

**School of Energy and Electronic Engineering**

Name: Yordanka Popova

Student ID: 782716

Project Title: Computer-synthesised Singing using Neural Networks

Course: BEng Computer Engineering

Supervisor: Branislav Vuksanovic

Academic year: 2018/19

****

DECLARATION

Declaration of Originality and Approval of Research Ethics

Project Title: Computer-synthesised Singing using Neural Networks

Student Name: Yordanka Angelova Popova

HEMIS Number: 123456

‘I certify that this is my own work, and it has not previously been submitted for any assessed qualification. I certify that School of Engineering research ethics approval has been obtained and the use of material from other sources has been properly and fully acknowledged in the text’.

Signed:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Dated: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ABSTRACT

This project presents a comparative study of the most recent speech generating Deep Learning architectures as a step towards an end-to-end singing synthesiser based on Machine Learning and existing music from any particular musical artist to allow the creation of new songs using old samples.

AKNOWLEDGEMENTS

This is optional. Please remove if you do not use it.

Contents

[DECLARATION i](#_Toc1131030)

[ABSTRACT ii](#_Toc1131031)

[AKNOWLEDGEMENTS iii](#_Toc1131032)

[Table of Figures v](#_Toc1131033)

[Table of Tables v](#_Toc1131034)

[SYMBOLS AND ABBREVIATIONS v](#_Toc1131035)

[1. INTRODUCTION 1](#_Toc1131036)

[2. LITERATURE REVIEW 2](#_Toc1131037)

[2.1 First Subchapter 2](#_Toc1131038)

[2.1.1 First Sub-subchapter 2](#_Toc1131039)

[2.2 Second Subchapter 2](#_Toc1131040)

[3. Analysis and design 2](#_Toc1131041)

[3.1 subchapter/section for requirements analysis 2](#_Toc1131042)

[3.2 subchapter/section for design 2](#_Toc1131043)

[4. Implementation/modelling/simulation 2](#_Toc1131044)

[5. TEST and Discussion 3](#_Toc1131045)

[4.1 Test 3](#_Toc1131046)

[4.2 Discussion 3](#_Toc1131047)

[6. PROJECT EVALUATION/CONCLUSION 3](#_Toc1131048)

[REFERENCES 3](#_Toc1131049)

[LIST OF ACHIEVEMENTS 3](#_Toc1131050)

[Project Achievements 3](#_Toc1131051)

[Learning Achievements 3](#_Toc1131052)

Table of Figures

[Figure 2.1. Figure caption should be meaningful, and self-explanatory. The first number of the figure should be the number of the MAIN chapter. 2](#_Toc508362883)

Table of Tables

[Table 3.1. Table Caption above the table. 3](#_Toc508362889)

SYMBOLS AND ABBREVIATIONS

List any symbols and abbreviations (ALPHABETICALLY) you have used in your work, eg.;

*CNN* Convolutional Neural Network

*HMM* Hidden Markov Model

TTS Text-to-Speech

Etc.

# INTRODUCTION

Background

Project motivation

Aims and objectives

# LITERATURE REVIEW

## 2.1 Neural Networks and Deep Learning

## 2.2 Text-to-Speech Synthesis

Text-to-Speech is the automatic conversion of written to spoken language and usually consists of

two parts – input in the form of text and waveform output.

Vocoding is an idea that allows a speech signal to be converted into some form of a representation that can be transmitted or stored, or in other words – parametrising speech.

### Training

Training is one of the main terms mentioned when it comes to Machine Learning. In this specific case, most Text-to-Speech systems share the same idea of “training” – extracting linguistic specifications from a collection of samples (also known as the “training data(set)”) and indexing them the new stored form of the speech, to allow synthesis to be performed after by having the linguistic specifications as input.

The linguistic specifications that the model decides to take into consideration can completely change the output produced. Some examples of context factors are preceding and following phonemes, position of segment in syllable, distance from stressed/accented syllable. Deciding on which ones to use and modelling the attention around them depends entirely on the desired output and vary completely from system to system. For example, in attempts to produce speech that is conditioned on emotions, different factors are considered as opposed to producing s

### 2.2.1 Concatenative methods

Before machine learning, concatenative methods were the gold standard for speech synthesis. This technique was first introduced all the way back in the early 90s and is used by the more successful singing synthesisers such as Vocaloid (INTERSPEECH, 2007), which is developed by the Yamaha corporation. Using a large database of samples extracted from singing voices. Even thought that has been a largely successful product, concatenative methods have their downsides. As mentioned, these systems must rely on extremely large databases in order to produce natural sounding speech or singing. As such, they are very rigid and are difficult to improve upon.

### 2.2.2 Generative methods

Statistical parametric speech synthesis has allowed for a better, more advanced way of synthesising speech, leveraging Deep Neural Networks (DNNs), resulting in a much more flexible and natural sounding speech.

These methods rely on numerical approximations to speech in their attempt to recreate how humans produce air signals. They are often called “HMM Synthesis” due to them using Hidden Markov Models or other very similar methods.

The advantage of these generative methods over concatenative ones is the ability to “iron out” inconsistencies in the data.

### 2.2.3 Datasets

In both described methods, huge amounts of data are required. Even though generative methods require “smaller” datasets, in all of the researched examples that usually means from 24 hour long recordings to 100 hours of singing when it comes to concatenative methods.

## 2.3 Vocal separation

# Analysis and design

## 3.1 subchapter/section for requirements analysis

## 3.2 subchapter/section for design

# TEST and Discussion

## 4.1 Test

Below are described the tests run, using the most popular open source implementations of each of the three architectures.

## Data preparation

After evaluating different methods of vocal separation on musical tracks, PhonicMind was identified as the best solution. It does not compromise on quality and produces the best possible outcome out of all the tested methods (filters, Neural Network implementations).

### 4.1.1 WaveNet Implementation

<https://github.com/ibab/tensorflow-wavenet>

The input to this architecture was WAV files, which made it the easiest to tend to. Traditionally, the WaveNet model was trained by Google’s DeepMind team using 44 hours of data from 109 different speakers as opposed to our dataset of 1 singer and 12 mins of audio. By comparison,

Training and generation with this particular model has been incredibly slow (especially compared to the later examples), which was to be expected with this particular architecture due to its generative nature.

Although the official WaveNet implantation was tested as a part of a Text-to-Speech system, this particular open source implementation is not conditioned on text, meaning the output is free-form speech generation in the form of a single second of audio. As a result, when put together, the samples formed non-existent but human-like words, that however did not resemble singing.

### 4.1.2 Tacotron Implementation

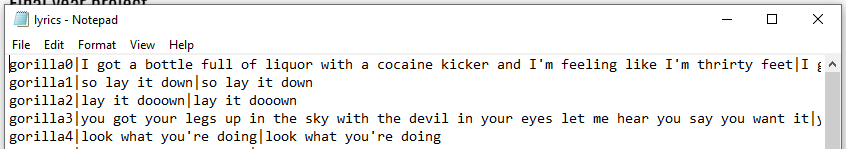
<https://github.com/keithito/tacotron>

### 4.1.2.1 Preprocessing

This current implementation of Tacotron uses indexed, transcribed audio files as a dataset. Due to the nature of the implementation, some extra work was needed.

To help with the data processing, a Python application was developed to serve three main purposes. For the first part of the data processing, the different phrases sung in the audio file have to be separated into new WAV files to allow the transcription of the different segments. Inspired by a method in Audacity, a threshold value was used to indicate silence and audio is split into chunks, removing the silence in between. Although the threshold had to be adjusted for different songs, this technique was largely successful.

For the second part, transcribing the newly separated segments, it was of big help to create an application that would index the files and allow for the manual input of the lyrics, creating the correct file format for the training set of the chosen implementation. This results in a text file with the following format:



This specific file format corresponds to what is described in a pre-processor file written originally for the Linda Johnson dataset, where all the different segments of the 24 hour long audio set are transcribed and included in a CSV file, where the first value is the segment file name, the second is the words said in the audio and the third is the clean version with no special characters that would intrude with the mapping of the rest of the characters .

Following that example, another pre-processor file was created to handle the files in this dataset, based on the same principle. This allows for the selection of a dataset when training the model, in this case a choice between the Linda Johnson 24 hour long data set,

### 4.1.3 Tacotron 2 Implementation

### 4.1.4 Training times

Training times depend on a few factors. The ones that have been considered in this case are hardware, dataset and architecture. To compare the three architectures, we measure their progress in steps, hence time/step is the measure of progress.

Initially, a university computer was used to setup one of the environments necessary to train the model. However, due to the hardware specs of the machine, a single step was taking up to 20 mins. Considering the models have been trained up to 250 000 steps, using those machines would have severely slowed down the progress on the project. Instead, a personal desktop was used at home with a newer generation processor that increased the tempo ever so slightly. To further increase the training speeds, TensorFlow GPU support was installed, which made a huge difference taking down the time to 1 step per second on average, depending on the model implementation. Even though that still required overnight runs to make significant progress, it sped up the process significantly.

The size of the dataset also helped with the relatively “short” training times, since compared to other datasets being 24 hours of audio, the Bruno Mars set used was only 12 mins long. This is also the reason why in most research studies the minimum used in terms of hardware is a Pascal architecture GPU worth up to 5000£.

Lastly, every different architecture took a different amount of time to train and generate audio. The most computationally heavy being the last implementation, that did not allow the simultaneous training of both the WaveNet model (especially due to its generative nature) and the Tacotron 2 model.

### 4.1.5 Overfitting

One of the biggest concerns and challenges with this project and in Machine Learning in general has been overfitting.

Due to the small datasets used to test the three systems described above.

What overfitting means in the context of machine learning is the model learning the detail and noise of its training set to such an extent that it starts to negatively impacting the performance of the newly generated data.

There are a few methods that could help when overfitting occurs, that have been tested in the duration of the project to try to maximise on the existing dataset. The most obvious solution to this problem is adding more data, which is the very first reason the total duration of the dataset was increased – processing more songs, increasing the context that the values were used in to allow the model to learn. However, that also introduces more noise, not all of it being helpful in out case – different pitches of the singing voice, different pace. In the end, adding more data proved to be just as harmful (e.g. in segments alignment) as it was helpful. Observation showed that adding more data increased the system’s capability to segment words, provided examples were given that added to the context of the word. Naturally, adding more data also increased pre-processing and training times.

Another method that was used was “early stopping”. All the implementations used created a checkpoint for every set number of steps, where the ability was there to listen to a produced sample and compare to previous iterations. This allowed for discovering the point from which overfitting starts occurring – after 60 000 steps on both of the Tacotron and Tacotron 2 implementations, proceeding with the training only introduced more noise to the generated singing.

## 4.2 Discussion

# CONCLUSION

Speech synthesis is a big area to explore within the given time.

The project was a success in terms of exploring the limitations and enablement of the different Machine Learning speech synthesising architectures. It provides a better insight into what is needed to create an end-to-end singing synthesising system that relies solely on musical tracks.

REFERENCES

Babuschkin, I. (2017). *A TensorFlow implementation of DeepMind's WaveNet paper* [online] GitHub. Available at: <https://github.com/ibab/tensorflow-wavenet>

LIST OF ACHIEVEMENTS

## Project Achievements

1. Successfully trained 3 different architectures using musical samples to produce a controlled output in the form of musical synthesis.

2. etc.

## Learning Achievements

1. I learned a new programming language – Python.
2. I learned how Neural Networks work.
3. I learned how to create and train Neural Networks using TensorFlow.
4. I learned how to breakdown complex Deep Learning architectures such as WaveNet and Tacotron.
5. I learned how to set up Deep Learning environments quickly and efficiently using Anaconda.