The Impact of Deep Learning on Speech Synthesis with Mobile Devices

Hannes Bohnengel
Technical University of Munich
hannes.bohnengel@tum.de

ABSTRACT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

1 INTRODUCTION

Virtual personal assistants (VPA) like Siri, Cortana or Google Now start having a huge impact on the way of interacting with electronic devices like smartphones or notebooks. Up to now the VPAs help only with rather simple tasks like search queries, starting phone calls or setting a clock, but according to a recent survey from the IT research firm Gartner [4], this will change in the near future. With the Facebook Messenger it is already possible to make purchases or to order an Uber car and new use cases are expected soon. The survey also states, that through the vast increase of devices in the scope of the Internet of Things (IoT) the way of interacting with machines will go towards minimal or zero touch. Instead of interacting through common touch-displays or buttons, the user simply speaks to the device, like to another person. To enable this, both Automatic Speech Recognition (ASR) and speech synthesis are essential technologies.

In this paper I will only focus on the speech synthesis part. A widely spread technique to synthesize human speech from a given text or from linguistic descriptions is Statistical Parametric Speech Synthesis (SPSS); also referred to as Statistical Parametric Speech Generation (SPSG) [11]. This technique is based on the usage of Hidden Markov Models (HMMs). Black *et al.* [5] show that it has several advantages over its predecessor, the concatenative speech synthesis, for example the flexibility in changing voice characteristics and a smaller memory footprint. However the quality of the generated speech still has potential for improvement. Due to over-smoothing the voice sounds muffled in comparison to natural speech.

This is where recent achievements in deep learning come in. Deep learning is usually referred to as a class of machine learning techniques that achieve tasks like feature extraction or pattern analysis by using many connected layers of non-linear information processing [8, 11]. Since 2006 advances in the training algorithms of Deep Neural Networks (DNNs) have enabled the field of deep learning applications to emerge [6]. Most machine learning models until then had used shallow structures, like for example HMMs, Gaussian Mixture Models (GMMs), Conditional Random Fields (CRFs) or Support Vector Machines (SVMs). In these structures only one layer is responsible for generating features out of the raw input signals. While achieving quite good results with rather simple problems, they reach their limit when it comes to more complex tasks like processing human language or natural images [8]. In the tutorial survey [8] the author also states four different approaches to improve speech synthesis through deep learning models, whereof three are dealing with SPSS. One of those three approaches is described in [15], where the authors implemented a part of the speech synthesis system by using a DNN and observed an improved performance in predicting output features. In [9] a more general approach is conducted by investigating what effects the deployment of a DNN on different parts of the SPSS system has. An improvement of the naturalness of the generated speech was one of the main results.

For implementing speech synthesis on resource-constrained devices like smartphones or tablets, SPSS is considered the best solution due to the trade-off between voice quality and acceptable footprint size [13]. Since the computational costs of SPSS are often high, some optimization steps like applying fewer conditional calls are conducted in [13] to make the HMM-based speech synthesis technique SPSS more suitable for mobile devices. Going one step further, in [6] an approach to adapt SPSS for embedded devices by using a deep learning model, an Auto Encoder (AE), is employed. Four tasks (syllabification, phonetic transcription, part-of-speech tagging and lexical stress prediction) are examined and tested with the use of this deep learning model. As results the authors highlight hugely reduced model sizes, higher training times, very close performance and a similar run time in comparison to the state-of-the-art models. This shows that the usage of deep learning models for speech synthesis on embedded systems is a reasonable step, not only to improve performance and voice quality, but also towards the independence on online databases for speech synthesis.

The remaining paper is structured as follows: Section 2 first states the motivation, why speech synthesis is a useful technology. Then it describes the conventional approach without deep learning models for speech synthesis and gives an overview of advantages and drawbacks of the used models and techniques. This is followed by a brief explanation of the probably most common used technique SPSS, where the paper [5] has been chosen as commonly cited reference. Thereafter two possibilities how SPSS can be improved by deploying deep learning models are characterized, wherefore the papers [9, 15] are reviewed. In Section 4 the motivation, why speech synthesis is important on embedded or mobile devices is given, followed by two examples on how speech synthesis can be implemented on an embedded system, once without [13] and once with deep learning models [6]. Finally Section 5 summarizes the essential points of this paper and gives some future directions.

2 CONVENTIONAL SPEECH SYNTHESIS

Short text with content of this chapter

2.1 Motivation & Approaches

Speech synthesis has emerged over the last 10 years due to a vast contribution by the global community of researchers and the increasing computational power for data processing. Its quality and naturalness has increased steadily and different approaches have been developed so far [12]. The typical applications like navigation systems in cars or telephone-based dialogue systems are nowadays

widely established. But also as reading aid for visually impaired people [1] or as in the case of the famous scientist Stephen Hawking, who has been using a synthesized voice to communicate since 1997 [3], speech synthesis has proven to be very useful. Another very interesting application of speech synthesis is shown in [XX]. The author proposed to introduce synthetic speech as means of communication between pilots, since there have been many accidents due to misunderstandings at radio-based communication.

According to [10] speech synthesis can be divided into three types: Canned speech, Context-to-Speech (CTS) and Text-to-Speech (TTS). Canned speech more or less is the play back of prerecorded spoken sentences or words with none or very little adjustments. A typical example are the announcements on train stations. Because of the high effort of recording everything (almost) exactly as it is played back, this approach is limited to only a few simple applications. With CTS the waveform is generated out of a linguistic description without any information of the respective text. In this way no natural language processing is required, but nevertheless CTS nowadays has not made any important impact. The last and most promising type is TTS.

A TTS system consists of a Natural Language Processing (NLP) part, where the text is analyzed and the word and sentence structure and accents are extracted. In the next step these accents are used to generate the prosody of the given text like duration, intensity and pitch. Then the created phonetic representations with prosody information are stringed together to a continuous stream of signal parameters. The last task, the speech generation, uses this stream to generate the respective waveform. This function block can be implemented in different ways. In [10] three general approaches are named as follows: Parametric Speech Synthesis (formant-based synthesis), Concatenative Speech Synthesis (unit-selection synthesis) and Statistical Parametric Speech Synthesis (HMM-based synthesis). The methods in brackets are the respective implementations, which are most commonly used.

The formant-based synthesis is the oldest approach. To generate a voice waveform, an excitation signal is fed into multiple formant filters which describe the characteristics of the human vocal tract. The output of the filters then forms the voice waveform. This technique is the only one which does not need any recorded speech, but instead generates the synthesized voice only by modeling the human vocal tract. The quality of the generated voice is the lowest in comparison to the other techniques, but therefore formant-based synthesizer have the smallest footprint and the voice characteristics can easily been modified by just changing the filter parameters [10].

With the development of concatenative speech synthesizer the quality of the generated speech improved tremendously. Very similar to CTS prerecorded speech is used as reference. Very basically said, the recorded speech is divided into units and these units are then stringed together to form the new speech signal according to a given text. Hereby the chosen size of the units determines both the footprint size and the voice quality. With larger units a higher voice quality can be achieved, but this also results in a much larger database. The challenge in unit-selection is to ensure, that the transissions between the units are as natural as possible [10]. In Section 2.2 some concepts on how to achieve this as well as the third implementation, the HMM-based synthesis, a specific instance of SPSS are described in detail.

2.2 HMM-based Speech Synthesis

In this section the HMM-based, the most recent approach for speech synthesis will be described further and both advantages and drawbacks compared to unit-selection synthesis will be highlighted. Therefore the work of Black *et al.* [5] will be taken as reference.

The quality of unit-selection synthesis directly depends from the quality of the prerecorded speech. But even with a database of excellent quality sporadic errors can still not be avoided totally. If a specific phonetic or prosodic part of a generated sentence is not well represented in the database the output quality of this sentence suffers immensely. To try to avoid this a huge effort in specifically designing the database for the required application can be performed, but still there is no guarantee that such bad joins happen. In addition the fact, that in unit-selection no or only very little adaptions of the voice characteristics are possible without an enormous increase of the database size, the ambition towards seamless speech synthesis leads to the HMM-based approach. There a statistic representation of some sets of speech segments is used to generate arbitrary synthetic speech.

Details about unit-selection synthesis ???

In Figure 1 the structure of a typical HMM-based synthesizer is shown. The whole system can be divided into two parts, the training and the synthesis part. The connection of these two parts is a set of context-dependent (what is that?) HMMs. In the training part spectrum and excitation parameters of the recorded speech are used to generate acoustic models represented by the HMMs. Thereby phonetic, linguistic and prosodic parameters are considered. Details about spectrum and excitation parameters? In comparison to a unit-selection system the database only is needed in the training part.

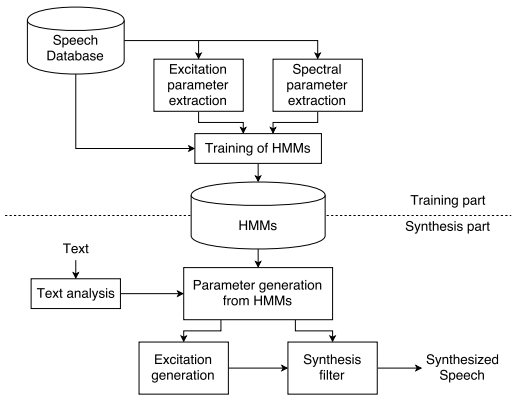

Figure 1: Function blocks of HMM-based synthesis [5]

In the synthesis part first the text which is to be synthesized is transformed to a sequence of parameters containing information about the context (what context?). According to this sequence the respective HMMs (what is that?) are concatenated in order to form an utterance HMM. Then after determining the state durations of the HMMs a sequence of coefficients is created, which finally is used to construct the speech waveform using a special filter (details?).

The main disadvantage of this approach compared to unit-selection synthesis is the quality of the synthesized speech. Three factors are accountable for the lack of quality: the vocoder, the modeling accuracy, and an effect called over-smoothing (details?).

However there are some essential advantage, which makes the HMM-based approach a competitive alternative to unit-selection synthesis. First, the voice characteristics can be modified without much effort Thus the implementation of different languages and the realization of different speaking styles with emotional emphasize is possible. Second, these just named aspects require a much smaller

database than in unit-selection synthesis, since only a statistic representation of speech segments rather than raw speech data is stored.

In Table 1 the up to here mentioned techniques are compared regarding the most prominent advantages and drawbacks.

Table 1: Comparison of speech generation methods [5, 10]

Technique	Advantages	Drawbacks
Formant-based	No prerecorded speech required	Very artificial and metallic voice
Unit-selection	Very high voice quality possible	Large database required
HMM-based	Adjustable voice and small footprint	Voice sounds muffled

3 SPSS WITH DEEP LEARNING MODELS

Although, the approach of SPSS has brought many advantages over unit-selection synthesis, as shown in the second chapter, the generated voice is still not as natural as desired. Therefore deep learning models recently have been used to further improve SPSS. Since DNNs have proven to be very effective in speech recognition, they have found their way into speech synthesis. With the use of DNNs it is possible to represent higher dimensional and correlated features in an efficient way as well as compactly modeling complex mapping functions. Exactly these properties can be used in SPSS, where different features representing the context (linguistic, prosody, etc.) have to be considered for the acoustic modeling [9].

In this section first different approaches of deploying deep learning models are investigated, as shown in [9] and then one specific approach is explained further, wherefore [15] is used as reference.

3.1 One specific approach for improvement

In the previous section we learned that the quality issue of HMM-based speech synthesis is caused by three aspects, the vocoder, the accuracy of acoustic models and the over-smoothing effect. Zen *et al.* [15] suggest a specific approach to eliminate one of these causes, the accuracy of acoustic models, by allocating this task to a DNN. In conventional HMM-based systems the mapping between context features (phonetic and linguistic properties) and speech parameters is done by decision tree based context clustering. Thereby the context-dependent HMMs are assigned to different clusters depending on the combination of contexts using binary decision trees. Each cluster is characterized by a specific set of speech parameters. In this way it is possible to estimate all HMMs in a robust way, with a typically sized training database.

However decision trees soon reach their limits when handling complex contexts. Only by increasing the size and in this way decreasing the efficiency of the decision tree more complicated contexts (*e.g.* XOR) can be dealt with.

In addition to that decision require partitioned input data with each partition having a different set of parameters. In this way whith less data per region overfitting (weak generalization) is likely to happen, which then results in a lack of quality. These downsides can be avoided by using a DNN instead of multiple decision trees.

But the use of a DNN also introduces two disadvantages: The first one arises in terms of computational power. Both in the training and in the prediction stage decision trees require much less operations (total amount and level of complexity) than DNNs. The second one has to do with the decision process in its basic form. With decision trees a binary question has to be answered, whilst a DNN consists of weighted neurons, which use non-linear activation functions (*e.g.* sigmoid, tanh, ReLU [7]) to determine their state. As consequence of that interpretable rules are far easier to produce with decision trees, than with DNNs.

In Figure 2 the structure of a DNN-based speech synthesis system is shown. First a sequence of input features is generated after analyzing the input text. These parameters contain numeric values like the number of words in a sentence or the duration of a phoneme as well as binary answers to questions like ”is the current phoneme *aa*?”. Then this parameter sequence is fed into the DNN where a mapping to output features is deployed by using forward propagation. The DNN has to be trained sometime before with pairs of input and output features from a database. In the following steps the speech parameters are extracted from the statistics of the output features and the voice waveform in turn generated from the speech parameters. this is done in the same way as in the HMM-based system. For this system the function blocks of text analysis, parameter generation, and waveform synthesis can be reused from a HMM-based system. Only the mapping from input features (*e.g.* linguistic contexts) to output features (spectral and excitation parameters) is implemented in a different way.

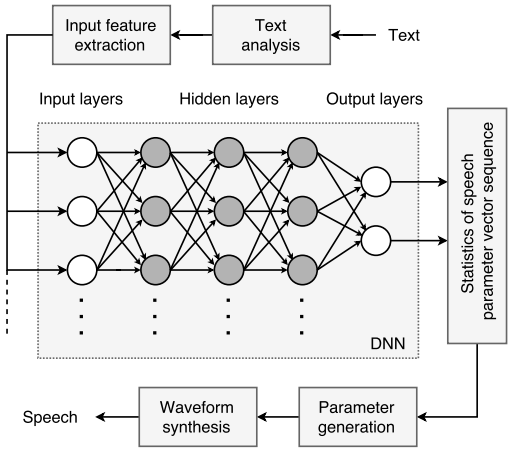

Figure 2: Speech synthesis based on DNN [15]

To compare the output of the above describe framework with that of a HMM-based, Zen *et al.* conducted some experiments with each a HMM-based and a DNN-based speech synthesis system in [15]. Therefore they used the same speech data in US English which includes about 33 000 utterances for both systems.

The HMM-based system used 2 554 questions for the decision tree-based context clustering. To influence the size of the decision trees the scaling factor α is used, where a high value results in a small decision tree and $\alpha = 1$ denotes a typical HMM-based system.

342 binary features (*e.g.* phonemes identities) and 25 numerical features (*e.g.* number of syllables in a word) represent the input features of the DNN-based system. As activation function the sigmoid function is used, since the authors experienced superior performance of this type in previous tests. In total one network with different number of layers (1, 2, 3, 4, or 5) and units per layer (256, 1 024, or 2 048) are used.

The objective evaluation showed that the DNN-based system achieved better performance in voiced/unvoiced classification and aperiodicity prediction, regardless of the number of layers or the

units per layer. At the Mel-cepstral distortion only DNNs with three or more layers outperformed the HMM-based system[1].

As subjective evaluation 173 test sentences each synthesized with the DNN- and the HMM-based systems were played back to a number of listeners, who then choose which synthesis they preferred. If no difference is perceived the option "neutral" can be selected. In this test the same number of parameters for both systems are used. For the structure of the DNN-based approach 4 layers with a different number of units per layer were applied. In Table 2 the outcome of the subjective evaluation is shown. Unmistakably the speech generated with the DNN-based systems were preferred, regardless of the amount of units per layer. The listeners described them as less muffled.

Table 2: Subjective scores (in %) of speech samples in [15]

HMM (α)	DNN (layers × units)	Neutral
15.8 (16)	38.5 (4 × 256)	45.7
16.1 (4)	27.2 (4 × 512)	56.8
12.7 (1)	36.6 (4 × 1 024)	50.7

In conclusion it can be stated, that a DNN-based approach to implement the acoustic model[2] of a speech synthesis system is a reasonable alternative to the conventional decision tree-based strategy. The improved performance for predicting spectral and excitation parameters and the more natural sounding voice both show the potential of the DNN-based approach for speech synthesis. However the higher computational costs both at training and prediction stage due to more complex arithmetic operations in the DNN-based approach indicate where future work should be focused.

3.2 Other ways for improvement

The previous section focused on the impact a DNN has by representing the acoustic model of a speech synthesis system. However deep learning models can be deployed in other parts of SPSS as well. In [9] different ways on how to implement a DNN into a speech synthesis are investigated. Therefore the different parts of a HMM-based speech synthesis system once are modeled with a DNN and once with the conventional technique and then the results are compared in an objective and a subjective fashion.

Hashimoto *et al.* concentrated on two core components of a speech synthesis system, the acoustic models and the speech generation part. In Figure 3 you can see the simplified structure of a speech synthesis system. In the first block the text is analyzed and the contextual features are extracted (A). These are then converted to static and dynamic acoustic features (B) by the acoustic models. The parameter generation block then uses these acoustic features to create speech parameters (C). In the last step the speech waveform is synthesized.

The two gray-colored blocks in Figure 3 are subject of two experiments conducted in [9]. The conventional approach for the acoustic models is the use of decision tree clustered HMMs, whereas the parameter generation usually is implemented by a Maximum Likelihood Parameter Generation (MLPG) algorithm. As seen in Section 3.1 the acoustic models can be represented by a DNN. Hashimoto *et al.* also included this approach in their experiments but furthermore used a DNN for the speech parameter generation task.

[1] **further explanation**
[2] **other word here???**

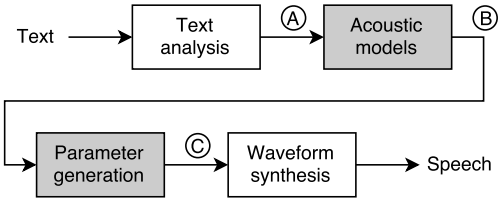

Figure 3: Simplified structure of speech synthesis system [9]

In Table 3 the systems resulting of the different combinations of the deployment of either conventional or DNN-based approach are shown. The systems I - IV are part of experiment 1 and all systems except system IV are part of experiment 2.

Table 3: Different systems within the experiments [9]

System Nr.	I	II	III	IV	V	VI
Experiments	1 & 2	1 & 2	1 & 2	1	2	2
Acoustic models	HMM	HMM	DNN	DNN	HMM	HMM
Speech generation	MLPG	DNN	MLPG	DNN	DNN + MLPG	MLPG + DNN

In experiment 1 all DNNs are equipped with three hidden layers and different number of units per layers (256, 512, or 1 024) and the size of the decision trees was controlled similar as in [15]. As objective evaluation only the Mel-cepstral distortion was used. Here the systems with a DNN (system II - IV) all outperformed system I in a similar way, as soon as they had 512 or more units per layer. For the subjective evaluation 20 sentences were played back to 8 listeners who had to assign a naturalness score between 1 and 5 (with 5 being the most natural). After applying the mean opinion score test method, system III was most preferred with 3.53 followed by system IV with 3.17 and system I with 3.08. The less natural sounding voice accoring to the listeners was produced by system II. From these results it can be concluded, that the speech generation part implemented with MLPG (system I & III) produces a more natural output than those with a DNN (system II & IV) [3].

Due to these outcomes experiment 2 introduces two new test cases, system V and system VI. For the parameter generation task a combination of MLPG and DNN is used, for which the respective input and output features where adapted. Again once an objective and a subjective evaluation like in experiment 1 were conducted. The objective evaluation basically showed the same trend as in experiment 1, as soon as 512 ore more units per layer were used, the Mel-cepstral distortion was less than with the conventional approach (system I). In the subjective evaluation the two new systems (V & VI) both were preferred over system I, with system VI being the most preferred system of this experiment. System III again was better and system II again was evaluated less natural than system I.

As conclusion, we see, that the experiments conducted in [9] confirmed the results of [15], that the decision tree-clustered HMMs can be replaced by a DNN to improve both the prediction performance as well as the naturalness of the generated speech. Beyond that Hashimoto *et al.* demonstrated, that it is worthwhile to go one step further and deploy an additional DNN as post-filtering block into the parameter generation task of the speech synthesis system to even further improve the above named results.

[3] **Reasons: parameter trajectories**

4 TTS ON MOBILE DEVICES

In this section two approaches to optimize speech synthesis for mobile devices are presented. In the first some optimization steps on a conventional HMM-based system are suggested [13] whereas the second approach introduces deep learning models to get rid of the dependence on internet-based services.

4.1 Motivation and Challenges

During the last 10 years, advances in technology have led to a tremendous spread of mobile devices, especially smartphones. According to [2] there are 2.1 billion smartphone users worldwide in 2016 which is forecasted to increase to almost 3 billion in 2020. In [2] it also states, that only in 2016 about 1.5 billion new smartphones were sold. From these numbers we can derive, that already today and with increasing significance in the near future, smartphones constitute an essential part of our daily life.

Especially on smartphones, where the visual output is restricted to a rather small screen, usually not bigger than 5 inch in diagonal, speech enabled interaction can improve the user experience in a significant way. In [13] the following three application scenarios are pointed out: Speech interaction can be used as an extension to existing communication channels, speech can be deployed as main output, if other outputs are restricted (*e.g.* while driving), or speech help visually impaired or blind people to interact with a system at all (*e.g.* screen readers).

However, when implementing a speech synthesis application on a mobile device, the increased computational power and storage capacity on recent smartphones can not be used as in desktop devices. One aspect is the power consumption. An application on a smartphone should use as less processing power as possible to avoid shortening the battery life unnecessarily [13]. Another issue is the restricted amount of main memory (16 - 32 MB RAM per app), which forces each application to be as resource-saving as possible. While accomplishing these demands, the output speech still needs to be processed in real-time, to ensure a good user experience [6].

4.2 Optimized HMM-based Synthesis

One approach to optimize HMM-based speech synthesis for mobile devices can be seen in [13]. Here Tóth *et al.* conduct a study to optimize a HMM-based speech synthesis system in terms of computational power, playback latency and footprint size while ensuring a reasonable quality of the synthesized speech. In the following this approach is explained further.

As already pointed out in Section 2.1 HMM-based speech synthesis offers the best trade-off between speech quality and footprint size. This surely makes it the most appropriate approach to be deployed in mobile systems. Tóth *et al.* indicate, that there already have been several approaches to optimize HMM-based speech synthesis, but they claim to be the first, who emphasize the optimization steps on resource constrained platforms to achieve a reduction in computational costs while still fulfilling real-time responsiveness.

To measure the processing time of the speech synthesis procedure, Tóth *et al.* divide the TTS-process in three steps: The loading of the HMM database[4] into main memory (1), the speech parameter generation (2), and the waveform synthesis (3). As test sentence a 17 seconds sequence is chosen, although it is unlikely that such a long utterance has to be synthesized often in real life (since in practice segments with 1 - 5 seconds are synthesized). But in this way it can be ensured, that shorter sequences are likely to be synthesized as desired, if this long sentence can be dealt with adequately.

[4] **explanation**

The following four approaches to optimize HMM-based TTS are suggested by Tóth *et al.*:

1. **Vocoder parameters**
2. **Size of decision trees**
3. **Streaming synthesis**
4. **Source code optimizations**

In the first approach the conventional Mel-Log Spectral Approximation (MLSA) filter used for speech parameter generation is replaced by a Line Spectral Pair (LSP) filter. In this way the performance in step (2) can be enhanced. Different orders of the LSP filter are tested to compare speech subjective quality, with an order of 18 being the conventional setup and 14, 12, and 10 being the optimization approaches.

The second approach dealt with the size of the decision trees. While lowering the number of nodes of a decision tree reduces the quality of synthesized speech, it also decreases the memory footprint and the computational costs. Therefore three different test settings were defined (see Table 4).

Table 4: Test settings [13]

	Nodes in decision trees	Footprint size
Original	6983	666 KB
Setting 1	4762	463 KB
Setting 2	2743	214 KB
Setting 3	1273	140 KB

General speech synthesis frameworks usually do not include the audio playback functionality to ensure platform independence. Implementing this part allows streaming synthesis. In this way one already synthesized segment is played back while the next segment is generated. While long segments cause an undesired latency, too short segments result in uncontinuous audio playback. The optimal segment size is determined at run-time to achieve a compromise between low as possible latency and low as possible computational effort.

Low-resource devices differ from high-end devices not only in the amount of available memory and the computing performance, but also in the chip architecture. Memory management, fixed- versus floating-point calculation and conditional call management are points to be aware of.

The above described optimizations were tested on three different smartphones with increasing CPU speed and main memory, starting with the "weakest": an Apple Iphone 3G, a Samsung Galaxy Spica (GT-i5700) and a HTC Desire (A8181). Therefore the above described optimization steps were tested one by one, since they do not affect each other noteworthy.

For the first approach a LSP filter with 12th order was chosen as the optimal setting in terms of latency and subjective speech quality. Concerning the size of the decision trees, setting 1 clearly proved to be the optimal setting, since the time to load the HMM database (1) was decreased to half as before without notable impaired speech quality. The streaming synthesis approach tremendously decreased step (1), with an improvement from the order of seconds to 8 ms without affecting the speech quality at all. Concerning the source code optimizations, almost no optimizations have been conducted. Only some conditional calls were removed, which resulted in a insignificant improvement of latency. In total the computation time could be decreased by 65% without a negligible decrease of speech quality.

4.3 Deep Learning-based Approach — ??? —

5 CONCLUSIONS

Here the core points will be repeated and concluded.
Some future aspects will be highlighted.
What should be done in the future?

See [14]

- Voice cloning
- Voice reconstruction
- Personalised speech-to-speech translation
- Articulatory-controllable speech synthesis

REFERENCES

[1] 2017. The Benefits of Text to Speech. (02 June 2017). http://www.readspeaker.com/benefits-of-text-to-speech/

[2] 2017. Number of smartphone users worldwide. (10 June 2017). https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[3] 2017. Stephen Hawking - The Computer. (02 June 2017). http://www.hawking.org.uk/the-computer.html

[4] 2017. Survey on usage of virtual personal assistants by Gartner. (22 May 2017). http://www.gartner.com/newsroom/id/3551217

[5] Alan W. Black, Heiga Zen, and Keiichi Tokuda. 2007. Statistical Parametric Speech Synthesis. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, Vol. 4. IV–1229–IV–1232. https://doi.org/10.1109/ICASSP.2007.367298

[6] Tiberiu Boroş and Stefan Daniel Dumitrescu. 2015. Robust Deep-learning Models for Text-to-speech Synthesis Support on Embedded Devices. In *Proceedings of the 7th International Conference on Management of Computational and Collective intElligence in Digital EcoSystems (MEDES '15)*. ACM, New York, NY, USA, 98–102. https://doi.org/10.1145/2857218.2857234

[7] Hoon Chung, Sung Joo Lee, and Jeon Gue Park. 2016. Deep neural network using trainable activation functions. In *2016 International Joint Conference on Neural Networks (IJCNN)*. 348–352. https://doi.org/10.1109/IJCNN.2016.7727219

[8] Li Deng. 2014. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing* 3 (January 2014). https://doi.org/10.1017/atsip.2013.9

[9] K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda. 2015. The effect of neural networks in statistical parametric speech synthesis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4455–4459. https://doi.org/10.1109/ICASSP.2015.7178813

[10] Florian Hinterleitner. 2017. *Quality of Synthetic Speech: Perceptual Dimensions, Influencing Factors, and Instrumental Assessment (T-Labs Series in Telecommunication Services)* (1st ed. 2017 ed.). Springer, Singapore. https://doi.org/10.1007/978-981-10-3734-4

[11] Z. H. Ling, S. Y. Kang, H. Zen, A. Senior, M. Schuster, X. J. Qian, H. M. Meng, and L. Deng. 2015. Deep Learning for Acoustic Modeling in Parametric Speech Generation: A systematic review of existing techniques and future trends. *IEEE Signal Processing Magazine* 32, 3 (May 2015), 35–52. https://doi.org/10.1109/MSP.2014.2359987

[12] David Suendermann, Harald Höge, and Alan Black. 2010. *Challenges in Speech Synthesis*. Springer US, Boston, MA, 19–32. https://doi.org/10.1007/978-0-387-73819-2_2

[13] Bálint Tóth and Géza Németh. 2012. Optimizing HMM Speech Synthesis for Low-Resource Devices. In *Journal of Advanced Computational Intelligence and Intelligent Informatics*. 327–334.

[14] Junichi Yamagishi. 2011. New and emerging applications of speech synthesis. (February 2011). www.cstr.ed.ac.uk

[15] Heiga Zen, Andrew Senior, and Mike Schuster. 2013. Statistical parametric speech synthesis using deep neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 7962–7966. https://doi.org/10.1109/ICASSP.2013.6639215

[16] Heiga Zen, Keiichi Tokuda, and Alan W. Black. 2009. Statistical parametric speech synthesis. (April 2009), 1039 - 1064 pages. https://doi.org/10.1016/j.specom.2009.04.004

Remarks starting here:

Page count estimation:

- Section 1 1 (incl. Abstract)
- Section 2.1 0.75
- Section 2.2 0.5
- Section 3.1 0.5
- Section 3.2 0.5
- Section 4.1 0.25
- Section 4.2 0.75
- Section 4.3 0.75
- Section 5 0.25
- References 0.25

Total: **6 pages ???**

To do:

- cite both long and short version of [16]
- check main title
- check (sub)section titles
- check references (bibtex warnings ???)
- check correct citing
- check numbers (written out or as digit)
- check for typos
- insert footnotes if further explanations are necessary
- keywords necessary ???
- check page count (6 pages)
- disable sreen mode on last draft
- explain acronyms first time they appear in abstract AND first time they appear in body

Remarks for section 2.2:

- (Relationship between the two approaches)
- (Hybrid approaches)
- Why there is need to further improve this technology?
- **Change functional diagram -> more basic & and only synthesis part ??**
- What is a HMM?
- Bring up MLPG (Maximum Likelyhood Parameter Generation)
- Bring up Decision Tree
- Robustness as additional advantage (source [10] in [15])

Questions:

(1) Why speech synthesis is important? What are its applications?
(2) What are the conventional techniques of speech synthesis? What are the drawbacks of such techniques?
(3) What is deep learning? What improvements do deep learning algorithms bring?
(4) How some algorithms are modified to suit speech synthesis?
(5) Why is it important to implement speech synthesis on embedded platform?
(6) An example of how speech synthesis can be implemented on embedded platform without deep learning.
(7) How the 3 can be combined?
(8) Future works.