Karlsruhe Institute of Technology

Fine-Grained Prosody Control in Neural TTS Systems

Bachelor's Thesis of

Moritz Behr

at the Department of Informatics
Institute for Anthropomatics and Robotics (IAR),
Interactive Systems Labs (ISL)

Reviewer:	Prof. Dr. Alex Waibel
Second reviewer:	Prof. Dr. Tamim Asfour
Advisor:	M.Sc. Stefan Constantin

11. July 2021 – 10. November 2021

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

. . . . . .

(Moritz Behr)

Abstract

Text-to-speech (TTS) systems are becoming increasingly important in recent years as various computer systems with speech interfaces are being integrated into everyday life. These systems often need to participate in dialogues with human users and convey important information via generated speech. As prosody is an important part of conveying information and emotion in spoken language, controlling the prosody of the generated speech can greatly improve the capabilities of these systems to take part in natural dialogues or enable them to more effectively convey emotional content, for example when displaying agitation during warnings of danger. In this thesis, a TTS model based on Fast Speech 2 [Ren+20] is proposed which allows for fine-grained prosody control by parameterizing pitch, volume and speech rate on word-level. Many TTS models are evaluated using only English Data, which leads to evaluation results with limited significance for other languages. To alleviate this, the proposed model is trained once with English data and once using German data. Subsequently, speech quality as well as fine-grained prosody control capabilities are evaluated in a survey. The evaluation shows that both models are capable of generating high-quality speech and that the fine-grained prosody control can be used to add emphases to generated speech with no reduction in comprehensibility and only slight a decrease in speech naturalness. The prosody parameters can be entered using the Speech Synthesis Markup Language.

Zusammenfassung

In den vergangenen Jahren gewannen Systeme für Text-to-Speech (TTS) vermehrt an Bedeutung, auch weil verschiedene Computersysteme mit Sprachinterface immer stärker in das Alltagsleben integriert werden. Diese Systeme müssen in der Lage sein, Dialoge mit Menschen zu führen und wichtige Informationen über generierte Sprache zu vermitteln. Da die Prosodie einer Äußerung einen großen Teil der vermittelten Emotion und Information enthält, ist die Fähigkeit, die Prosodie der generierten Sprache zu kontrollieren, von großer Wichtigkeit. Kontrolle über die Prosodie kann die Fähigkeit des Systems, an natürlichen Dialogen teilzunehmen, stark ausbauen und es ermöglichen, emotionale Informationen zu vermitteln. So kann die Prosodie beispielsweise genutzt werden, um Aufregung zu signalisieren, wenn das System vor Gefahren warnt. In dieser Arbeit wird, basierend auf Fast Speech 2 [Ren+20], ein TTS-Modell vorgestellt, das wortgenaue Prosodiekontrolle erlaubt, indem Tonhöhe, Lautstärke und Sprechgeschwindigkeit auf Wort-Level parametrisiert werden. Viele TTS-Modelle werden nur auf englischen Daten evaluiert, was zu eingeschränkter Aussagekraft für andere Sprachen führt. Um dies zu vermeiden, wird das Modell sowohl mit deutschen als auch mit englischen Daten trainiert. Anschließend werden die Sprachqualität und die wortgenaue Prosodiekontrolle mittels einer Umfrage evaluiert. Die Evaluationsergebnisse zeigen, dass beide Modelle in der Lage sind, Sprache von hoher Qualität zu generieren und dass die wortgenaue Prosodiekontrolle genutzt werden kann, um der generierten Sprache klar erkennbare Betonungen hinzuzufügen. Das Hinzufügen der Betonungen führt zu keiner Reduzierung der Verständlichkeit und die Natürlichkeit der generierten Sprache wird nur geringfügig verringert. Die Speech Synthesis Markup Language kann benutzt werden, um die Parameter zur Prosodiekontrolle einzugeben.

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

The field of natural language processing (NLP) is one of the most important areas of research for the ongoing undertaking of creating computer systems that are able to communicate with humans via natural language. These systems are increasingly part of our lives in the form of robots, smart-house systems and digital personal assistants, to only name a few. Increasingly, the form of communication with these systems switches from written language to spoken language, as it is more efficient and can be more easily integrated into everyday activities. We can ask our smartphone about the weather forecast for next week, ask our car which song is currently playing on the radio or tell our smart home device to close the living room shutters. Therefore, Text-to-speech (TTS) systems are one of the subfields of natural language processing which are of increasing importance as these intelligent systems now need to be able to generate comprehensive and natural speech. TTS systems are complex to construct, but much progress has been made during the last few years, a big step being the move from traditional TTS methods to deep learning-based TTS methods.

Since then, many successful neural TTS models were developed that are able to generate very natural and comprehensible speech. However, a problem of these systems is that they tend to generate speech with rather flat prosody as they learn the average prosody of the speech data they were trained on. Even when ignoring the sometimes flat prosody, explicitly controlling the prosody of the generated speech would be an important step towards being able to participate in a natural dialogue with a human, as prosody plays a big part in dialogues, conveying meaning as well as emotion. More precisely, prosody control would be very useful for error correction in dialogue systems, as the computer would be able to ask natural sounding questions when encountering ambiguities it cannot solve without additional input from the human. It would also prove useful when the human attention has to be drawn to urgent matters or sources of danger, as prosody control could be used to convey the appropriate urgency. In these situations, fast inference times are also of importance, as the usefulness of dialogue systems dwindles when generating the next utterance takes too long.

1.2. Goal of this Work

This work aims at creating a deep learning-based TTS system with the ability to control prosody on word-level and capable of being trained on data of various languages. Additionally, the model should have low inference times, so that it may be used in environments in which only limited processing power is available, without losing the ability to be used in a fluent dialogue.

The created TTS system will be trained and evaluated by conducting a survey on the comprehensibility and naturalness of the generated speech and the perceptibility of emphases which were explicitly added to the generated speech samples using fine-grained prosody control parameters. Additionally, the model will be trained for a language other than English and evaluated in the same manner so as to discern if the model is also able to utilize its prosody control capabilities when trained for other languages. The model will be based on a modified version of the Fast Speech 2 [Ren+20] TTS system, as Fast Speech 2 fulfills the prerequisite of fast inference times as it is non-autoregressive and also already provides prosody control on utterance-level.

1.3. Structure

Chapter 2 will be used to quickly convey the prerequisites needed to understand the rest of the work, and is split up into two main sections. The first of which, section 2.1, covers the needed basics of deep learning, ranging from basic perceptrons and multilayer perceptrons to more complicated architectures like convolutional neural networks. After the deep learning basics, TTS basics are covered by section 2.2, starting at the traditional methods like concatenative systems and then discussing the modern deep learning-based methods in more detail and presenting different TTS architectures while also mentioning their respective strengths and weaknesses. After that, chapter 3 presents relevant works that are either used in the proposed model or discuss a similar subject as this thesis. In chapter 4 the proposed model will then be described in much detail, starting with explanations of the original Fast Speech 2 model in section 4.1, dealing with each module of the architecture in a separate subsection. Section 4.2 is then used to go over the training process of a Fast Speech 2 model and in section 4.3 the proposed model is presented by describing all modifications made to the Fast Speech 2 architecture. In chapter 5, the evaluation of the proposed model is presented. First, the evaluation method is described in detail in section 5.1 and the used datasets and hyperparameters are then discussed, together with the general training process, in section 5.2. The evaluation results are then finally presented in section 5.3 with a subsection for each trained model. Subsequently, the discussion of the evaluation results is done in section 5.4. Lastly, the thesis is concluded in chapter 6, where the goals and respective results of the thesis are summed up in section 6.1 and potential further work is outlined in section 6.2.

2. Fundamentals

This chapter provides a short overview of the concepts and technologies used in this work and is made up of two main parts. The first covers the basics of machine learning with artificial neural networks, the second deals with the fundamentals of text-to-speech systems.

2.1. Artificial Neural Networks

Artificial neural networks (ANNs) are biologically inspired computing systems made up of many simple computing units called nodes or artificial neurons. The nodes of the networks we are concerned with are still based on the perceptron which was popularized by Frank Rosenblatt in the 1950s [Ros57]. An overview of the perceptron architecture is shown in Figure 2.1.

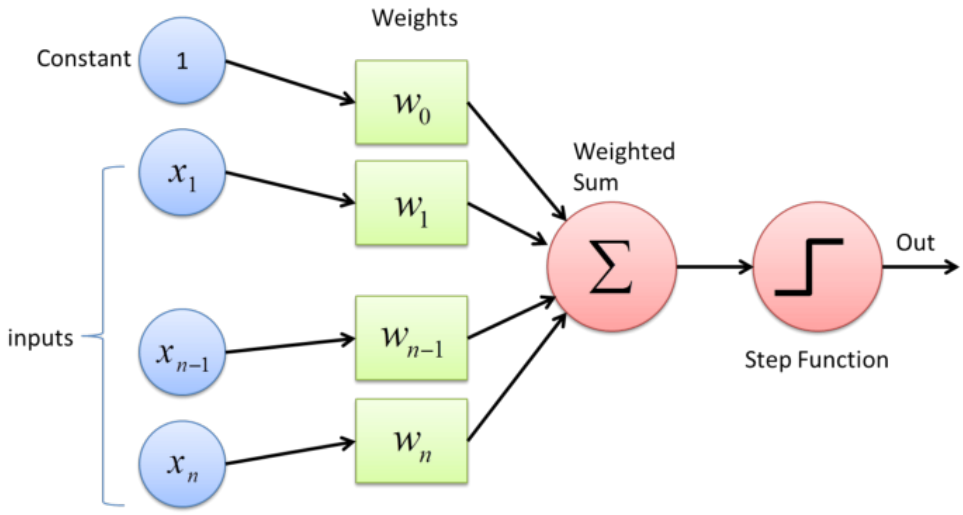


Figure 2.1.: A Perceptron [Sha17]

A node of a modern ANN usually has multiple inputs and a single output value. The output of the node is specified by the application of a chosen activation function to the weighted sum of all inputs. These weights are the parameters by which the output of the node can be controlled. However, the weights are not set by hand but are set during the training process, which tries to estimate the optimal weights to make the output of the node estimate a goal function.

The output is often interpreted as a binary value by assigning all negative outputs to one class and all positive outputs to the other class. As seen in Figure 2.1, one of the inputs of the node is special as it is a constant value of one. This input is called the *bias* and it is needed to shift the node's decision boundary away from the origin. Without it, the node would not be able to change its output for an input of only zeros. But still, just as the original perceptron, a single node of a modern ANN is severely limited as it can only linearly separate the input data into two classes which makes it impossible for it to learn most functions and even basic ones like the XOR function [MP69].

2.1.1. Multilayer Perceptron

To overcome the limitations of a single artificial neuron, multiple nodes can be connected to form networks, the simplest form of which are multilayer perceptrons (MLPs). MLPs are made up of multiple layers, each consisting of many nodes. Since MLPs are fully connected, each node in layer n is connected to every node in layer $n+1$ as shown in Figure 2.2.

Just as with every input of a single perceptron, each of those connections has an associated weight. The weights of all connections between two layers can be represented as a matrix of real numbers and the input, output and intermediate activations can be represented as vectors of real numbers to make mathematical notation more comfortable.

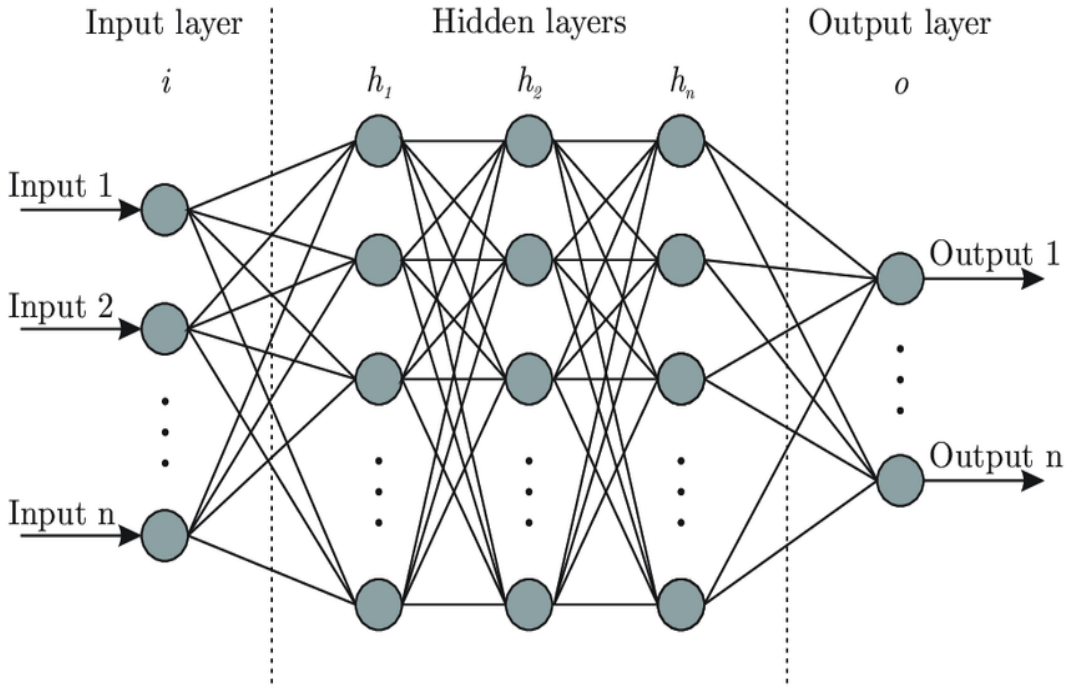


Figure 2.2.: The structure of a multilayer perceptron [Shu19]

Given an input $x \in \mathbb{R}^n$, a matrix of weights $W \in \mathbb{R}^{nxm}$ and an activation function $f(x) : \mathbb{R} \to \mathbb{R}$, the activation of a single layer can be described as $A = F(W * x)$ with F being a function that applies the activation function f to each number in a given vector. Computing the output of an ANN is called a *forward pass* because the input values are passed through the network layer by layer. The activations of the nodes in the last layer are the output of the network.

2.1.2. Backpropagation

Neural networks, like MLPs, of sufficient size are universal function approximators [Cyb89]. However, to make a given ANN estimate a goal function, the weights of the ANN need to be set to appropriate values. The process of finding these values is called *training*. A widely used technique to train ANNs is the backpropagation algorithm [RHW85; RHW86].

Backpropagation can find suitable weights for an ANN in an iterative process by estimating the deviation of the network's output from the goal function's output using a loss function. Then, the gradients of the loss function with respect to the weights of the network are computed in an efficient way. Lastly, the weights can be adjusted in the direction opposite of the gradient to reduce the deviation.

A single iteration of the backpropagation algorithm starts by computing a forward pass through the network for the given input data. The activation values of each layer are stored for later use. Then, the error is computed by applying the loss function to the output of the network and the desired output for the current input as given by the training data. Now the derivative of the loss function with respect to the activation of the output layer can be computed. Assuming a network with i layers, this is written as:

$$\frac{\partial L}{\partial A_i}$$

with L being the loss function and A_i the activation of the i-th layer, which is also the output of the network. This is why we made sure to store the activation values of each layer during the forward pass. To make the computation of gradients for layers further from the output tractable, the algorithm makes use of the chain rule:

$$(f \circ g)' = (f' \circ g) \cdot g'$$

This means that we can use the already calculated gradients for computing the gradients of the next layer. In this case *next* layer means the layer which is one step further from the output, as the algorithm computes the gradients for each layer starting from the output of the network, ending at the input. This is why this step is also called *backward pass*.

For example, as we have already computed the gradients of the loss function with respect to the output activation values, we can now calculate the gradients for the weights of the connections between the last and second to last layer. Applying the chain rule gives us:

$$\frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial A_i} \cdot \frac{\partial A_i}{\partial W_i}$$

This means that we can use the already calculated gradients for A_i and just need to compute one more simple derivative to get the gradients for the weights W_i. These gradients can now be used to change the values in W_i in a way to decrease the error. This is done by changing the weights in the opposite direction of the gradient:

$$W_i \leftarrow W_i - \alpha \frac{\partial L}{\partial W_i}$$

with α being the *learning rate* parameter, which controls the extent of the change. This process can now be continued until the gradients for all weights have been computed and the weights have been changed accordingly. Now the next iteration of the algorithm can start by performing a forward pass for further input data. Performing the algorithm for every datum in the training data is called one *epoch*. The training process can be continued for a set number of epochs or, alternatively, the output of the loss function can be monitored until it reaches acceptable values.

2.1.3. Convolutional Neural Networks

When dealing with more complicated inputs like images or long sequences, the usefulness of standard MLPs is limited. Recurrent neural networks (RNNs) or convolutional neural networks (CNNs) can be used in these cases to construct more effective models for the problem at hand. As TTS systems are dealing with complicated sequences like texts, phoneme sequences, spectrograms and raw waveform data, many modern neural TTS models and all TTS models mentioned in this work use at least one of these types of networks. But since the main model we are concerned with only makes use of CNNs, we will focus on them.

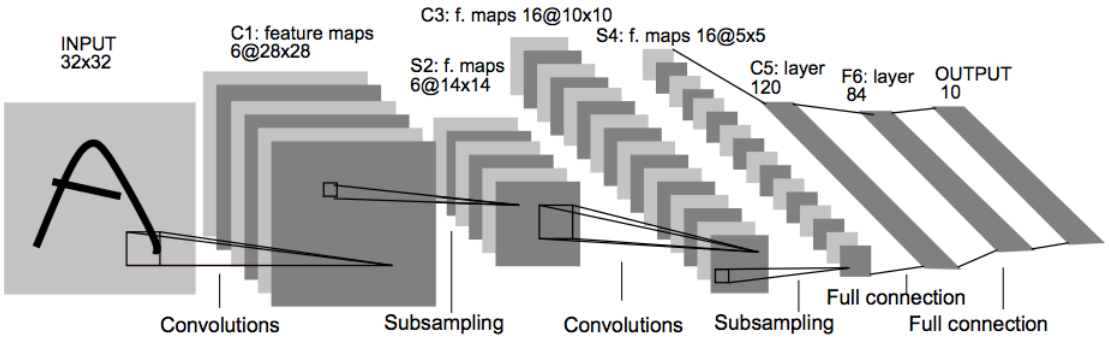


Figure 2.3.: Overall structure of a typical CNN used for image recognition [LeC+98]

CNNs were originally introduced as time delay neural networks by Alex Waibel in 1987 [Wai87] and published in 1989 [Wai+89]. Their architecture generally consists of three types of layers: convolutional layers, subsampling layers and fully connected layers.

In the first part of a CNN, convolutional and subsampling layers are used in successive pairs. In the second part, fully connected layers like in a MLP are used to compute the final output of the CNN. The general structure of a CNN can be seen in Figure 2.3. Note the pairs of convolutional and subsampling layers in the first part of the network and the fully connected layers at the output.

In the convolutional layers, the weights are grouped in *kernels*, which are tensors of the same dimensionality as the input data of that layer. While having the same amount of dimensions, the kernels are usually much smaller in size compared to the input data. Each kernel is used to create a so called *feature map* by carrying out a discrete convolution of the input and the kernel. During this process the kernel is moved across the input in steps of constant length. The length of these steps is also called *stride*. At each step the sum of the element-wise multiplication of the kernel and the input at the current location is calculated and stored in the feature map. See Figure 2.4 for an example of how a feature map is calculated with kernel size two by two and a stride of one. Together, the feature maps of all kernels of a convolutional layer form the output of that layer.

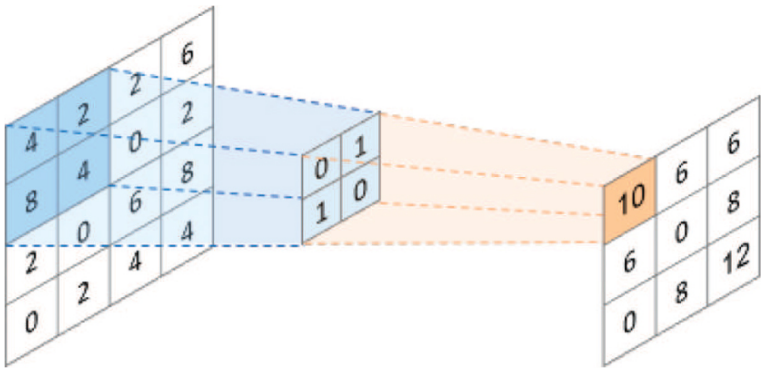


Figure 2.4.: A feature map is computed in a convolutional layer [TCW20]

In most CNNs, subsampling layers are then used to reduce the dimensions of the preceding convolutional layer's output. A common type of subsampling layer are local pooling layers. They work by combining local clusters of the input, for example a two by two pixel area of an image, into a single value. This can be done in various ways, such as computing the maximum or the mean of the local cluster. An example of a max pooling layer is shown in Figure 2.5. Many variables can be used to change the behaviour of the pooling layer. These include the size of the local cluster, the usage of padding at the edges of the input and the stride size.

After several pairs of convolutional and subsampling layers, the n-dimensional output of the last subsampling layer is flattened and fed into the first fully connected layer of the CNN. These layers are architecturally identical to the already described MLPs. The output of the last fully connected layer is also the output of the whole CNN.

The training process of a CNN is very similar to that of a MLP, since the backpropagation algorithm can be applied to all types of layers as long as the derivative of their application to input data can be computed which is the case for both convolutional and subsampling layers.

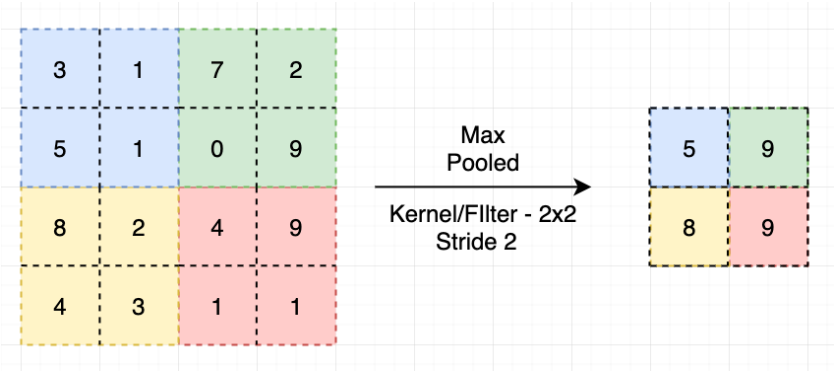


Figure 2.5.: An example of max pooling as used in subsampling layers [Ran20]

A distinctive feature of CNNs is their *shift invariance*. This means that a CNN which was successfully trained to detect a pattern in its input will still be able to detect that same pattern when its location is shifted within the input. This is because the weights in the kernels of the convolutional layers remain constant while the kernel is moved through the input. The subsampling layers are used to gradually reduce the dimensions of the feature maps while the convolutional layers detect higher level features in the feature maps until they are small enough to be efficiently processed by a fully connected network, which then can detect non-linear combinations of these high-level features.

Even though CNNs are most prominently used in computer vision tasks, their application area is not limited to them. They can also be applied to one dimensional inputs like plain text, which is the input for most TTS systems. They are also more suited for parallel computation than RNNs, since RNNs generally need the output of the last time step to compute the next one. CNNs are therefore often used in TTS systems, as they make achieving real-time inference possible even when using large networks with many millions of connections. Real-time meaning the duration of the resulting waveform.

2.2. Text-to-Speech Systems

As the name suggests, TTS systems produce audio of human speech from text inputs. The desired output is a natural sounding recitement of the given text. Until recently, there has been no application of deep learning in this area and classic methods like concatenation-based techniques or source-filter systems have been used to synthesize speech from text. This changed in 2016 when DeepMind showed that neural networks were capable of generating raw waveforms of complicated audio like speech when proposing their WaveNet model [Oor+16].

2.2.1. Traditional TTS Methods

Concatenative speech synthesis and source-filter systems are two traditional TTS approaches that have been used since the 1970s [Gro74; RRT05]. Concatenation-based systems, also called unit selection systems, work by combining (concatenating) fragments of pre-recorded speech in a way that produces the desired output. If working on a sufficiently sized database of high-quality recordings, these systems generally produce very natural-sounding speech as little digital signal processing (DSP) is involved. However, there can occur audible glitches at the points of concatenation. Another downside are the big, handcrafted databases from which recordings can then be fetched and concatenated. These databases are needed at runtime and are often gigabytes in size and contain many hours of recorded speech, an example being the CMU Arctic speech database for unit selection systems [KB04].

An important distinction between different unit selection systems is the resolution of the basic recordings that are being concatenated. These range from the basic building blocks of spoken language like phones or diphones to whole sentences and phrases, with the middle ground being syllables. While TTS systems that are being built for narrow purposes like transit schedule announcements can be built using a database of recorded phrases, this would not be practical for general-purpose TTS systems, as either the database would need to be unrealistically large or the system would be too limited by the few phrases stored in the database. On the other hand, when the resolution is very high, for example on diphone-level, the amount of sonic glitches typically increases as many more concatenations are necessary. This can, however, be overcome with DSP techniques, as diphone-based concatenative TTS systems like MBROLA [Dut+96] show. Generally, the choice of unit resolution is specific to the area of application and language and therefore varies even among general-purpose concatenative TTS systems [HB96].

Another important traditional approach to building TTS systems are source-filter systems. They are based on the observation that the sound of human speech, described in the terms of acoustic engineering, can be adequately specified by the describing the parameters of the source and filters that form the resulting sound waves. In the case of human speech, the source are the human vocal cords and the filters being the current state of the vocal tract.

Source-Filter systems do not use recordings of human speech during the synthesis process but synthesize sounds based on handcrafted acoustic models. These acoustic models can be based on a set of rules which determine how the sound source, which models the glottal pulse of the human vocal cords, is filtered in such a way as to model the formant resonances of the vocal tract. This approach is called formant synthesis [Smi10]. The acoustic model in articulatory synthesis on the other hand is based on the geometry of the vocal tract and its changes during articulation. From the geometry and the position of the source within it, the resulting filters can be derived [Bir14].

2.2.2. Deep Learning-Based TTS Methods

Even though both concatenative and source-filter systems are capable of synthesizing highly intelligible speech, concatenative systems tend to sound emotionless and source-filter systems mostly generate robotic-sounding speech [Sax17]. Additionally, both types of systems are complex to construct since large and well-maintained databases are needed for concatenative systems and the design of the filter rules for source-filter systems proved to be complicated, requiring extensive knowledge in acoustical engineering. Another disadvantage of traditional systems is that their attributes like speaker or spoken language cannot be easily changed.

Most of these problems can be easily solved by using deep learning models to synthesize speech directly from text. These are easier to built as no huge database is needed during runtime and small imperfections in the training data don't have a huge negative impact on the final synthesis result. Also, no expert knowledge in acoustics is required to design any rules or acoustical models. Deep learning models can also be easily adapted to new or enhanced data, as the information in an already trained model can be leveraged when retraining it on new or additional data. This means that, for example, when building a TTS system for a speaker or language for which little data is available, pretrained systems can be utilized in an approach called *transfer learning* [Tu+19]. One can even utilize sentence representations learned by models in other areas of NLP during training of TTS model. [Hay+19].

It still took until 2016 that researchers were able to create a generative deep learning model of high-quality raw audio waveforms. The first of these models was DeepMind's WaveNet [Oor+16] which was able to generate realistic human speech from linguistic features like *mel spectrograms*. Mel spectrograms are modified spectrograms of audio signals acquired by applying short-time Fourier transforms, with their frequency axis converted to a logarithmic mel scale and it's amplitude dimension, usually represented by color, converted to decibels [Rob20]. The mel scale is a scale for pitch designed in such a way so that pairs of pitches of equal distance on the mel scale sound equally distant to a human listener [SVN37]. E.g. a pitch of 1 000 mels sounds equally distant to a pitch of 2 000 mels as a pitch of 500 mels sounds to a pitch of 1 500 mels. This results in a logarithmic scale when displayed in hertz, as increasingly large intervals are needed to make human listeners perceive equal pitch increments. See Figure 2.6 for an example of a mel spectrogram.

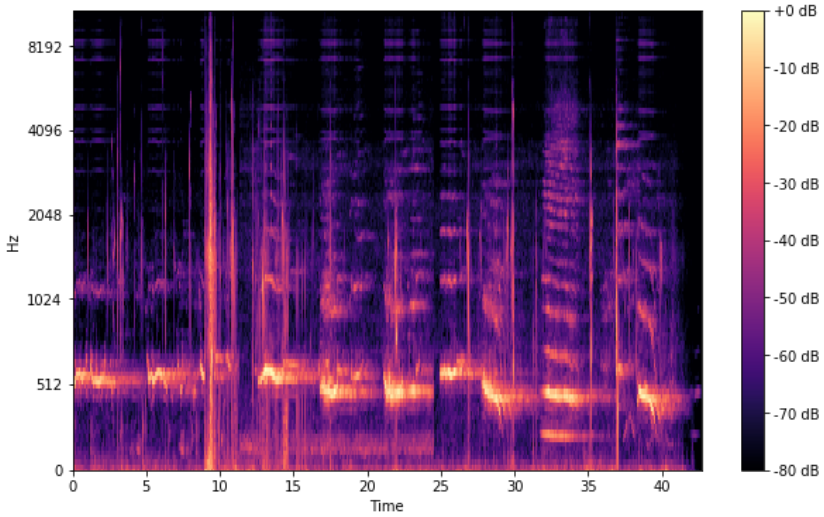


Figure 2.6.: A mel spectrogram [Gar19]

Often, the y-axis is still displayed in a logarithmic hertz scale to make it easier to read, as is the case in the given example.

While some early deep learning-based TTS models, like 2017s Tacotron [Wan+17], work end-to-end, meaning a single model generates audio waveforms directly from text, these systems either suffer from difficult and long training processes or only use deep learning models to generate mel spectrograms and then use hard-coded algorithms, like the Griffin-Lim algorithm [GL84], to reconstruct these spectrograms into audio signals. To solve this, many neural TTS systems emerged that worked by combining two separate deep learning models. The first model generates a mel spectrogram of appropriate human speech from a given text input, the second model then generates raw audio waveforms from that mel spectrogram. The first model in this design is called the TTS model, while the second model, which generates the audio from mel spectrograms, is called the (neural) *vocoder*. One of these early TTS systems was the successor of Tacotron, Tacotron 2 [She+18]. It used a WaveNet conditioned on mel spectrograms as its vocoder model. According to its evaluation, this model achieved speech quality comparable to professionally recorded speech and was easier to train as the vocoder would not have to be retrained when making changes to the TTS like switching speakers or languages.

Figure 2.7 shows the high-level architecture of Tacotron 2. As many sequence-to-sequence models, it is split up into what is called an *encoder-decoder* architecture. This means that the input sequence is first converted into an encoded internal representation by the encoder, shown in blue in the figure, which tries to encapsulate all relevant information of the input in a compact way. The decoder, shown in orange, then predicts the output sequence from that internal representation. The model also has a *attention* feature, shown in gray, which means that it is able to weight different parts of the internal representation differently for each part of the output sequence.

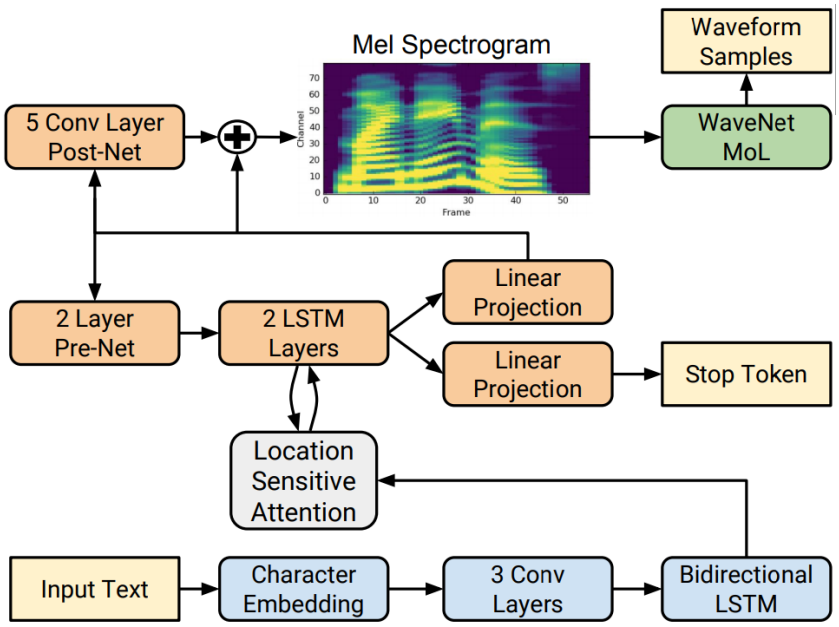


Figure 2.7.: Architectural overview of Tacotron 2, a modern TTS model [She+18]

As seen in the figure, the output of Tacotron 2 is a mel spectrogram that acts as input for the vocoder. The vocoder then converts the mel spectrogram into raw audio waveform. In this case, the vocoder used is WaveNet.

Later TTS models focused on reducing training complexity while further improving audio quality. Another problem of the first deep learning TTS systems were slow inference times as both models, TTS and vocoder, were using *autoregressive* architectures, which makes parallelization difficult as every part of the output sequence depends on all parts that came before it. For example, Tacotron 2 makes use of many RNN-type networks called LSTMs [HS97] which struggle to model the very long dependencies in speech [Li+19] and work autoregressively. Later models [Li+19] tried to alleviate this problem by using the Transformer architecture [Vas+17], which improved training times but still worked in an autoregressive manner. The 2019 TTS model Fast Speech [Ren+19] was the first non-autoregressive TTS model which improved inference times significantly but still suffered from long training times due to its teacher-student architecture which still used autoregressive parts but only during training. Fast Speech 2 [Ren+20], which is the main TTS model used in this work, improved the architecture of Fast Speech by removing the need for a teacher model and therefore eliminating all autoregressive parts from the network. This resulted in faster training, much faster inference and higher audio quality than Tacotron 2 or Transformer TTS. But this came at the cost of some additional work before being able to train the model, as text-audio-alignment information for the training data is needed. For further inquiry, an example of text-audio alignment data is given in Appendix A.

Vocoder models also improved since WaveNet. Non-autoregressive versions of WaveNet like WaveGlow [PVC19] provided considerably better inference times but were huge networks with Waveglow having around 90 million parameters [KKB20]. Still, WaveGlow finally made real-time end-to-end inference possible when combined with a fast TTS model like Tacotron 2.

Other vocoder systems solved the inference time problem by utilizing the GAN architecture [Goo+14] which is a generative architecture originally designed to generate image data but used by models like MelGAN [Kum+19] and HiFi-GAN [KKB20] to generate audio from mel spectrograms. While MelGAN improved the training and inference times of WaveGlow manyfold, it did so with reduced audio quality. HiFi-GAN further improves both while also improving audio quality over WaveGlow [KKB20]. HiFi-GAN is also the vocoder used in this work.

3. Relevant Works

There exist several other works concerned with prosody control in neural TTS systems. Fast Speech 1 [Ren+19] and Fast Speech 2 [Ren+20] can control the speech rate, pitch variance and volume of the output, but the control is not fine-grained and only the prosody of the whole utterance can be controlled. Before that, multiple works, for example by Skerry-Ryan et al. [Ske+18], dealt with prosody transfer, which is the task of transferring the prosodic parameters of one utterance to another one. In layman terms, this is a "try say it like that" task. Wang et al. [Wan+18] then designed a model to learn a clustered latent space of style-embeddings, so-called global style tokens, from which a prosody style can be chosen during inference. There have also been systems that allowed fine-grained prosodic transfer instead of transfers on utterance-level [Kli+19]. The drawback of these approaches that perform unsupervised learning of a prosodic latent space is that all variance apart from linguistic content, and sometimes speaker identity, is learned that way. This includes differences in noise level, recording setup and voice quality among other unwanted variables.

There have also been papers proposing TTS models that work towards direct prosody control instead of prosody transfer. One of these is CHiVE [Ken+19] which samples slightly randomized variations from the prosody latent space to increase naturalness. While it succeeded in providing minor prosody variations that proved to increase naturalness, it did not offer a possibility to control these variations. Raitio et al. proposed a TTS model which did offer control of several intuitive prosodic features, but did not provide fine-grained control, as the prosody was only parameterized for the whole utterance and also the audio quality suffered slightly even when not using the prosody parameters[RRC20]. One of the most capable models in the realm of prosody control is the Tacotron 2 based model proposed by Sun et al. [Sun+20]. While providing fine-grained prosody control, the model itself is based on Tacotron 2 and therefore autoregressive, which comes with the aforementioned drawbacks concerning training and inference times and occasional audio glitches like word repetitions or words skips [Ren+20]. As the used method to control prosody involves a variational autoencoder and latent space, the same drawbacks that were mentioned above apply here too. Thus, not only prosody parameters are embedded in the latent space but all information that is not speaker identity or phonetic content. There is therefore no guarantee that the embedding only represents the desired prosody parameters. The proposed model also had no mentioned method to easily input text with the desired prosody parameters.

The TTS model proposed in this work is based heavily on the Fast Speech 2 model[Ren+20] and more precisely on the implementation[1] of Fast Speech 2 by Chung-Ming Chien and Chie-yu Huang for their 2021 Paper on multi-speaker TTS [Chi+21]. The implementation includes support for multiple vocoder models. HiFi-GAN [KKB20] is the one used for this work, the implementation is based on the one of Jungil Kong [2]. For training the English model, the LJ Speech dataset [IJ17] was used, downloaded from Keith Ito's website[3]. The German model was trained using the German part of the CSS10 dataset [PM19], downloaded from the author's GitHUb repository [4].

[1] https://github.com/ming024/FastSpeech2
[2] https://github.com/jik876/hifi-gan
[3] https://keithito.com/LJ-Speech-Dataset/
[4] https://github.com/Kyubyong/css10

4. Model

In this chapter, the proposed TTS system for fine-grained prosody control is presented. First, the unmodified architecture of the Fast Speech 2 model, which the proposed model is heavily based on, is described in detail. After that, the training process is outlined. Lastly, the used implementation and modifications thereof are discussed.

4.1. Fast Speech 2

Fast Speech 2 [Ren+20] is a non-autoregressive TTS model based on Fast Speech [Ren+19] which was proposed in 2020. During runtime, it takes character sequences as input and converts them into a mel spectrogram of a recitement of the given text. It is designed in a way that tries to reduce the one-to-many mapping problem of text-to-speech synthesis, which describes the fact that for a given text there are many possible audio variations that would pose as valid recitements of the text. It tries to alleviate this fact by adding information about speech variation as additional input to the decoder, which narrows down the possible audio variations. During training, this information, which includes duration, pitch variation and energy (volume), is extracted from the goal waveform. During inference, this information is provided by predictor networks, which are also trained during the main model's training process. This also enables the model to control the prosody of the resulting speech, as the predicted prosody information is interpretable and can be modified in a way to purposefully alter pitch, duration or energy of the utterance.

Fast Speech 2 utilizes an encoder-decoder architecture based on network modules with modified Transformer [Vas+17] architecture. The encoder-decoder architecture can be seen in Figure 4.1, which generally shows the overall architecture of Fast Speech 2. As shown, the model can be split up into the embedding network, the encoder network, the variance adaptor module and a decoder network. In the following sections, the architecture of each of these network parts will explained in detail.

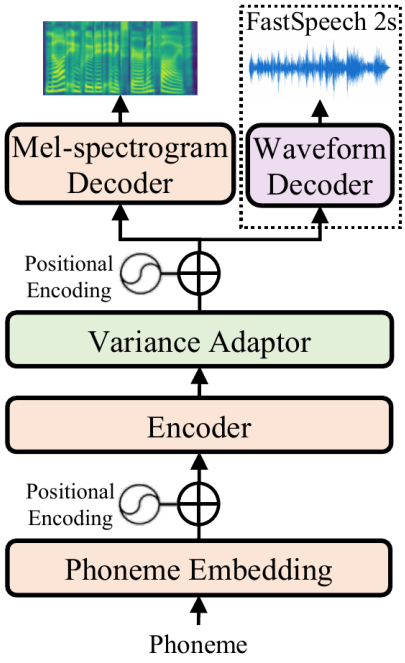


Figure 4.1.: Architectural overview of Fast Speech 2 [Ren+20]

4.1.1. Embedding Architecture

Before being fed into the model, any given text is converted into a phoneme sequence. This sequence is then turned into a phoneme embedding sequence by the embedding layer, which is the first part of the model. Embedding layers are a common first part in neural networks, as they turn the very high dimensional but sparse one-hot encoded vectors that encode the given character or phoneme sequence into dense vectors of fixed sizes. This embedding space evolves during the training process to group related phonemes in clusters, which then enables the rest of the network to perform better as the embedding now includes useful features.

The phoneme embedding is then extended with positional encoding information, which is a representation of the location of single phonemes in the whole sequence. The positional encoding enables parts of the encoder network that are based on the Transformer architecture to make use of the order of the sequence.

4.1.2. Encoder Architecture

Now, the enhanced phoneme embedding sequence is being fed into the encoder. The encoder consists of four *feed-forward Transformer* (FFT) blocks, which are network modules with modified Transformer architecture and have already been used in the original Fast Speech model. The structure of a single FFT block can be seen in Figure 4.2.

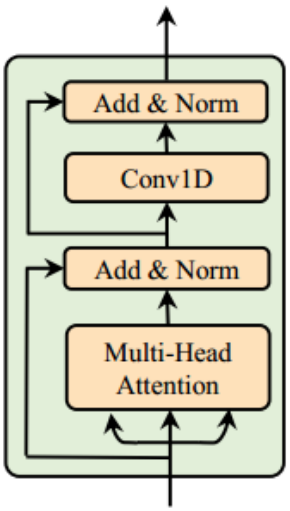


Figure 4.2.: Structure of a single FFT block [Ren+19]

As shown, a single FFT consists of a self-attention network and a one-dimensional CNN of two layers. The self-attention part is multi-headed and thus able to extract cross-position information from the phoneme-sequence. The original Transformer architecture uses fully-connected layers instead of CNN layers. The decision to switch to CNNs was made since in phoneme and spectrogram sequences, adjacent hidden states are more closely related than distant ones and CNNs are better suited to detect local patterns independent of their location. However, the residual connections and layer normalization steps used in the original architecture are carried over.

4.1.3. Variance Adaptor

The variance adaptor's purpose is to add variance information to the phoneme hidden sequence in order to alleviate the one-to-many mapping problem inherent in speech synthesis and make it possible for the decoder to know which speech variant is the right one to decode to. The variance information used in the original Fast Speech 2 paper is threefold: phoneme duration, pitch and energy. Phoneme duration acts on phoneme-level and describes how many frames in the mel spectrogram a phoneme sounds. Pitch, or more precisely: pitch contour, acts on frame-level and represents the F_0 frequency of each frame. Lastly, energy represents the frame-level magnitude of the mel spectrogram and correlates with speech volume. Each of these variance parameters is predicted by a separate module.

As seen in the architectural overview of the variance adaptor in Figure 4.3, the phoneme hidden sequence is first fed into the duration predictor, a two-layer one-dimensional CNN with a single fully connected layer at the output. This submodule then predicts, on a logarithmic scale, how many mel frames correspond to each phoneme in the given hidden sequence.

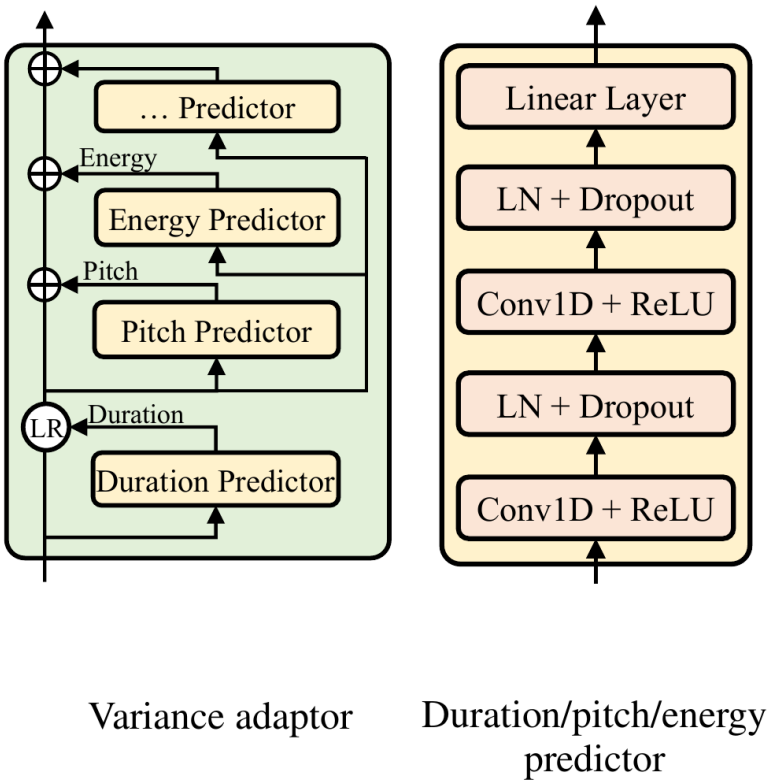
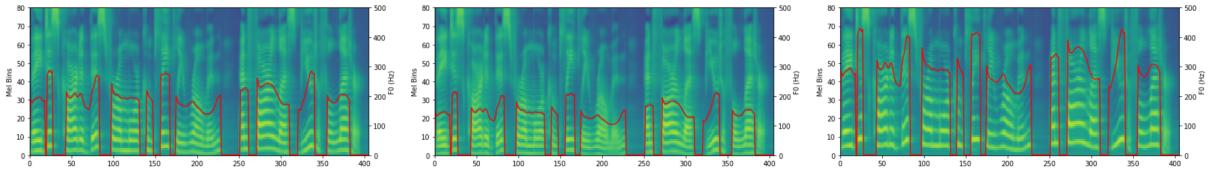


Figure 4.3.: Architecture of Fast Speech 2's variance adaptor and its submodules [Ren+20]

Subsequently, the phoneme hidden sequence and the output of the duration predictor are fed into the length regulator module, denoted as *LR* in Figure 4.3. The length regulator is then used to solve the mismatch between the length of the phoneme sequence and the mel spectrogram sequence but can also control speech rate. As the duration predictor gives a prediction d_n for every phoneme, describing how many mel frames correspond to it, the length regulator can match the sequence lengths by expanding the hidden state of each phoneme d_n times. To control the speech rate, the length regulator can expand each hidden state $x \cdot d_n$ times, with x being a scaling factor. For example, an x value of 0.8 would result in each phoneme corresponding to 20 percent less mel frames and the resulting speech would consequently be 20 percent faster.

The expanded phoneme hidden sequence is now consecutively fed into the pitch and energy predictors. Both predictors share their architecture with the duration predictor but differ in their training targets. The pitch predictor is trained to predict a pitch spectrogram [Sun+13; HT15] for the given hidden phoneme sequence, which can then be converted into pitch contour information, assigning each frame's pitch F_0 one of 256 possible values on a logarithmic scale. The embedded pitch information is then added to the expanded hidden sequence.



(a) $\hat{F}_0 = F_0$ (b) $\hat{F}_0 = 0.75F_0$ (c) $\hat{F}_0 = 1.50F_0$

Figure 4.4.: Effects of different pitch control parameters on the resulting spectrogram [Ren+20]

The energy predictor works similar, but instead of pitch contour, it predicts the L2-norm of the amplitude of each mel spetogram frame as one of 256 values on a uniform scale. The embedded energy information is now also added to the expanded hidden sequence and this final hidden sequence now leaves the variance adaptor and is handed over to the decoder. As shown in Figure 4.4, the predicted formant frequencies can be controlled by a multiplicative parameter, which can flatten or exaggerate the predicted prosody contour. The same method allows control of the predicted amplitudes. Note that these parameters, just like the duration control parameter mentioned earlier, can only control the prosody of the whole utterance and are therefore not fine-grained.

4.1.4. Decoder Architecture

The decoder is architecturally very similar to the encoder. It also consists of four FFT blocks, with the difference being that its output is a mel spectrogram with 80 dimensions. As the FFT blocks operate non-autoregressively, the decoder can convert the expanded hidden sequence into a mel spectrogram sequence in parallel.

As shown in Figure 4.1, the Fast Speech 2 paper presents two different decoder options. The one seen in the top-left, denoted as *Mel-spectrogram Decoder* is the one we are concerned with and which was described in the preceding paragraph. This decoder corresponds to the common mode of operation in neural TTS systems, in which the output of the model is a mel spectrogram that can then be converted into audio by a vocoder. However, the paper also proposes an alternative architecture, called *Fast Speech 2s* [Ren+20] in which the mel spectrogram decoder is replaced by an alternative decoder, as seen in the top-right of Figure 4.1, in which it is denoted as *Waveform Decoder*.

Using the waveform decoder turns the model into a true end-to-end TTS system, converting text into audio without any intermediate outputs. But, as one can infer from the presented results in the paper, this does not come with any remarkable advantages over the default Fast Speech 2 model. Inference times are slightly reduced for Fast Speech 2s but audio quality, while still good, drops below that of state-of-the-art autoregressive models like Tacotron 2. Also, one of the main advantages of splitting up the TTS pipeline into two models, which is the reusability of a trained vocoder model, is now absent. This results in

a many-fold increase in training time when switching from Fast Speech 2 to Fast Speech 2s.

While still being an exciting work towards end-to-end TTS models, the disadvantages described above make using Fast Speech 2s impractical given the goals of this work, and so the default mel spectrogram decoder will be used.

4.2. Training Process of the TTS System

Training the Fast Speech 2 model requires additional information, as the variance adaptor needs to be trained alongside the TTS model. This additional information includes phoneme-level information about pitch, volume and duration. Still, only text-audio pairs are needed as training data. The missing information is acquired from alignment data that assigns every phoneme in the training data it's corresponding time frame in the respective audio file. The model can then lookup the pitch and volume during the phoneme's time frame to get phoneme-level data about volume and pitch.

If the training data does not have alignments included, alignment applications like the Montreal Forced Aligner (MFA) [McA+17] can be used to obtain alignment data for the used dataset. To obtain alignment data, MFA needs a pronunciation dictionary for grapheme to phoneme (G2P) conversion and an acoustic model to predict the relationship between phones and time frames in the audio file. Fortunately, MFA includes several prebuilt pronunciation dictionaries and pretrained acoustic models. Alternatively, a G2P model for dictionary creation and/or an acoustic model can be trained on the given dataset.

When the alignments are ready, the preprocessing process creates a mel spectrogram from the audio file of each datum. These can then act as training goals. During this process the phoneme-level duration, volume and pitch information is also derived from the alignment data as described above. This information is then used to train the variance adaptor's submodules. During training, the correct variance information as derived from the alignments is added to the hidden phoneme sequence. The duration, pitch and energy predictors all try to predict the respective variation information and are trained using the actual values as goal. During inference, the variance information of the predictors is then used.

4.3. Implementation and Modifications

The used implementation[1] of Fast Speech 2 is mostly faithful to the description in the paper. There are only three deviations from the paper that are worth mentioning. For one, the addition of a post-net adapted from Tacotron 2 that tries to ease the reconstruction process for the vocoder by passing the predicted mel spectrogram sequence through a five-layer CNN and adds the predicted residual to the sequence.

[1] https://github.com/ming024/FastSpeech2

Second, the implementation added the possibility to change the resolution of pitch prediction as well as the resolution of energy prediction from frame-level to phoneme-level, which may result in better prosody [Chi+21]. The third change is that the pitch predictor directly uses F_0 as target instead of predicting pitch spectrograms. This was also the case in early versions of the Fast Speech 2 paper.

The implementation is written in Python and uses Pytorch [Pas+19] as deep learning API. It includes code from different vocoder implementations, the one used here is an implementation of HiFi-GAN[KKB20][2]. Both implementations were published on GitHub under the MIT License, which allowed the author of this thesis to use them as basis for the model proposed in this work. In the following subsections, all additions and modifications are described that were needed to achieve the goal of fine-grained prosody control in an English model and a German model.

4.3.1. Modifications for German Model

To make training a Fast Speech 2 model for German language possible, the usable symbols and method of G2P conversion had to be updated and a new set of text cleaners had to be added. The usable symbols originally only included characters used in the English language and the ARPABET phonetic transcription codes [Kla01]. These were changed to character sets that included special characters used in German writing and a phonetic transcription code for the German language created by Prosodylab [GHW11]. As described in section 4.2, alignment data is needed for training the model. Like in the original paper, MFA was used to generate alignment for the training data. MFA also has the ability able to create G2P models, which was used to create a G2P model that was then integrated into the German Fast Speech 2 version for G2P conversion of text inputs during inference.

4.3.2. Modifications for Fine-Grained Prosody Control

To achieve the goal of prosody control on word-level, several changes had to be made. First, all control parameters needed to be turned into vectors of the desired resolution. As we aim for word-level control, these vectors need to contain one parameter per word in the input. As Fast Speech 2 internally works on phoneme-level (in the used implementations even the predictors for pitch and energy), the parameter vectors now need to be extended to the length of the phoneme sequence. During this extension process, we need to keep track of word borders inside the phoneme sequence, so as to apply to each phoneme the parameter set for the word it is part of.

During the G2P conversion, the amount of phonemes a_w added to the phoneme sequence is noted for each word w, which then enables us to extend the parameter vectors by expanding the parameter for each word w exactly a_w times.

[2] https://github.com/jik876/hifi-gan

After this, the control parameter vectors are on phoneme-level resolution and can be applied to the respective variance prediction.

As described above, the duration prediction consists of one value for each phoneme, representing the amount of mel frames corresponding to it. An element-wise multiplication can now be applied to the vector of duration control parameters and the duration prediction. This scales the duration of each phoneme according to the duration parameter set for the word it is part of, and thus results in word-level duration control.

As for the pitch and energy prediction, they also contain one value for each phoneme. The values in the pitch prediction describe the formant frequency f_0 for each phoneme, while the values in the energy prediction contain information about the mel spectrogram amplitudes for each phoneme. These can also be controlled on word-level by applying an element-wise multiplication to the prediction vectors and the respective control-value vectors. The result is word-level control of pitch-variance and mel spectrogram amplitude, which approximates speech volume.

Just like the single pitch control parameter in the original Fast Speech 2 model, the word-level pitch parameter can only control pitch-variance and cannot specifically increase or decrease pitch. This restriction can be overcome, however, by directly changing the predicted pitch F_0 of the targeted words. This additional control for absolute pitch was implemented by allowing special keyword-strings in the vector of pitch control parameters. For example, the keyword *HIGH* will make the system set the pitch F_0 of the targeted word to be ten percent higher than the maximum pitch F_0 of the whole utterance. This can be used to force specific words to be emphasized by high pitch.

Controlling the prosody of the utterance on word-level by entering dozens of parameters, three values for each word, proved to be impractical, time-consuming and error-prone. To improve the user interface for prosody control, support for the Speech Synthesis Markup Language [TI97] (SSML) was added. SSML is an XML-based markup language used in TTS applications, with tags that allow the user to add prosody information to the input. This includes support for adding pitch, speech rate and volume parameters and a tag to emphasize certain parts of the text.

The added SSML support allows users to enter SSML strings that contain prosody information, with the following tags being currently supported to specify the desired prosody: <prosody range> for the pitch parameter, <prosody rate> for the duration parameter, <prosody volume> for the energy parameter and <emphasis> which can used to automatically set the control parameters in a way so as to emphasize the words inside the tag. The emphasis tag is currently handled by setting slightly lower duration and energy parameters for all other parts of the text and using a higher duration parameter, higher energy parameter and the *HIGH* keyword as pitch parameter for the words inside the tag.

As the many other tags included in SSML are currently ignored by the parser, the model can only be said to support a dialect of SSML. Integration of additional tags or full SSML support will for now be left for future work. An example of a valid SSML input that uses various prosody tags is given below.

> <prosody rate=1.2> Would you please </prosody> give me the <emphasis> red </emphasis> <prosody volume=-20%> cup over there? </prosody>

Given the above string, the model will synthesize speech with the desired prosody. The first three words will be about 20 percent slower than the second three words. The word *red* will be emphasized by slow and loud speech, as well as by increased pitch. Finally, the last three words will have reduced volume compared to the rest of the utterance. As seen in the example, the tags support both percentage parameters and decimal parameters.

5. Evaluation

The goal of the evaluation is twofold. The first goal is to assess whether the added capabilities for fine-grained prosody control can be used to add clear emphases to the synthesized speech and to what extent the fine-grained control parameters affect synthesis quality. The second goal is to investigate whether the prosody control parameters of the modified Fast Speech 2 model can be used to control prosody of a language other than English.

5.1. Evaluation Method

Two Fast Speech 2 models were trained for evaluation: one model for English speech and one for German speech. These models were then used to synthesize two versions of several texts from the respective test set. One version was synthesized without utilizing the prosody control parameters, the other version was synthesized using a set of control parameters that aimed at emphasizing a certain word or part of the text. Then, a survey was conducted to obtain Mean Opinion Score (MOS) [CP06] values for naturalness, comprehensibility, and, for the speech samples with explicitly added emphasis, perceptibility of that emphasis. Intelligibility evaluation by calculating the word error rate, which is the fraction of words that are substituted, skipped or deleted during speech synthesis, was not used for evaluation as non-autoregressive TTS models like Fast Speech 2 rarely display these errors, which are typical for autoregressive models [Ren+20]. Nevertheless, any intelligibility problems would still cause low comprehensibility scores and thus not go undetected.

To establish a baseline for naturalness and comprehensibility, ground-truth audio samples taken from the datasets were also evaluated by the participants. The evaluation results will thus allow for an analysis of the performance of the Fast Speech 2 models regarding naturalness and comprehensibility of the synthesized speech compared to the ground-truth quality.

The first goal, determining whether fine-grained prosody control can add emphases and how it affects speech quality, can be achieved by analyzing the results regarding emphasis perception and comparing the results of synthesized samples without added prosody and samples with added prosody. This will establish whether the prosody control parameters affect speech quality and if the added emphasis is easily perceptible. The second goal of the evaluation, determining if the modified Fast Speech 2's fine-grained prosody control can be used in languages other than English, can be attained by comparing the results of prosody-controlled outputs of the German and English models.

5.2. Evaluated Models

In this subsection, the training process of the evaluated German and English Fast Speech 2 models is described. This includes information about the used datasets and data preprocessing, as well as the used hyperparameters. Lastly, the selection and synthesis processes for the evaluation samples are outlined.

5.2.1. Datasets

The well-known LJ Speech dataset by Keith Ito [IJ17] was used for training the English model, as it is known for its high quality and suitability for TTS model training [Dun19]. The dataset is made of up approximately 24 hours of recorded speech, split into 13 100 short audio clips with durations between one and ten seconds. All audio clips are recordings of a single female speaker reading extracts from seven non-fiction books that are in the public domain. The recordings were made by the LibriVox project [Kea14] and curated by Keith Ito for the dataset.

The required alignment data was generated by the MFA program [McA+17], using the Librispeech pronunciation dictionary for the English language [Pan+15] and the English acoustic model that is included with MFA, which is also trained on the Librispeech corpus.

For the German model, the German part of the CSS10 dataset [PM19] was used for training. It contains roughly 16 hours of speech recordings, distributed over 7 500 audio clips with a duration mostly between two and fifteen seconds. Like LJ Speech, CSS10 is a single speaker dataset built from readings of public domain books. Due to lower sampling rates, the audio quality is somewhat worse compared to the high-quality audio of LJ Speech but still acceptable. Also, the recording environment changes between recordings of different books, which leads to different noise-levels and slight changes in voice quality between samples.

As the audio samples of the CSS10 dataset are taken from audiobooks which are recited in a remarkably calm way, pauses are often noticeably longer than in everyday speech which would lead to a less natural speech synthesis result and even caused some problems during training as the model struggled to converge because of the highly varying pause durations. To fix this problem, the pause durations in the audio data were adjusted to a maximum of 300 milliseconds. This was achieved by analyzing the audio data using a voice activity detection algorithm called WebRTC VAD[1] and stunting all pauses longer than 300 milliseconds to a random length between 250 and 300 milliseconds.

Just like for the LJ Speech dataset, MFA was used to generate text-audio alignments for the German training data. During alignment, the German acoustic model by Prosodylab[GHW11], trained on the Globalphone dataset [Sch02] and the German pronunciation

[1] https://webrtc.org/

dictionary [2] by Prosodylab were used.

The dictionary was also utilized to train a G2P model for conversion of German words from graphemes to phonemes of the Prosodylab phone set for German. This G2P model was used during inference in order to convert words that were not found in the dictionary to phoneme sequences compatible with the trained Fast Speech 2 model.

5.2.2. Training Preparations

Before training, preprocessing of the training data was required. From the text-audio pairs and the respective alignments, much information is derived. First, mel spectrograms for all audio samples are generated for later use as training targets for the decoder. Then, all texts are converted into phoneme-sequences using the G2P model already needed for the creation of the alignment data. Lastly, the alignment data and respective mel spectrograms are used to obtain phoneme-level duration, pitch F_0 and spectrogram amplitude information. These can easily be gathered by looking up the mel frames corresponding to a given phoneme. The amount of frames is the duration information, while the amplitude and pitch F_0 of those frames are the energy information and pitch information, respectively.

5.2.3. Training Configuration

Both models were trained on a server with an Intel 4124 CPU, 32 gigabytes of memory and a single NVIDIA RTX Titan GPU. The hyperparameters for the English model are as follows. In the encoder, four FFT blocks were used with hidden phoneme embeddings and self attentions of size 256. Two attention heads were used. The kernel size in the one-dimensional two-layer CNN of the FFT blocks was 9, with input and output sizes of 256 and 1024, respectively, for the first layer while input and output sizes of 1024 and 256 were used in the second layer. The decoder is parameterized in a similar manner but uses six FFT blocks instead of four. A dropout rate of 0.2 is applied to both the encoder and the decoder network. The variance predictor was set to a kernel size of 3 in its one-dimesional CNNs, with input and output sizes of 256. A dropout rate of 0.5 is applied here.

These hyperparameters mostly match the ones described in the paper. Only slight adjustments were made, like the increased amount of FFT blocks and reduced hidden state dimensions in the decoder when compared to the encoder. Also, a slightly higher dropout rate was applied to the encoder and decoder, as it originally was set to 0.1. A batch size of 64 was used and the model was trained for 500 000 steps.

The German model's hyperparameters are very similar, but the dropout rate in the encoder and decoder was increased to 0.3 to achieve better generalization on the smaller German dataset. Also, a lower batch size of 48 was used as the german audio samples are longer on average, which led to increased GPU memory requirements. The German model was trained for 400 000 steps. For both models, the training time was about 72 hours.

[2] https://github.com/prosodylab/prosodylab.dictionaries

5.2.4. Synthesis of Evaluation Samples

The texts from which the evaluation samples were synthesized were taken from the test part of the dataset. Eight texts that are complete sentences with lengths ranging from four to twenty words were randomly chosen from each dataset. From these texts, speech was now synthesized using the model for the respective language. Now, one to two suitable words were chosen and now speech was synthesized using prosody control parameters to emphasize these words. The emphasis was either achieved by using the <emphasis> tag or by using a similar prosody control configuration as the <emphasis> tags creates, but instead of increasing pitch for the emphasized part, the pitch was lowered. The more appropriate approach was selected for each sentence before synthesizing. No decision was revised. This decision process may be automated in later works, as it can be based on the pitch F_0 in the default synthesis at the location of the words that should be emphasized.

All evaluation samples, except for one, were only synthesized once. One sample[3], however, only contained noise and was synthesized a second time. The behavior could not be reproduced. It was maybe caused by the short length of the input text, which was only four words long and is the lower limit appearing in the training data.

Additionally, five audio samples of complete sentences from each dataset were chosen and added to the other samples to later establish a baseline for evaluation. The order of the samples was randomized.

5.3. Evaluation Results

The results of the evaluation survey are presented in this section. The presented mean opinion scores (MOS) give a numerical estimation of speech comprehensibility, naturalness and, if present, perceptibility of emphasis. Scores are given on a Likert scale [Lik32] of 1 to 5, with 5 being the best score. Later, the comprehensibility and naturalness scores are combined into a general MOS. Obviously, MOS values are inherently subjective and heavily depend on the expectations of the participant. Therefore, the variances of MOS values for single samples as well as overall variances have to be analyzed additionally.

MOS values were calculated as arithmetic mean of the respective scores:

$$MOS_x = \sum_{n=1}^{N} s_{xi}$$

with:

x being the sample evaluated
s_{xi} being the score given to x by participant i
N being the number of scores taken into account for this MOS

[3] Deutsch07.wav, see Appendix A for more information

The survey included seven participants, all of which are German native speakers and fluent in English.

5.3.1. English Model

The MOS evaluating comprehension and naturalness for the ground-truth samples are shown in Table 5.1. The samples are denoted as GT 1 through 5. As expected for ground-truth samples, both comprehension and naturalness scored quite high with a total MOS of 4.45 for comprehension, while naturalness received a slightly lower 4.34. Variance was generally low for both measures, except for sample GT 3 which had a variance of 0.9 across its comprehension scores. This might be due to its length, as the corresponding text, *"A thorough inspection would have involved washing and cleansing the back, and this is not practical in treating an acutely injured patient."*, is the longest text used in the evaluation. As the participants of the survey are no English native speakers, this rather long and complicated sentence might have caused the differences in English proficiency between the participants to become more apparent and influence their comprehensibility rating.

	GT 1	GT 2	GT 3	GT 4	GT 5	All
Comprehension MOS	4,43	4,28	4,28	4,71	4,57	4,45
Comprehension Variance	0,61	0,57	0,90	0,24	0,29	0,49
Naturalness MOS	4,13	4,00	4,42	4,57	4,57	4,34
Naturalness Variance	0,47	0,33	0,28	0,61	0,28	0,40

Table 5.1.: Comprehension and naturalness MOS and respective variances for English ground-truth samples

Now, we will inspect the MOS results for synthesized samples with default prosody parameters as shown in Table 5.2. In the table, the synthesized samples are called S 1 through S 8. Comprehension scores have decreased slightly but are still high with an overall MOS of 4.32. The decrease is expected even when evaluating a well-trained model, as the model has to generalize from the training data. Generalization enables it to perform well on unseen data, but comes at the cost of a small hit to performance on all data, including training data and of course unseen data such as these texts from the test set.

Naturalness scores decreased a bit more compared to ground-truth samples, receiving a total MOS of 4.0. This is probably due to small audio artifact, which sometimes are audible even in samples from well performing TTS systems. These do not hinder speech comprehension but can sound unnatural and therefore lead to reduced scores on naturalness.

Both variances are higher than the ones seen in the ground-truth results, which is partly because values further from the limits of the scale can be more spread out across it. Additionally, some of the participants might be more sensitive to the mentioned minor audio artifacts while others don't notice or deem them too minor to cause a score reduction. This would also explain a slightly higher variance.

	S1	S2	S3	S4	S5	S6	S7	S8	All
Comprehension MOS	4,43	4,29	4,00	4,43	4,43	4,29	4,14	4,57	4,32
Comprehension Variance	0,62	0,90	0,66	0,61	0,61	0,90	0,80	0,28	0,62
Naturalness MOS	4,14	4,57	4,43	3,86	4,00	4,00	3,29	3,71	4,00
Naturalness Variance	0,47	0,61	0,61	0,47	0,33	0,66	0,23	0,57	0,58

Table 5.2.: Comprehension and naturalness MOS and respective variances for English synthesized samples with default prosody

The last results for the English model are presented in Table 5.3. These again show MOS values for comprehension and naturalness, but now for samples synthesized using prosody parameters in order to add an emphasis to part of the utterance. An additional MOS evaluates the perceptibility of the added emphasis. The texts used here are the same as in the evaluation of synthesized samples with default prosody parameters, so the S1 sample in Table 5.2 is synthesized from the same input, apart from prosody control parameters, as the S1 E sample in Table 5.3.

Comprehension MOS have increased when compared to the default prosody samples, with an overall MOS of 4.46. Interestingly, this is even higher than the comprehension MOS of the ground-truth samples. This may be due to slightly increased audio durations as the emphasized words are spoken somewhat slower, which could enhance comprehension. However, the added emphasis and increased comprehensibility came at the cost of a lower naturalness MOS of 3.82. Still, while slightly decreasing naturalness, the emphases seemed to be well perceptible as the perceptibility MOS across all samples is 4.13.

Variances are in line with previous observations with a 0.39 variance in comprehension MOS, 0.69 naturalness MOS variance in 0.76 variance in the emphasis perceptility scores. This further indicates that the emphases were generally well perceptible. Upon further inquiry, it can be noted that, while well perceptible, the addition of emphasis might have slight robustness problems, as sample S8 E scores only 3.0 on perceptibility, which is the reason for the increased variance in the perceptibility MOS when compared to naturalness and comprehension.

	S1 E	S2 E	S3 E	S4 E	S5 E	S6 E	S7 E	S8 E	All
Comprehension MOS	4,42	4,28	4,42	4,71	4,57	4,57	4,42	4,28	4,46
Comprehension Variance	0,61	0,57	0,28	0,23	0,28	0,28	0,28	0,90	0,39
Naturalness MOS	3,85	4,00	3,57	3,85	3,57	4,28	3,85	3,57	3,82
Naturalness Variance	0,47	0,66	1,61	0,80	0,28	0,90	0,80	0,28	0,69
Emphasis MOS	4,28	4,85	4,71	4,85	3,57	3,42	4,28	3,00	4,13
Emphasis Variance	0,57	0,14	0,23	0,14	0,62	0,28	0,57	0,33	0,76

Table 5.3.: MOS and variance for comprehension, naturalness and perceptibility of the explicitly added emphasis for English synthesized samples with modified prosody

Generally, the English model seems to perform well and provide comprehensibility comparable to that of ground-truth samples but at a slight reduction in naturalness. Adding prosody parameters in order to emphasize parts of the input results in perceptible, mostly even well perceptible, emphases and increases in comprehensibility compared to ground-truth levels. However, this reduces naturalness again, but the naturalness difference between prosody controlled synthesis and default synthesis is smaller than the one difference in naturalness observed between ground-truth and default synthesis. The result regarding comprehensibility and naturalness can be summarized in three numbers by combining naturalness and comprehensibility scores into a combined MOS. Using this combined MOS, gound-truth samples received a score of 4.40, synthesized samples reached 4.16 and synthesized samples with explicit prosody scored 4.14.

The corresponding texts for each sample used in the evaluation of the English model are listed in Appendix B.

5.3.2. German Model

Table 5.1 shows the MOS results regarding comprehension and naturalness for the German ground-truth samples. The overall MOS for comprehensibility is very high with a score of 4.82 with low variance of 0.21, showing an increase over the 4.45 comprehension score of the English model. This is very likely due to the participants being native German speakers, which makes comprehension of German speech inherently easier than comprehension of other languages, even when speaking them fluently, and explains the increase over the English ground-truth comprehensibility MOS.

As for the naturalness MOS, it is comparable to the respective English MOS with a score of 4.28 and a low variance of 0.33. Interestingly, the gap between comprehensibility and naturalness scores is much more pronounced in the German dataset.

	GT 1	GT 2	GT 3	GT 4	GT 5	All
Comprehension MOS	4,85	4,71	4,85	4,85	4,85	4,82
Comprehension Variance	0,14	0,57	0,14	0,14	0,14	0,21
Naturalness MOS	3,85	4,42	4,42	4,42	4,28	4,28
Naturalness Variance	0,47	0,28	0,28	0,28	0,23	0,33

Table 5.4.: Comprehension and naturalness MOS with respective variances for German ground-truth samples

The MOS results for synthesized samples with default prosody parameters are shown in Table 5.5. Comprehension scores are still very high with a score of 4.69. But, as with the English model, the comprehension score decreased slightly compared to the ground-truth samples. The overall higher comprehensibility over the English model can still probably be traced back to the participants being native speakers. Comprehension score variance is nearly unchanged at 0.25.

Naturalness scores provide a different picture. The overall MOS decreased to 3.59 from the ground-truth value 4.28 with variance also increasing to 0.72. The decrease in naturalness score is much more pronounced than in the English model. This may be explained by various properties of the German dataset. For one, the English dataset is consists of 24 hours of speech recordings, which is a 50 % increase compared to the 16 hours of the German dataset. This might explain why the German model seems to generalize worse than the English one. Also, the recording parameters of the contained speech recordings vary within the German dataset, leading to variations in noise level and even voice quality. This additional variation is not present in the English dataset, which may lead to improved training results.

Lastly, the results for synthesized prosody-cotrolled samples inferred by the German model are shown in Table 5.6. These, as always, include MOS results for comprehensibility and naturalness but also an MOS rating the perceptibility of the added emphasis. As in the English results, the texts used here are the same as in the evaluation of the synthesized samples with default prosody parameters, the only difference being the prosody control parameters.

	S1	S2	S3	S4	S5	S6	S7	S8	All
Comprehension MOS	4,43	4,71	4,86	4,86	4,57	4,71	4,57	4,86	4,69
Comprehension Variance	0,28	0,23	0,14	0,14	0,28	0,23	0,62	0,14	0,25
Naturalness MOS	4,14	2,58	4,43	3,42	3	4,14	3,57	3,43	3,59
Naturalness Variance	0,81	0,28	0,61	0,28	0	0,47	0,61	0,28	0,72

Table 5.5.: Comprehension and naturalness MOS and respective variances for German ground-truth samples

As observed in the English model, comprehension MOS values have increased when compared to the default prosody samples, now scoring 4.71. But, unlike the English model, the German model does not achieve better comprehensibility with emphases than ground-truth samples did. Comprehension MOS values might only be slightly lower than for ground-truth samples, but this still reinforces the assumption that the German model performs not quite as good as the English one.

The similarities continue as the naturalness MOS is also worse for prosody-controlled samples than samples synthesized with default parameters, with the emphasized samples scoring 3.29 against 3.59 for unemphasized samples. It seems, that the emphases, while well perceptible, either reduce audio quality or, more likely, do not entirely sound like natural emphases. This is probable as the model was not trained to produce natural sounding emphases and the parameters used to change the model's behavior in order to create a desired emphasis were hard-coded.

Still, like in the English model, the emphases are generally well perceptible, while naturalness scores decrease slightly. The samples received a MOS for perceptibility of 3.75. This score matches the pattern that emerged in the previous results, and the German model shows similar behavior as the English model did, but generally performs slightly worse. The same possible reasons described before apply here too. Variances for comprehensibility and naturalness are quite low, with values of 0.35 and 0.42, respectively. While the variance of comprehension is unremarkable, 0.42 is the lowest overall naturalness variance measured, which suggests that, while the synthesized prosody-controlled samples are not highly natural, no outliers occurred that sounded particularly unnatural.

The German model overall also performed well but earned slightly worse scores than the English model did. While Comprehensibility was always very good, even comparable to ground-truth, naturalness occasionally suffered somewhat. As observed in the English model, this got worse when utilizing prosody control parameters. The generated emphases were generally perceptible, with all samples except for one achieving MOS values greater

	S1 E	S2 E	S3 E	S4 E	S5 E	S6 E	S7 E	S8 E	All
Comprehension MOS	4,43	4,71	5,00	4,86	4,57	4,71	4,57	4,86	4,71
Comprehension Variance	0,62	0,57	0,00	0,14	0,62	0,23	0,62	0,14	0,35
Naturalness MOS	3,57	3,00	4,00	3,42	2,86	3,00	3,14	3,29	3,29
Naturalness Variance	0,61	0,00	0,66	0,62	0,14	0,33	0,14	0,23	0,42
Emphasis MOS	4,29	4	4,29	4,14	2,71	4,14	3,29	3,14	3,75
Emphasis Variance	0,57	0,66	0,23	0,47	0,57	0,47	0,23	0,47	0,73

Table 5.6.: MOS and variance for comprehension, naturalness and perceptibility of the explicitly added emphasis for German synthesized samples with modified prosody

than three. The result can again be summarized in three numbers. When combining naturalness and comprehensibility scores into a combined MOS, ground-truth samples received a MOS of 4.55, synthesized samples of 4.14 and synthesized samples with explicit prosody control scored 4.0.

The corresponding texts for each sample used in the evaluation of the German model are listed in Appendix C.

5.4. Discussion

The presented results achieved the first evaluation goal, as the evaluation indicates that the fine-grained prosody controls described in this work can be utilized to add well perceptible emphases to synthesized utterances with the loss of audio quality being within tolerable limits. The second evaluation goal was achieved by showing that the prosody control of Fast Speech 2 does indeed work for other languages than English as shown by the evaluated German model, which was able to add perceptible emphases to synthesized samples with perceptibility MOS values comparable to those of the English model. All prosody controlled samples were still very comprehensible and naturalness scores only dropped less than ten percent, which is well within limits as the prosody control parameters were hard-coded and only rudimentary adaptions to the current sample were made.

Another factor that might impair the model's ability to drastically alter, as is needed for adding clear emphases, is the lack prosodic variance in the training data. When training a default TTS model without prosody control, it is desirable for the training data to have little variance in speech volume and speech rate. Also, extreme prosodic features like strong emphases are avoided so as not to obscure the training goal of natural and comprehensible everyday speech.

As the used datasets are intended for use in TTS model training, these points hold true for them. While beneficial for the training of default TTS models, the evaluated models thus have trouble generalizing across a broad range of prosodic parameters as very little variance is provided during training. This is especially noticeable for the energy parameter, as volume is the dimension of prosody control with the least amount of variance in the training data. While volume can be controlled by tweaking the energy parameter, this is only possible within quite narrow limits and changes in the parameter result in loss of speech quality much more quickly than it does for the other two prosody parameters. Thus, the little volume variance in the training data is presumably responsible for a large part of the naturalness reduction in the synthesized samples with added emphases.

As for the score differences between the English and German models, some of them might be explained by the native language of the participants, as German native speakers might tend to award higher comprehension scores to German speech than to English speech. The lower naturalness scores of the German model on the other hand might have to do with dataset differences, as the German dataset is about 50 % smaller than the English dataset and the included samples display more variation in noise level and recording setup than the English dataset.

Variance of MOS was high at times, reaching values of over 0.7 in Table 5.3, Table 5.5 and Table 5.6. This shows that, while MOS can approximate objective quality assessments from multiple objective scores, the participants sometimes evaluate the samples quite differently, which can get lost in the single MOS value. An important cause of the rating differences are probably varying expectations among participants. While a person with prior knowledge in computer science or an interest in technology may know or suspect that constructing a TTS system is a hard task and therefore can be impressed easier, someone without this knowledge might compare the samples to the naturalness of in-person small talk and is then easier to disappoint.

6. Conclusion

This final chapter gives a brief overview of the proposed model and attained evaluation results, followed by a conclusion to the questions that were aimed to be answered in this thesis. Finally, possible future works based on the achieved results are outlined.

6.1. Conclusion

In this work, a modified TTS model based on Fast Speech 2 was proposed to achieve fine-grained prosody control in a modern, deep learning based TTS system. A German and an English model were trained and evaluated to discern whether the modified model is capable of synthesizing high-quality speech while enabling fine-grained prosody control and if the variance adaptor of the modified Fast Speech 2 model performs well on languages other than English.

The modified Fast Speech 2 model was mostly realized by extending the existing variance adaptor in Fast Speech 2, which originally only allowed for utterance-level prosody control. These modifications made it possible to use word-level prosody parameters to control the internal embeddings for speech rate, pitch and volume created by the variance adaptor. Additionally, a parser for parts of the Speech Synthesis Markup Language was added to make the input of complex prosody parameters for a given text more practical and user-friendly and allow for later integration of the model into workflows involving SSML.

The evaluation of both trained models confirmed that the proposed model is able to synthesize high-quality speech samples both in German and English, with the English and German model receiving an overall MOS of 4.16 and 4.15, respectively, which is close to the ground-truth MOS of 4.40 for English samples and 4.55 for German samples. It was further shown that fine-grained prosody control is achieved by the model as it was able to add various, well perceptible emphases to the synthesized samples by varying speech rate, pitch and volume. This claim is supported by the evaluation, in which emphasis perceptibility received a MOS of 4.13 for the English model and 3.75 for the German model.

As both the English and German model were able to add emphases to the synthesized speech using fine-grained prosody control, it can be confirmed that the proposed model can be used to control prosody in languages other than English. As mentioned, the modified model was able to synthesize high-quality speech while controlling prosody on word-level. However, the speech quality suffered somewhat when compared to samples obtained using the default parameters. But the deviation was within tolerable limits as MOS values for

comprehension even increased slightly, while the overall MOS for naturalness decreased by 0.18 for English samples with emphasis and by 0.69 for German samples with emphasis. Alleviating these minor drawbacks of fine-grained prosody control is left for future work.

6.2. Future Work

There are multiple pathways for further enhancements of the results of this work. One major area of enhancement are further modifications to the model, allowing for more precise prosody-control or increased user-friendliness. The other area are the used datasets, as experiments with other or even new datasets may lead to possible fixes for the decrease in speech quality when making big changes to the prosody parameters.

Model Enhancements

A possible way to enable even more precise prosody control is leveraging the fact that, internally, the model works with phoneme-level prosody parameters. As of now, only word-level prosody parameters can be controlled, which are then extended and applied to all phonemes of the corresponding word. It would be worth researching, whether applying some smoothing to the parameters on prosody-level in order to create less harsh changes between word borders in the prosody sequence leads to more natural sounding results. Also, being able to directly control prosody parameters on phoneme-level might useful for some users in certain situations.

As for user-friendliness, a more complete integration of SSML would also be a desirable enhancement for the model, as it would allow for better integration into SSML environments and more possibilities for automation. Introducing extensive error checking on the input SSML string would also further increase user-friendliness.

More diverse prosody control parameters could be introduced. This is achievable either by adding more variance predictors to the model or by handcrafting sets of prosody parameters that correlate with certain desired prosody types. These handcrafted parameter sets may also be templates that are then automatically adjusted to fit the given text. For example, the <emphasis> tag might be automated in such a way, so that the default pitch F_0 of the words that shall be emphasized are examined to decide whether increasing or decreasing pitch for emphasis would sound more natural.

Lastly, the variance adaptor of Fast Speech 2, which is shown in Figure 4.3, can be modified to contain more than three variation predictors, which would consequently increase the amount of controllable prosody dimensions. Possible additions are emotional content and strength of emphasis. However, training goals for any new prosody predictors are needed and therefore, these additional prosody dimensions would have to be known for the training data.

Alternative Datasets

Another possibility for future work would be to retrain the German model using a larger and higher quality TTS dataset. The used CSS10 dataset is rather small, containing only 16 hours of speech recordings, and has further drawbacks as the long pauses caused by a very pronounced audio-book-style speech pattern make it harder to synthesize natural speech. Additionally, suboptimal sound quality and varying noise levels and recording environments worsen the problem. The recently released Open German Voice Dataset[1] might be a good alternative and could potentially close the observed gap in audio quality between the English model and the German model.

Major improvements in the sound quality of samples with extensive prosody control may be possible if the model is trained on a dataset with high variance in the used prosody parameters, including speech rate, pitch and volume. These variances are usually kept at a minimum in commonly used TTS datasets, as it makes training TTS models without variance adaptors harder. But as Fast Speech 2 can use the embeddings of these prosody parameters during training, it may even ease the training process in this case. Also, training a model on such a dataset probably leads to an increase in prosody control capabilities as the variance adaptors will be able to generalize better and will therefore able to better predict embeddings, especially for extreme values. The higher prosody variance may be achieved by data augmentation or by constructing a new dataset specifically for this purpose.

[1] https://github.com/thorstenMueller/deep-learning-german-tts

Bibliography

[Bir14] P Birkholz. *VocalTractLab–Towards high-quality articulatory speech synthesis.* 2014.

[Chi+21] Chung-Ming Chien et al. "Investigating on incorporating pretrained and learnable speaker representations for multi-speaker multi-style text-to-speech". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2021, pp. 8588–8592.

[CP06] Min Chu and Hu Peng. *Objective measure for estimating mean opinion score of synthesized speech.* US Patent 7,024,362. 2006.

[Cyb89] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[Dun19] Arsenii Dunaev. "A Text-to-Speech system based on Deep Neural Networks". PhD thesis. Informatics Institute, 2019.

[Dut+96] Thierry Dutoit et al. "The MBROLA project: Towards a set of high quality speech synthesizers free of use for non commercial purposes". In: *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96.* Vol. 3. IEEE. 1996, pp. 1393–1396.

[Gar19] Dalya Gartzman. *Getting to Know the Mel Spectrogram.* `https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0`. Accessed: 2021-10-02. 2019.

[GHW11] Kyle Gorman, Jonathan Howell, and Michael Wagner. "Prosodylab-aligner: A tool for forced alignment of laboratory speech". In: *Canadian Acoustics* 39.3 (2011), pp. 192–193.

[GL84] Daniel Griffin and Jae Lim. "Signal estimation from modified short-time Fourier transform". In: *IEEE Transactions on acoustics, speech, and signal processing* 32.2 (1984), pp. 236–243.

[Goo+14] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[Gro74] M Grosswirth. "Leachim The teaching Robot". In: *Datamation* 20.8 (1974), pp. 64–67.

[Hay+19] Tomoki Hayashi et al. "Pre-Trained Text Embeddings for Enhanced Text-to-Speech Synthesis." In: *INTERSPEECH.* 2019, pp. 4430–4434.

[HB96] Andrew J Hunt and Alan W Black. "Unit selection in a concatenative speech synthesis system using a large speech database". In: *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*. Vol. 1. IEEE. 1996, pp. 373–376.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "LSTM can solve hard long time lag problems". In: *Advances in neural information processing systems* (1997), pp. 473–479.

[HT15] Keikichi Hirose and Jianhua Tao. *Speech Prosody in Speech Synthesis: Modeling and generation of prosody for high quality and flexible speech synthesis*. Springer, 2015.

[IJ17] Keith Ito and Linda Johnson. *The LJ Speech Dataset*. `https://keithito.com/LJ-Speech-Dataset/`. 2017.

[KB04] John Kominek and Alan W Black. "The CMU Arctic speech databases". In: *Fifth ISCA workshop on speech synthesis*. 2004.

[Kea14] Jodi Kearns. "Librivox: Free public domain audiobooks". In: *Reference Reviews* (2014).

[Ken+19] Tom Kenter et al. "CHiVE: Varying prosody in speech synthesis with a linguistically driven dynamic hierarchical conditional variational network". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3331–3340.

[KKB20] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. "HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis". In: *Advances in Neural Information Processing Systems* 33 (2020).

[Kla01] Aldebaro Klautau. "ARPABET and the TIMIT alphabet". In: *URL: https://web.archive.org/web/20160603180727/http://www.laps.ufpa.br/aldebaro/papers/ak_arpabet01.pdf* (2001).

[Kli+19] Viacheslav Klimkov et al. "Fine-grained robust prosody transfer for single-speaker neural text-to-speech". In: *arXiv preprint arXiv:1907.02479* (2019).

[Kum+19] Kundan Kumar et al. "MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis". In: *Advances in Neural Information Processing Systems* 32 (2019).

[LeC+98] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[Li+19] Naihan Li et al. "Neural speech synthesis with transformer network". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 6706–6713.

[Lik32] Rensis Likert. "A technique for the measurement of attitudes." In: *Archives of psychology* (1932).

[McA+17] Michael McAuliffe et al. "Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi." In: *Interspeech*. Vol. 2017. 2017, pp. 498–502.

[MP69] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. 1969.

[Oor+16] Aaron van den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[Pan+15] Vassil Panayotov et al. "Librispeech: an asr corpus based on public domain audio books". In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 5206–5210.

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[PM19] Kyubyong Park and Thomas Mulc. "CSS10: A Collection of Single Speaker Speech Datasets for 10 Languages". In: *Interspeech* (2019).

[PVC19] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. "Waveglow: A flow-based generative network for speech synthesis". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 3617–3621.

[Ran20] Kartikeya Rana. *Pooling Layer - Short and Simple*. https://ai.plainenglish.io/pooling-layer-beginner-to-intermediate-fa0dbdce80eb. Accessed: 2021-10-20. 2020.

[Ren+19] Yi Ren et al. "FastSpeech: fast, robust and controllable text to speech". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 2019, pp. 3171–3180.

[Ren+20] Yi Ren et al. "FastSpeech 2: Fast and High-Quality End-to-End Text to Speech". In: *International Conference on Learning Representations*. 2020.

[RHW85] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[Rob20] Leland Roberts. *Understanding the Mel Spectrogram*. https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53. Accessed: 2021-10-01. 2020.

[Ros57] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton*. Report, Project PARA, Cornell Aeronautical Laboratory, 1957.

[RRC20] Tuomo Raitio, Ramya Rasipuram, and Dan Castellani. "Controllable neural text-to-speech synthesis using intuitive prosodic features". In: *arXiv preprint arXiv:2009.06775* (2020).

[RRT05] Philip Rubin, Gordon Ramsay, and Mark Tiede. "The History of Articulatory Synthesis at Haskins Laboratories". In: *Auditory-Visual Speech Processing* (2005).

[Sax17] Utkarsh Saxena. "Speech synthesis techniques using deep neural networks". In: *blog post* (2017).

[Sch02] Tanja Schultz. "Globalphone: a multilingual speech and text database developed at karlsruhe university". In: *Seventh International Conference on Spoken Language Processing*. 2002.

[Sha17] Sagar Sharma. *What the hell is perceptron*. `https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53`. Accessed: 2021-09-01. 2017.

[She+18] Jonathan Shen et al. "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 4779–4783.

[Shu19] Lavanya Shukla. *Designing Your Neural Networks*. `https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed`. Accessed: 2021-10-15. 2019.

[Ske+18] RJ Skerry-Ryan et al. "Towards end-to-end prosody transfer for expressive speech synthesis with tacotron". In: *international conference on machine learning*. PMLR. 2018, pp. 4693–4702.

[Smi10] Julius Orion Smith. *Physical audio signal processing: For virtual musical instruments and audio effects*. W3K publishing, 2010.

[Sun+13] Antti Santeri Suni et al. "Wavelets for intonation modeling in HMM speech synthesis". In: *8th ISCA Workshop on Speech Synthesis, Proceedings, Barcelona, August 31-September 2, 2013*. ISCA. 2013.

[Sun+20] Guangzhi Sun et al. "Fully-hierarchical fine-grained prosody modeling for interpretable speech synthesis". In: *ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2020, pp. 6264–6268.

[SVN37] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. "A scale for the measurement of the psychological magnitude pitch". In: *The journal of the acoustical society of america* 8.3 (1937), pp. 185–190.

[TCW20] Yun-Cheng Tsai, Jun-Hao Chen, and Jun-Jie Wang. "Predict forex trend via convolutional neural networks". In: *Journal of Intelligent Systems* 29.1 (2020), pp. 941–958.

[TI97] Paul Taylor and Amy Isard. "SSML: A speech synthesis markup language". In: *Speech communication* 21.1-2 (1997), pp. 123–133.

[Tu+19] Tao Tu et al. "End-to-end text-to-speech for low-resource languages by cross-lingual transfer learning". In: *arXiv preprint arXiv:1904.06508* (2019).

[Vas+17] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[Wai+89] Alex Waibel et al. “Phoneme recognition using time-delay neural networks”. In: *IEEE transactions on acoustics, speech, and signal processing* 37.3 (1989), pp. 328–339.

[Wai87] Alex Waibel. “Phoneme recognition using time-delay neural networks (Technical Report TR-I-0006)”. In: *Japan: Advanced Telecommunications Research Institute* (1987).

[Wan+17] Yuxuan Wang et al. “Tacotron: Towards end-to-end speech synthesis”. In: *arXiv preprint arXiv:1703.10135* (2017).

[Wan+18] Yuxuan Wang et al. “Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5180–5189.

A. Alignment Data Example

Alignment data generated by MFA [McA+17] are stored in plain text using the following format. The example below contains alignment data on word-level for the text *"in being comparatively* modern". As shown, the time interval during which each word is heard is stored in the data, with *xmin* and *xmax* denoting the respective start time and end time in seconds. Phoneme-level alignment data is stored equivalently.

```
item []:
    item [1]:
        class = "IntervalTier"
        name = "words"
        xmin = 0
        xmax = 1.8995
        intervals: size = 5
        intervals [1]:
            xmin = 0
            xmax = 0.14
            text = "in"
        intervals [2]:
            xmin = 0.14
            xmax = 0.41
            text = "being"
        intervals [3]:
            xmin = 0.41
            xmax = 1.27
            text = "comparatively"
        intervals [4]:
            xmin = 1.27
            xmax = 1.83
            text = "modern"
        intervals [5]:
            xmin = 1.83
            xmax = 1.8995
            text = ""
```

B. English Evaluation Sentences

Below, the sentences used in the evaluation of the English model are listed, preceded by their respective ID. The words that were emphasized during evaluation of emphasis perceptibility are written in capital letters.

```
GT 1: He was generally supposed to be a surgeon.

GT 2: It should be remembered that in every big job there are some imperfections.

GT 3: The court itself can best undo what the court has done.

GT 4: This was at least in part because of the close physical confines in
      which some of the work had to be done.

GT 5: As the motorcade approached elm street.


S 1(E): A Dallas police car and several motorcycles at THE REAR kept
        the motorcade together.

S 2(E): A thorough inspection would have involved washing and cleansing the back,
        and this is NOT practical in treating an acutely injured patient.

S 3(E): SOME rooms remained quite empty and unoccupied,
        while others were full to overflowing.

S 4(E): And had certain unattractive features, including a LOW ceiling
        with exposed conduits and beams.

S 5(E): Decent clothing and bedding, and a diet SUFFICIENT to support him.

S 6(E): In the DEPLORABLE situation in which many of them now are.

S 7(E): Every prison containing FEMALE prisoners was to have a matron
        who was to reside constantly in the prison.

S 8(E): Partly because of the greater speed and comfort of travel AND partly
        because of the greater demands made on the president.
```

C. German Evaluation Sentences

Below, the sentences used in the evaluation of the German model are listed, preceded by their respective ID. The words that were emphasized during evaluation of emphasis perceptibility are written in capital letters.

GT 1: Flüchtig hingesehen, erschienen die weißen Linien wie ein Flug Wildgänse,
die in einer Landschaft über Baum und Hügel hinflogen.

GT 2: Meister Floh hatte während dieser Zeit seine natürliche Gestalt angenommen
und war spurlos verschwunden.

GT 3: Und dieser schwarze Strick, der die Dicke eines Männerarms hat,
wird noch heute in einer Lacktonne im Tempel von Kioto aufbewahrt.

GT 4: Gehab dich so lange wohl, meine herzgeliebte Ulla. Bald bin ich wieder hier.

GT 5: Die Frau starrte die schlafende Katze an, aber die gemalte Katze
hielt die Augen geschlossen und blinzelte nicht.

S 1(E): Und HANAKES Gesicht wurde wieder höflich und freundlich und unbeschrieben
wie eine weiße Eierschale.

S 2(E): Das ist das Bild, das ich HIER malen will.

S 3(E): SEHR weise gesprochen, sagte die alte Dame.

S 4(E): Die ihre Blüten und Blätter aus der tiefsten Tiefe emporrankten und
auf ANMUTIGE Weise ineinander verschlangen.

S 5(E): Der andere aber musste SEINE Schulstellung aufgeben und wurde Polizist.

S 6(E): Ich höre IHN nicht.

S 7(E): Entsetzt ließ er die RUDER ins Wasser fallen.

S 8(E): Ich habe zwar nie einen solchen Baum gesehen, ich KENNE aber seine
Rindenschrift wie die Linien meiner Hand.