reference encoder (Skerry-Ryan et al., 2018). The reference encoder is a simple, yet useful technique to capture the global speaking style. The reference encoder can be also used as the basis for a variational autoencoder in the future, as demonstrated by Zhang et al. (2018).

To represent prosody at a fine level, ConEx uses discrete embeddings, as they are easier to control and interpret. This is because the effect of a limited number of discrete embeddings can easily be tested individually, which is not the case for continuous embeddings[1]. An additional advantage is that a simple autoregressive prior network can predict such discrete embeddings easily when generating speech. This AR prior network also opens up further opportunities for editing prosody while maintaining a consistent speaking style. This will be discussed further in section 3.6.

To learn discrete prosody embeddings, a vector quantized variational autoencoder (van den Oord et al., 2017) is used. This is a powerful generative model which has shown great results in modeling images, music and speech. Since it does not suffer from the issue of posterior collapse, it is easier to train than a regular VAE. Furthermore, the VQ-VAE also has a strong theoretical background, as demonstrated by Henter et al. (2018).

3.2 Related works

Speech synthesis is a highly active research area with many researchers from academia and technology advancing the state of the art continuously. As such, similar ideas often

[1]The use of discrete embeddings does not allow the combination and scaling of embeddings, as is the case when using global style tokens (Wang et al., 2018). Combining and scaling style tokens can have complex effects on prosody, which makes it harder to obtain a certain prosodic phenomenon.

emerge simultaneously[1]. ConEx, the proposed model, shares some similarities with recent works. This section details these similarities and differences between ConEx and these related works.

- Sun et al. (2020) introduced the idea of using a VQ-VAE to learn discrete fine-grained prosody embeddings. Furthermore, the authors also showed that an autoregressive prior can be trained to predict these discrete embeddings at inference time. ConEx also uses a VQ-VAE and an autoregressive prior. However, their work uses the autoregressive Tacotron 2 architecture and is thus slower than ConEx. Furthermore, their model does not provide a way to control the global speaking style. Lastly, the authors did not publish experiments on controlling prosody, while controllability is the main focus of ConEx.

- At the end of 2020, researchers from the Tacotron team introduced Parallel Tacotron, a non-autoregressive model similar to FastSpeech (Elias et al., 2020). The model uses a variational autoencoder to capture the prosody and ease the one-to-many mapping problem. The learned prosody embeddings are thus continuous, in contrast to the discrete embeddings that ConEx learns. Elias et al. experimented with both a global VAE and with a fine-grained phoneme level VAE. However, they did not combine these VAEs. The authors also did not show how prosody can be controlled with their model. Instead, more emphasis was put on the quality of the synthesized speech.

- Sun et al. (2020) proposed a hierarchical VAE model that operates on utterance, word, and phoneme level to achieve multilevel control of prosody. Their prosody modeling scheme is added to the autoregressive Tacotron 2 architecture. The VAE produces continuous latent embeddings. They showed that these embeddings could be used to control prosody on word and phoneme level. In particular, they employed a training scheme so that the dimensions of the prosody embeddings correspond to energy, duration, and F_0. ConEx aims to achieve fine-grained prosody control by employing discrete prosody embeddings, which could represent different prosodic phenomena. Furthermore, ConEx aims to control the global speaking style, which is something Sun et al. did not cover in their research.

- Lastly, Chien and Lee (2021) extended the FastSpeech architecture with different prosody modeling components. For neural-based prosody embeddings, they employed a VQ-VAE. Furthermore, they also introduced a hierarchical model that uses word and phoneme-level prosody embeddings for natural-sounding, accurate prosody. Their work does not include global style modeling. Furthermore, the authors also do not experiment with controlling the prosody using the discrete prosody embeddings. Instead, they introduce multiple methods for predicting the prosody automatically using the input text.

[1] Note that many of these related works were published during the development of this thesis.

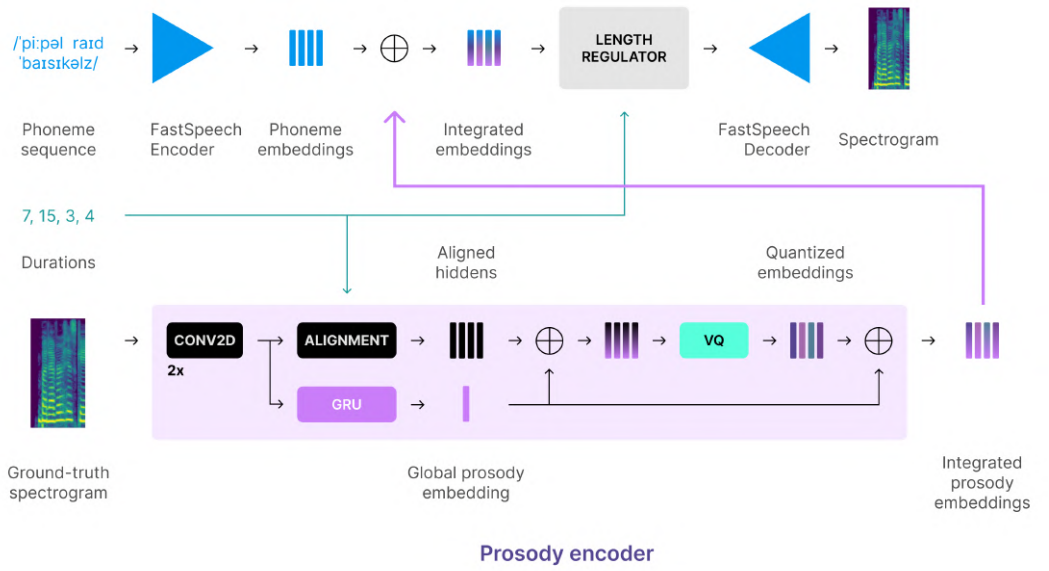


Figure 3.1: The complete architecture of ConEx.

3.3 Model architecture

This section gives an overview of the ConEx architecture. The architecture largely follows the FastSpeech structure from the original paper by Ren et al. (2019). However, ConEx adds a residual encoder after the encoder and uses the duration predictor introduced in FastSpeech 2 (Ren et al., 2020). Figure 3.1 depicts the complete ConEx architecture. Appendix A contains an overview of all the hyperparameters of the proposed model.

3.3.1 TTS components

Encoder & decoder The encoder and decoder each consist of four FFT blocks as in FastSpeech (see section 2.4.2). The encoder converts an input phoneme sequence to a sequence of hidden phoneme embeddings. The decoder transforms a sequence of embeddings to output mel spectrogram frames.

Duration predictor & length regulator The length regulator overcomes the length mismatch between the input phoneme sequence and the output mel spectrogram by duplicating the phoneme embeddings according to their durations. These durations are learned by the duration predictor.

3.3.2 Prosody encoder

ConEx introduces a prosody encoder consisting of two components: a reference encoder and a VQ-VAE. The reference encoder aims to capture the global speaking style, while the VQ-VAE creates quantized fine-grained prosody embeddings. These fine-grained embeddings can be used to control the local prosody.

Reference encoder

The goal of the reference encoder is to represent the speaking style of an entire utterance in a single fixed-length vector. The reference encoder in ConEx is based on the reference encoder by Skerry-Ryan et al. (2018). The reference encoder takes a mel spectrogram as its input and outputs a continuous global prosody embedding.

The global reference encoder consists of two 2D convolutional layers, one GRU layer, and a final projection. The convolutional layers use a kernel size of 3 and are each followed by a *rectified linear unit (ReLU)* activation function. These layers downsample the input mel spectrogram. The GRU layer transforms the downsampled sequence into a single embedding. To integrate the global prosody embedding with the phoneme embeddings, the prosody embedding is first projected to match the dimensionality of the phoneme embeddings. Finally, the prosody embedding is added to all phoneme embeddings.

Quantized fine-grained prosody encoder

To capture prosody at the phoneme level, ConEx employs another encoder. This fine-grained prosody encoder uses a VQ-VAE to learn discrete fine-grained embeddings. Such a prosody encoder was first introduced by Sun et al. (2020), but ConEx more closely follows the implementation of Chien and Lee (2021). The fine-grained encoder takes a mel spectrogram as input and outputs one discrete prosody embedding per phoneme embedding.

The fine-grained prosody encoder consists of two 2D convolutional layers, an alignment layer, two linear projection layers, a vector quantization layer, and then a final projection. The convolutional layers are shared with the global reference encoder and thus have the same hyperparameters.

The convolutional layers first downsample the input mel spectrogram frames. The alignment layer then aligns these to the phoneme embedding sequence, so that there is one prosody embedding per phoneme embedding. This is achieved through phoneme-wise mean pooling, i.e., all downsampled mel spectrogram frames that correspond to a certain phoneme embedding are averaged together. Then, two linear projection layers followed by ReLU activation functions map the hiddens to a latent 3D space.

Next, *vector quantization (VQ)* is performed, following van den Oord et al. (2017). Figure 3.2 shows this vector quantization process. The VQ-VAE contains a fixed number of 3D discrete embeddings in a codebook. These codebook embeddings are randomly initialized and then updated during training. To quantize the latent sequence obtained after the projection, each latent from the sequence is replaced by the closest embedding from the codebook. Denote the latent sequence by $\boldsymbol{z}$, the latent representation corresponding to the n^{th} phoneme by $\boldsymbol{z}_n$, and an embedding from the codebook by $\boldsymbol{e}_k$. The index of the closest embedding is then found by computing:

$$k = \text{argmin}_j \| \boldsymbol{z}_n - \boldsymbol{e}_j \|_2^2 \tag{3.1}$$

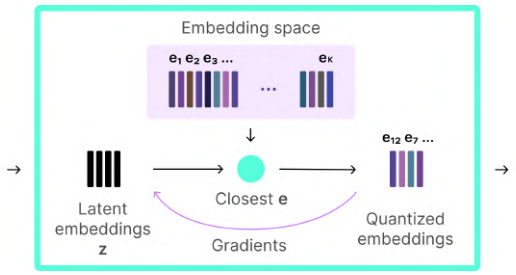


Figure 3.2: The vector quantization process. The purple arrow indicates the copying of the gradients during training, as described in the next section. Figure based on Sun et al. (2020).

Lastly, the final projection layer maps the 3D discrete embeddings to the dimensionality of the phoneme embeddings, and the fine-grained embeddings are added to the phoneme embeddings.

Autoregressive prior

When generating speech for an unseen phoneme sequence, a ground-truth mel spectrogram is not available. Thus, the reference encoder and fine-grained prosody encoder cannot be used directly. Instead, an existing global prosody embedding is used to determine the speaking style of the generated speech. However, local prosody embeddings cannot simply be transferred to different phoneme sequences, as the local prosody is linked to the phonemes.

To solve this issue, an autoregressive prior model is added to ConEx. This component predicts the fine-grained prosody embeddings from the new phoneme sequence and the given global prosody embedding. It is also autoregressive, i.e., it uses the previously predicted prosody embeddings to generate the next one. This helps the generate more coherent prosody.

Why is this model called a "prior"? (optional reading)

Altosaar (2016) shows that generating data such as speech can be viewed probabilistically as follows:

1. Model a probability distribution over certain variables of interest: $p(\boldsymbol{x})$.
2. Sample the distribution to generate plausible values for the modeled variables: $\boldsymbol{x}_i \sim p(\boldsymbol{x})$.

Let $\boldsymbol{x}$ denote the variables of interest and $p(\boldsymbol{x})$ their distribution. For example, if $\boldsymbol{x}$ represents mel spectrogram variables, then $p(\boldsymbol{x})$ can be sampled to generate mel spectrogram frames.

Generative models with latent variables do not model $p(\boldsymbol{x})$ directly, but instead model joint probability distribution of observed variables $\boldsymbol{x}$ and some latent variables $\boldsymbol{z}$, i.e. they model $p(\boldsymbol{x},\boldsymbol{z})$. This joint distribution can be written as $p(\boldsymbol{x},\boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$. $p(\boldsymbol{x}|\boldsymbol{z})$ is called the *likelihood* and $p(\boldsymbol{z})$ is called the *prior*.

Now data can now be generated as follows:

1. Model the likelihood and prior probability distributions.
2. Sample $\boldsymbol{z}_i \sim p(\boldsymbol{z})$.
3. Sample $\boldsymbol{x}_i \sim p(\boldsymbol{x}|\boldsymbol{z} = \boldsymbol{z}_i)$ from the likelihood conditioned on the sampled $\boldsymbol{z}_i$.

A VQ-VAE is a generative model with discrete latent variables. The likelihood is parametrized by the decoder. The prior is taken to be fixed during training and can be parametrized by a prior model.

The autoregressive prior consists of a single LSTM layer and a linear projection layer. The prior takes the phoneme embeddings with the added global prosody embedding as input. For every phoneme embedding, it outputs a categorical distribution over the quantized fine-grained prosody embeddings in the codebook. The fine-grained prosody embeddings are then again transformed to the dimensionality of the phoneme hiddens by the linear projection layer.

3.4 Training

The training procedure for ConEx has two steps: First, the TTS model is trained jointly with the duration predictor, the reference encoder, and the fine-grained prosody encoder. Second, the autoregressive prior is trained. This section details how the two steps are carried out.

3.4.1 Training the TTS model and prosody encoder

In the first training step, the FastSpeech encoder takes a phoneme sequence and the corresponding phoneme durations as input and produces phoneme embeddings. The reference encoder and fine-grained prosody encoder take the ground-truth mel spectrogram as input. The fine-grained prosody encoder also uses the phoneme durations to align the input with the phoneme embeddings. The output global prosody embedding and fine-grained embeddings are added to the phoneme embeddings. The duration predictor uses these integrated embeddings to predict the duration of each phoneme. However, this predicted duration is not used by the length regulator during training. Instead, the ground-truth duration is used. Finally, the expanded embeddings are transformed to mel spectrogram frames by the decoder.

To update the weights of the ConEx model during training, a loss needs to be backpropagated. The target of the TTS model is the ground-truth mel spectrogram. As such, the *mean absolute error (MAE)* or L1 loss is calculated between the generated and ground-truth mel spectrogram. This loss is used to update the parameters of the encoder, the decoder, the reference encoder, and the fine-grained encoder. However, the fine-grained encoder contains a vector quantization layer, and there is no real gradient defined for

equation 3.1. To overcome this, the gradients of the decoder are simply copied to before the VQ layer, as is done by van den Oord et al. (2017). The copied gradients are shown in figure 3.2.

This still leaves the issue of training the VQ layer, as the method above simply copies the gradients as if the VQ layer were not there. Thus, to update the quantized embeddings in the codebook, a separate VQ loss is used, as proposed by van den Oord et al. (2017). This loss consists of two components: a *quantization loss* and a *commitment loss*. The quantization loss pushes the quantized embeddings towards the latent vectors before the VQ layer. The commitment loss conversely pushes the latent vectors closer to the quantized embeddings, to ensure that the embedding space does not expand too fast. The VQ loss is computed for a sequence of N latent vectors as:

$$L_{\text{VQ}} = \sum_{n=1}^{N} \left(\| \text{sg}[\boldsymbol{z}_n] - \boldsymbol{e}_n \|_2^2 + \beta \| \boldsymbol{z}_n - \text{sg}[\boldsymbol{e}_n] \|_2^2 \right) \tag{3.2}$$

where sg$[\cdot]$ represents the stop gradient operator, which ensures that its operand is not updated (i.e. turning it into a constant during backpropagation). The first term in equation 3.2 is the quantization loss and the second term is the commitment loss. The hyperparameter β controls the importance of the commitment loss. ConEx uses a low value of $\beta = 0.05$.

Finally, to train the duration predictor, a separate *mean squared error (MSE)* or L2 loss is used. This error is calculated between the ground-truth phoneme durations and the predicted phoneme durations.

3.4.2 Training the autoregressive prior model

In the second training step, the autoregressive prior model is added and only the weights of this prior model are updated. The AR prior model takes the phoneme embeddings integrated with a global prosody embedding as input. The quantized fine-grained prosody embeddings, generated during the first training step, are used as the targets of the AR prior model. An MSE loss is calculated between the target fine-grained prosody embeddings and the predicted fine-grained prosody embeddings. This loss is backpropagated through the LSTM to update the weights of the prior model.

3.5 Inference

To generate speech, two inputs are needed: a phoneme sequence and a global prosody embedding. The phoneme sequence determines the content of the output speech and the global prosody embedding determines the speaking style of the output speech. The global prosody embedding can be input directly, or a mel spectrogram can be input to the reference encoder to obtain a global prosody embedding. Figure 3.3 shows the architecture during inference.

First, the FastSpeech encoder transforms the phoneme sequence into a sequence of phoneme embeddings. Then the global prosody embedding is added to all phoneme

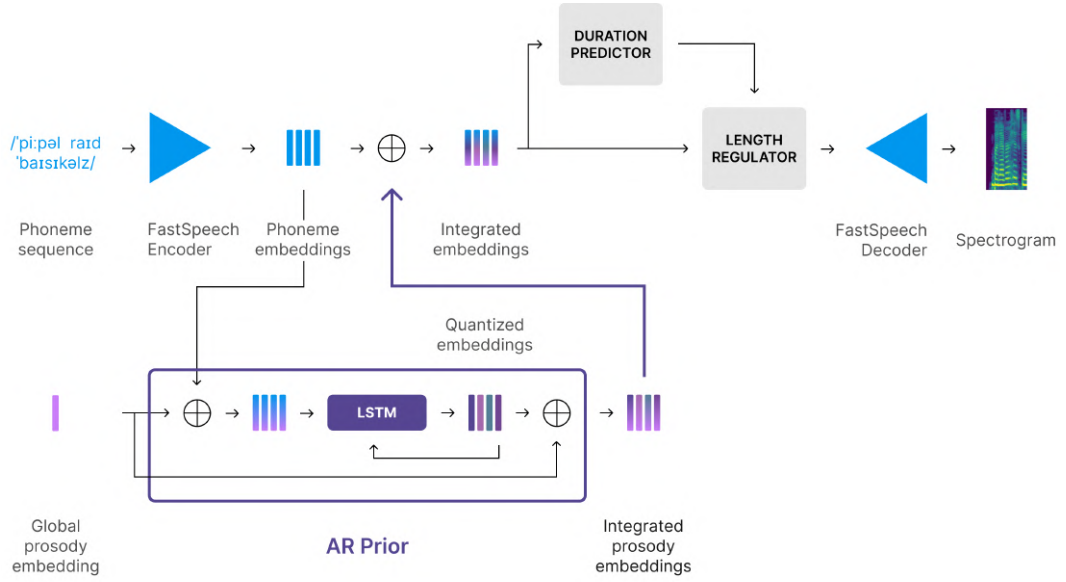


Figure 3.3: The ConEx architecture at inference time. The global prosody embedding can be input directly or can be generated by the reference encoder (see 3.3.2). The quantized embeddings are predicted autoregressively by the LSTM.

embeddings. If a mel spectrogram is input instead of a global prosody embedding, then the reference encoder first transforms this mel spectrogram to a global prosody embedding (after which it is added to the phoneme embeddings). Next, the autoregressive prior model takes these integrated embeddings and generates fine-grained prosody embeddings one by one[1]. These are added to the integrated embeddings. The duration predictor uses these embeddings to predict the duration for each embedding in the sequence. Then the length regulator copies the embeddings according to the predicted durations. Finally, the decoder transforms the embeddings to output mel spectrogram frames.

3.6 Controllability

The aim of ConEx is to obtain control over the global speaking style and the local prosody of synthesized speech. This section details how this controllability is achieved through. It describes the use of the global prosody embedding to achieve control over the speaking style, and the use of the autoregressive prior to make adjustments to the local prosody of the generated utterance.

3.6.1 Controlling the global speaking style

As described above, a global prosody embedding is required to generate speech for a new phoneme sequence. This global prosody embedding can be obtained by choosing

[1]Note that the fine-grained prosody encoder is never used during inference.

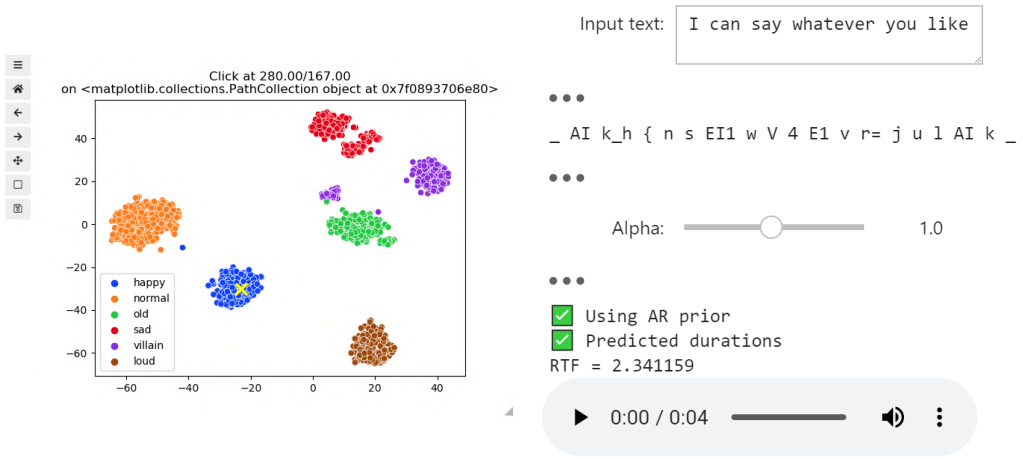


Figure 3.4: The process of generating speech in a certain style. The image on the left shows a t-SNE visualization of the global prosody embeddings learned from the Will dataset. The user selected a point in the blue cluster. Next, the user provided an input text (which was converted to phonemes) and selected the speed (controlled by alpha). Next, an output sample was produced using the input text and the selected prosody embedding.

a global prosody embedding from one of the training utterances (since they were generated during training). In this case, the style from the training utterance is transferred to the newly generated utterance. Alternatively, a new speech fragment[1] that was not included in the dataset can also be used. In that case, the reference encoder first creates a global prosody embedding from the new speech fragment's mel spectrogram. The generated speech will then have the speaking style of the speech fragment.

To facilitate choosing a suitable global prosody embedding, the global prosody embeddings generated during training can be visualized in a t-SNE plot[2] (van der Maaten and Hinton, 2008). Depending on the different speaking styles in the dataset, t-SNE typically shows some clusters of similar prosody embeddings. These clusters can then be used to choose similar, or completely different, speaking styles. Figure 3.4 shows this process for the Will dataset (which will be introduced in 4.1.1).

3.6.2 Making local prosody adjustments

The autoregressive prior is used to generate the quantized fine-grained prosody embeddings during inference. One fine-grained prosody embedding represents the prosody of a single phoneme. By changing the fine-grained prosody embeddings, the prosody can be changed locally. This section describes a novel method to make edits to the local

[1] A mel spectrogram suffices, i.e. a corresponding phoneme sequence is not necessary.

[2] *t-distributed Stochastic Neighbor Embedding (t-SNE)* is a dimensionality reduction technique that can be used to visualize high-dimensional vectors (such as embeddings) by mapping them (in a non-linear manner) to 2D/3D points. The technique aims to display similar vectors close to each other and dissimilar vectors far away from each other, thus typically creating clusters of similar vectors.

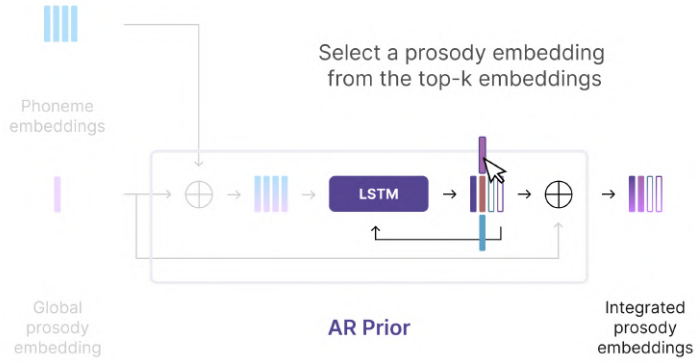


Figure 3.5: Controlling local prosody using the fine-grained embeddings predicted by the autoregressive prior model. Once a prosody embedding is selected from the top-k options, the following prosody embeddings are predicted autoregressively.

prosody. This method uses the predictions of the autoregressive prior model to alter the fine-grained prosody embeddings.

The fine-grained prosody embeddings are discrete, so they cannot be scaled or combined. To change the prosody, the individual embeddings instead have to be swapped out for other embeddings. This means that to obtain a desired prosody, the effect of the quantized embeddings in the codebook on the perceived prosody has to be known. Interpreting the quantized embeddings is impractical and time-consuming, and is only possible with a limited codebook size. Furthermore, swapping the fine-grained prosody embeddings for random embeddings could also hurt the naturalness of the utterance as a whole, since the chosen prosody embedding might not fit in.

Instead, the predictions of the autoregressive prior model can be used to swap fine-grained embeddings. As detailed before, this model outputs a categorical distributions over the different embeddings in the codebook. Normally, the most probable embedding would be used. To control the prosody, however, a user can choose any of the top-k predictions. This embedding is then used by the prior model to predict the next fine-grained embeddings, as the prior model is autoregressive. k can be set depending on how many different prosody options are desired. There exists a trade-off for this choice, as using a large k offers more diverse options, at a cost of possible reduced naturalness.

Figure 3.5 shows how the process of choosing a new fine-grained embedding works. First, the AR prior generates a complete fine-grained embedding sequence. This sequence is used to generate the first output speech. If the prosody of a certain phoneme has to be changed, the categorical distribution predicted by the AR prior model for that phoneme is used to show k options. When choosing an option, the rest of the fine-grained prosody embedding sequence is generated autoregressively. This sequence can then be decoded to speech again. The fine-grained prosody embeddings before the altered phoneme remain unchanged. Since generating speech using the proposed architecture is fast, this process allows local adjustments to prosody to be carried out relatively quickly. Figure

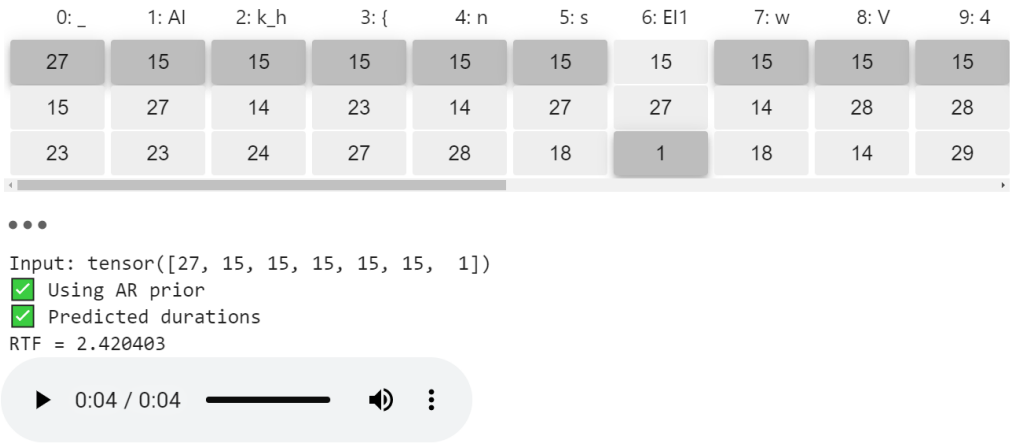


Figure 3.6: A user interface for making edits to the local prosody. The boxes show the indices of the three best fine-grained prosody embeddings (as predicted by the AR prior model). The user changed the prosody of phoneme 6 by clicking on a different embedding. The remainder of the prosody embedding sequence was then predicted autoregressively, and a new speech sample was generated.

3.6 shows a user selecting a different fine-grained embedding for a phoneme using the novel method.

3.7 Implementation

Successfully implementing a deep neural network is no mean feat, as modern neural network architectures can be highly complex and contain many components. All of these components need to be carefully implemented so that they use the right inputs and produce a correctly shaped output. Since neural networks operate on vectors, the inputs and outputs can be difficult to comprehend and debug. Furthermore, the flow of the loss signal through the network also needs to be considered, as this signal is used to update the model parameters, i.e. to train the model. Lastly, training neural networks requires large amounts of data, and thus also entails complicated pipelines to collect, transform, and utilize that data for training and evaluation.

Luckily, several mature frameworks for machine learning exist and make implementing neural networks more manageable. Frameworks such as PyTorch[1] and Tensorflow[2] provide high-level APIs to create and train neural networks. Basic neural network components such as projection layers, but also LSTMs, are included in these frameworks. Furthermore, these frameworks also offer automatic differentiation engines, which can automatically compute the gradients necessary to update the model parameters.

The ConEx model introduced in this chapter was implemented in Python using PyTorch (Paszke et al., 2019). Furthermore, it was implemented as an addition to ESPnet, an open-source speech processing toolkit (Hayashi et al., 2020). The implementation of

[1] `http://pytorch.org/`
[2] `https://www.tensorflow.org/`

the FastSpeech encoder, decoder, and length regulator was adapted from the implementation of FastSpeech in ESPnet. The implementation of the vector quantization layer is based on an open source PyTorch implementation by Subramanian (2020).

That being said, the model was not implemented in one go. Instead, it was developed iteratively by starting from the ESPnet FastSpeech implementation and progressively adding components to model and control prosody. For example, first only a global prosody encoder was implemented in the model. 4.3 describes this earlier iteration of ConEx, as it is used for an initial experiment on the use of a VQ-VAE for global prosody embeddings. The iterative process allowed some of the basic model components (such as the VQ-VAE) to be implemented and tested before other more advanced components were added.

Other than the model itself, the user interface described in section 3.6.1 and 3.6.2 was implemented in a Jupyter notebook[1]. This interface can be used to generate speech for any input text, while controlling the global speaking style and the local prosody. Additionally, a data pipeline was implemented using Python and Bash scripting. This pipeline allows ConEx to be trained using the datasets introduced in section 4.1.1. This proved challenging, as for example the phoneme durations had to be transformed and adapted, without any documentation. Finally, all the experiments were also implemented in Python, mostly in Jupyter notebooks.

3.8 Conclusion

This chapter introduced ConEx, a new neural network-based model for controllable expressive speech synthesis. Using ConEx, speech can be generated in a certain speaking style and the prosody of the generated speech can be adjusted locally. ConEx thus aims to solve the issue of controlling both global style and local prosody, as these both contribute to expressive speech.

ConEx is based on the non-autoregressive FastSpeech(Ren et al., 2019) architecture, but adds a prosody encoder that explicitly models prosody. The prosody encoder of ConEx consists of a reference encoder (modeled after Skerry-Ryan et al. (2018)) which encodes the global style, and a vector quantized variational autoencoder (inspired by (Sun et al., 2020)) which creates quantized fine-grained prosody embeddings. Both components use ground-truth mel spectrograms as inputs. The prosody encoder and the FastSpeech components are trained jointly.

When generating speech, ConEx takes a phoneme sequence and a global style embedding as inputs. The global embedding can be obtained from any reference speech clip. This global embedding thus enables control over the global speaking style. The fine-grained prosody embeddings, however, cannot be obtained from any reference clip, since the fine-grained embeddings are linked to the phonemes. Instead, a separate network, called the autoregressive prior, predicts the fine-grained embeddings at inference

[1] https://jupyter.org/

time. It is trained using the phoneme embeddings and global style embedding as inputs and the prosody embeddings from the VQ-VAE as targets.

To make adjustments to the local prosody of an utterance, the fine-grained prosody embeddings can be manipulated. The autoregressive prior model outputs the most probable fine-grained prosody embeddings for each phoneme. Instead of choosing the most probable embedding, any embedding of the top-k embeddings can be used, thus changing the prosody locally. The following prosody embeddings are generated autoregressively, and should thus ensure that the output speech is still consistent and natural.

The proposed model was implemented as an extension to ESPnet, a popular speech processing toolkit. It was implemented in Python using the PyTorch machine learning framework. Furthermore, a data pipeline and user interface for controlling the prosody were also implemented.

Chapter 4

Experiments

This chapter details the experiments that were carried out to test the ConEx architecture proposed in the previous chapter. In particular, this chapter aims to evaluate if the proposed architecture can be used to control the global speaking style and to make local adjustments to the prosody of the generated speech. Furthermore, this chapter also describes an initial experiment that was conducted using a simplified version of the architecture. This initial experiment evaluates the use of a VQ-VAE for learning prosody embeddings.

To conduct experiments on a TTS model, the model first needs to be trained. This chapter first describes the training set-up and the datasets used. Next the methodology and the evaluation metrics are discussed. Then the initial experiment using a simplified architecture is described and the results are discussed. Finally, two experiments are conducted on the complete ConEx architecture. The first experiment gauges the ability to control the global speaking style by performing style transfer. The second experiment aims to evaluate if fine-grained prosody embeddings allow control over the local prosody, in the way described in section 3.6.

4.1 Training set-up

This section first describes the two datasets used in the experiments and relevant training details. Then, the section details the hardware used to train the models.

4.1.1 Datasets

To train a deep neural network like ConEx, a great deal of training data is needed. Since this work is limited to single-speaker TTS, datasets containing a few dozen hours of speech recordings are necessary. For the different experiments described in this chapter, two different single-speaker datasets were used.

The first speech dataset is a proprietary dataset provided by Acapela Group. The dataset consists of 23.5 hours of speech data in US English. To obtain this data, a professional voice actor (Will) recorded 13686 utterances in one of seven styles. Of the seven styles, six

were used for the experiments: *normal, happy, sad, old man, villain,* and *from afar/loud*. In what follows, this dataset will be referred to as the "Will dataset".

The second dataset is the often-used 2013 Blizzard Challenge dataset (King and Karaiskos, 2013). This dataset consists of 147 hours of speech data taken from 49 audiobooks narrated by Catherine Byers. Since the recordings come from narrations of classic novels, the utterances are highly expressive. However, the dataset does not include any additional style, prosody, or genre annotations.

For all experiments, phonemes were used as input to the encoder instead of characters. This is done to disambiguate the pronunciation of the input words and ease the text-to-speech mapping problem (see section 2.2.1). Acapela Group provided the software needed to convert text to phonemes.

In addition to this software, Acapela Group also provided the durations of the phonemes in the recordings. These durations are used to train the model's duration predictor as well as align the ground-truth mel spectrogram to the phoneme embeddings for learning fine-grained prosody embeddings.

Before training the models, the datasets were first split into three distinct sets: a training, validation, and test set. The <phoneme, audio> pairs in the training set were used to train the model. The validation set was used to track the performance of a model during training and to choose the best versions of the model. Finally, the test set (also called the holdout set) contains examples that were only used for evaluation purposes after training.

4.1.2 Training details

All models used for the experiments were trained for 400.000 steps. Training one model takes 42-48 hours using the hardware described in the next section. The Adam optimizer (Kingma and Ba, 2014) was used ($\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-4}$) with the learning rate schedule described in Vaswani et al. (2017).

As detailed in section 3.4.2, the autoregressive prior model should not be trained together with the TTS architecture. Instead, it was trained after the TTS model for 25.000 steps. The same optimizer and learning schedule were used. This process took around 3h30 on the hardware described in the next section.

As explained in section 2.5.3, VQ-VAEs do not suffer from the posterior collapse issue, which plagues regular VAEs. However, another problem occurred when training the VQ-VAE in the ConEx model. The number of active prosody embeddings collapsed on multiple occasions, as shown in figure 4.1. This issue, called *index collapse* (Kaiser et al., 2018), can occur when an embedding is close to many encoder outputs initially. In that case, the VQ loss will push the encoder outputs even closer to this embedding and the embedding closer to the outputs. This feedback loop causes only a limited number of discrete embeddings to represent the data, resulting in an underused latent space.

To combat this issue, Kaiser et al. (2018) proposed decomposing the encoder outputs. However, such a method was not used in this work. Instead, by initializing the em-

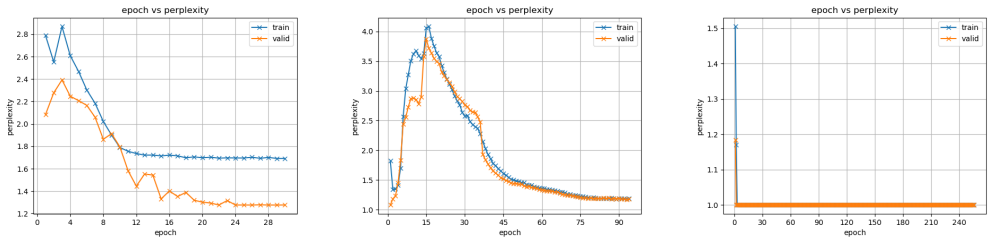


Figure 4.1: Index collapse occurring during training of the VQ-VAE in ConEx on three separate occasions. The perplexity metric indicates how many embeddings are active on average.

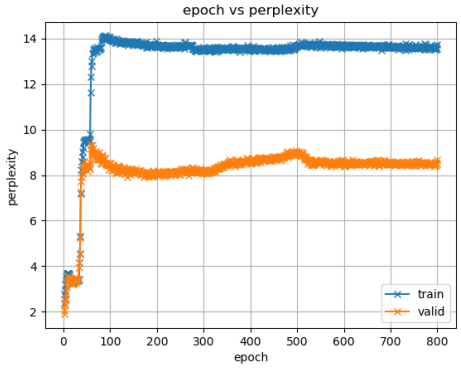


Figure 4.2: The perplexity during VQ-VAE training with the embeddings initialized close to the initial encoder outputs.

beddings using a normal distribution close to the initial encoder outputs, the issue of index collapse was prevented. The result was that around 8 embeddings were active on average for the validation set, as shown in figure 4.2.

4.1.3 Hardware

Training deep neural networks requires a powerful GPU. The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government. Concretely, a single NVIDIA P100 GPU (SXM2 at 1.3 GHz with 16 GB GDDR) GPU was used to train the models.

Generating speech using the model does not require a GPU or special powerful hardware. Thus, an Intel Core i73610QM CPU (at 2.3 GHz with 4 cores and 8GB RAM) was used for generating speech for the various experiments.

4.2 Methodology & evaluation

The main aim of the experiments is to evaluate the controllability of the proposed model, i.e., whether the global speaking style and the local prosody can be controlled. Two

kinds of experiments are conducted: global style transfer experiments and local prosody control experiments. This section discusses the methodology behind these experiments, and how they will be evaluated.

4.2.1 Global style transfer

The goal of style or prosody transfer is to generate a speech utterance in the style of a reference speech utterance using a latent representation of the reference utterance. The text of the generated utterance and the reference utterance do not have to match. The latent representation is a global, fixed-length vector and represents a certain speaking style.

To evaluate style transfer, two speech outputs, X and Y, are generated. Speech sample X is generated conditioned on the style embedding from a reference utterance A. Speech sample Y is a baseline speech utterance not conditioned on the style embedding. Then the style transfer can be evaluated by comparing the two speech outputs to the reference utterance. If the style transfer was successful, then X should be more similar to A than Y. The similarity can either be measured objectively or subjectively.

Objective measures for evaluating style transfer have been proposed by Skerry-Ryan et al. (2018) and Tits (2020). Skerry-Ryan et al. adapt metrics from general audio processing to measure differences[1] in properties relating to prosody, and Tits proposes additional objective measures. In this work, the *Mel Cepstral Distortion (MCD)* and the F_0 MSE are used.

Mel Cepstral Distortion This metric, originally introduced by Kubichek (1993), is a measure of the difference between two speech samples. It is defined in terms of the *mel frequency cepstral coefficients (MFCCs)* of the speech samples. It is computed as:

$$\text{MCD} = \frac{1}{T}\sum_{t=0}^{T-1}\sqrt{\sum_{k=1}^{K}(c_{t,k}-c'_{t,k})^2}$$

where $c_{t,k}$, $c'_{t,k}$ are the k^{th} MFCCs of the t^{th} frame of the two speech samples. The first coefficient ($c_{t,0}$) is skipped.

F_0 Mean Squared Error This metric measures the distance between the F_0-contours of two speech samples. It is defined as:

$$F_0\,\text{MSE} = \frac{1}{T}\sum_{t=0}^{T-1}(F_{0\,t}-F'_{0\,t})^2$$

These metrics directly compare corresponding MFCCs and F_0 values respectively. This implies that the speech sequences being compared have to be of the same length. In the

[1] When using metrics that measure the difference between the reference utterance and the generated utterance, then successful style transfer means that the difference between X and A is smaller than the difference between Y and A.

case of comparing a generated speech sequence to a reference speech sequence, this requirement is not met. The speech sequences thus have to be aligned before calculating the metrics.

To align the sequences, *Dynamic Time Warping (DTW)* can be used. However, aligning the sequences changes the phoneme durations and thus the rhythm of the speech. This means that the metrics can not show if the rhythm of the generated speech sample matches the reference speech utterance.

Style transfer can also be evaluated subjectively by conducting a listening test. A listening test for style transfer is usually structured as a AXY discrimination test, as proposed by Skerry-Ryan et al. (2018). This entails that the human taking the test has to choose if the style of speech sample X is more similar to the reference A, or if Y is closer. Usually, a 5 or 7-point scale is used to obtain finer results. Conducting such a listening test is out of scope of this work, as it is costly to conduct properly.

4.2.2 Control over local prosody

In the second kind of experiment, the aim is to evaluate the level of control over local prosody. To evaluate this, different qualitative tests are carried out, whereby the effects of changing the fine-grained prosody embeddings are analyzed. This approach is similar to the controllability experiments carried out by Sun et al. (2020); Zhang et al. (2020).

In ConEx, local prosodic information is encoded in the fine-grained (phoneme-level) prosody embeddings, which are generated by the autoregressive prior model at inference time. These fine-grained prosody embeddings can be swapped to change the prosody, as described in section 3.6.2. The tests aim to answer three questions relating to this procedure:

- Can the fine-grained prosody embeddings be swapped to obtain the desired local prosody?
- How local are the effects of changing a fine-grained prosody embedding?
- How diverse are the top-k embeddings proposed by the AR prior model?

Furthermore, the effect of the AR prior model on the prosody is also tested by comparing it against a random sampling scheme. This test aims to determine if the AR prior model can learn to predict the fine-grained prosody embeddings, such that the resulting speech sounds natural. The test should also offer insight into the effect of the different fine-grained prosody embeddings, as the random sampling scheme can also generate embeddings that are not natural for a certain sentence.

Quantitative results could additionally be obtained by labeling the prosody of a test utterance before and after a change, and seeing if the desired effect was achieved. However, as mentioned in section 2.5, it is not easy to label prosody completely. Even complex prosody labeling systems still suffer from inter-annotator disagreement, because human

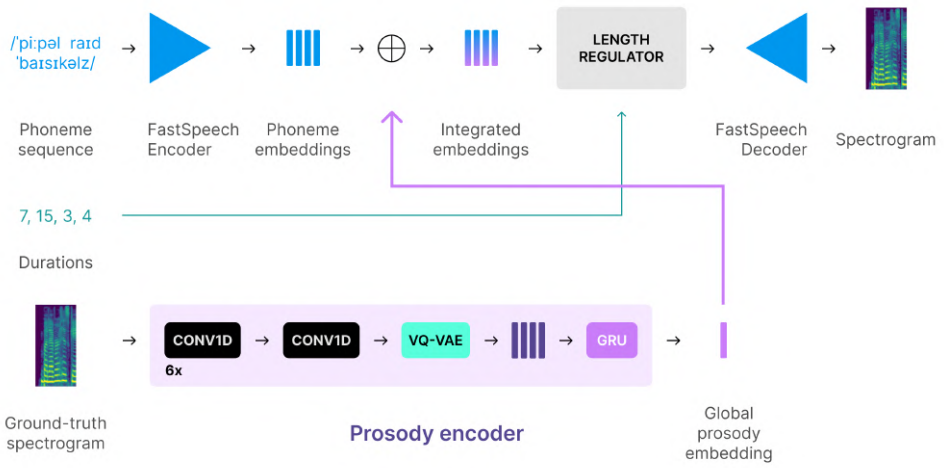


Figure 4.3: Architecture of the global prosody embedding model.

annotators can perceive and interpret prosodic phenomena differently. The experiments thus do not use any ground-truth prosody labels for evaluation.

Lastly, a subjective listening experiment could be carried out, as is common in speech synthesis research. However, most listening tests evaluate the naturalness of the generated outputs (Sun et al., 2020; Zhang et al., 2020), while controllability is of greater interest in this work. Thus, a listening experiment is not conducted in this work.

4.3 Initial experiment results

The section details the set-up and results of the initial experiment, which used a simplified version of ConEx. This experiment is a stepping stone to the experiments on the proposed ConEx architecture in the next section. First, this simplified architecture is described and then the results of the experiment are discussed.

The aim of the initial experiment was to assess whether a vector quantized variational autoencoder (introduced in section 2.5.3) could be used to learn global style embeddings, and if these embeddings could be used for global style transfer. The Will dataset (described in section 4.1.1), was used, as the six distinct speaking styles make it well-suited for evaluating the learned global prosody embeddings and global style transfer.

Simplified model architecture

This experiment did not use the full ConEx architecture described in section 3.3. Instead, only a single VQ-VAE was used to encode the prosody globally. Figure 4.3 shows this simplified architecture. Implementing this simplified architecture first, proved to be valuable for the development of the complete ConEx model.

The architecture of the global prosody encoder is similar to that of the reference encoder (Skerry-Ryan et al., 2018), but includes a vector quantization layer. This layer acts as a

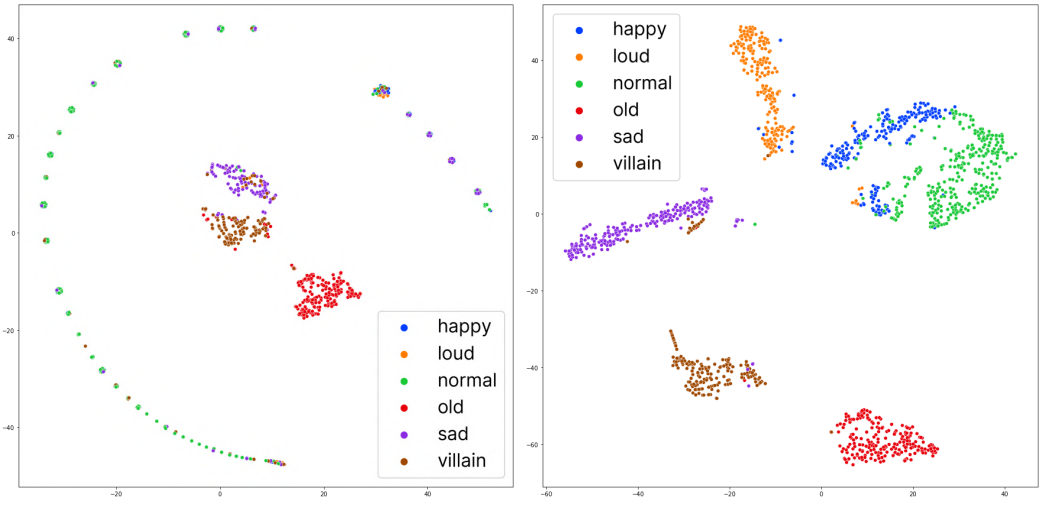


Figure 4.4: t-SNE visualization of the global prosody embeddings and the different speaking style they represent. Left: the embeddings after 12 hours of training. Right: the embeddings after the training was completed (after 42 hours).

bottleneck for the information that can flow from the ground-truth mel spectrogram to the decoder.

The global prosody encoder consists of 6 one-dimensional convolutional layers with 128 hidden channels, a kernel size of 4, and a stride of 2, followed by one 1-D convolutional layer with 64 channels, a kernel size of 3, and a stride of 1. After this, the downsampled mel spectrogram frames are quantized using a codebook of 32 vectors with 64 dimensions. Finally, a 64-width GRU layer is used to compress the sequence of quantized latents to a fixed-length prosody embedding.

Qualitative results

First, the learned global prosody embeddings are inspected visually by using a t-SNE visualizations. Figure 4.4 shows a t-SNE visualization of the learned embeddings during and after training. After 12 hours of training, the embeddings could differentiate between three of the six speaking styles in the dataset; the *old, sad,* and *villain* utterances already appear in different clusters in the t-SNE visualization. After the training was completed, all of the six speaking styles in the dataset are separated into their own clusters in the t-SNE plot with minimal overlap.

Next, to evaluate the model outputs, random examples from the holdout dataset were used to generate speech utterances. The Griffin-Lim algorithm (Griffin and Lim, 1984) was used as the vocoder. For this test, the model used the ground-truth audio to create the global prosody embedding. Thus, the prosody of the generated utterances should match the prosody of the original utterances. Generated speech samples confirm that the model can indeed reproduce the different speaking styles. Readers are encouraged

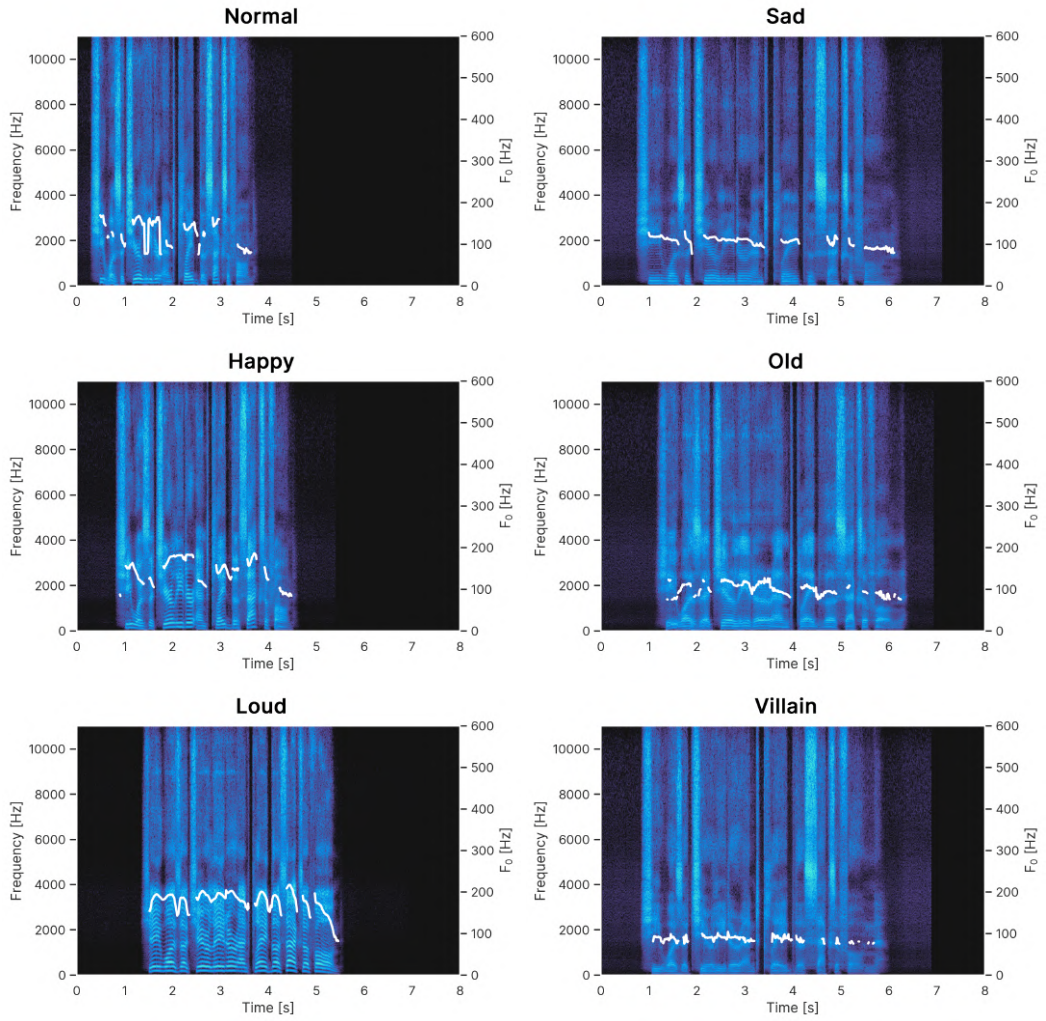


Figure 4.5: Mel spectrograms with F_0 contours for the sentence "She was a cheerleader and played the saxophone" in the six distinct speaking styles of the Will dataset. The mel spectrograms were generated using the simplified model.

to listen to these audio samples on the demo page[1].

To evaluate if these learned embeddings allow for global style transfer, speech utterances were generated using the same phoneme sequence and six different prosody embeddings. The prosody embeddings were chosen to be good representatives for one of the six speaking styles. The mel spectrograms generated by the model were converted to waveforms using a pretrained parallel WaveGAN vocoder (Yamamoto et al., 2020). The vocoder was trained on the LibriTTS corpus (Zen et al., 2019) by Tomoki Hayashi[2] [3].

[1] `https://bit.ly/conex-samples`

[2] `https://github.com/kan-bayashi/ParallelWaveGAN`

[3] Because the vocoder was trained on a different corpus than the TTS model, the voices sound a little distorted and some artifacts are audible in the generated utterances. However, the speech still sounds

Speaking style	**Mean MCD (↓)**		**Mean F_0 MSE (↓)**	
	AX	*AY*	*AX*	*AY*
Happy	3.74	**3.60**	**7274.82**	7562.63
Loud	**5.13**	6.14	**6559.18**	8024.88
Old	**3.55**	3.85	**4653.62**	6001.33
Sad	**4.45**	4.56	**3506.03**	5627.45
Villain	3.84	**3.75**	**3202.86**	6014.79

Table 4.1: Mean MCD and mean F_0 MSE values for the difference between the samples generated using a global prosody embedding and the reference sample (AX) and the difference between baseline samples and the reference sample (AY). The samples were generated using the simplified model trained on the Will dataset.

The demo page shows the results of this style transfer test for the phrase "She was a cheerleader and played the saxophone". Listening to the audio samples reveals that the six styles are transferred well. The 'happy' style is the least distinct style, but this is also the case in the reference audio samples.

Figure 4.5 shows the mel spectrogram and F_0 or pitch contour of the six generated speech samples shown on the demo page. The figure clearly shows that the *sad*, *old*, and *villain* speaking styles cause the phonemes to be much longer. The pitch also changes between the samples; the *normal* and *happy* speaking style have a large pitch range, the *old* and *villain* samples have a low pitch, and the *sad* speaking style's pitch always bends down at the end of a word. Finally, the energy also differs between the speaking styles; the figure shows that the *loud* utterance retains a high energy level throughout the clip[1].

Quantitative results

To evaluate global style transfer quantitatively, an objective AXY test was carried out as described in section 4.2.1. In particular, five global prosody embeddings were extracted from reference samples (the A's), each representing a different speaking style. Speech samples (the Y's) were generated using all of the input phoneme sequences in the hold-out set combined with the extracted global prosody embeddings. The baseline set (the Y's) consisted of all of the input phoneme sequences in the hold-out set mapped to speech with a *normal* speaking style.

Table 4.1 shows the results of the AXY objective test using the simplified model. All utterances generated using a global style embedding, except for the *happy* and *villain* utterances, score better than the baseline on both metrics. These results thus indicate that the simplified model is capable of transferring the style of a reference utterance to the generated speech, by using a global prosody embedding.

The higher MCD value for *happy* utterance is likely due to the fact that the *happy* style is very similar to the baseline *normal* style. Because of this, the MFCCs of the *normal* baseline is closer to the MFCCs of the reference *happy* utterance on average. This is

more natural than when using the Griffin-Lim algorithm.

[1]The energy or loudness is marked by the colors, with a lighter color representing more energy.

also supported figure 4.4, as both styles are very close and even have some overlap in the t-SNE visualization. The MCD value for *villain* is more surprising, since the F_0 MSE seems to indicate that the pitch is predicted very well. This could indicate that the energy of the *villain* utterances is not transferred well.

4.4 ConEx experiments results

This section describes the results of the experiments carried out using the ConEx model architecture. Both the Will and Blizzard dataset (described in section 4.1.1), were used in the experiments. As mentioned before, the main goal is to assess the controllability ConEx offers.

The proposed architecture uses global prosody embeddings to enable control over the global speaking style. Specifically, style transfer can be performed to change the global speaking style. Thus, this section first describes the qualitative and quantitative results of style transfer experiments.

ConEx also uses an autoregressive prior model to predict fine-grained prosody embeddings when generating speech. The sequence of fine-grained embeddings can be altered as described in 3.6.2. Thus, after the style transfer experiments, the results of using this local prosody control method are described.

4.4.1 Global style transfer

Qualitative results

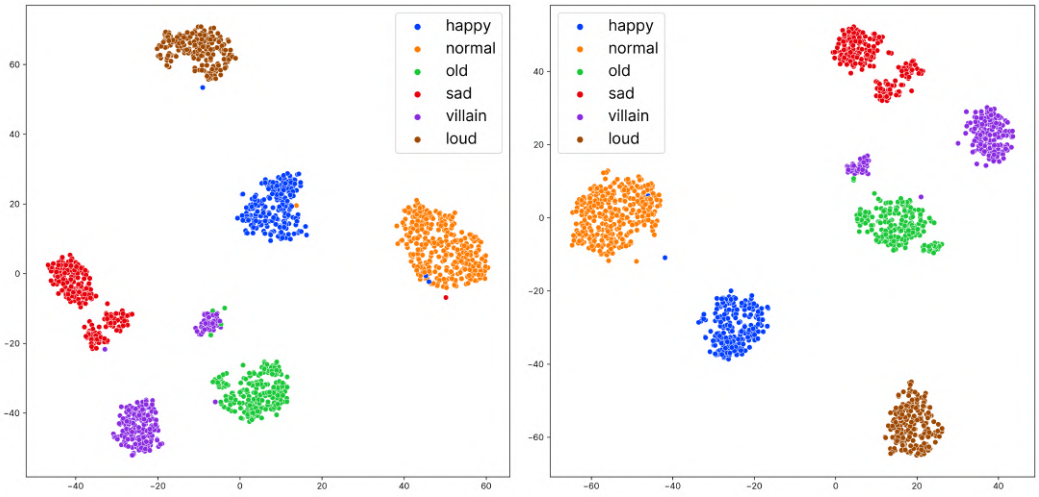


Figure 4.6: t-SNE visualization of the global prosody embeddings learned from the Will dataset. The colors indicate the different speaking styles in the Will dataset. Left: the embeddings after 6 hours of training. Right: the embeddings after the training was completed (after 48 hours).

Figure 4.6 shows a t-SNE visualization for the learned global prosody embeddings when training ConEx on the Will dataset. The styles in the Will dataset were separated into one

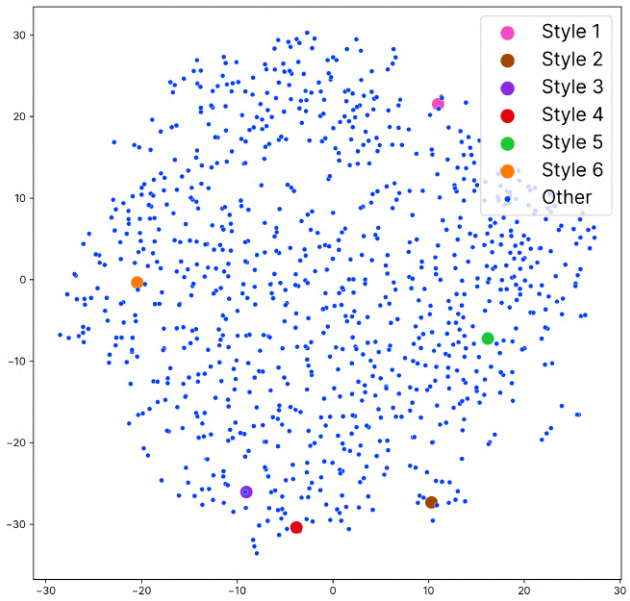


Figure 4.7: t-SNE visualization of the global prosody embeddings learned from the Blizzard dataset (after the training was completed). The six global prosody embeddings that were chosen for the style transfer experiments are marked in different colors.

or more clusters per different speaking style in the dataset. What is more, this separation occurred earlier during training than when training the simplified model. There is also fewer overlap between styles.

Figure 4.7 shows the same figure, but when the ConEx model was trained on the Blizzard dataset. This dataset does not contain distinct labeled speaking styles, and thus, the figure can not be colored according to speaking styles. The t-SNE visualization shows no distinct clusters, which indicates that the global speaking styles in the dataset are not very distinct.

As in the initial experiment, the next test consists of comparing the generated samples with the ground-truth samples. For this test, the global and fine-grained embeddings are extracted from the ground-truth mel spectrograms. As such, the prosody of the generated samples should match the prosody of the ground-truth sample precisely. Listening to some random generated samples, confirms that the model successfully encoded the global speaking style and the fine-grained prosody, both when trained on the Will dataset and on the Blizzard dataset.

To evaluate the global style transfer qualitatively, speech samples were generated in the style of some reference samples. To do so, the global style embeddings of the reference samples were used. For the Will dataset, one global prosody embedding was chosen per speaking style in the dataset (six in total). For the Blizzard dataset, six reference utterances with distinctive prosody were chosen manually. Figure 4.7 shows the chosen global prosody embeddings in the 2D t-SNE space. The distances between the points in the t-SNE plot correspond largely to subjective listening experience, as style 2, 3, and 4

indeed sound the most alike.

Three input sentences were mapped to speech in the different styles. A parallel WaveGAN vocoder (Yamamoto et al., 2020), pre-trained on the LibriTTS corpus (Zen et al., 2019)/footnoteThe LibriTTS corpus consists of many speakers, both male and female, and can thus be used as the vocoder for the Will and the Blizzard dataset. However, since the exact voices of Will and Catherine Byers are not included in the dataset, the generated audio samples distort their voices., was used. The resulting audio samples are included on the demo page.

Listening to the audio samples reveals that the distinct speaking styles of the Will dataset were all transferred well. The chosen styles were also transferred when using the Blizzard dataset, but not as well as the speaking styles of the Will dataset. Styles 4, 5, and 6 can be recognized well in the generated samples, but style 1, 2, and 3 sound rather alike and miss some of the characteristics of the reference samples. In particular, it appears that the speaking rate and rhythm do not seem to be transferred. This is for example apparent for style 2, which is fast and agitated, while the generated samples are not.

The results for the test phrase "She was a cheerleader and played the saxophone" are visualized in figure 4.8 and 4.9 for the Will and the Blizzard dataset, respectively. The figures show the mel spectrograms and the F_0 contours. Note that the pitch contours are rather noisy and thus hard to analyze.

The mel spectrograms of the different styles in the Will dataset resemble the mel spectrograms from the initial results strongly (see figure 4.5). The same prosodic patterns can be noted. For the Blizzard dataset, variations in the pitch contours and phoneme durations can be noted, although it is harder to discern specific styles visually, as they are less distinct compared to the speaking styles in the Will dataset

Quantitative results

Finally, an objective AXY test was carried out to evaluate global style transfer with the ConEx model. Tables 4.2 and 4.3 show the mean MCD and F_0 MSE values. For both datasets, the results show that the samples generated using the global prosody embedding are closer to the reference sample than the baseline samples (created using a neutral style).

The only exception is the F_0 MSE for the fourth style for the Blizzard dataset, although the mean MCD is significantly smaller. These good results further indicate that the problems noted in the qualitative analysis for the Blizzard dataset are due to phoneme durations, as the durations are altered by DTW (see section 4.2.1).

4.4.2 Fine-grained prosody control

This section describes the experiments that were carried out to evaluate fine-grained prosody control with the ConEx model. Specifically, section 3.6.2 introduced a new method for editing local prosody by choosing fine-grained prosody embeddings from the outputs of the autoregressive prior. This section first evaluates the AR prior and

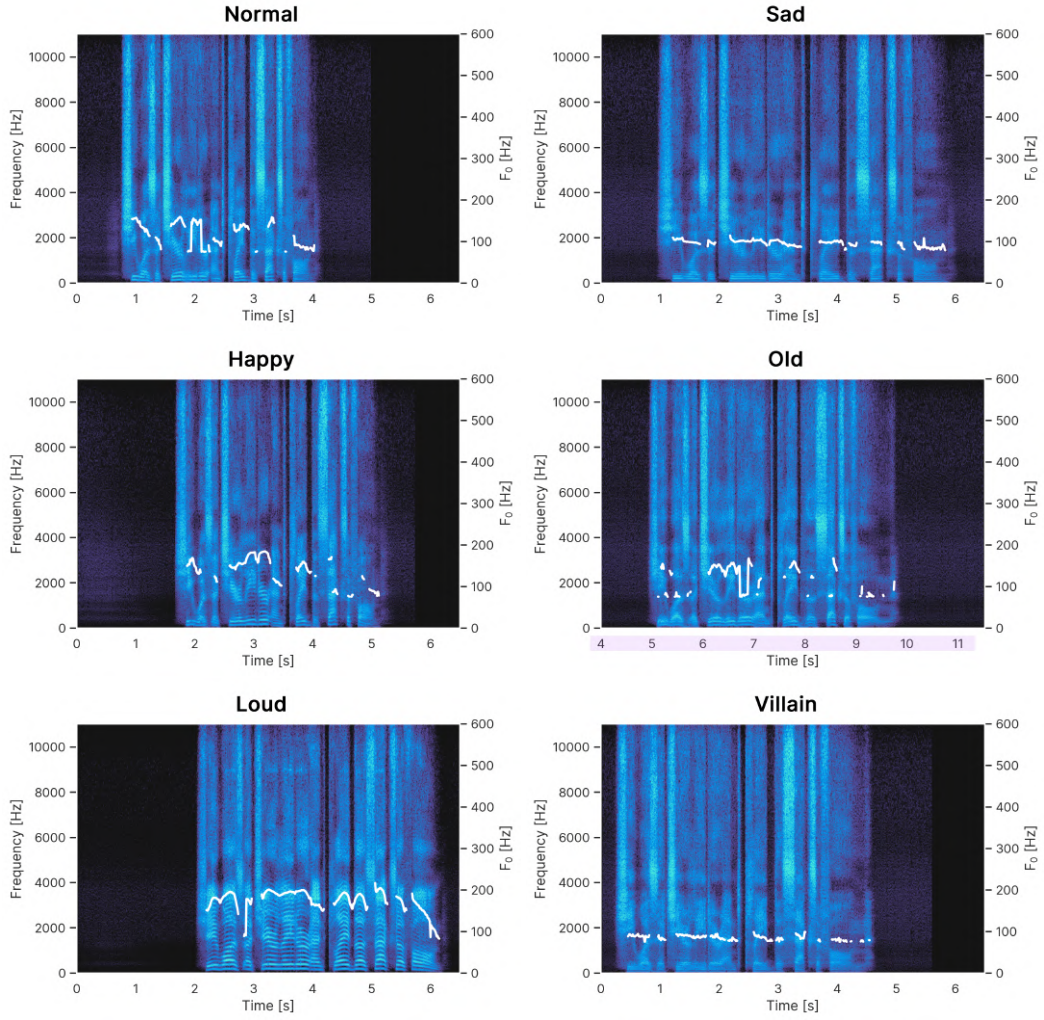


Figure 4.8: Mel spectrograms with F_0 contours for the sentence “She was a cheerleader and played the saxophone” in the distinct speaking styles of the Will dataset. The mel spectrograms were generated using the ConEx model.

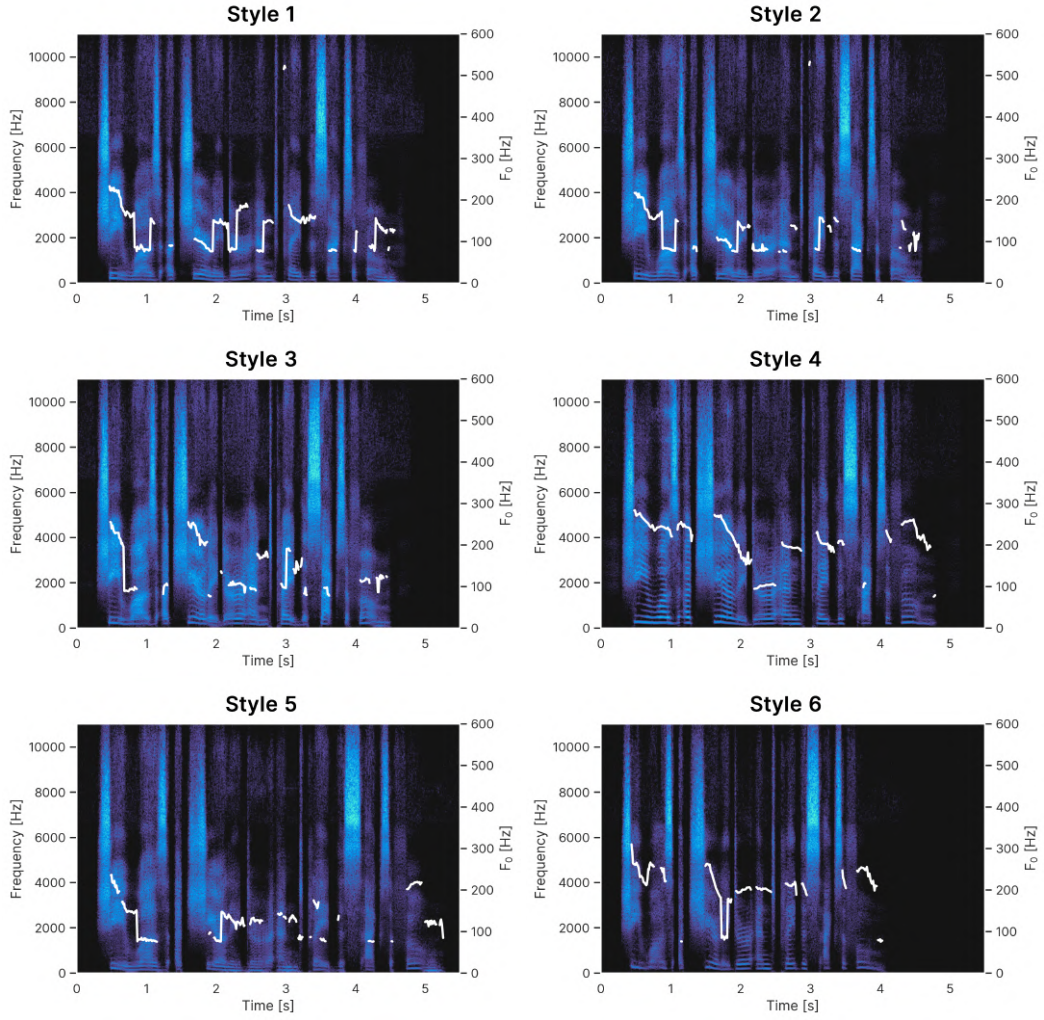


Figure 4.9: Mel spectrograms with F_0 contours for the sentence “She was a cheerleader and played the saxophone” in different speaking styles extracted from the Blizzard dataset. The mel spectrograms were generated using the ConEx model.

Speaking style	**Mean MCD (↓)**		**Mean F_0 MSE (↓)**	
	AX	*AY*	*AX*	*AY*
Happy	**3.61**	3.97	**6992.11**	7270.97
Loud	**5.30**	6.61	**8102.28**	10089.48
Old	**2.55**	3.05	**3721.09**	5635.80
Sad	**4.16**	4.63	**3090.39**	5820.60
Villain	**3.32**	3.83	**1164.74**	9991.15

Table 4.2: Mean MCD and mean F_0 MSE values for the difference between the samples generated using a global prosody embedding and the reference sample (AX) and the difference between baseline samples and the reference sample (AY). The samples were generated using the ConEx model trained on the Will dataset.

Speaking style	**Mean MCD (↓)**		**Mean F_0 MSE (↓)**	
	AX	*AY*	*AX*	*AY*
Style 1	**3.49**	3.70	**11270.39**	11806.36
Style 2	**3.44**	3.56	**11469.32**	11338.54
Style 3	**3.29**	3.74	**14614.33**	14778.82
Style 4	**3.22**	3.75	15728.65	**14835.17**
Style 5	**3.56**	3.67	**9979.27**	10403.00
Style 6	**3.79**	4.04	**15230.49**	15340.72

Table 4.3: Mean MCD and mean F_0 MSE values for the difference between the samples generated using a global prosody embedding and the reference sample (AX) and the difference between baseline samples and the reference sample (AY). The samples were generated using the ConEx model trained on the Blizzard dataset.

the learned fine-grained embeddings. Then it describes the tests used to evaluate the controllability and the results. Finally, the results of the local prosody transfer test is described.

AR prior & fine-grained prosody embeddings evaluation

When generating speech using ConEx, the fine-grained prosody embeddings are generated using an autoregressive prior model. A first experiment compared this autoregressive prior to randomly sampling fine-grained prosody embeddings, without taking into account any prior information. For the experiment, all example phrases in the hold-out set were used to generate speech samples, first using random sampling of the fine-grained prosody embeddings, then using the autoregressive prior model. Some random output samples are included on the demo page.

The results for the Blizzard dataset show that random sampling lead to incoherent speech, with strange prosody that changes heavily during the samples. The samples also show that words in these samples were emphasized often. This result indicates that the fine-grained prosody embeddings indeed encode local prosody not determined by the global prosody embedding. Furthermore, using the fine-grained prosody embeddings from the autoregressive prior improved the prosody dramatically. This finding confirms

the usefulness of an autoregressive prior, as proposed by Sun et al. (2020).

The samples from the Will dataset paint a different picture. Here, it seems that there is less variety in prosody for the samples generated by randomly sampling fine-grained prosody embeddings. This could indicate that most of the prosody is already determined by the different speaking styles (which are captured by the global prosody embeddings), and that there is very limited prosodic variation within the styles. Unfortunately, this means that local prosody edits will have limited effects, as there is less room for changing the prosody.

Controllability

For local prosody control, experiments were carried out to test the local prosody control method introduced in section 3.6.2. First, an output speech utterance was generated using the top fine-grained prosody embeddings predicted by the AR prior model. Then, the prosody embeddings for a certain phoneme were edited by selecting one of the other top-3 embedding options. The experiments used the parallel WaveGAN vocoder for the Blizzard dataset and Griffin-Lim was used for the Will dataset. The results of these tests are now detailed, starting with the Blizzard results.

The first test input is "I didn't say he stole the money". This sentence is interesting since each word can be emphasized to alter the meaning of the sentence[1]. The goal of this first test was thus to assess if the local prosody could be edited to emphasize each word individually. Thus, seven samples were generated.

For the Blizzard dataset, the local prosody editing technique was successful at emphasizing 4 out of the 7 words. The initial predictions of the AR prior model lead to the emphasis on "didn't". By choosing a different fine-grained embedding for the phoneme corresponding to "I", this phoneme was emphasized. However, the result was not very clear. As such, a pause was also added to the phoneme sequence, but this resulted in less natural speech. Next, "say" was emphasized. An alternative version was also generated by also changing the beginning of the sample. Figure 4.10 shows both changes visually. Lastly, "stole" was emphasized.

"he", "the", and "money" could not be emphasized by choosing different fine-grained prosody embeddings from the top 3. Changing the prosody embedding of the phoneme embedding corresponding to "he" lead to a change in prosody for "say". The edit's effects are were thus not local. Furthermore, to investigate if the "o" in "money" could be emphasized at all by changing the fine-grained prosody embeddings, every one of the 32 fine-grained embeddings was used for that phoneme. The results show that two embeddings indeed result in an emphasized "o". However, the majority of the codes caused changes in the prosody of "stole".

The results for the second example sentence ("Whenever you feel like criticizing anyone, he told me, just remember that all the people in this world haven't had the advantages

[1] For example, when saying "I didn't say he stole the money", the speaker indicates that he previously said something different, e.g. that the other person borrowed the money

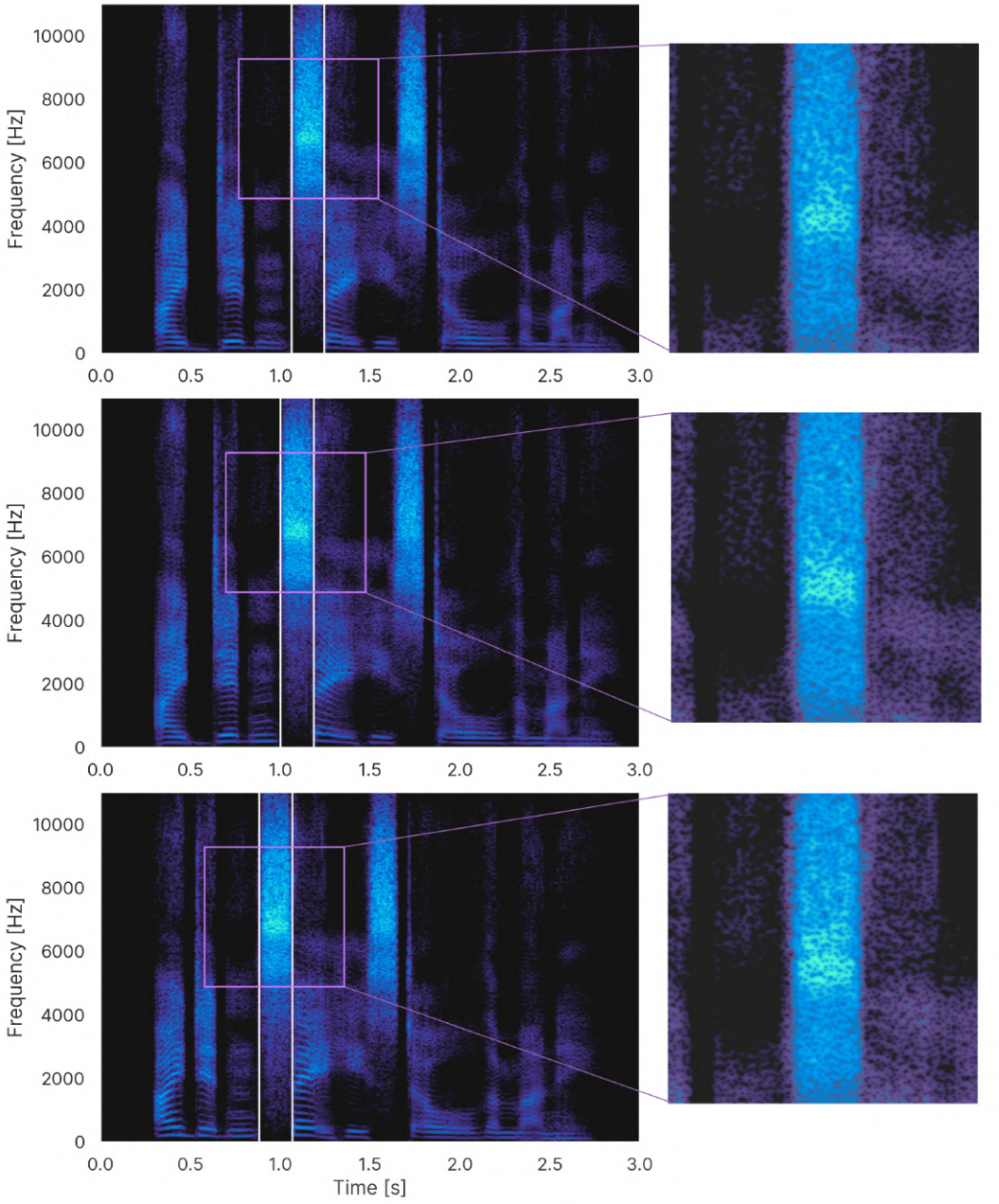


Figure 4.10: Mel spectrograms for the sentence “I didn’t say he stole the money” generated by the ConEx model trained on the Blizzard dataset. The top image shows the resulting spectrogram when using the predictions from the AR prior. The middle image shows the sample, where the fine-grained prosody embedding corresponding to the “a” in “say” was replaced by a different prosody embedding, resulting in more emphasis on the word. The bottom image shows an alternate edit, where the first phonemes of the sentence were also altered. The differences are subtle, but easy to discern when listening.

that you had") for the Blizzard dataset show that the diversity of the the top-3 fine-grained prosody embeddings can sometimes be limited. When the prosody embedding for the phoneme corresponding to the "a" in "anyone" was changed for the second option, the output speech only changed slightly: the boundary between "criticizing" and "anyone" disappeared, making the words sound more linked.

Furthermore, the third prosody embedding option for that phoneme caused a change in a previous phoneme: the pitch of "ci" in "criticizing" was altered. This means that changing the fine-grained prosody embeddings does not only result in local changes. This complicates the process of making targeted edits to local prosody.

The third example sentence ("This would have changed the grand result of the war") also showed this effect when paired with the Blizzard dataset. When the fine-grained prosody embeddings were swapped for the phonemes corresponding to "wa" in "war", the prosody of "sult" in "result"changed. The fine-grained prosody embeddings thus not only effect their corresponding phonemes, but also other prior phonemes.

The results for the Blizzard dataset thus show that effects of the edits are not always local. How can this happen, given that each fine-grained prosody embedding is only added to the single corresponding phoneme embedding? The answer is that the ConEx decoder relies on self-attention to transform the embeddings into mel spectrogram frames. This self-attention mechanism allows the decoder to use information from multiple embeddings to generate a spectrogram frame.

When using the model trained on the Will dataset, the first example sentence already showed that the proposed technique fails to make edits to the local prosody. The top predictions from the AR prior model all resulted in the same prosody. Even when using the "happy" speaking style, which is more varied than the other styles in the dataset, the technique failed to offer local prosody control.

To investigate the reason, all fine-grained embeddings were tried out for the phoneme corresponding to the "a" in "say" and the "o" in "stole". The results showed that most embeddings resulted in the same prosody. Only very few embeddings changed the prosody noticeably, and mostly affected the phoneme duration. Figure 4.11 shows the effect of a prosody embedding. This lack of diverse prosody embeddings can be explained by the fact that the Will dataset contains less variety in fine-grained prosody, as the voice actor focused on speaking in a certain global speaking style, while not varying other speech characteristics much. Thus, the fine-grained prosody embeddings do not encode diverse prosodic phenomena, and the embeddings thus do not result in great changes. Since many embeddings do not have any effect, the top options predicted by the AR prior model all result in the same prosody.

For the second and third example sentence, the AR prior technique also failed to change the prosody. However, in the experiments a fine-grained prosody embedding was found that results in shortening the phoneme. Audio samples using this prosody embedding are included on the demo page. Thus, when a dataset contains fewer fine-grained differences in prosody, the local prosody can be controlled by changing the fine-grained prosody embeddings directly (instead of using AR prior predictions). To make this

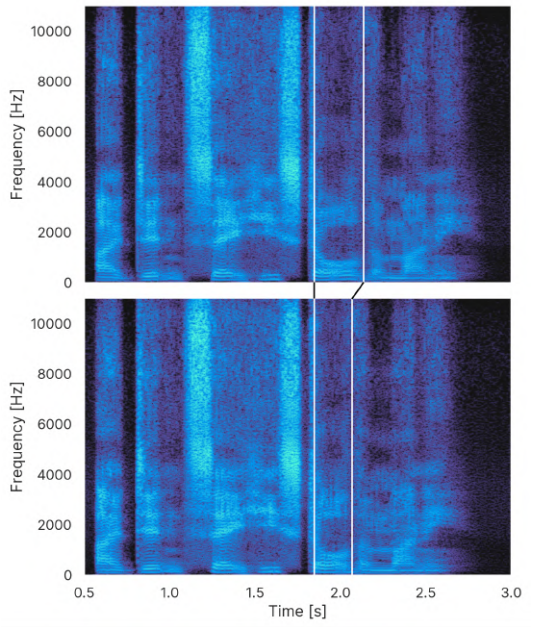


Figure 4.11: Mel spectrograms for the sentence “I didn’t say he stole the money”. The top image shows the resulting spectrogram when using the predictions from the AR prior. The bottom shows an edited version, where the fine-grained prosody embedding corresponding to the “o” in “stole” was replaced by a different prosody embedding, causing a the phoneme duration to decrease.

process more intuitive, the number of fine-grained embeddings could be reduced; the effects of the prosody embeddings could then be assessed more easily. The downside of such a process is, that it could lead to reduced naturalness, but this is less of a problem if the effects of the fine-grained prosody embeddings are only small.

4.5 Conclusion

Multiple experiments were carried out, with as main objective to gauge the controllability of the proposed ConEx model. Two datasets were used: a proprietary dataset (Will) with distinct speaking styles, and a dataset with expressive speech gathered from audiobooks (Blizzard).

An initial experiment used a simplified architecture with a global prosody encoder based on a vector quantized variational autoencoder. After training the model on the Will dataset, the experiments showed that the discrete embeddings learned by the VQ-VAE can represent the different speaking styles in the dataset. Furthermore, the embeddings can be used for performing global style transfer. This way, the global speaking style of a generated speech utterance can be controlled.

The global style transfer experiment for the ConEx architecture showed that style transfer could be performed using the global prosody embeddings learned by the model. When the model was trained on the Will dataset, the quantitative and qualitative analyses showed that the distinct styles in the dataset could be transferred successfully. When

using the proposed architecture with Blizzard dataset, global style transfer was less successful. First, the dataset does not contain very distinct speaking styles, making choosing styles harder. Next, only some styles were recognizable in the output speech. The model seemed to fail to transfer the rhythm characteristics of the reference sample to the generated speech. However, the objective metrics still indicated that other aspects of the generated speech were closer to the reference sample than a baseline.

The ConEx model uses an autoregressive prior to predict fine-grained prosody embeddings at inference time. When comparing the predicted fine-grained prosody embeddings with randomly generated prosody embeddings, the experiments showed that predicted embeddings lead to more natural generated speech. The AR prior is thus a useful addition to the model. Random sampling especially lead to very unnatural speech for the model trained on the Blizzard dataset. The effect of random sampling was significantly smaller for the model trained on the Will dataset. The largest variations were noticeable for the *normal* and *happy* speaking styles, but even then, the changes were only limited to the phoneme durations. This indicates that there is less fine-grained variation in this dataset.

To evaluate the proposed technique to make local prosody edits, multiple speech utterances were created and edited. The results on the Blizzard dataset showed that the method could indeed change the prosody to emphasize words, change the intonation, etc. However, editing the prosody of a phoneme sometimes also changed the prosody of other phonemes, i.e., the results of the edits were not local. This makes the process of editing prosody less predictable and more complicated. Capturing and editing prosody at a higher level (syllable or word-level), could make the method more reliable and intuitive.

Furthermore, the AR prior did not always suggest options that led to diverse prosody. Instead, some options had the same effect, or even no effect. This issue was largest when using the model trained on the Will dataset. Using that dataset, the prosody seemed to be locked to the global prosody style, which made editing local prosody impossible in many cases. Using a dataset with enough fine-grained prosodic diversity is thus important when using the proposed technique. Furthermore, a mechanism could be introduced to rank diverse options higher. However, there is always a trade-off between naturalness and diversity.

Chapter 5

Conclusion

5.1 Overview of this thesis

In this thesis, ConEx, a novel model for controllable expressive speech synthesis, was proposed. ConEx brings control over global speaking style and over local prosody to a neural-based text-to-speech system. In particular, global style transfer can be used to generate speech with different speaking styles, and a novel method was proposed to enable editing prosody at the phoneme level.

ConEx builds on the architecture of FastSpeech, a non-autoregressive text-to-speech model. ConEx adds a prosody encoder, which models prosody at both global and phoneme level. This prosody encoder consists of two parts: a reference encoder is used to extract global prosody embeddings, and a vector quantized variational autoencoder is used to learn discrete fine-grained prosody embeddings.

At inference time, a global prosody embedding from a reference speech sample can be used to control the speaking style of the output speech. An autoregressive prior model is used to predict the fine-grained prosody embeddings based on the phoneme embeddings and the global prosody embedding. To make a local edit to the prosody of the output speech, a fine-grained prosody embedding can be replaced by another embedding with a high probability of fitting, as predicted by the AR prior.

The ConEx architecture was implemented using PyTorch as an extension to ESPnet, an open-source speech processing toolkit. Furthermore, a user interface was developed, to allow users to select a global prosody embedding from a t-SNE plot, and make local prosody edits by the proposed technique.

Multiple experiments were carried out, with as main objective to gauge the controllability of the proposed ConEx model. Two datasets were used for the experiments: the proprietary Will dataset, which contains speech samples from a professional voice actor in six distinct speaking styles, and the Blizzard dataset, which contains expressive speech from readings of audio books performed by Catherine Byers.

The experiments showed that model succeeds at providing control over both global and local prosody. However, the level of control depends strongly on the dataset used. In

particular, the Will dataset, with its distinct styles, proved to be great for controlling the global speaking style, but editing prosody locally was often impossible. Whereas when ConEx was trained on the Blizzard dataset, local edits could be made to the prosody of the synthesized speech. This is due to the dataset containing more fine-grained differences in prosody. Control over the global speaking style was also possible, but limited, since the global speaking styles in the Blizzard dataset are not diverse.

Experiments also showed some limitations to the proposed local editing technique. First, edits did not always have local results, which makes the editing process more cumbersome. Related to that, when trying to change a certain phoneme, multiple edits are sometimes needed. Lastly, when there is little diversity in the fine-grained prosody embeddings, the auto-regressive prior top-k predictions tend to have the same effect, which makes editing the local prosody impossible using the proposed technique.

5.2 Future work

In order to overcome some of the limitations found in relation to the use of ConEx, and to further improve the model, several improvements could be analyzed in future work:

- Deep learning models, such as ConEx, rely on large amounts of data. In speech synthesis research, there are only a few datasets with expressive speech. A new dataset, which includes both large differences in global speaking style and varied fine-grained prosody, could improve the controllability over the global speaking style and local prosody.
- The ConEx decoder could further be conditioned on a speaker embedding, such that the model can generate speech in different voices. In order for this to succeed, the dataset should include many voices, each with enough diversity in speaking style and fine-grained prosody.
- The proposed local editing technique could be improved by learning and manipulating fine-grained embeddings at a higher level, such as syllable or word level. Furthermore, a different ranking scheme for the top-k options could be used to offer more diverse prosody embedding options. A parameter could be introduced to control the trade-off between naturalness and diversity.
- Implementing a decomposed vector quantization scheme, as proposed by Kaiser et al. (2018), could make the training of the VQ-VAE more stable and prevent index collapse issues. This would replace the somewhat arbitrary initialization scheme used in this work.
- ConEx uses a simple reference encoder for learning global prosody embeddings. More advanced models have been proposed, which try learn representations with disentangled and interpretable dimensions. For example, a Gaussian mixture VAE could be used, as proposed by Hsu et al. (2018). These disentangled dimensions can then be directly manipulated to control the global speaking style, instead of having to rely on style transfer.

- The experiments in this work focused on testing the controllability of the proposed model. In order to convert the mel spectrogram outputs the model generated into waveforms, a vocoder was used that was trained on a different dataset. As such, the generated audio samples distorted the voice of the speakers and led to unnatural audio. By training a neural-based vocoder on the dataset used, more natural results can be obtained. Furthermore, using modern vocoder architectures such as MelGAN (Kumar et al., 2019) can speed up the vocoding many times, leading to almost real-time speech synthesis.

- In order to carry out the experiments, a Jupyter notebook was implemented which allowed selecting a global prosody embedding from a t-SNE visualization, and changing fine-grained prosody embeddings by selecting one of the top-3 predictions. This proof-of-concept could be expanded into a real user interface for controlling expressive speech synthesis. User experience research could offer insights into what such a user interface should look like to make controlling the global speaking style and local prosody as intuitive and efficient as possible.

Appendices

Appendix A

Model hyperparameters

Name	Value
Phoneme embedding dimension	128
Encoder layers	4
Encoder conv1D kernel size	3
Encoder conv1D channels	1536
Encoder attention heads	2
Encoder dropout	0.2
Decoder layers	4
Decoder conv1D kernel size	3
Decoder conv1D channels	1536
Decoder attention heads	2
Decoder dropout	0.2
Duration predictor conv1D layers	2
Duration predictor conv1D kernel size	3
Duration predictor conv1D channels	128
Duration predictor dropout	0.2
Postnet layers	5
Postnet conv1d channels	256
Postnet conv1d filter size	5
Postnet dropout	0.5
Reference encoder conv2D layers	2
Reference encoder conv2D kernel size	3
Reference encoder conv2D channels	32
Reference encoder conv2D stride	2
Reference encoder GRU units	32
Number of discrete FG prosody embeddings	32
Discrete FG prosody embedding dimension	3
VQ-VAE beta	0.05
FG alignment dropout	0.2

Table A.1: ConEx hyperparameters

Bibliography

C. Kuang and R. Fabricant. *User Friendly: How the Hidden Rules of Design are Changing the Way We Live, Work & Play*. Ebury Publishing, 2019. ISBN 9780753551530. URL https://books.google.be/books?id=qM-QDAAAQBAJ.

João Medeiros. How Intel gave Stephen Hawking a voice. *Wired*, jan 2015.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. FastSpeech: Fast, robust and controllable text to speech. In *Advances in Neural Information Processing Systems*, volume 32, pages 3171–3180. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/f63f65b503e22cb970527f23c9ad7db1-Paper.pdf.

RJ Skerry-Ryan, Eric Battenberg, Ying Xiao, Yuxuan Wang, Daisy Stanton, Joel Shor, Ron Weiss, Rob Clark, and Rif A. Saurous. Towards end-to-end prosody transfer for expressive speech synthesis with Tacotron. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4693–4702, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/skerry-ryan18a.html.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 6306–6315. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf.

G. Sun, Y. Zhang, R. J. Weiss, Y. Cao, H. Zen, A. Rosenberg, B. Ramabhadran, and Y. Wu. Generating diverse and natural text-to-speech samples using a quantized fine-grained VAE and autoregressive prosody prior. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6699–6703, 2020.

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

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009. ISBN 0131873210.

Tim Capes, Paul Coles, Alistair Conkie, Ladan Golipour, Abie Hadjitarkhani, Qiong Hu, Nancy Huddleston, Melvyn Hunt, Jiangchuan Li, Matthias Neeracher, Kishore Prahallad, Tuomo Raitio, Ramya Rasipuram, Greg Townsend, Becci Williamson, David Winarsky, Zhizheng Wu, and Hepeng Zhang. Siri on-device deep learning-guided unit selection text-to-speech system. In *Proc. Interspeech 2017*, pages 4011–4015, 2017. doi: 10.21437/Interspeech.2017-1798. URL http://dx.doi.org/10.21437/Interspeech.2017-1798.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio, 2016.

Aäron van den Oord and Tom Walters. WaveNet launches in the google assistant. https://deepmind.com/blog/article/wavenet-launches-google-assistant, October 2017. Accessed: 2021-5-1.

Yuxuan Wang, R.J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. In *Proc. Interspeech 2017*, pages 4006–4010, 2017. URL http://dx.doi.org/10.21437/Interspeech.2017-1452.

Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ-Skerrv Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 4779–4783. IEEE, 2018. doi: 10.1109/ICASSP.2018.8461368. URL https://doi.org/10.1109/ICASSP.2018.8461368.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. FastSpeech 2: Fast and high-quality end-to-end text to speech, 2020.

Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal Forced Aligner: Trainable text-speech alignment using kaldi. In *Proc. Interspeech 2017*, pages 498–502, 2017. doi: 10.21437/Interspeech.2017-1386. URL http://dx.doi.org/10.21437/Interspeech.2017-1386.

Kim Silverman, Mary Beckman, John Pitrelli, Mari Ostendorf, Colin Wightman, Patti Price, Janet Pierrehumbert, and Julia Hirschberg. ToBI: A standard for labeling english prosody. 01 1992.

Yuxuan Wang, Daisy Stanton, Yu Zhang, R. J. Skerry-Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Fei Ren, Ye Jia, and Rif A. Saurous. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. *CoRR*, abs/1803.09017, 2018. URL http://arxiv.org/abs/1803.09017.

Ya-Jie Zhang, Shifeng Pan, Lei He, and Zhen-Hua Ling. Learning latent representations for style control and transfer in end-to-end speech synthesis. *CoRR*, abs/1812.04342, 2018. URL http://arxiv.org/abs/1812.04342.

Wei-Ning Hsu, Yu Zhang, Ron J. Weiss, Heiga Zen, Yonghui Wu, Yuxuan Wang, Yuan Cao, Ye Jia, Zhifeng Chen, Jonathan Shen, Patrick Nguyen, and Ruoming Pang. Hierarchical generative modeling for controllable speech synthesis. *CoRR*, abs/1810.07217, 2018. URL http://arxiv.org/abs/1810.07217.

Isaac Elias, Heiga Zen, Jonathan Shen, Yu Zhang, Ye Jia, Ron J. Weiss, and Yonghui Wu. Parallel tacotron: Non-autoregressive and controllable TTS. *CoRR*, abs/2010.11439, 2020. URL https://arxiv.org/abs/2010.11439.

Younggun Lee and Taesu Kim. Robust and fine-grained prosody control of end-to-end speech synthesis. *CoRR*, abs/1811.02122, 2018. URL http://arxiv.org/abs/1811.02122.

Guangyan Zhang, Ying Qin, and Tan Lee. Learning Syllable-Level Discrete Prosodic Representation for Expressive Speech Generation. In *Proc. Interspeech 2020*, pages 3426–3430, 2020. doi: 10.21437/Interspeech.2020-2228. URL http://dx.doi.org/10.21437/Interspeech.2020-2228.

Guangzhi Sun, Heiga Zen, Ron J. Weiss, Yonghui Wu, Yu Zhang, and Yuan Cao. Fully-hierarchical fine-grained prosody modeling for interpretable speech synthesis. In *ICASSP*, 2020.

Chung-Ming Chien and Hung-Yi Lee. Hierarchical prosody modeling for non-autoregressive speech synthesis, 2021.

Tuomo Raitio, Ramya Rasipuram, and Dan Castellani. Controllable neural text-to-speech synthesis using intuitive prosodic features. *Interspeech 2020*, Oct 2020. doi: 10.21437/interspeech.2020-2861. URL http://dx.doi.org/10.21437/interspeech.2020-2861.

Adrian Łańcucki. FastPitch: Parallel text-to-speech with pitch prediction, 2021.

Yuxuan Wang and RJ Skerry-Ryan. Expressive speech synthesis with tacotron. https://ai.googleblog.com/2018/03/expressive-speech-synthesis-with.html, March 2018. Accessed: 2021-5-5.

Diederik P. Kingma and M. Welling. Auto-encoding variational Bayes. *CoRR*, abs/1312.6114, 2014.

Gustav Eje Henter, Jaime Lorenzo-Trueba, Xin Wang, and Junichi Yamagishi. Deep encoder-decoder models for unsupervised learning of controllable speech synthesis, 2018.

Jennifer Williams and Simon King. Disentangling Style Factors from Speaker Representations. In *Proc. Interspeech 2019*, pages 3945–3949, 2019. doi: 10.21437/Interspeech.2019-1769. URL http://dx.doi.org/10.21437/Interspeech.2019-1769.

Jaan Altosaar. *Tutorial - What is a Variational Autoencoder?*, August 2016. URL https://doi.org/10.5281/zenodo.4462916.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Tomoki Hayashi, Ryuichi Yamamoto, Katsuki Inoue, Takenori Yoshimura, Shinji Watanabe, Tomoki Toda, Kazuya Takeda, Yu Zhang, and Xu Tan. Espnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7654–7658. IEEE, 2020.

A.K Subramanian. PyTorch-VAE. https://github.com/AntixK/PyTorch-VAE, 2020.

Simon King and Vasilis Karaiskos. The Blizzard Challenge 2013. In *Proceedings Blizzard Workshop 2013*, September 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

Lukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables. *CoRR*, abs/1803.03382, 2018. URL http://arxiv.org/abs/1803.03382.

Noé Tits. *Controlling the Emotional Expressiveness of Synthetic Speech - a Deep Learning Approach*. PhD thesis, 12 2020.

R. Kubichek. Mel-cepstral distance measure for objective speech quality assessment. In *Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, volume 1, pages 125–128 vol.1, 1993. doi: 10.1109/PACRIM.1993.407206.

Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203, 2020. doi: 10.1109/ICASSP40776.2020.9053795.

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

Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, and Aaron Courville. Melgan: Generative adversarial networks for conditional waveform synthesis, 2019.