UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Oleh Matsuk

Multi-speaker Text-to-speech Synthesis in Estonian

Master's Thesis (30 ECTS)

Supervisor: Mark Fišel, PhD

Tartu 2021

Multi-speaker Text-to-speech Synthesis in Estonian

Abstract:
Text-to-speech synthesis is a challenging problem, but in recent years it has obtained convincing solutions in the form of neural network models. Specialized model architectures have been proposed to affect speaker identity features of the synthesized speech without training separate models, thus reducing the requirements for data volume and training time. In this work we implement and train a recently proposed neural architecture with limited amount of Estonian speech data to obtain a model capable of multi-speaker text-to-speech synthesis. Consequently, we evaluate the overall quality of the synthesized speech and the model's ability to assume speaker identity features for speakers both seen and unseen in training. We evaluate and compare the results between multiple models trained with different sets of training data.

Keywords: text-to-speech, multi-speaker, neural networks, Tacotron 2, speaker imitation.

CERCS: P176 Artificial intelligence

Mitmehäälne Kõnesüntees Eesti Keeles

Lühikokkuvõte:
Kõnesüntees on keeruline probleem, millele on viimastel aastatel arendatud mitmeid veenvaid lahendusi tehisnärvivõrkude mudelite abil. On välja pakutud spetsiaalseid mudelite arhitektuure, mis mõjutavad sünteetilise kõneleja tämbrit ilma eraldiseisvaid mudeleid treenimata, vähendades andmestiku suuruse ja treeningaja nõudeid. Käesolevas teoses implementeeritakse ja treenitakse hiljuti väljapakutud tehisnärvivõrgu arhitektuuril põhinev mudel kasutades limiteeritud eestikeelset andmestikku, mis on võimeline genereerima mitmeisikulist kõnesünteesi. Sellele järgnevalt hinnatakse kõnesünteesi kvaliteeti ja mudeli kohanemisvõimet erinevate andmestikus esinenud ja mitteesinenud kõnelevate isikute jaoks. Seejärel hinnatakse ja võrreldakse tulemusi erinevatel andmestikel treenitud mudelite vahel.

Võtmesõnad: kõnesüntees, mitmehäälne, tehisnärvivõrgud, Tacotron 2, kõneleja imiteerimine.

CERCS: P176 Tehisintellekt

Contents

1 Introduction

Text-to-speech synthesis is the process of generating audio that imitates human speech from a provided textual annotation. Text-to-speech systems have many practical applications, mainly as accessibility tools for people with limited abilities, as well as in the areas of education, telecommunications, multimedia, and others. Nowadays such systems are quite common and easily available, although the options may be limited depending on the choice of language. In addition, there is still room for improvement in terms of how natural the synthesized speech sounds, capturing the right stress and intonation, etc. One such challenge is the task of multi-speaker text-to-speech synthesis, which introduces an additional requirement of controlling the speaker identity, i.e., the voice, of the synthesized speech.

Traditionally text-to-speech systems have been built concatenatively, by combining prerecorded phoneme combinations. Unsurprisingly such methods produce speech of limited naturalness and the resulting models cannot be easily modified or reused. Recently proposed neural network models have been able to achieve state-of-the-art results in speech synthesis, outperforming previous methods [van den Oord et al., 2016, Arik et al., 2017b, Wang et al., 2017]. Machine learning techniques, especially deep learning, have proven to be invaluable when applied to complex tasks which cannot be convincingly solved with traditional methods that rely on domain knowledge and algorithmic approach. In such cases the data-driven approach is preferable since it can greatly alleviate the task-specific complexity. In the task of speech synthesis deep learning also has the advantage of relatively soft data requirements compared to the traditional methods. Modern text-to-speech neural networks can be trained on annotated speech data obtained from various sources and collected for different purposes, as long as the quality of the audio and annotations as well as the volume of the data are sufficient. In addition, such systems tend to be more robust when facing unfamiliar input and can be parameterized to customize the synthesized speech in various ways.

There are many features of the human speech that can affect how subjectively natural it sounds depending on the context and expectations of the human listener. Some of the most important of these are *prosody* (patterns of stress and intonation) and *speaker identity* (features by which utterances of different speakers can be distinguished). For this reason controlling these features within the synthesized speech has been a research focus in the field of neural text-to-speech, with some systems already achieving quite promising results [Arik et al., 2017a, Skerry-Ryan et al., 2018, Wang et al., 2018, Jia et al., 2018]. The mentioned methods eliminate the need to train separate neural networks with different properties hard-wired into them. Instead, the speech synthesis is controlled with additional input parameters, thus greatly increasing the scalability and reusability of the trained models.

Such deep learning frameworks are freely applicable to different target languages, the

main obstacle in applying them to new languages being the requirement for considerable amount of training data in the language of choice. With multi-speaker synthesis this requirement becomes more severe since sufficient amount of examples must be provided for a multitude of different identified speakers.

In this work we replicate a state-of-the-art deep learning framework for multi-speaker text-to-speech synthesis. We reuse an existing implementation of the basic text-to-speech model from [Shen et al., 2017] and extend it with our own implementation of multi-speaker synthesis following the description from [Jia et al., 2018]. We perform multi-stage training of the neural networks with the limited amount of speech data in the Estonian language available to us, which includes a dedicated dataset for text-to-speech synthesis and a supplementary "noisy" dataset compiled for automatic speech recognition. We consequently study the overall quality and naturalness of the synthesized speech achieved with the main text-to-speech data and whether it is improved or affected by the additional speech recognition training examples. We similarly investigate the success of speaker imitation in a multi-speaker synthesis setting and its correlation with the relevant properties of the training data such as number of speakers, amount of data per speaker and quality of the sample audio. To assess the synthesized speech in these subjective metrics we facilitate and conduct manual evaluation with the help of independent native speakers using a specially developed web-based environment. Finally, we present and interpret the obtained results.

In the second chapter we will list relevant text-to-speech neural network architectures and describe in detail the state-of-the-art multi-speaker text-to-speech model from [Shen et al., 2017] and [Jia et al., 2018]. In the third chapter we will describe the utilized data, the implementation details of the model, and the training process. In the fourth chapter we will introduce the human evaluation testing conditions, report and analyze the results.

2 Model Architecture

2.1 Neural Text-to-speech Models

WaveNet Text-to-speech neural networks have been evolving quite rapidly in the recent years. One of the earliest successful models that proved the viability of text-to-speech generation with neural networks was Google DeepMind's WaveNet [van den Oord et al., 2016]. It was able to significantly outperform previous text-to-speech techniques in subjective naturalness while simultaneously relaxing the previous requirements for domain knowledge and manual feature engineering.

WaveNet's main contribution and novelty over previous research is attributed to tackling the complexity of fully probabilistic generation of audio signal. In computing, audio is stored and processed as a one-dimensional sequence of values corresponding to the amplitudes of the signal waveform at each sampling instance (see Figure 1). To preserve fidelity of the sound continuous audio signal must be sampled at a high rate (for example, modern audio formats like MP3 require a sampling rate of 44100 Hz, i.e., 44100 discrete samples per second).

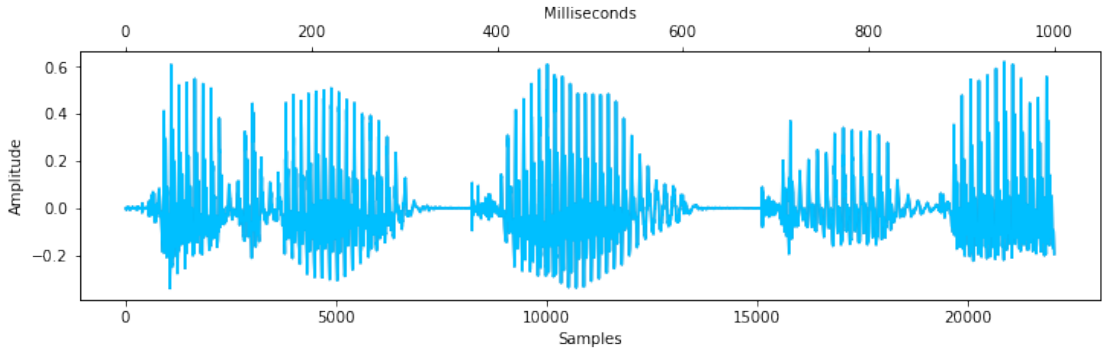

Figure 1. Audio waveform at 22050 sampling rate.

Traditional methods for handling sequential data in deep learning involve *recurrent neural networks* (RNNs). RNNs process sequences element by element: at each step a recurrent layer performs a computation based on the current input element and an output from the previous step. From the probabilistic point of view, each RNN output prediction $\mathbf{y}_{t+1}$ is conditioned on all the previously seen elements in the input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_t$, which is crucial for learning dependencies between the sequence elements. In RNNs this contextual information is maintained by passing an additional output from each processing step to the next. This process is called *unrolling* (see Figure 2) and it results in the computational graph of the RNN to grow in size proportionally to the length of the processed sequence. Standard RNNs tend to struggle with handling the

contextual information in lengthy sequences since the updates in the neural network which facilitate learning cannot be signalled over too many computational steps, thus preventing the model from learning based on non-recent context (known as the *vanishing gradient problem*). Because of this it is more common to use enhanced RNN layers, such as *long short-term memory* (LSTM) [Hochreiter and Schmidhuber, 1997], which alleviates the issue by allowing the context (usually called *memory*) to pass through the sequence with minimum number of operations. However, even these dedicated RNNs do not scale well enough to handle the sequence lengths typical for audio signal data.

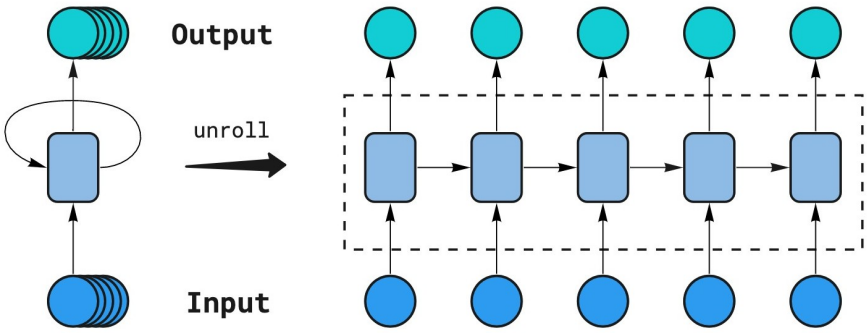

Figure 2. Recurrent neural network unrolling over the input sequence.

WaveNet tackles these issues by replacing the recurrent layers with its newly introduced *dilated causal convolution* layers (see Figure 3). *Convolutional* layers apply a special type of mapping between specific elements of the input and the corresponding computed output elements based on their relative location (this is called a *convolution*). For example, in image processing, where convolutional neural networks (CNNs) are most widely used, convolutions allows processing groups of collocated pixels to extract higher-level features of the image. Similarly, in natural language processing CNNs can be used to extract features from adjacent words or characters. Dilated causal convolutions extend this idea with the notion of *dilation*, i.e., skipping over a fixed number of elements in the convolution, thus increasing the size of the convolutional window. WaveNet applies stacks of such convolutions over the sequence, allowing to establish dependencies between the output elements and their predecessors in the input sequence, such that each layer of convolution exponentially increases the number of elements considered in the prediction.

Deep Voice While providing a convincing solution for the audio synthesis itself, WaveNet still has a number of flaws which limit its practical application. First of all, WaveNet's synthesis is conditioned on specific linguistic and audio features which are extracted from the input using existing text-to-speech systems. This means that

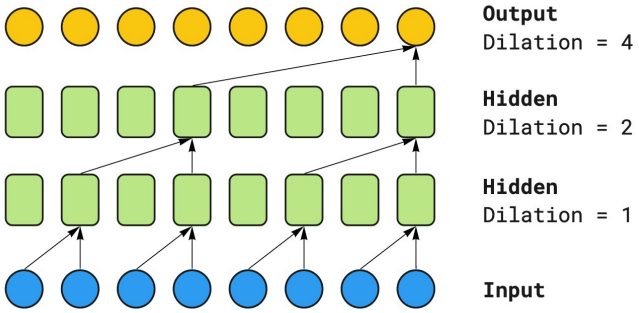

Figure 3. WaveNet's dilated causal convolutions. With each convolutional layer the dilation is doubled, which results in exponential growth of convolutional window proportional to the number of stacked layers.

WaveNet cannot be trained directly on annotated speech and requires knowledge and usage of additional dependencies. Secondly, the original WaveNet's per-sample synthesis tends to be too slow and cannot be used in real-time applications.

Deep Voice [Arik et al., 2017b], introduced by the Baidu Silicon Valley Artificial Intelligence Lab, aimed to solve these exact issues. Similar to previous text-to-speech methods, Deep Voice requires a number of explicit text-to-speech-specific features for conditioning the audio synthesis. However, rather than expecting them to be provided beforehand with external dependencies, Deep Voice includes specialized fully neural modules capable of predicting the features as part of its architecture. Specifically this includes a grapheme-to-phoneme module, a segmentation module, a phoneme duration module and a fundamental frequency module. Together with the actual audio synthesis module, for which Deep Voice uses a modified version of WaveNet, they form a complete end-to-end text-to-speech system (see Figure 4).

The introduction of Deep Voice signified a considerable leap in the field of neural text-to-speech since it was now possible to train a text-to-speech system from scratch with nothing but a set of annotated speech samples. The subsequent improvements over the original Deep Voice came in the form of DeepVoice 2 [Arik et al., 2017a], which focused on extending the model with multi-speaker synthesis, and DeepVoice 3 [Ping et al., 2017], which largely reworked and simplified the original architecture.

2.2 Tacotron 2

As the base model for basic text-to-speech synthesis in this work we utilize the currently state-of-the-art *Tacotron 2* [Shen et al., 2017].

The original *Tacotron* [Wang et al., 2017] was presented shortly after the release of Deep Voice by a team of researchers from Google. Same as Deep Voice, Tacotron allows

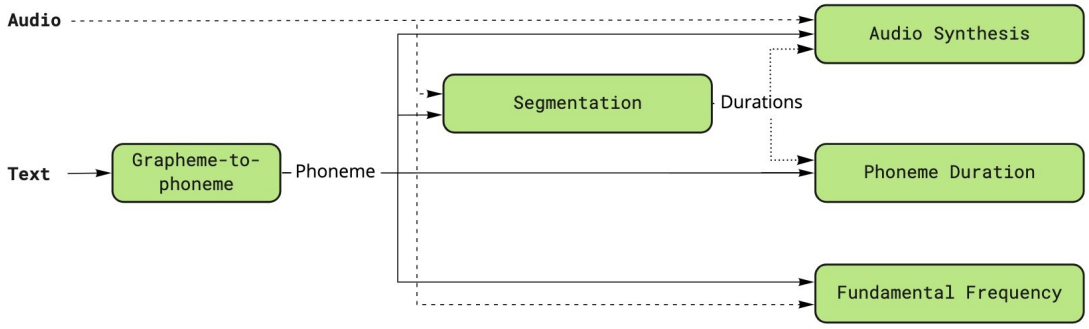

Figure 4. High-level Deep Voice architecture (data flow between modules during training).

for end-to-end text-to-speech synthesis directly from annotated speech with no additional dependencies. Unlike Deep Voice, Tacotron aims to reduce the amount of separate modules since, according to its developers, the errors in separately trained modules tend to compound and such a complex architecture is harder to modify and extend. After a relatively short time the model was improved with the introduction of Tacotron 2 in [Shen et al., 2017]. While both systems are fairly similar, Tacotron 2 simplifies over the original architecture by replacing the custom layers introduced along with the model in [Wang et al., 2017], such as a recurrent *CBHG* layer, with the commonly known and widely used LSTM.

Sequence-to-sequence At the core of Tacotron 2 lies a recurrent character-to-spectrogram prediction network based on the *sequence-to-sequence* architecture from [Sutskever et al., 2014], also known as *encoder-decoder*. Sequence-to-sequence is a popular high-level RNN architecture, especially widely used in the area of natural language processing. As the name suggests, the architecture is applicable for learning data mappings where both inputs and outputs take the shape of sequences.

A sequence-to-sequence neural network consists of two recurrent modules, an *encoder* and a *decoder*, which perform sequence processing in two distinct steps (see Figure 5). First, the recurrent encoder network fully consumes the input sequence $\{\mathbf{x}\}_{i=1}^{T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ and outputs a fixed length context embedding $\mathbf{c}$ (usually the memory of the recurrent cell after the last encoder timestep T). Secondly, the decoder starts generating the output sequence $\{\mathbf{y}\}_{j=1}^{S} = \{\mathbf{y}_1, \ldots, \mathbf{y}_S\}$ element by element. Each decoder output $\mathbf{y}_j$ is generated as the highest probability prediction conditioned on all the previous outputs and the encoding of the input sequence

$$\underset{\mathbf{y} \in \mathbb{Y}}{\operatorname{argmax}} P(\mathbf{y}_j \mid \{\mathbf{y}_1, \ldots, \mathbf{y}_{j-1}\}, \mathbf{c})$$

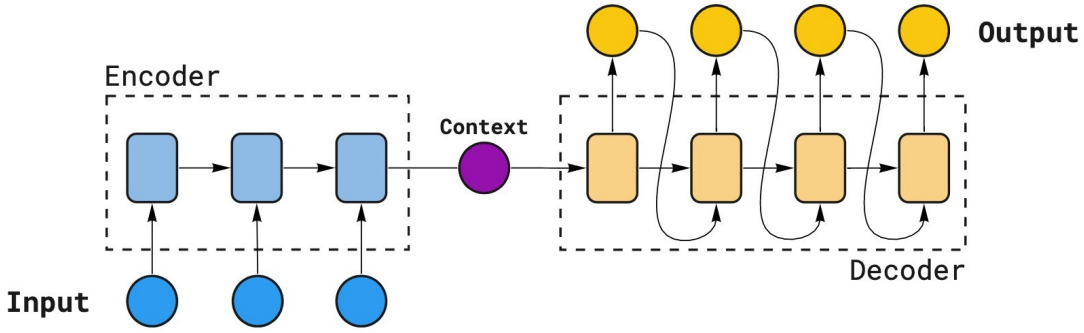

Figure 5. Sequence-to-sequence architecture.

On the network level, a decoder output $\mathbf{y}_j$ is computed by the recurrent cell at decoder timestep j as a function of the previously predicted output element $\mathbf{y}_{j-1}$ and the current cell memory, initialized with the context $\mathbf{c}$ on the first decoder step, i.e.,

$$\mathbf{y}_j = f(\mathbf{y}_{j-1}, \mathbf{c})$$

During training, instead of the generated $\mathbf{y}_{j-1}$ the model is given the ground truth element $\hat{\mathbf{y}}_{j-1}$ to facilitate learning by avoiding the compounding of errors in case of an incorrectly predicted element (this technique is called *teacher forcing*).

The advantage of this architecture over simply generating the target sequence simultaneously with consuming the input is that it does not impose a dependency between the lengths of the input and the output sequences: the encoding output dimensionality is fixed and independent from the input length, while the decoding process can continue until the decoder network itself marks the end of the generated sequence with a special output value.

Attention The fixed size of the context in the sequence-to-sequence architecture is also its well recognized limitation. Since the context represents an encoding of the entire input sequence and should contain sufficient information for the decoding process, it may not be scalable enough to support especially lengthy inputs. To mitigate this issue the sequence-to-sequence architecture is enhanced with an *attention* mechanism, originally introduced in [Bahdanau et al., 2014].

In this architecture (see Figure 6) the encoder network is not constrained to output only a limited context vector $\mathbf{c}$. Instead it produces an encoding sequence $\{\mathbf{e}\}_{i=1}^{T} = \{\mathbf{e}_1, \ldots, \mathbf{e}_T\}$ of the same length as the input $\{\mathbf{x}\}_{i=1}^{T}$ (commonly the outputs of the recurrent encoder cell after each input element $\mathbf{x}_i$). At each decoder timestep j the encodings $\{\mathbf{e}\}_{i=1}^{T}$ are used to evaluate the *alignment* $\alpha_{j,i}$ between the current decoder

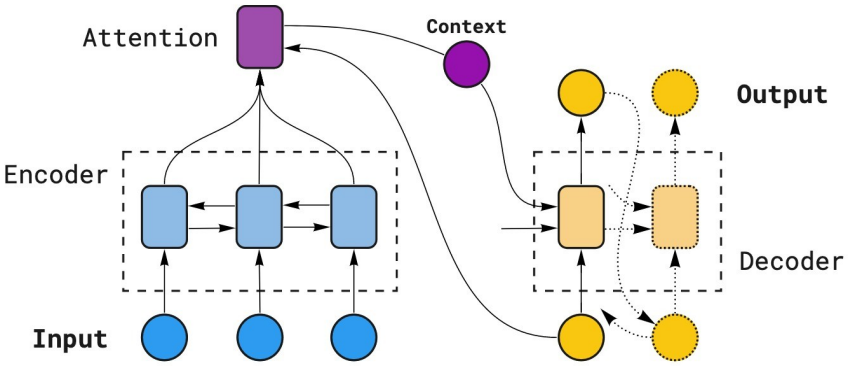

Figure 6. Sequence-to-sequence with attention.

input $\mathbf{y}_{j-1}$ and the elements of the input sequence $\{\mathbf{x}\}_{i=1}^{T}$ corresponding to the encodings

$$\alpha_{j,i} = a(\mathbf{y}_{j-1}, \mathbf{e}_i)$$

Now instead of relying on the previously constant context $\mathbf{c}$ in the decoder, the attention mechanism computes a timestep-specific attention context $\mathbf{c}_j$ as the sum of the encoder outputs $\{\mathbf{e}\}_{i=1}^{T}$ weighted by the softmax-normalized alignment scores

$$\mathbf{c}_j = \sum_{i=1}^{T} \alpha_{j,i}\mathbf{e}_i$$
$$\mathbf{y}_j = g(\mathbf{y}_{j-1}, \mathbf{c}_j)$$

Intuitively, the alignment scores represent how relevant each input element is to the currently produced output. In the example of text-to-speech, the network learns to predict which textual characters in the input sentence should be considered the most when producing a specific segment of the audio. Thus, the expected behaviour for the attention mechanism is to learn a mostly diagonal alignment between the input and the output sequences (see Figure 7), at least when applied to the tasks like machine translation or text-to-speech synthesis.

As the encoder, [Bahdanau et al., 2014] also suggests using a bidirectional LSTM (BiLSTM) which consists of two LSTM layers stacked together consuming the input sequence in the normal and reversed order respectively. In this architecture the encodings are formed as a concatenation of the outputs of the forwards and backwards RNNs $\mathbf{e}_i = [\overrightarrow{\mathbf{e}}_i; \overleftarrow{\mathbf{e}}_i]$. This allows the encodings to include extra contextual information from processing both the preceding and the following elements of the sequence.

The attention mechanism allows the decoder to access the entire input sequence during decoding, which removes the previously mentioned length constraint, while simultaneously training the model to focus the attention on the relevant parts of the input data throughout the synthesis process.

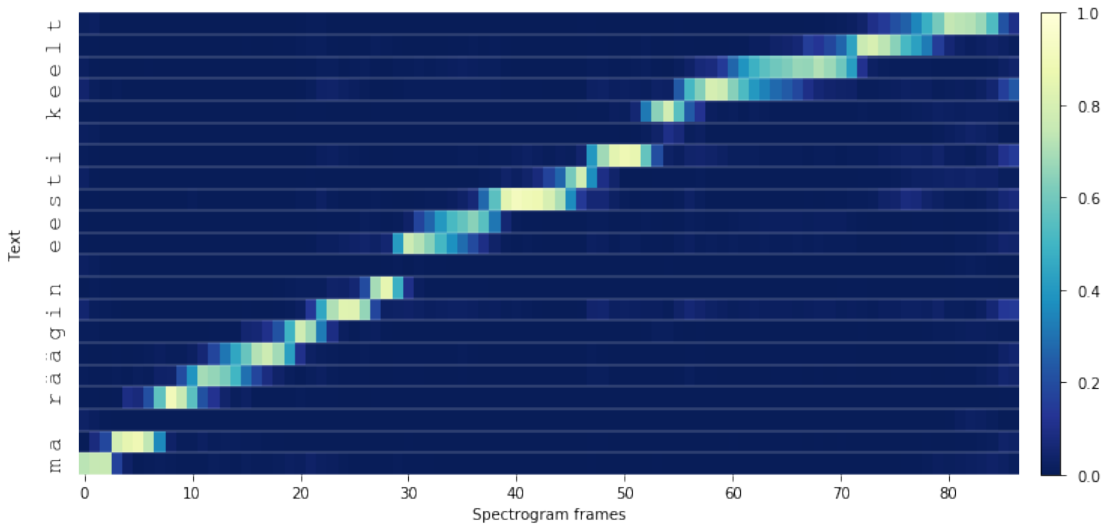

Figure 7. Attention alignment during text-to-speech synthesis.

Mel spectrograms As previously mentioned, directly decoding text into audio signal is not feasible in practice, which is why analogous text-to-speech models tend to split it into smaller steps, enrich the decoder with various intermediary features, etc.

Another approach, utilized by the Tacotron model, is to instead use a more compact audio representation through *mel-scale spectrograms*. The fundamental idea behind *spectrograms* is based on the fact that a signal waveform is comprised of a set of simple signals of different frequencies. A spectrogram decomposes a signal into the amplitude values of varying signal frequencies changing over time (see Figure 8). This can be achieved using the *short-time Fourier transform* operation (STFT). STFT applies a sliding window to the continuous signal amplitude function and performs Fourier transform to determine the strengths of the composing frequencies at each window (also called *frame*).

When dealing with audio signals intended for human perception it is also useful to consider that humans do not distinguish changes in frequencies at different frequency heights at the same rate. In practice, at higher frequencies humans report signals to be more similar than at lower ones even when they differ by the same value in the linear frequency scale. This property is encapsulated by the *mel scale*, which applies a

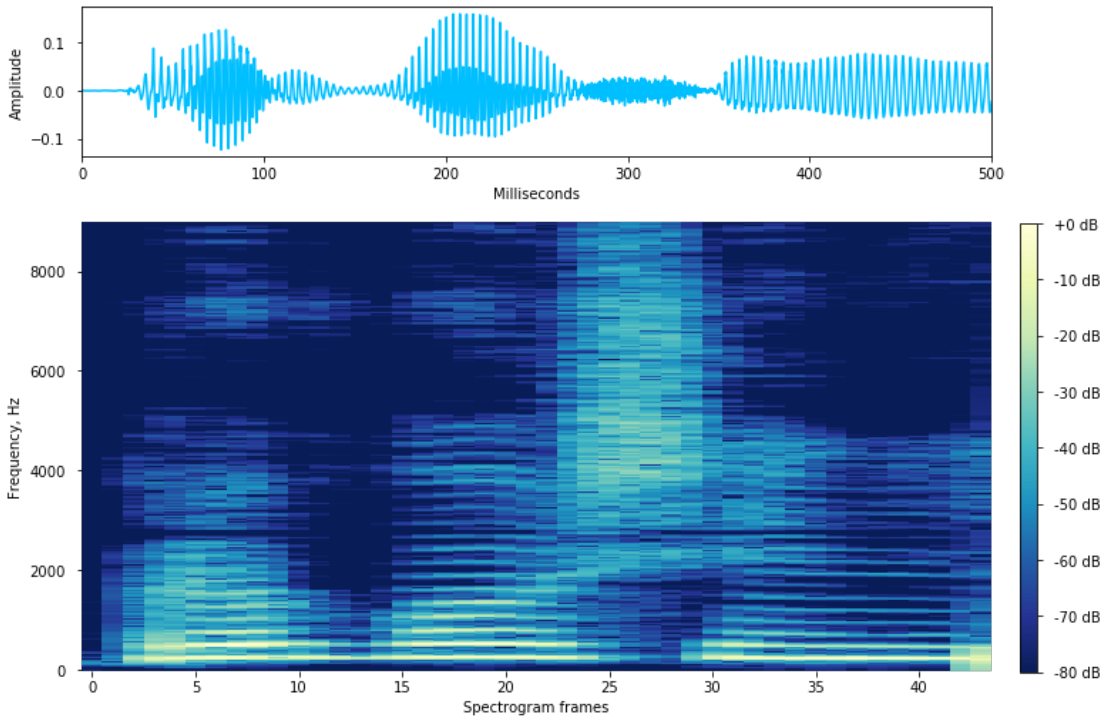

Figure 8. Spectrogram decomposition of a waveform.

non-linear transformation of the frequency scale that better matches the sensitivity of human perception (see Figure 9).

With the frequency axis grouped into discrete bins (known as *channels*), the complete mel-scale spectrogram representation of an audio signal forms a matrix $\mathbf{M}$, where $\mathbf{M}_{t,c}$ is the strength of the underlying signal at the mel frequency channel c during the time frame t (see Figure 10).

The original waveform can be reconstructed from the spectrogram with various techniques, such as the *Griffin-Lim* reconstruction algorithm or by a dedicated neural network. At the same time it is much more compact in the temporal axis, which makes it more suitable for sequence generation, since the temporal dependencies in the sequence are much easier to preserve and learn. For example, a waveform sampled at 22050 Hz converted into a spectrogram with 1024 samples per frame and 256 hop length has approximately 86 spectrogram frames per second. Assuming an 80-channel mel scale, each second of audio is represented as an 86×80 spectrogram matrix, as opposed to a sequence of 22050 waveform samples, which amounts to overall data compression factor of around 3.2 (while sequence length is reduced by a factor of 256).

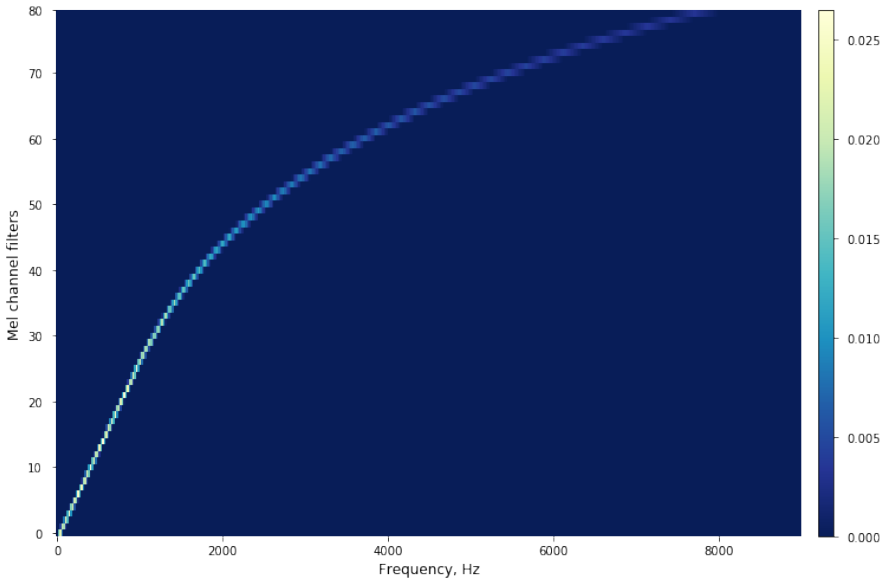

Figure 9. 80-channel mel-scale transform matrix with the maximum frequency capped at 8000 Hz.

End-to-end text-to-speech model The full Tacotron 2 model from [Shen et al., 2017] consists of two semi-independent components (see Figure 11): a recurrent sequence-to-sequence with attention prediction network (a variation of the previously described neural architecture) which predicts mel-scale spectrograms directly from sequences of textual characters, and a waveform generator conditioned on the predicted spectrogram frames (called *vocoder*).

The prediction network expects a sequence of textual characters as its input. The encoder transforms them into embeddings based on an embeddings lookup layer, passes them through multiple convolutional layers with small convolution window (the original length of the sequence is preserved), and finally applies a bidirectional recurrent LSTM. The outputs of the LSTM are used as the encodings for the attention mechanism. Tacotron 2 uses a special *location-sensitive* attention [Chorowski et al., 2015], which stores the alignment scores produced on previous decoder steps (this allows the mechanism to maintain a history of assigned attention resulting in more uniform distribution of attention across the input sequence). During decoding, the attention mechanism calculates a compact attention context based on the encoder outputs, the current decoder input, as well as the location information stored in the attention module. The decoder consists of a preprocessing network (*prenet*), made of multiple linear transformations, and several recurrent LSTM cells. Each decoder cell's state is enriched with the computed attention context. The decoder outputs are processed with an additional convolutional postprocess-

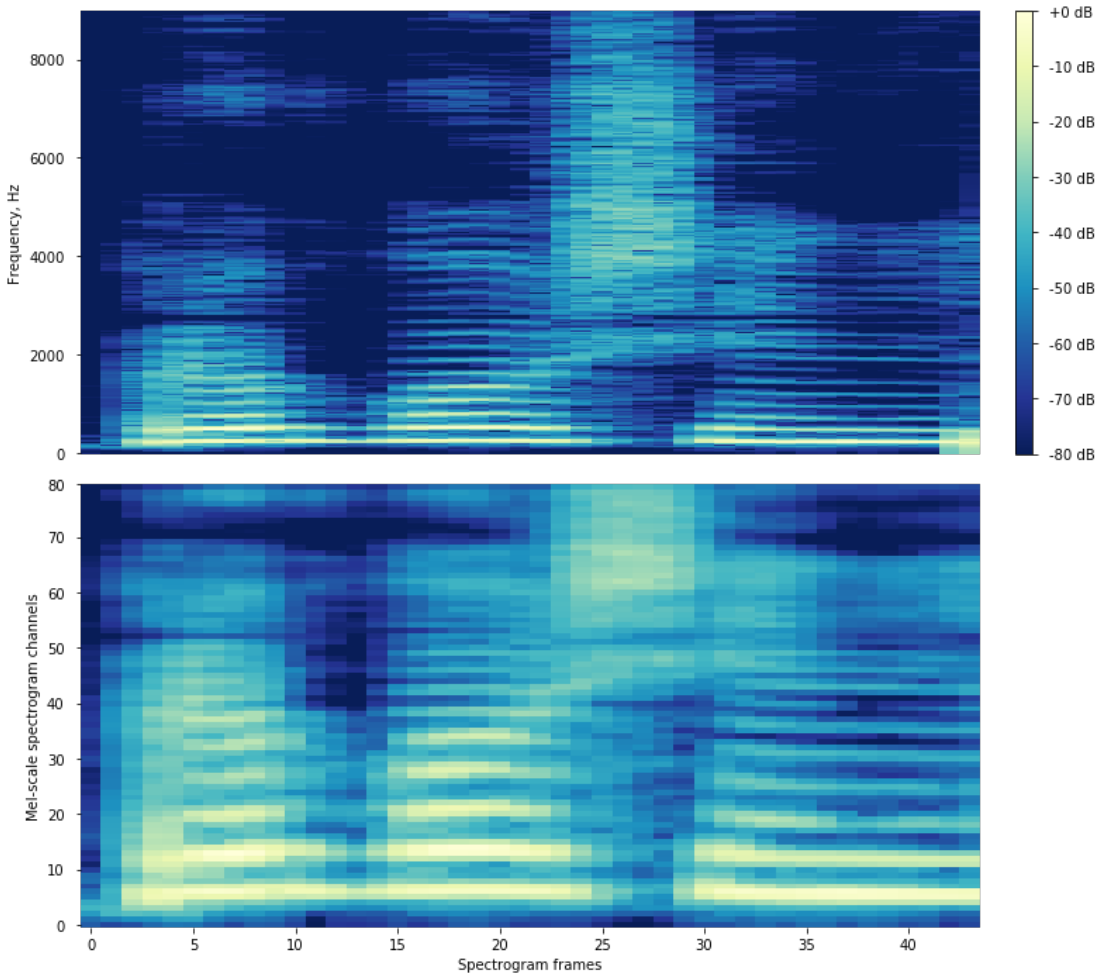

Figure 10. Spectrogram in linear scale and 80-channel mel scale.

ing network (*postnet*), which results in the final spectrogram predictions. In addition, the decoder produces a separate *gate* value, which signifies whether the decoding should end at this step or continue. The character-to-spectrogram prediction network is trained on the combined spectrogram and gate prediction loss.

In the original Tacotron the vocoder is implemented with the Griffin-Lim algorithm, while in Tactotron 2 it is replaced with a modified WaveNet making the model fully neural. Such vocoder can be trained independently from the prediction network, which further simplifies re-training the text-to-speech system under new conditions, e.g., with a different target language.

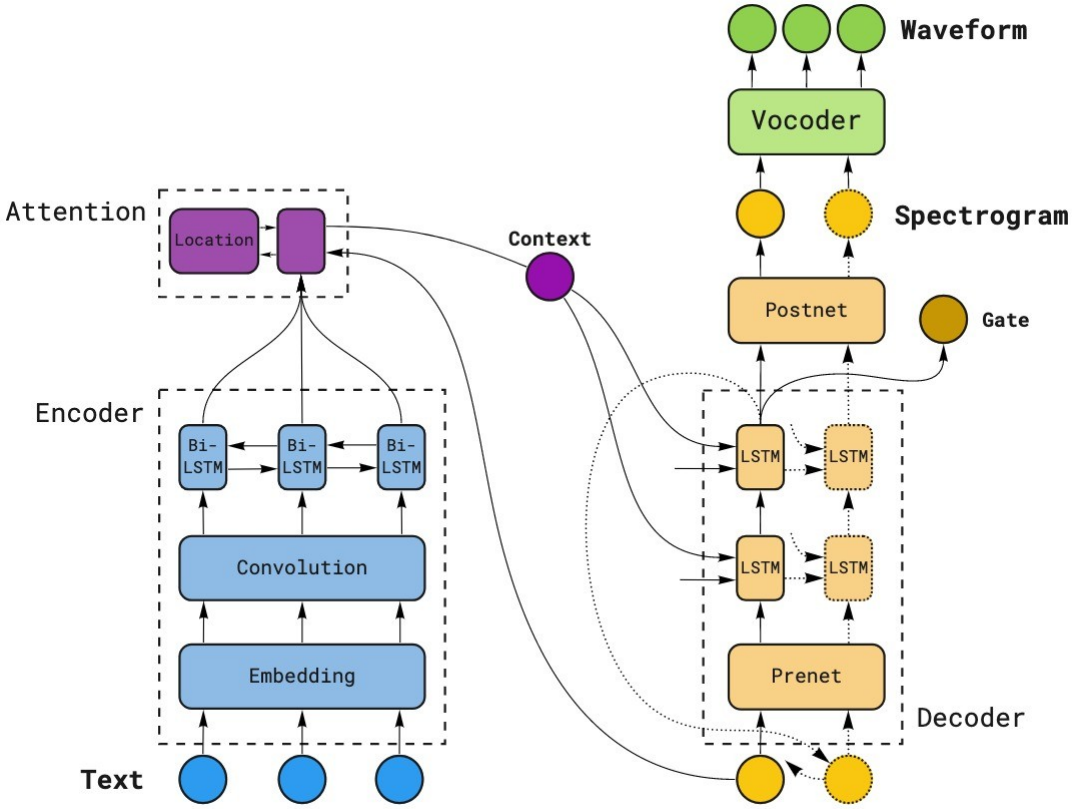

Figure 11. Tacotron 2 architecture.

2.3 Multi-speaker Tacotron

One of the factors that add to the overall complexity of text-to-speech generation is the absence of definitively correct speech output given a concrete textual input. Some features of the speech, such as intelligibility, completeness of the target sentence, naturalness and quality of the sound, can be considered the integral objectives for the task of text-to-speech synthesis. At the same time features such as patterns of stress and intonation (*prosody*) and voice of the speaker (*speaker identity*), which can also lead to considerable unpredictability of the output, can technically be freely interpreted by the model in the task of basic text-to-speech. Unsurprisingly, a lot of research efforts have subsequently been dedicated to controlling and affecting these parameters to improve the robustness and applicability of text-to-speech systems, including the follow-up research for the Tacotron model.

Prosody imitation A Tacotron extension proposed in [Skerry-Ryan et al., 2018] aims to control the synthesized speech prosody by capturing the prosody of an input audio sample. An additional *reference encoder* module is trained to compress the provided audio into a compact prosody embedding. Since in training the model is provided with the target audio as the reference encoder input, it is especially important to limit the size of the embedding, otherwise the model might end up trivially producing the provided reference as the text-to-speech output, forcing the main synthesis pipeline obsolete. During synthesis the embedding is concatenated with each annotation encoder output, thus enriching subsequent decoding with prosody information. The model is shown to successfully imitate prosody from example audio and allow supplying text annotation and prosody as independent input parameters.

In [Wang et al., 2018] the speech prosody is formalized as a set of *style tokens*. A randomly initialized set of fixed style embeddings is defined with the purpose of modelling the overall prosody of the speech as their arbitrarily weighted sum, which is then similarly used for decoding enrichment. The resulting model allows to parameterize prosody either by providing an example audio, which is mapped to a set of contributing styles by the network, or by specifying the weights of the styles directly. While individual styles are unlabeled and are not assigned any distinct meaning at initialization, they are shown to each learn a separate prosodic style, which is apparent from the resulting parameterized audio.

Speaker identity imitation Our work aims to replicate the model proposed in [Jia et al., 2018], which focuses on capturing and modelling the speaker identity assumed in the synthesis to solve the task of multi-speaker text-to-speech. As a simple variation of this problem, a model trained on a dataset of annotated speech containing multiple speakers with speaker labels can be taught to distinguish between the speakers seen in training, and subsequently parameterized to assume their identities based on a fixed lookup table of speaker identity tokens. This technique is also used in [Skerry-Ryan et al., 2018] together with the reference prosody encoder to make sure that the learned prosody is independent from the speaker identity. In practice, these two features are often hard to disentangle due to the nature of the data, since in most multi-speaker speech datasets the same speakers tend to follow the same prosodic styles, which causes prosody and speaker identity to be learned simultaneously as a single feature. While such straightforward training of speaker parameterization is a viable approach to the problem, multi-speaker capabilities of the resulting model are limited only to the fixed set of seen speakers. In addition, compiling datasets with sufficient training examples per multiple speakers can be especially challenging. [Jia et al., 2018] proposes an architecture which removes these limitations and in theory allows synthesizing speech for speakers unseen by the trained model, thus considerably reducing the data requirements. Furthermore, assuming a new speaker identity requires only a few seconds of example audio as opposed to tens

of minutes needed in the training.

In order to achieve this the base Tacotron 2 architecture is extended with an additional *speaker encoder* module. Similar to the previous research, the speaker encoder processes the provided audio sample and outputs a compact speaker identity embedding to enrich the text-to-speech decoding. In order to force the speaker encoder to learn only the disentangled features pertaining to speaker identity and to avoid the constraint of speaker recognition, the speaker encoder is trained on the task of *speaker verification* independently of the main model.

Speaker verification Speaker verification is the task of verifying whether a speech sample belongs to a specific speaker given other labeled examples of their speech. Training a neural network for this purpose requires utilizing a dedicated custom loss function. For this the model relies on the *generalized end-to-end loss* (GE2E), introduced in [Wan et al., 2017].

GE2E loss is designed to evaluate the similarity loss between a set of vectors belonging to different clusters. Assuming that each speech utterance can be mapped to a compact vector representation and is assigned a cluster label according to the speaker affiliation, G2E2 loss computed for the respective speech embeddings will represent the similarity between the speech utterances of the same speaker and the dissimilarity between those of different ones.

G2E2 loss is normally computed on a batch of N speakers with M speech samples each. Each embedding vector $\mathbf{d}_{ji}$ corresponds to the i-th speech utterance of the j-th speaker. GE2E is based on evaluating the cosine similarity between the L2-normalized embedding vectors. First, for each speaker j a centroid of all their M respective utterance embeddings is computed as

$$\boldsymbol{\mu}_j = \mathbb{E}[\mathbf{d}_{j*}] = \frac{1}{M}\sum_{m=1}^{M}\mathbf{d}_{jm}$$

Then, a similiraty matrix $\mathbf{S}_{ji,k}$ is calculated as the scaled cosine similarity between each pair of utterance embedding $\mathbf{d}_{ji}$ and speaker centroid $\boldsymbol{\mu}_k$ as

$$\mathbf{S}_{ji,k} = w\cos(\mathbf{d}_{ji}, \boldsymbol{\mu}_k) + b,$$

where w and b are learnable scalar parameters. Additionally, to improve training stability, when similarity is computed for the centroid $\boldsymbol{\mu}_k$ and an utterance $\mathbf{d}_{ki}$ of the same speaker k, the centroid is re-calculated without considering $\mathbf{d}_{ki}$ as

$$\boldsymbol{\mu}_k^{-i} = \frac{1}{M-1}\sum_{\substack{m=1 \\ m\neq i}}^{M} \mathbf{d}_{jm}$$

$$\mathbf{S}_{ji} = \begin{cases} w\cos(\mathbf{d}_{ji}, \boldsymbol{\mu}_k^{-i}) + b & \text{if } k = j \\ w\cos(\mathbf{d}_{ji}, \boldsymbol{\mu}_k) + b & \text{otherwise} \end{cases}$$

The loss per each embedding vector $\mathbf{d}_{ji}$ is defined as

$$L(\mathbf{d}_{ji}) = -\mathbf{S}_{ji,j} + \log\sum_{k=1}^{N}\exp(\mathbf{S}_{ji,k})$$

The first component of the loss accounts for the similarity of the embedding to other embeddings of the same speaker, and the second one represents its similarity to embeddings belonging to other speakers. The loss should push the embedding towards its own speaker cluster center and away from the cluster centers of all the other speakers (see Figure 12).

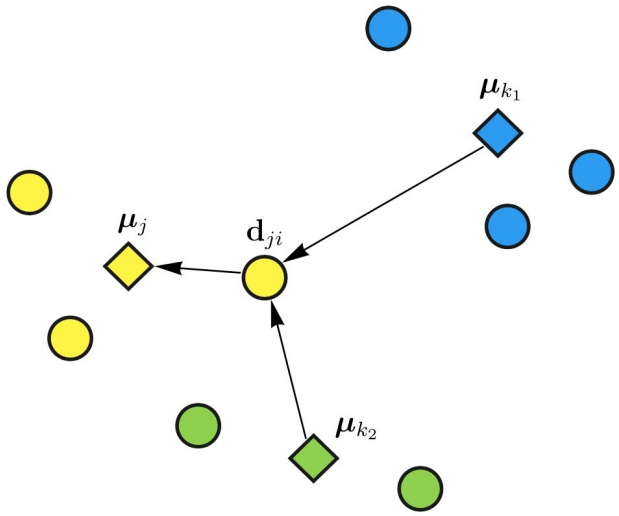

Figure 12. Generalized end-to-end loss effect on the embeddings.

The overall GE2E loss is computed as the cumulative loss for all the embeddings

$$L(\mathbf{S}) = \sum_{j,i} L(\mathbf{d}_{ji})$$

Intuitively, a neural network trained on this loss function should learn to extract features from speech that are the most useful for identifying and distinguishing different speakers.

End-to-end multi-speaker text-to-speech model The overall multi-speaker model proposed in [Jia et al., 2018] consists of the basic Tacotron 2 model combined with an additional speaker encoder module (see Figure 13). The speaker encoder is trained independently of the main text-to-speech prediction network, and requires a multi-speaker dataset of speech with speaker labels. Since it is not required to have textual annotations for this data, the speaker encoder is potentially free to use datasets not suitable for the text-to-speech training.

The speaker encoder is trained to produce a speaker embedding based on a provided audio sample. The network module itself follows a simple RNN architecture consisting of multiple LSTM layers followed by a linear layer to output an L2-normalized embedding vector of fixed dimensionality from the last output of the recurrent cell. Before being passed to the encoder, the audio samples are converted to the mel spectrogram representation to facilitate their processing by the recurrent layers. In addition, the audio (and the corresponding spectrogram) is split into fragments using a sliding window of 0.8s with 0.4s step. The final speaker embedding for the utterance is computed as an average of the embeddings produced from its fragments.

Within the end-to end model the speaker embedding produced by the speaker encoder is concatenated with each output of the text-to-speech network's encoder module. These enriched encoder outputs are then examined by the attention mechanism during decoding just as in the base text-to-speech architecture, allowing the decoder to access the speaker identity context and utilize it during spectrogram prediction.

The speaker encoder is first pre-trained on the task of speaker verification using the described GE2E loss. Once pre-trained, it can be used to train the actual end-to-end text-to-speech model. During text-to-speech training, the speaker encoder is given the ground truth audio. At this phase its parameters are frozen to prevent the previously mentioned trivial outputting of the provided input data by the network. In theory, the model trained in this setting should be able to improve its spectrogram predictions by matching the speaker features of the target audio.

During inference, the resulting model allows to synthesize speech according to separately specified text annotation and voice, with the latter provided as a short reference speech sample.

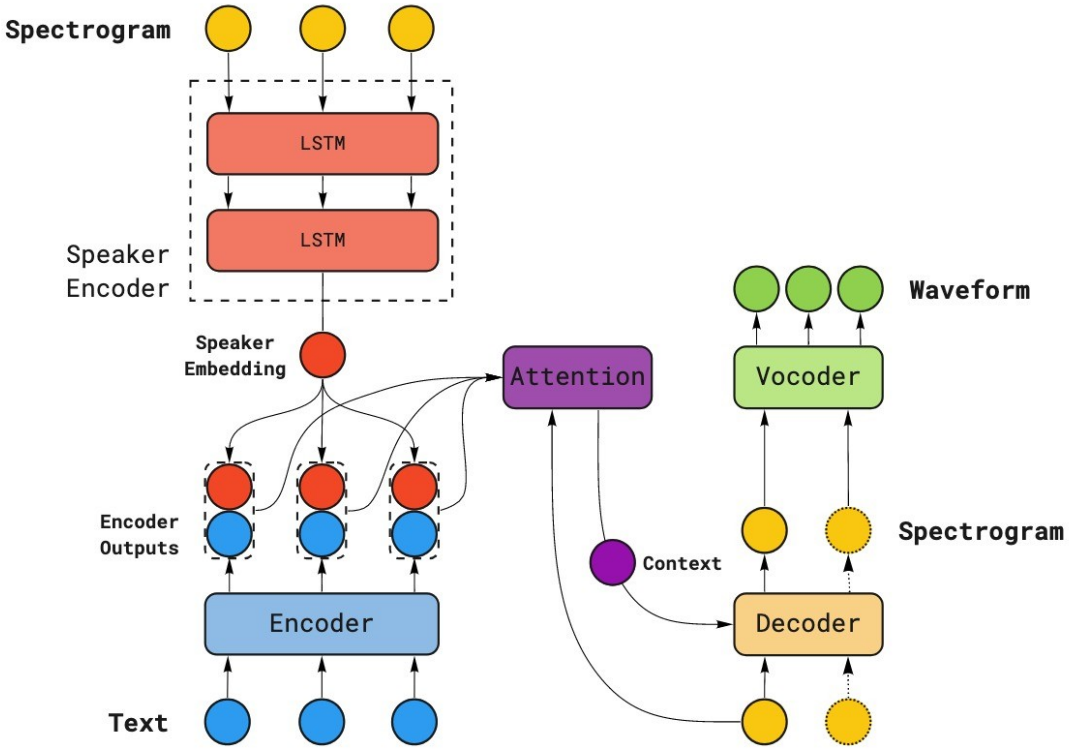

Figure 13. Multi-speaker Tacotron 2 architecture.

3 Experiments

3.1 Implementation Details

The full multi-speaker Tacotron 2 model used in this research closely follows the original work. The base Tacotron 2 model is based on the available implementation in PyTorch [1]. The multi-speaker extension of the base model has been manually implemented according to the original architecture from [Jia et al., 2018]. GE2E loss for speaker verification training is also manually implemented as described in [Wan et al., 2017].

The full text-to-speech framework includes a preprocessing frontend for the textual input. First, input text is filtered to only contain allowed characters, which includes letters of the Estonian alphabet (Latin-based alphabet with unique "õ", "ä", "ö", "ü", "š" and "ž"), space and certain punctuation characters. Supporting numerical characters is intentionally left out of scope since spelling of numbers in Estonian is non-trivial due to declension. The full set of supported characters forms a fixed list which allows assigning each character a unique sequential identifier. Letter characters are converted to lower case and consecutive spaces are collapsed into one for further simplification. The entire input is tokenized into characters and mapped to IDs, resulting in a sequence of integers.

All audio is converted to 80-channel mel-scale spectrograms using a sampling rate of 22050, Hann function frame windows of length 1024 with hop length of 256 samples, and maximum frequency of 8000 Hz used for the transformation to mel scale.

Hyperparameters for the implemented model are provided in Table 1.

The encoder module accepts a batch of B zero-padded preprocessed input sequences of max length T. It consists of a lookup table embedding layer, 3 convolutional layers over the temporal axis and a bidirectional recurrent LSTM. The BiLSTM forwards the outputs of all timesteps, resulting in an encodings tensor of shape ($B\times$) $T\times 2\cdot$ LSTM cell size (outputs per directional layer are concatenated).

The implemented speaker encoder module accepts 80-channel mel spectrograms of the target speaker samples and produces 128-dimensional speaker embeddings. It consists of 3 LSTM layers each followed by a fully connected projection layer. The projected output of the last recurrent step is L2-normalized to obtain the final embedding. The embedding dimensionality is reduced from the original 256 used in [Jia et al., 2018] since we found that during speaker verification training on the available data the encoder struggled to utilize all of the provided dimensions.

In the speaker encoder, an input spectrogram is initially broken into 72-frame fragments (roughly corresponding to 0.8s of audio) with a sliding window of 36 frames. The corresponding embeddings per input spectrogram are averaged to produce its final embedding. The final encodings are computed by broadcast-concatenating the original encoder outputs with the produced speaker embedding. To account for this, the dimen-

[1] Tacotron 2 implementation by NVIDIA: `https://github.com/NVIDIA/tacotron2`

Encoder	Embedding(512) $\rightarrow$ 3$\times$Conv1D(filters=512, window=5, padding=2) $\rightarrow$ BatchNorm1D $\rightarrow$ ReLU $\rightarrow$ Dropout(0.5) $\rightarrow$ BiLSTM(256)
Speaker Encoder	3$\times$LSTM(768) $\rightarrow$ Linear(128) $\rightarrow$ L2-Norm
Decoder Prenet	2$\times$Linear(256) $\rightarrow$ ReLU $\rightarrow$ Dropout(0.5)
Attention RNN	LSTM(1024) $\rightarrow$ Dropout(0.1)
Attention	Conv1D(filters=32, window=31, padding=15) $\rightarrow$ Linear(128), Linear(128), Linear(128) $\Rightarrow$ Sum $\rightarrow$ Tanh $\rightarrow$ Linear(1) $\rightarrow$ Softmax
Decoder RNN	LSTM(1024) $\rightarrow$ Dropout(0.1) $\Rightarrow$ Linear(80), Linear(1) $\rightarrow$ Sigmoid
Decoder Postnet	4$\times$Conv1D(filters=512, window=5, padding=2) $\rightarrow$ BatchNorm1D $\rightarrow$ Dropout(0.5) $\rightarrow$ Tanh $\rightarrow$ Conv1D(filters=80, window=5, padding=2) $\rightarrow$ Dropout(0.5)

Table 1. Model implementation hyperparameters.

sionality of all the dependent layers in the computational graph is also increased by 128 compared to the original implementation.

The attention module includes a location-sensitive mechanism which stores the attention scores produced at the previous decoder step, as well as the sum of all attention scores produced since the start of decoding (initialized with zeros according to the input sequence length T). During attention calculation these scores are used to compute a timestep-specific location token, which is obtained by concatenating the scores into a $T\times2$ tensor and upsampling it with a length-wise convolutional layer.

At each decoder step the attention module computes the attention context based on the enriched encoder outputs, a timestep-specific decoder query and a computed location token. In order to compute the attention scores all of the listed inputs are passed through respective linear projections, resulting in unified dimensionality, and summed up (the decoder query projection is broadcast-summed). The $\tanh$ activation of the sum tensor is then passed through a final projection layer and normalized with softmax to obtain a T-length vector of attention scores. The location-sensitive mechanism is updated with the newly produced scores and the final attention context is calculated as a weighted sum of the encodings.

The decoder performs 80-channel mel spectrogram prediction one spectrogram frame per timestep (alternatively it is also possible to output a fixed number of frames at each decoder step).

The decoder prenet expects a single spectrogram frame as its input. It consists of two fully connected layers with ReLU activation and 50%-chance dropout for better

generalization.

In the decoder, the prenet output is concatenated with the previous attention context and processed by an LSTM attention RNN. The output of this RNN is then used as the decoder query for the attention module. The decoder query concatenated with a newly computed attention context is passed to an LSTM decoder RNN, the output of which is then again concatenated with the attention context. The resulting tensor is passed though an 80-channel projection layer to produce the raw decoder output, which is consequently used as the input for the next decoder step. In addition, the same tensor is also mapped into a single sigmoid-activated gate value with another projection layer. During inference, if the gate exceeds the threshold of 0.5 the decoding is stopped. As mentioned before, during training the next step is instead initialized with the ground truth mel spectrogram frame, and the gate prediction is ignored (since the length of the target spectrogram is known).

The raw decoder spectrogram predictions of shape $S \times 80$ (assuming the decoding terminated after S steps) are passed through a postnet consisting of 5 convolutional layers. The postnet outputs are then added to the raw decoder outputs to produce the final mel predictions.

3.2 Datasets

For the text-to-speech training we had access to two datasets of annotated speech in Estonian - *Kõnekorpus* and *Common Voice*.

Kõnekorpus [Rätsep et al., 2020] contains 65 hours of studio-recorded high audio quality speech samples of 11 labeled speakers. The total speech duration per speaker is distributed quite unevenly: 3 of the speakers have around 15 hours each, while the others have roughly 2 hours of speech per speaker. In total there are 43K samples with an average duration of 5.5s. The dataset has been specifically created for text-to-speech training and perfectly suits the basic task.

Common Voice [Ardila et al., 2019] is a community effort project managed by Mozilla. It consists of speech recordings submitted and validated by volunteers. Although the main focus of the project is neural speech recognition, our assumption is that it can be also used to facilitate text-to-speech training. Predictably, the dataset has some inherent flaws due to being fully community contributed. The quality of the audio is significantly lower than with studio recordings. Various background noise, arbitrary sounds and padding around the actual speech in the audio are quite common, although the validations put in place guarantee the uttered speech to match the annotations. At the time we accessed it the Estonian corpus of Common Voice contained 19 hours of speech submitted by 430 unique speakers. It consists of 10.5K samples of 6.6s average duration. As expected, per-speaker submissions are quite short: only 56 out of 430 speakers have more than 5 minutes of speech, amounting to about 7.5 hours. We use both the full Common Voice data and a subset of these 56 speakers (referred to as filtered

Common Voice) to examine if especially underrepresented speakers can still contribute to improving the trained model.

In addition, we use *VCTK* dataset [Veaux et al., 2017] exclusively for the task of speaker verification. It contains 43 hours of studio-quality expressive speech of 108 speakers in English. We include it into the experiments to verify that a speaker encoder module can be separately trained on a different language dataset and be successfully reused to facilitate multi-speaker speech synthesis, since theoretically speaker identity features should not be dependent on neither content nor even language of the example speech.

3.3 Training

Speaker encoder training We first train multiple speaker encoder models on the task of speaker verification using different dataset variations (see Table 2): only Kõnekorpus, Kõnekorpus with full Common Voice, Kõnekorpus with filtered Common Voice, and VCTK. The reasoning behind training and comparing these models is to analyze how the supplementary data from the non-specialized Common Voice dataset affects the baseline Kõnekorpus speaker encoder. The datasets in question have vastly different characteristics relevant to speaker verification training. Kõnekorpus contains a large amount of speech data in high quality, but it is unevenly distributed between only a small number of speakers. Common Voice includes a considerably larger selection of speakers, but they have much less per-speaker representation and an overall low audio quality. VCTK is characterized by high quality and large number of speakers with reasonable representation, although it does not match the target language.

	Speaker encoder training data
K	Kõnekorpus
K + CV	Kõnekorpus + Common Voice
K + fCV	Kõnekorpus + filtered Common Voice
VCTK	VCTK

Table 2. Trained speaker encoder model variations.

The models are trained against the described GE2E loss with initial $w = 10.0$ and $b = -5.0$ using a simple stochastic gradient descent optimizer with learning rate 0.01 and gradient clipping, as described in [Wan et al., 2017], for a total of 600K iterations. For batching at each step we sample 32 speakers with 10 utterances each (except during the Kõnekorpus model training where instead we sample 10 speakers with 32 utterances, since the dataset only contains 11 speakers in total). During training we separately record validation metrics for seen and unseen speakers (5 speakers from Kõnekorpus and 10 speakers from Common Voice respectively).

Although the absolute values of GE2E loss are not very informative due to being scaled (with learnable scalar w and b), we can still conclude that the models trained with smaller numbers of speakers tend to overfit to the seen data and are consequently worse at predicting unseen speakers (see Figure 14). From the VCTK model results, where both groups of validation data are effectively unseen, it is also apparent that unseen speaker samples correspond to worse results regardless of prior knowledge - perhaps because of added difficulty due to inherent audio quality.

Figure 14. Mean per-utterance speaker encoder validation loss.

To gain insight into how the models prioritize the two described components of GE2E loss, we introduce and track some custom metrics. In order to measure the quality of clusterization in a cosine similarity-based embedding space, for a set of produced embeddings we measure mean cosine similarity between the same-cluster embeddings (*compactness*) and negative mean cosine similarity between cross-cluster pairs of embeddings (*distinctiveness*), with the target for both being 1.0 (see Figure 15). On the seen speaker utterances we again observe the pure Kõnekorpus model to overfit, yielding the best results in both compactness and distinctiveness. Surprisingly, while failing to clusterize unseen speakers it achieves the best distinctiveness score for this group. A possible explanation could be that after quickly minimizing the compactness loss due to having only a small number of easily distinguishable speakers, the model

trains to efficiently minimize distinctiveness loss by learning to produce more distinct embeddings overall and better utilizing the available embedding dimensionality. By simply having higher mean deviation in the produced embedding space the Kõnekorpus model can achieve higher distinctiveness score for the given utterances regardless of how well they are clusterized. As expected, models trained on Kõnekorpus enriched with Common Voice data tend to generalize better, proportionally to the amount of additional data. There is also no evidence that withholding speakers with especially few speech examples (filtered Common Voice) has any advantages over providing all the available speakers based on these metrics. Finally, the VCTK model achieves the best unseen speaker compactness with the worst distinctiveness by a large margin, which puts the usefulness of the trained model under question. We investigate how these speaker encoder metrics correspond to the quality of actual multi-speaker speech synthesis in the follow-up evaluation experiments.

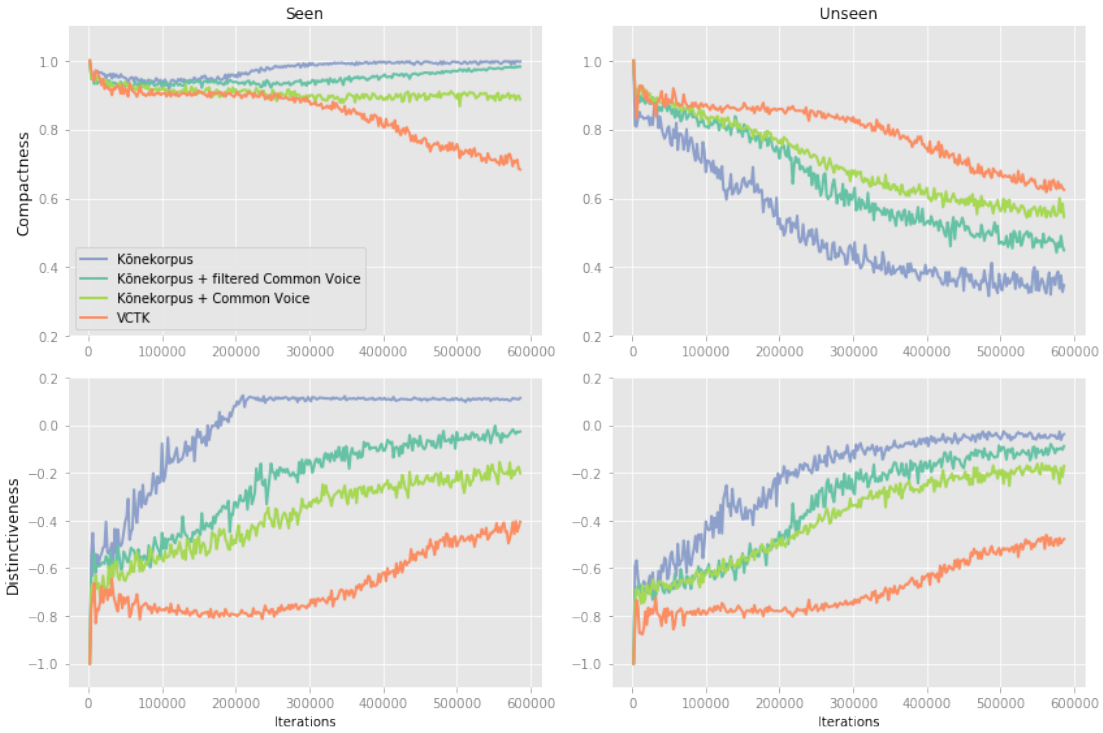

Figure 15. Speaker encoder validation compactness and distinctiveness

We also inspect the evolution of the trainable scalars w and b used for scaling the embedding similarity in GE2E loss, to better understand their role in directing the model training. Curiously, we discover that the parameter b remains constant throughout the training process, making it obsolete. We consequently verify this by expanding

the formula (see Appendix III). The behavior of parameter w seems to be related to the tradeoff in minimizing the loss components representing either compactness or distinctiveness (see Figure 16), where the increase of w corresponds to the weight increase of the compactness component. As the training begins the produced embeddings are arbitrarily distributed across the embedding space, causing the lack of compactness to contribute the majority of loss. At the maximum point of w as the predicted clusters start to pull together, the distinctiveness loss becomes the major component. Eventually w stabilises indicating the component contribution to equalize.

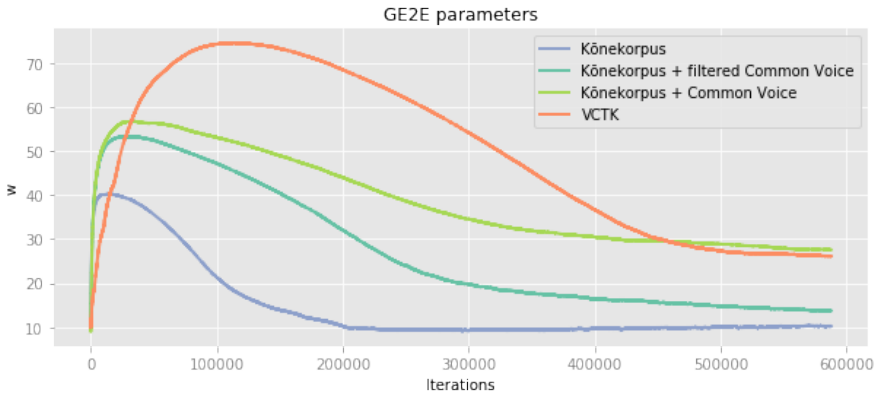

Figure 16. GE2E loss trainable parameter w

Overall we observe the trained speaker encoders to successfully separate audio for speakers with largely distinct voices (especially male versus female), but struggle with more similar ones (see Figure 17).

Text-to-speech training We train several variations of the full multi-speaker text-to-speech models using different datasets to test our assumptions (see Table 3). To check whether the supplementary "noisy" data from Common Voice can improve text-to-speech quality we train a baseline text-to-speech model with Kõnekorpus speaker encoder using only Kõnekorpus data. We compare it against models with additional data used in both speaker encoder and text-to-speech model training (specifically filtered Common Voice and full Common Voice data added to the baseline Kõnekorpus data). In addition, to investigate how the utilized pre-trained speaker encoder affects the resulting models in isolation, we train multiple models using the same set of all the available data in text-to-speech training (Kõnekorpus plus full Common Voice) and different variations of speaker encoders described earlier.

The models are trained on mean squared error loss from both raw and postnet-corrected mel predictions, as well as binary cross entropy loss from gate predictions. We use Adam optimizer with learning rate of 10^{-3} and weight decay of 10^{-6}, with the batch

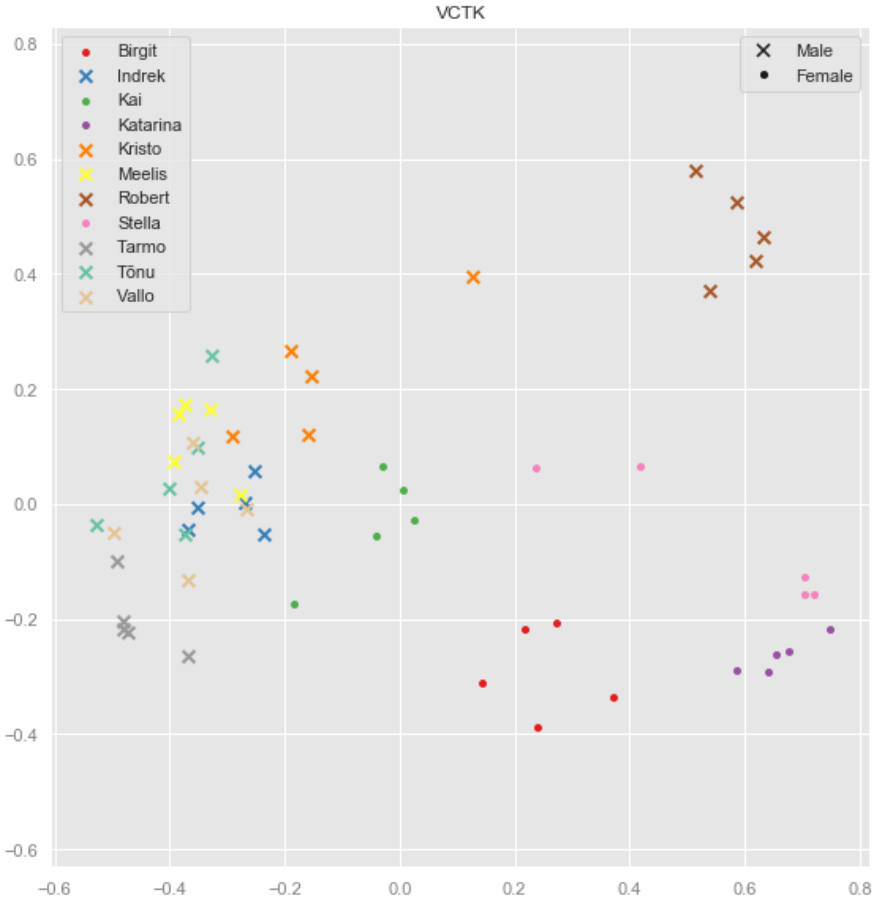

Figure 17. PCA-transformed embeddings of unseen Kõnekorpus speaker utterances

Text-to-speech training data	Speaker encoder training data
K	K
K + fCV	K + fCV
K + CV	K + CV
K + CV	K
K + CV	K + fCV
K + CV	VCTK

Table 3. Trained text-to-speech model variations.

size set to 32, continuing for approximately 200K iterations. During training the basic convergence of the model can be tracked by simply observing the produced attention scores, which eliminates the need to perform spectrogram-to-waveform synthesis.

As vocoder we use a readily available model called WaveGlow [2], pre-trained for mel spectrogram-to-waveform synthesis and provided together with the used Tacotron 2 implementation.

[2]Pre-trained vocoder by NVIDIA: `https://github.com/NVIDIA/waveglow`

4 Results

4.1 Evaluation

The nuanced nature of the human speech makes it hard to define objective computable metrics for its quality and naturalness. For this reason speech quality is most commonly measured and compared based on *mean opinion score*, which is an average of subjective opinion scores assigned by human evaluators. Understandably, it provides limited reliability when comparing results from different experiments due to a large number of varying factors such as the set of evaluators, testing conditions, etc., but more conclusive results can be obtained by performing a comparative mean opinion score evaluation of multiple models as part of the same experiment.

We perform a comparative mean opinion score evaluation of the trained text-to-speech models using 8 independent Estonian native speakers as evaluators. We synthesize 150 speech samples per model, half of which have speaker identity parameterized with speaker samples of seen speakers (from Kõnekorpus), and other half - of unseen speakers (from Common Voice). The target sentences used for the test data are extracted from an Estonian newspaper and tend to have complex language with frequently included proper nouns and uncommon terms. The evaluators assign a 1-5 score (with step 0.5) in two defined metrics: *speech naturalness* and *speaker similarity*. For evaluating speech naturalness we instruct the evaluators to consider intelligibility, clarity and sound quality. We also ask them to compare the content of the speech sample with the displayed target sentence and limit the score proportionally, since we find that in some cases the models output only a small part of the textual input followed by silence or noise. To evaluate speaker similarity we provide the original speaker sample used to parameterize the synthesis along with the synthesized sample and ask the evaluators to only score the similarity of speakers, ignoring all the other factors considered for speech naturalness. To conduct and facilitate the experiment under the specified conditions we prepare a simple web application (see Figure 18). Throughout the evaluation experiment the samples synthesized by the compared models are provided in a random order without disclosing the underlying models to the evaluators.

4.2 Results and Analysis

Speech naturalness We first notice that the models are not guaranteed to provide a valid speech output for all combinations of provided textual input and voice sample. In a number of cases the synthesized audio is characterized by silence, white noise or completely incomprehensible sounds. To measure the success rate of the synthesis we synthesize 400 samples per model and manually mark failed samples. We consequently provide only successful unmarked samples for independent evaluation and report both the raw evaluation scores and scores scaled by the estimated model success rate (since

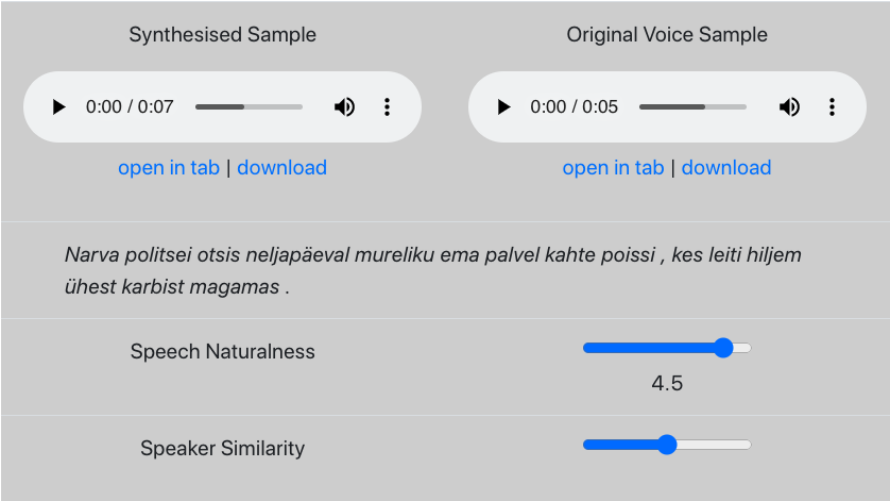

Figure 18. Web-interface for mean opinion score evaluation.

the resulting raw scores are inflated due to failed sample filtering).

Based on the estimated success rates (see Table 4) we can deduce that overall the trained models tend to struggle with unseen speakers: we estimate 96% success rate on seen speakers and only 59% on unseen. We can speculate that the unbalanced per-speaker distribution of the provided data is not suitable for training a fully robust multi-speaker model and more data should be provided for a larger number of distinct speakers. Moreover, it is apparent that the multi-speaker parameterization of the trained models is deeply entangled with the core text-to-speech mechanism, causing the models to fail completely in case of unexpected speaker identity input rather than handling the basic synthesis with unsuccessful speaker imitation. Lastly, the extremely high variance of per-model synthesis success rate suggests the overall low robustness of the models, as well as significant effect of the random testing input assigned to them on the probability of successful synthesis.

The evaluated mean opinion score for synthesized speech naturalness on seen speakers is 2.82 on average, with the highest scoring model achieving 3.16 (see Table 5). On unseen speakers the average score is 1.83 and the best individual model is assigned 2.40. The scores are significantly lower than the ones presented in the replicated state-of-the-art experiments, but they are still reasonable and are mostly related to the low robustness of the models and intentionally challenging testing input. At the same time around 17% of the assigned individual scores in speech naturalness are ≥ 4.0, signifying that the models are capable of synthesizing convincingly natural sounding speech under specific conditions.

We confirm that providing supplementary speech recognition data in training leads to better speech naturalness results. For example, the K text-to-speech with K speaker

Text-to-speech	Speaker encoder	Seen	Unseen
K	K	0.89	0.68
K + fCV	K + fCV	0.99	0.90
K + CV	K + CV	0.99	0.54
K + CV	K	0.95	0.37
K + CV	K + fCV	0.98	0.33
K + CV	VCTK	0.97	0.72
		0.96	0.59

Table 4. Estimated synthesis success rate.

encoder and K + CV text-to-speech with K speaker encoder models, which used no supplementary data at certain stages of training, are assigned scores worse by a significant margin compared to the models with enriched training data.

The comparison of fully and partially enriched models seems to suggest that while in regards to text-to-speech training the best results correspond to the models with full available data (K + CV text-to-speech), speaker encoders enriched with filtered speaker recognition data (K + fCV speaker encoder) can outperform fully enriched models. Based on this we can conclude that including training data for speakers with extremely low amount of examples is not particularly useful during the speaker encoder training.

We notice the model with the experimental VCTK speaker encoder to show results which are comparable or even superior to the models with speaker encoders trained on Estonian corpora, confirming our earlier assumption about language independence of speaker encoder training.

		Scaled		Raw	
Text-to-speech	Speaker encoder	Seen	Unseen	Seen	Unseen
K	K	2.30	2.17	2.46	2.73
K + fCV	K + fCV	2.92	**2.40**	2.94	2.55
K + CV	K + CV	3.04	1.66	3.07	2.22
K + CV	K	2.50	1.41	2.58	2.10
K + CV	K + fCV	**3.16**	1.32	**3.21**	1.97
K + CV	VCTK	3.09	2.31	3.15	**2.82**
		2.82	1.83	2.90	2.40

Table 5. Mean opinion scores in speech naturalness.

We also investigate per-speaker scores for the speakers seen in training (see Table 6), and see no noticeable improvement or even correlation between the volume of training examples provided for a specific speaker and the resulting speech naturalness. Given that the minimum speaker representation in this comparison is 2.5h, this might suggest that a

few hours of data (or less) per speaker is sufficient for the text-to-speech model and any additional data yields diminishing returns in terms of specific speaker identity synthesis.

Speaker	Duration	Mean	Best
Katarina	17.1h	3.14	3.59
Robert	16.8h	2.66	3.35
Kristo	12.3h	2.99	3.40
Stella	4.2h	2.31	3.21
Birgit	2.5h	3.26	3.56

Table 6. Raw mean opinion scores in speech naturalness per seen speaker.

Speaker similarity As before, we report both the raw mean opinion scores in speaker similarity and the scores scaled by the synthesis success rate (see Table 7). In addition, to gain a better perspective in how the quality of speaker identity replication is conditioned on the success of basic synthesis we also provide the scores corresponding to samples achieving at least 3.0 mean opinion score in speech naturalness.

The received average scaled mean opinion score in speaker similarity is 3.37 for seen speakers and only 1.68 for unseen. At the same time the mean score exclusively for samples with higher speech naturalness is 3.78 for seen and 2.55 for unseen speakers, which confirms the interdependent nature of the two metrics.

The perceived speaker similarity for speakers unseen in training is significantly lower for all models, highlighting the dependence on the prior knowledge and low degree of generalization.

The unseen speaker scores suggest that the models are capable of imitating high-level speaker features (such as gender affiliation) given just a short speech example, but fail to reproduce more subtle features that allow for the vast diversity of human voices, unless the speaker is recognized. At the same time the decent results for seen speakers indicate relative success in achieving a simpler fixed set multi-speaker model, which notably is not parameterized directly by a pre-defined speaker token, but rather by a random example utterance that needs to be matched with the prior knowledge.

Curiously, the models trained with the pure Kõnekorpus speaker encoder tend to underperform in seen speaker synthesis based on the scaled scores, although they are expected to be overfitted to these speakers. However, since the scaled scores take into consideration the overall model robustness, they do not accurately represent the best speaker similarity achievable by the models. Indeed, the higher-naturalness scores for the same models confirm their superior ability to imitate the seen speakers over more generalized models.

Surprisingly, we also achieve the best unseen higher-naturalness speaker similarity score with one of the Kõnekorpus speaker encoder models, which cannot be easily

Text-to-speech	Sp. enc.	Scaled		Raw		Naturalness >= 3.0	
		Seen	Unseen	Seen	Unseen	Seen	Unseen
K	K	3.05	1.86	3.31	2.27	**3.92**	2.51
K + fCV	K + fCV	3.45	**2.18**	3.48	**2.31**	3.77	2.63
K + CV	K + CV	3.51	1.56	3.54	2.04	3.81	2.46*
K + CV	K	3.23	1.42	3.34	2.12	3.82	**2.72***
K + CV	K + fCV	**3.59**	1.32	**3.65**	1.95	3.82	2.52*
K + CV	VCTK	3.44	1.87	3.51	2.20	3.64	2.54
		3.37	1.68	3.47	2.15	3.78	2.55

Table 7. Mean opinion scores in speaker similarity. Scores marked with * are computed with low confidence level.

justified. We consequently notice that the number of individual scores available for computing certain mean scores in the higher-naturalness section are especially small (≤ 10), and therefore they should be disqualified due to low confidence level.

We notice that based on the higher-naturalness scores the choice of utilized text-to-speech training data seems to be the deciding factor for speaker similarity, while the importance of speaker encoder seems negligible. For example, we notice almost equivalent scores for K + CV text-to-speech models with speaker encoder variations K, K + fCV and K + CV, which are trained on data with significantly different properties.

We observe the VCTK speaker encoder model to fall behind in the higher-naturalness similarity score for seen speakers, which is to be expected due to its lack of exposure to the speakers in speaker encoder training. Still, the difference is small enough to confirm the relatively low importance of pre-trained speaker encoders overall in the text-to-speech framework, since the model manages to achieve comparable results thanks to having the seen speakers present during text-to-speech training.

5 Conclusion

Multi-speaker text-to-speech synthesis is a complicated problem with rapidly evolving state-of-the-art solutions based on neural network training. Text-to-speech models have a wide range of practical applications as well as research value for the field of deep learning in general.

In this work we replicate a recent state-of-the-art model for multi-speaker text-to-speech synthesis and apply it to the Estonian language. Concretely, we re-purpose an existing text-to-speech model implementation and extend it to allow for multi-speaker synthesis with our own implementation following relevant research. We train the models using the available Estonian speech data and perform an independent evaluation of the results by native speakers.

Without particular optimization efforts we manage to train models with reasonable performance in both speech naturalness of basic text-to-speech synthesis and speaker similarity in the multi-speaker setting, confirming the potential and reusability of the replicated state-of-the-art architecture. Admittedly, the obtained models still have noticeable flaws in robustness and unseen speaker imitation.

We study the effects of varying training data characteristics on the resulting models and experimentally confirm that the measures of naturalness and speaker imitation quality for multi-speaker text-to-speech models can be improved by means of enrichment from different sources, which in our case includes a supplementary "noisy" dataset for speaker recognition.

We also find that the low reliability and high subjectiveness of the available evaluation methods leads to difficulty in comparing and evaluating text-to-speech models. Nevertheless, we confirm some part of our assumptions, as well as notice and highlight certain unexpected properties of the studied neural network frameworks.

The future work may include a more dedicated effort in improving the models with various data preprocessing and hyperparameter optimization, as well as replication and comparison for the peer and follow-up state-of-the-art research in text-to-speech synthesis.

References

[Ardila et al., 2019] Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F. M., and Weber, G. (2019). Common Voice: A Massively-Multilingual Speech Corpus. *arXiv e-prints*, page arXiv:1912.06670.

[Arik et al., 2017a] Arik, S., Diamos, G., Gibiansky, A., Miller, J., Peng, K., Ping, W., Raiman, J., and Zhou, Y. (2017a). Deep Voice 2: Multi-Speaker Neural Text-to-Speech. *arXiv e-prints*, page arXiv:1705.08947.

[Arik et al., 2017b] Arik, S. O., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Ng, A., Raiman, J., Sengupta, S., and Shoeybi, M. (2017b). Deep Voice: Real-time Neural Text-to-Speech. *arXiv e-prints*, page arXiv:1702.07825.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv e-prints*, page arXiv:1409.0473.

[Chorowski et al., 2015] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28, pages 577–585. Curran Associates, Inc.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

[Jia et al., 2018] Jia, Y., Zhang, Y., Weiss, R. J., Wang, Q., Shen, J., Ren, F., Chen, Z., Nguyen, P., Pang, R., Lopez Moreno, I., and Wu, Y. (2018). Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis. *arXiv e-prints*, page arXiv:1806.04558.

[Ping et al., 2017] Ping, W., Peng, K., Gibiansky, A., Arik, S. O., Kannan, A., Narang, S., Raiman, J., and Miller, J. (2017). Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning. *arXiv e-prints*, page arXiv:1710.07654.

[Rätsep et al., 2020] Rätsep, L., Piits, L., Pajupuu, H., Hein, I., and Fišel, M. (2020). Neural Speech Synthesis for Estonian. *arXiv e-prints*, page arXiv:2010.02636.

[Shen et al., 2017] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R., Saurous, R. A., Agiomyrgiannakis, Y., and Wu, Y. (2017). Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. *arXiv e-prints*, page arXiv:1712.05884.

[Skerry-Ryan et al., 2018] Skerry-Ryan, R., Battenberg, E., Xiao, Y., Wang, Y., Stanton, D., Shor, J., Weiss, R. J., Clark, R., and Saurous, R. A. (2018). Towards End-to-End Prosody Transfer for Expressive Speech Synthesis with Tacotron. *arXiv e-prints*, page arXiv:1803.09047.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *arXiv e-prints*, page arXiv:1409.3215.

[van den Oord et al., 2016] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. *arXiv e-prints*, page arXiv:1609.03499.

[Veaux et al., 2017] Veaux, C., Yamagishi, J., and Macdonald, K. (2017). Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit.

[Wan et al., 2017] Wan, L., Wang, Q., Papir, A., and Lopez Moreno, I. (2017). Generalized End-to-End Loss for Speaker Verification. *arXiv e-prints*, page arXiv:1710.10467.

[Wang et al., 2017] Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. A. (2017). Tacotron: Towards End-to-End Speech Synthesis. *arXiv e-prints*, page arXiv:1703.10135.

[Wang et al., 2018] Wang, Y., Stanton, D., Zhang, Y., Skerry-Ryan, R., Battenberg, E., Shor, J., Xiao, Y., Ren, F., Jia, Y., and Saurous, R. A. (2018). Style Tokens: Unsupervised Style Modeling, Control and Transfer in End-to-End Speech Synthesis. *arXiv e-prints*, page arXiv:1803.09017.

Appendix

I. Synthesized speech samples

Samples of synthesized speech can be accessed at https://tacotron-samples-ee.herokuapp.com.

II. Source code

Source code for this work can be accessed at https://github.com/GeronimoOM/multi-speaker-tacotron2.

III. Speaker verification loss expansion

$$\text{Let } \mathbf{S}_{ji} = \begin{cases} w\cos(\mathbf{d}_{ji}, \boldsymbol{\mu}_k^{-i}) + b & \text{if } k = j \\ w\cos(\mathbf{d}_{ji}, \boldsymbol{\mu}_k) + b, & \text{otherwise} \end{cases}$$

$$= \begin{cases} w\cos(\mathbf{d}_{ji}, \boldsymbol{\mu}_k^{-i}) & \text{if } k = j \\ w\cos(\mathbf{d}_{ji}, \boldsymbol{\mu}_k), & \text{otherwise} \end{cases} + b$$

$$= \mathbf{S}^*_{ji} + b$$

$$\text{Then } L(\mathbf{d}_{ji}) = -\mathbf{S}_{ji,j} + \log \sum_{k=1}^{N} \exp(\mathbf{S}_{ji,k})$$

$$= -(\mathbf{S}^*_{ji,j} + b) + \log \sum_{k=1}^{N} \exp(\mathbf{S}^*_{ji,k} + b)$$

$$= -\mathbf{S}^*_{ji,j} - b + \log \sum_{k=1}^{N} \left[\exp(\mathbf{S}^*_{ji,k}) \exp(b)\right]$$

$$= -\mathbf{S}^*_{ji,j} - b + \log \left[\exp(b) \sum_{k=1}^{N} \exp(\mathbf{S}^*_{ji,k})\right]$$

$$= -\mathbf{S}^*_{ji,j} - b + b + \log \sum_{k=1}^{N} \exp(\mathbf{S}^*_{ji,k})$$

$$= -\mathbf{S}^*_{ji,j} + \log \sum_{k=1}^{N} \exp(\mathbf{S}^*_{ji,k})$$

- thus b is obsolete.

IV. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Oleh Matsuk**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Multi-speaker Text-to-speech Synthesis in Estonian,
   *(*title of thesis)

   supervised by Mark Fišel.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Oleh Matsuk
14/01/2021