National University of Singapore

Master of Technology in Intelligent Systems

ISY 5007 Capstone Project in Intelligent

AI Name Reader Project Report

(Version 1.1, September 2021)

ISS Advisor: Professor. Wang Aobo

Submitted by Team 10: Vincent Ng, Zhou Zhe and Dong Xiaoguang

Table of Contents

1 Project Background

The AI Name Reader project is sponsored by Republic Polytechnic.

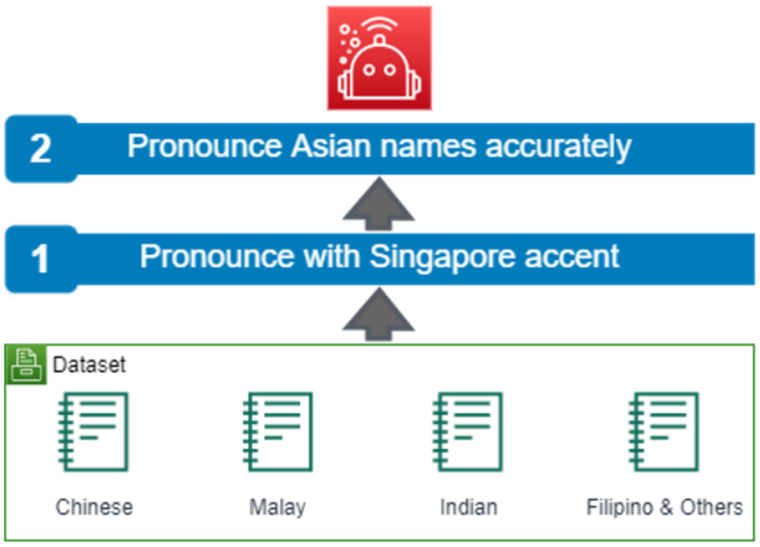
During Republic Polytechnic Graduation Ceremonies every year, student names are to be read by a staff. The staff faces significant stress reading hundreds of student names on stage. Other than that, it is easy to make mistakes when pronouncing long names or regional names from different ethnicities. Mispronouncing of student name is embarrassing for both the student and the school.

To solve this problem, Republic Polytechnic's current solution is to pre-record the names for verification and playback during ceremonies using a name reading system. This process indeed helps, but the additional recording task is tedious and time consuming.

Under such situation, Republic Polytechnic is looking for a more intelligent solution which can read the names as well as requiring less human supervision. Thus, this project is to develop a Text-to-Speech (TTS) voice model that can pronounce the Asian names of the students with a natural Singaporean voice.

2 Project Objective

To adapt to local student composition and English-speaking accent, the AI Name Reader is designed for 2 objectives.

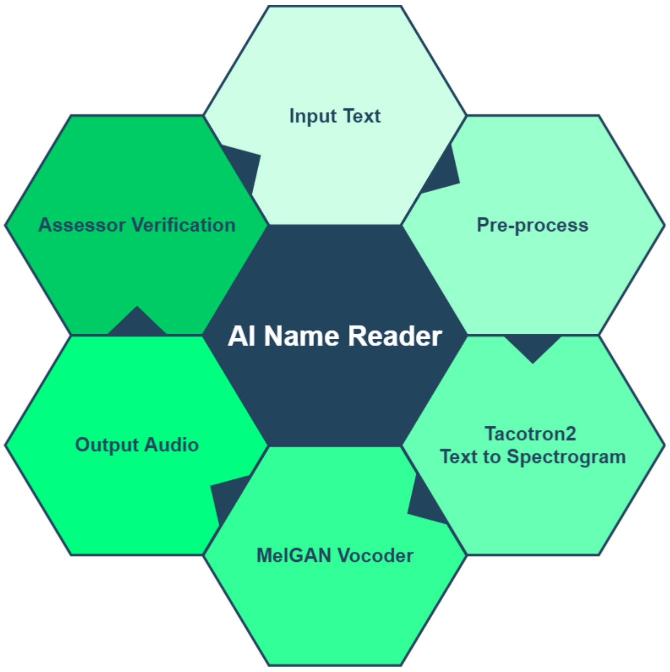

2 Pronounce Asian names accurately
1 Pronounce with Singapore accent
Dataset
Chinese
Malay
Indian
Filipino & Others

The first objective is to read the English sentences with the local Singapore accent. As the system is to be used during graduation ceremony, the pronunciation needs to be localized, or even better to be like a real Singaporean speaking. This is further separated into two parts. The first part is to train the system to pronounce like a human rather than a robot. On top of it, the second part is to let it have some Singapore accent. In this way, the synthesized audio will feel more natural to the audience.

The second objective is to pronounce Asian names accurately. Prior to developing the system, our team has studied on the ethnic composition of students from previous graduation batches. The finding is that largest student demographics are of Chinese ethnicity, followed by Malays and Indians. The rest are Filipino and others. That's also why the objective focuses on Asian name pronunciation. Among the Chinese names, some of them are in hanyu pinyin spelling while others are in Teochew, Hokkien and Cantonese dialects. For Malay and Indian names, they also have their own spellings and pronunciations. Some of them use abbreviation and periods as well.

3 Technical Design

The AI Name Reader follows normal AI project life cycle and consists of two main models which are the Tacotron 2 text-to-spectrogram model and MelGAN vocoder model.



3.1 Deep Learning TTS Model

The AI Name Reader is built based on Coqui TTS which is a library for advanced Text-to-Speech model training and audio generation. It's built on the latest research and development efforts, was designed to achieve the best trade-off among ease-of-training, speed and quality.

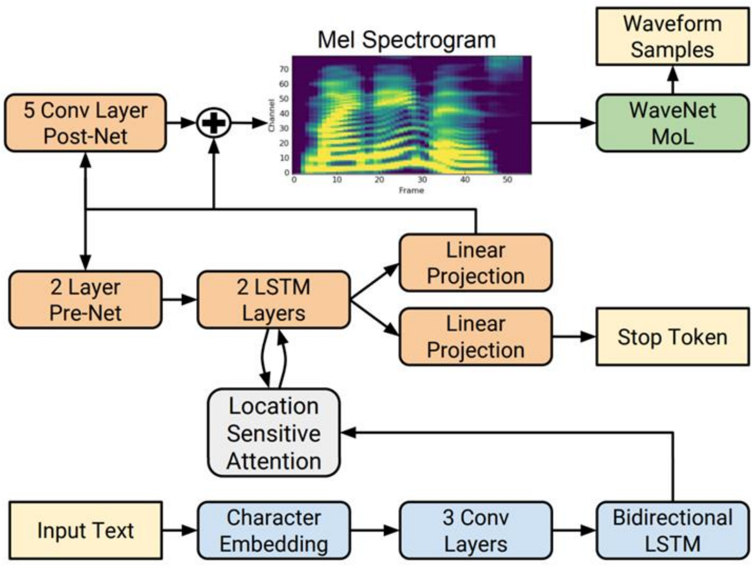
Among the Text-to-Spectrogram and Vocoder models, Tacotron 2 and MelGAN are selected in this project.

Prior to being sent into training the model, both text and audio are pre-processed. After training, an output audio is generated. The assessor will verify the audio quality and if there is an unacceptable discrepancy in terms of name reading, the assessor will update the audio by recording the names which may be used as final output as well as being used for future training input.

3.2 Tacotron 2 Text-to-Spectrogram

Tacotron 2 is a neural network for speech synthesis directly from text. The system is composed of a recurrent sequence-to-sequence feature prediction network that maps character embedding to mel-scale spectrograms, followed by a modified WaveNet model acting as a vocoder to synthesize time domain waveforms from those spectrograms.

The network is composed of an encoder and a decoder with attention. The encoder converts a character sequence into a hidden feature representation which the decoder consumes to predict a spectrogram. Input character embedding are passed through a stack of 3 convolutional layers each containing 512 filters, followed by batch normalization and ReLU activations. The output of the final convolutional layer is passed into a single bi-directional LSTM layer containing 512 units (256 in each direction) to generate the encoded features.

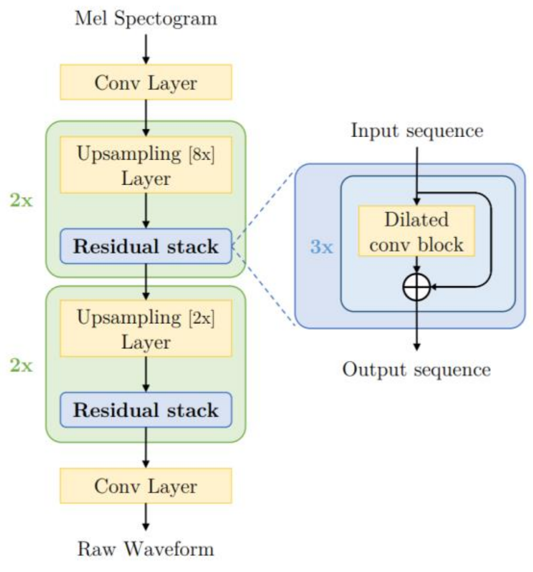


The encoder output is consumed by an attention network which summarizes the full encoded sequence as a fixed-length context vector for each decoder output step. Attention probabilities are computed after projecting inputs and location features to 128-dimensional hidden representations. Location features are computed using 32 1-D convolution filters of length 31.

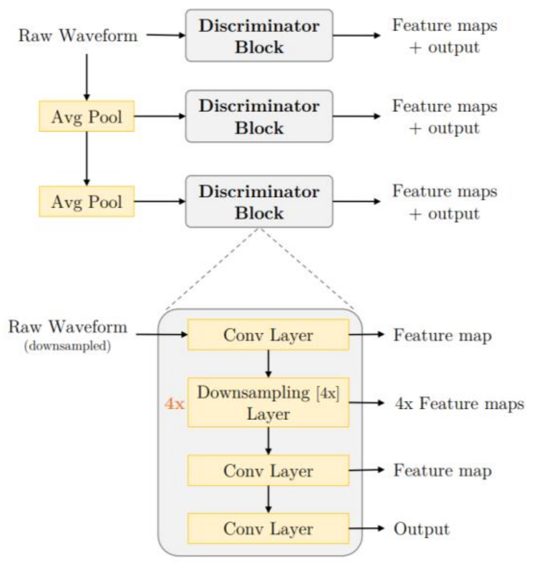
The decoder is an autoregressive recurrent neural network which predicts a Mel Spectrogram from the encoded input sequence one frame at a time. The prediction from the previous time step is first passed through a small pre-net containing 2 fully connected layers of 256 hidden ReLU units. The pre-net output and attention context vector are concatenated and passed through a stack of 2 uni-directional LSTM layers with 1024 units. The concatenation of the LSTM output and the attention context vector is projected through a linear transform to predict the target spectrogram frame. Finally, the predicted Mel Spectrogram is passed through a 5-layer convolutional post-net which predicts a residual to add to the prediction to improve the overall reconstruction. Each post-net layer is comprised of 512 filters with shape 5 × 1 with batch normalization, followed by tanh activations on all but the final layer.

3.3 MelGAN Architecture

MelGAN structure is to generate high quality coherent waveforms. It is developed on top of previous GANs model. Audio modeling is decomposed into two stages. The first stage is Tacotron 2 where models generate the intermediate spectrogram representation given text as input. The second stage, MelGAN, transforms the intermediate spectrogram representation to audio.



(a) Generator



(b) Discriminator

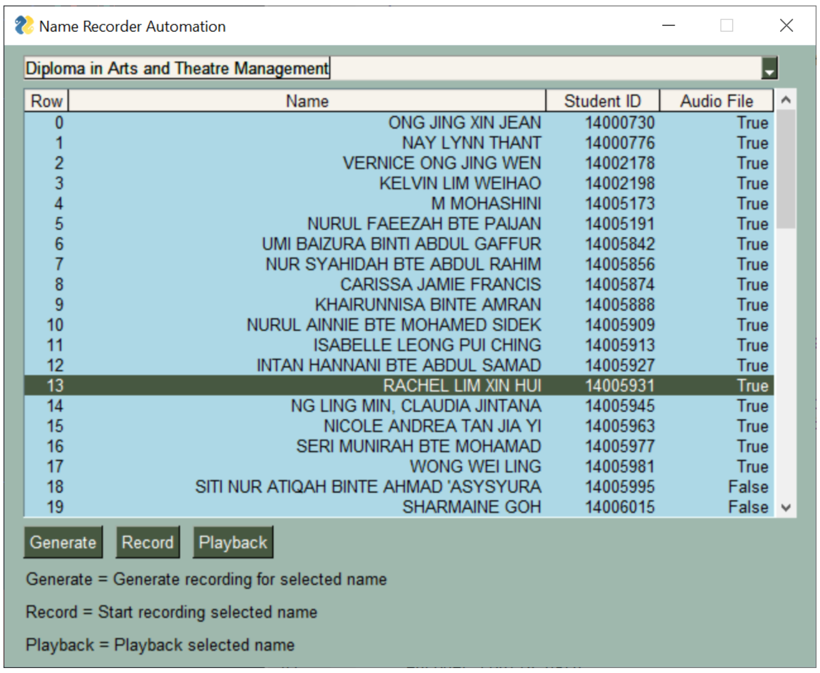
The MelGAN Architecture uses a pair of generative adversarial networks in unsupervised training.

The first part is the Generator. In this model, the generator is a fully convolutional network. Its input is a Mel Spectrogram and output is a raw waveform. The in total 256 upsampling is done in 4 stages (8x, 8x, 2x and 2x. Each residual dilated convolution stack has three layers with dilation 1, 3 and 9 with kernel-size 3, having a total receptive field of 27 timesteps. Weight normalization is used in all the layers in Generator.

Second part is the Discriminator which plays a role of scoring the synthetic audio against the real one. By minimizing the L1 distance between the fake data and the real data, the model is trained to produce a synthetic audio more and more similar to the real one, which also known as the ground-truths.

3.4 AI Name Reader GUI

Following sponsor's request, the team keeps UI simple and direct which will not need much time to learn and easy to get started.



Row	Name	Student ID	Audio File
0	ONG JING XIN JEAN	14000730	True
1	NAY LYNN THANT	14000776	True
2	VERNICE ONG JING WEN	14002178	True
3	KELVIN LIM WEIHAO	14002198	True
4	M MOHASHINI	14005173	True
5	NURUL FAEEZAH BTE PAIJAN	14005191	True
6	UMI BAIZURA BINTI ABDUL GAFFUR	14005842	True
7	NUR SYAHIDAH BTE ABDUL RAHIM	14005856	True
8	CARISSA JAMIE FRANCIS	14005874	True
9	KHAIRUNNISA BINTE AMRAN	14005888	True
10	NURUL AINNIE BTE MOHAMED SIDEK	14005909	True
11	ISABELLE LEONG PUI CHING	14005913	True
12	INTAN HANNANI BTE ABDUL SAMAD	14005927	True
13	RACHEL LIM XIN HUI	14005931	True
14	NG LING MIN, CLAUDIA JINTANA	14005945	True
15	NICOLE ANDREA TAN JIA YI	14005963	True
16	SERI MUNIRAH BTE MOHAMAD	14005977	True
17	WONG WEI LING	14005981	True
18	SITI NUR ATIQAH BINTE AHMAD 'ASYSYURA	14005995	False
19	SHARMAINE GOH	14006015	False

Generate = Generate recording for selected name

Record = Start recording selected name

Playback = Playback selected name

From the dropdown list, user can select student major, with which selected, the Row Numbers, Student Names, Student IDs and Audio File Indicators are shown in the dashboard. To start with, all the Audio File Indicators are shown as 'False', which means none of the names has been generated with its audio file. User may highlight one of the names and click on 'Generate' to let the trained model generate an audio file based on its text. After generating, the indicator in Audio File column will show as 'True' meaning audio file of this student exists.

Once the audio file is existing user can click on ‘Playback’ to play the audio. This may be used during graduation ceremony as well as assessor verification. In case of any issue in the generated audio, the assessor may use the ‘Record’ function to record his/her voice into the system. This recorded voice may then be used in graduation ceremony instead of the model generated audio. Newly recorded audio will be merged with current training data for future training of improved voice models.

4 Training Data Preparation

Two sets of input training data are sourced. The first set is the National Speech Corpus from IMDA. It contains 1TB of audio recordings in Singapore accent. This input is used for training the Baseline model. The second set is to be used after training on the first set. It comprises around 25k utterances of name recordings from past Graduation Ceremonies in Republic Polytechnic.

4.1 Training Data Analysis

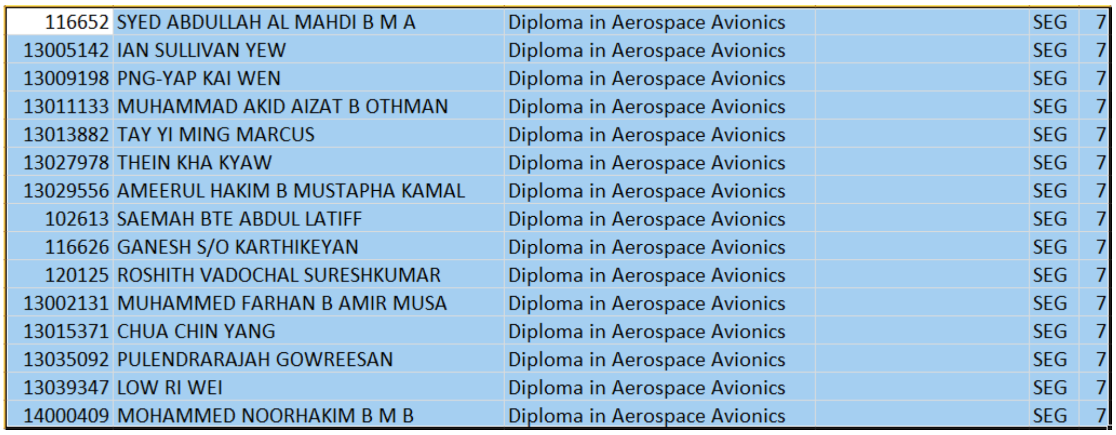
Our team has analyzed 5 years Republic Polytechnic graduation ceremony name list since 2016 and found that 2016-2019 lists are too inconsistent for training. There are in total more than 10 speakers with various recording environments. Comparing to which, the 2021 list is more suitable for training as there are only 2 speakers, 1 male and 1 female. Thus, our team decides to use the female speaker recordings as the training data which contains 2632 utterances from male speaker and 2394 utterances for female speaker.

As for the National Speech Corpus, our team also needs to find a proper speaker’s audio to train the Baseline model as the Corpus is as large as 1TB but fragmented across more than 2000 speakers with different accents. Within these speakers, most of them have only 200 to 400 utterances. After searching in the Corpus, the team eventually settled on a female speaker having 6033 utterances of book readings with a professional and neutral tone.

As the amount of cohesive training data (Singapore female accent) is relatively small (370mins from NSC and 75mins from RP), the approach of utilizing both datasets using two stages of training and combing both using transfer learning will help in achieving the maximum training coverage with our limited dataset. Most TTS deep-learning datasets have between 10-85hrs of training audio, compared to our 1.25hr if only using the RP training set.

4.2 Text Pre-Processing

The input text of student names comes in csv format. There are in total 6 columns of data which includes student number, student name, student major and so on. The names are in Chinese, Malay, Indian, Filipino and others. Among the names, our team has done several key text pre-processing as described in below sections.



116652	SYED ABDULLAH AL MAHDI B M A	Diploma in Aerospace Avionics		SEG	7
13005142	IAN SULLIVAN YEW	Diploma in Aerospace Avionics		SEG	7
13009198	PNG-YAP KAI WEN	Diploma in Aerospace Avionics		SEG	7
13011133	MUHAMMAD AKID AIZAT B OTHMAN	Diploma in Aerospace Avionics		SEG	7
13013882	TAY YI MING MARCUS	Diploma in Aerospace Avionics		SEG	7
13027978	THEIN KHA KYAW	Diploma in Aerospace Avionics		SEG	7
13029556	AMEERUL HAKIM B MUSTAPHA KAMAL	Diploma in Aerospace Avionics		SEG	7
102613	SAEMAH BTE ABDUL LATIFF	Diploma in Aerospace Avionics		SEG	7
116626	GANESH S/O KARTHIKEYAN	Diploma in Aerospace Avionics		SEG	7
120125	ROSHITH VADOCHAL SURESHKUMAR	Diploma in Aerospace Avionics		SEG	7
13002131	MUHAMMED FARHAN B AMIR MUSA	Diploma in Aerospace Avionics		SEG	7
13015371	CHUA CHIN YANG	Diploma in Aerospace Avionics		SEG	7
13035092	PULENDRARAJAH GOWREESAN	Diploma in Aerospace Avionics		SEG	7
13039347	LOW RI WEI	Diploma in Aerospace Avionics		SEG	7
14000409	MOHAMMED NOORHAKIM B M B	Diploma in Aerospace Avionics		SEG	7

4.2.1 Remove Trailing Single Letters (Recursive)

Some of the names are too long so that their name inputs come with a shorten form. For example, in the name ‘MOHAMMAD ZUL ILLIASYAH B M R’, the letters ‘B M R’ are the first letters of the remaining names. To handle this kind of situation, the pre-process has removed the trailing single letters to end up with ‘MOHAMMAD ZUL ILLIASYAH’.

4.2.2 Remove Trailing BIN/BINTE

In Malay names, BIN/BINTE/BTE/BINTI means ‘son of’ or ‘daughter of’. On their own without any text behind, they carry no meaning and are not read in the recordings. Thus, the pre-process has removed them as well. For example, the name ‘UR TASNEEM AQILAH BTE M R’ will become ‘UR TASNEEM AQILAH’.

4.2.3 Unwrapping Acronyms/Short-forms

There are many short-forms in the names such as ‘S/O’ meaning ‘Son of’, ‘D/O’ meaning ‘Daughter of’, ‘B’ meaning ‘Bin’, ‘BTE’ is same as ‘Binte’, ‘MOHD’, ‘MD’, ‘M D’ standing for ‘MOHAMMAD’. The pre-process has replaced them as well.

4.2.4 Remove Punctuation

This is a standard language process workflow which is to remove all the punctuations leaving only the name contents.

4.2.5 Standardize Spellings

Due to same name keyed in by different people, the spellings are various. For example, the name Mohammad can be found in the list with spelling of ‘Mohamad’, ‘Mohamed’ or ‘Mohammed’. They are to be standardized as they are all pronounced the same.

4.3 Audio Pre-Processing

As the audio recordings are not labeled by speaker, our team needs to sort them by manual listening. Furthermore, all the listed names must be matched to the audio files. Then amplitude normalization and silence truncation are carried out. All the audio files are in 16-bit, mono PCM WAV format, between 1 to 10 seconds each, with a sample rate of 22050 Hz.

5 Models Training Process

Once the training audio files and text are pre-processed, the models’ training can be started. Using the model architecture designed in section 3.0, the models’ training of our proposed application mainly consists of two separate deep neural networks which are separated into three training sessions as shown below:

The first is a Text-to-Spectrogram deep neural network model (e.g., Tacotron 2) that is used to compute acoustic features from given text input and the intermediate feature representation (Mel Spectrogram) will be generated accordingly. There are two models are trained in two phases called ‘Baseline model’ and ‘Name model’, which are tried to learn Singapore accent and Asian name pronunciation.

- ✓ Phase 1 (Baseline model): Text-to-Spectrogram training using training data from Singapore Nation Speech Corpus
- ✓ Phase 2 (Name model): Using transfer learning, fine-tune Text-to-Spectrogram model to pronounce Asian names.

The second is a neural vocoder model which tries to generate high quality audio files learnt from acoustic features.

Prior to conducting above trainings, the training environment prerequisites must first be fulfilled. The hardware, drivers and software environment must be configured correctly with GPU computation, including the training data folder structures and hyperparameters.

5.1 Training Setup

5.1.1 Coqui TTS Library

The trainings of our proposed models in AI Name Reader application are established and conducted on top of Text-to-Speech library called Coqui TTS, which is built on the latest research and designed to achieve the best trade-off among ease-of-training, speed and quality. The reason why we choose the Coqui TTS library is because it’s designed for easy-to-use, extensible for new models and support those high-performance models from latest research projects.

5.1.2 Environment Setup

Hardware

The hardware specifications are consisting of the following items for our models’ training:

- ✓ GPU: NVIDIA GeForce RTX 3070 (8GB Memory Size, 184 TMUS and GDDR6)
- ✓ CPU: 8GB Memory Size
- ✓ Hard disk: Recommended 10G above

Software

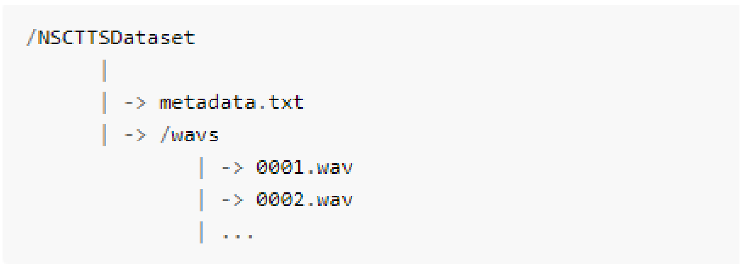
As our training environment is configured on Windows 10 64-bit operating system. The following software components are required to be installed accordingly.

- ✓ Python 3.8.x runtime
- ✓ NVIDIA CUDA Toolkit 10.1
- ✓ NVIDIA cuDNN v7.6.5 (This is a GPU-accelerated library of primitives for deep neural networks.)
- ✓ eSpeak NG (latest x64 bit version)
- ✓ Microsoft Visual C++ 14.0++ for Python
- ✓ coqui-ai TTS library source code (v.0.0.15)

Training Data Formatting

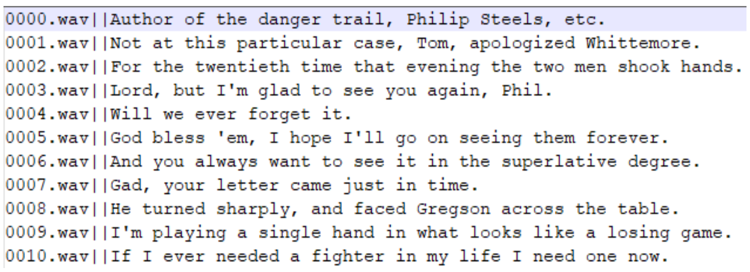
The training data folders must be created and organized together with speech audio files and transcriptions using the specified hierarchical format shown below, which consists of a 1) TTS data root folder, 2) transcription file (metadata.csv) and 3) sub-folder 'wavs' for storing all training data files.

- ✓ Folder Structure:



```
/NSCTTSDataset
       |
       | -> metadata.txt
       | -> /wavs
               | -> 0001.wav
               | -> 0002.wav
               | ...
```

- ✓ Metadata text file (metadata.csv):

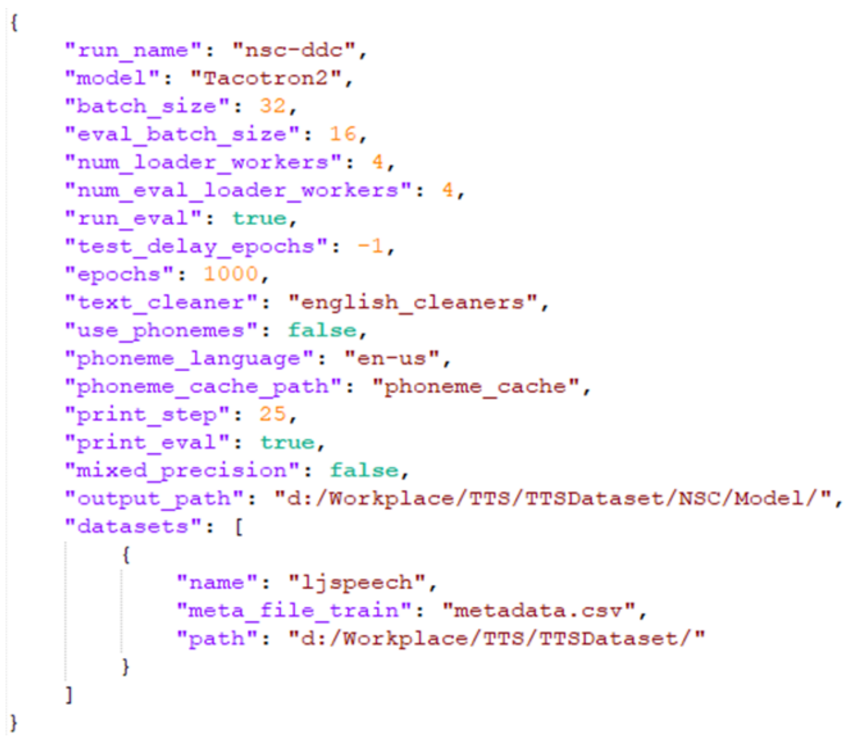


```
0000.wav||Author of the danger trail, Philip Steels, etc.
0001.wav||Not at this particular case, Tom, apologized Whittemore.
0002.wav||For the twentieth time that evening the two men shook hands.
0003.wav||Lord, but I'm glad to see you again, Phil.
0004.wav||Will we ever forget it.
0005.wav||God bless 'em, I hope I'll go on seeing them forever.
0006.wav||And you always want to see it in the superlative degree.
0007.wav||Gad, your letter came just in time.
0008.wav||He turned sharply, and faced Gregson across the table.
0009.wav||I'm playing a single hand in what looks like a losing game.
0010.wav||If I ever needed a fighter in my life I need one now.
```

Each line of transcription consists of audio file name and actual text script together with delimiter symbol '||'.

Hyperparameters' Configuration

The Coqui TTS library is well extensible to support other models, which also abstract hyperparameters into corresponding configuration classes as well as supportive JSON files. The JSON format is used for storing those hyperparameters for our model training and usage. The below are showing the simplified JSON format for Tacotron 2 model training.



```
{
    "run_name": "nsc-ddc",
    "model": "Tacotron2",
    "batch_size": 32,
    "eval_batch_size": 16,
    "num_loader_workers": 4,
    "num_eval_loader_workers": 4,
    "run_eval": true,
    "test_delay_epochs": -1,
    "epochs": 1000,
    "text_cleaner": "english_cleaners",
    "use_phonemes": false,
    "phoneme_language": "en-us",
    "phoneme_cache_path": "phoneme_cache",
    "print_step": 25,
    "print_eval": true,
    "mixed_precision": false,
    "output_path": "d:/Workplace/TTS/TTSDataset/NSC/Model/",
    "datasets": [
        {
            "name": "ljspeech",
            "meta_file_train": "metadata.csv",
            "path": "d:/Workplace/TTS/TTSDataset/"
        }
    ]
}
```

There are many other hyperparameters available for executing or fine-tune model training. They can be adjusted according to actual training performance results or training resources. The important basic attributes are used and described in the below table:

S/No	Name	Description
1	run_name	It's the name to indicate the current model training.
2	model	This is the actual model's name that is used to train.
3	batch_size	Batch size for training. Lower values than 32 might cause hard to learn attention. It is overwritten by 'gradual_training'.
4	num_loader_workers	number of training data loader processes.
5	num_val_loader_workers	number of evaluation data loader processes.

6	run_eval	It's a flag to switch on/off the evaluation.
7	test_delay_epochs	The value '-1' is assigned to disable testing.
8	output_path	It's output folder path of generated model.
9	datasets	It's a list of datasets that are used for training. The inner attributes have to be provided in order to lookup training data correctly.
10	epochs	total number of epochs to train.
11	lr	Initial learning rate (default: 0.0001).
12	wd	Weight decay weight (default: 0.000001)
13	save_step	Number of training steps expected to save training stats and checkpoints. (default: 10000)

5.2 Text-To-Spectrogram Training (Phase 1)

The baseline Text-to-Spectrogram model is designed and trained with Tacotron 2 to learn Singapore accent using Singapore National Speech Corpus (NSC). There are 6033 utterances with aligned text scripts by single speaker extracted from NSC. 370 minutes of recordings are trained for Baseline model.

The below table shows the batch script for executing Baseline model training:

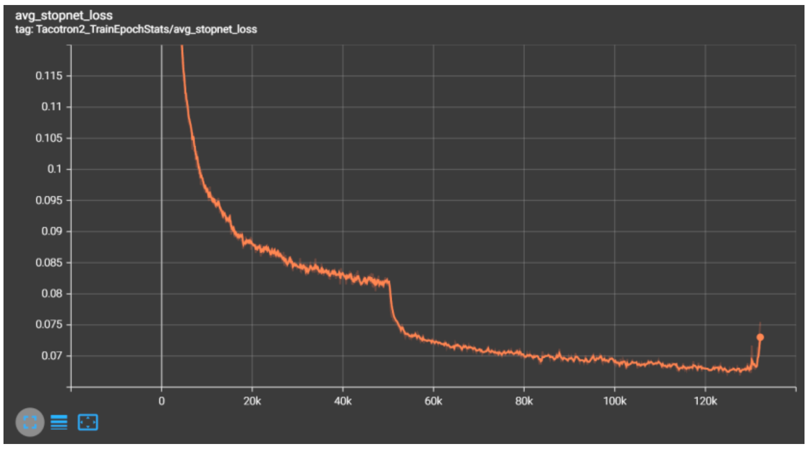
```
set PYTHONIOENCODING=UTF-8
set PYTHONLEGACYWINDOWSSTDIO=UTF-8
set PHONEMIZER_ESPEAK_PATH=C:\Program Files\eSpeak NG\espeak-ng.exe
scripts\python.exe ./TTS/bin/train_tacotron.py --config_path "d: \TTS_Data\NSC2021\config.json"
```

The following list describes the training process flow:

- ✓ The well-prepared training audio and text files are passed to training process.

- ✓ The checkpoint model will be generated and saved in output folder every number of training step (default: 10000), including training statistical file.
- ✓ Model generated audio files will be saved in output folder (test_audios) regularly, which is used by trainer to manually listen and compare with training audio files.
- ✓ Hyperparameters will be adjusted or pre-trained models are replaced by other models with better performance. The training will be restarted.
- ✓ The best model (best_model.pth.tar) will be created in output folder after number of training steps (default: 10000). This can be used for verifying the performance of generated model to decide if the training should be stopped. The best or any checkpoint models will be used as baseline model for transfer learning in phase 2.

The full Phase 1 training on GPU RTX 3070 takes for 40 hours. The best Baseline model was converged after 130,000 iterations as shown in below figure:



5.3 Text-To-Spectrogram Training (Phase 2)

Once Phase 1 training completed and the desired baseline model generated, the transfer learning technique will be applied to Name model in Phase 2 Text-to-Spectrogram training. Those Asian names should be pronounced natively after completing Phase 2 training using name recordings from past graduation ceremonies in Republic Polytechnic. Total 2394 utterances with aligned text scripts recorded from single speaker that are used for training (75 minutes of recordings).

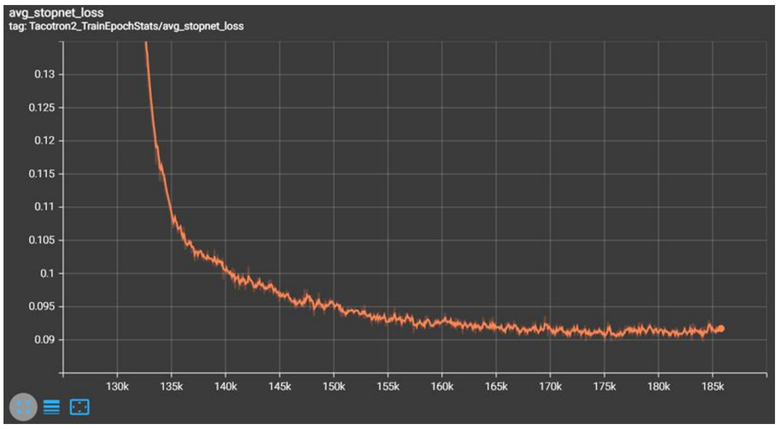
The execution of training script will be same except adding one more script parameter like ‘restore_path’ that points to baseline model tar file.

```
set PYTHONIOENCODING=UTF-8
set PYTHONLEGACYWINDOWSSTDIO=UTF-8
set PHONEMIZER_ESPEAK_PATH=C:\Program Files\eSpeak NG\espeak-ng.exe
scripts\python.exe ./TTS/bin/train_tacotron.py --config_path "d: \TTS_Data\GC2021\config.json" --
restore_path " d: \TTS_Data\NSC2021\Model\nsc-ddc-June-18-2021_09+57PM-
b8b79a5e\checkpoint_130000.pth.tar"
```

The following list describes the training process flow for Phase 2:

- ✓ The training data of name recordings and text files are passed to training process.
- ✓ The similar checkpoint model will be generated and saved in output folder every number of training step (default: 10000), including training stats file.
- ✓ Model generated audio files will be saved in output folder (test_audios) regularly, which is used by trainer to manually listen and compare with training audio files.
- ✓ Hyperparameters can be optimized for better training performance. The training will be restarted.
- ✓ The best model (best_model.pth.tar) will be created in output folder after number of training steps (default: 10000). This can be used for verifying the performance of generated model to decide if the training should be stopped.
- ✓ The best or desired checkpoint models can be used as Name model for pronouncing Asian name with a Singapore accent (Chinese, Malay, Indian, Pilipino and others).

The best Name model was generated after 50,000 iterations after transfer learning from Phase 1. The full Phase 2 training on GPU RTX 3070 takes 15 hours. The total training time together with Phase 1 is 55 hours. The following figure illustrates the training performance:



5.4 Vocoder Training

The Generative Adversarial Networks for Conditional Waveform Synthesis (Multiband MelGAN) is used as a vocoder model to train and learn from 2394 utterances of name recordings from past graduation ceremonies for converting Mel Spectrogram to high quality audio files. As the training is built on Coqui TTS library that provide the pluggable framework for supporting various of models regardless of TTS or vocoder models. The vocoder training process is similar to preceding TTS model training except independent configuration JSON file. In our training for vocoder model, transcription files are not required as Multiband MelGAN directly train on Mel Spectrogram and verify the generated audio file using discriminator networks. In the end, it is able to yield a high-quality text-to-speech synthesis model without additional distillation or perceptual loss functions.

The below show the sample of execution batch script for training vocoder model:

```
set PYTHONIOENCODING=UTF-8
set PYTHONLEGACYWINDOWSSTDIO=UTF-8
set PHONEMIZER_ESPEAK_PATH=C:\Program Files\eSpeak NG\espeak-ng.exe
scripts\python.exe ./TTS/bin/train_vocoder_gan.py --config_path
"d:/TTS_Data/GC2021/vocoder_config.json" --restore_path
"d:\TTS_Data\GC2021\vocoder_model\multiband-melgan-June-23-2021_05+40PM-
b8b79a5e\checkpoint_175000.pth.tar"
```

The training process flow for vocoder training is same as preceding model training. The best is used for inverting intermediate waveform to raw audio files.

5.5 Final TTS Model

The final full TTS models consists of preceding trained TTS and vocoder models. The below Figure illustrates the folder structure for the two models that we use them for AI Name Reader application:

```
tts_models
+---model
|       checkpoint_180000.pth.tar
|       config.json
|       ...
|
\---vocoder
        best_model_199776.pth.tar
        config.json
        ...
```

The file named ‘checkpoint_18000.pth.tar’ is our final Text-to-Spectrogram model (Tacotron 2) that will be integrated with AI Name Reader GUI for pronouncing Singapore accent Asian names.

The file named ‘best_model_199776.pth.tar’ is our final vocoder model (Multiband MelGAN) that will be used to generate high quality audio files from AI Name Reader application.

The final TTS model can run in real-time on CPU and generate audio file about 1 second.

6 TTS Model Validation and Verification

Here, the TTS model validation is not same as the model training performance evaluation with high accuracy. The stable TTS model is not meant to produce a high-quality audio file. In our work, we used one of most popular TTS evaluation metrics called Mean Opinion Score (MOS) test that is a subjective measurement used for quality assessment of audio generated, using human testers on a scale of 1-5. (4.3-4.5 is considered excellent quality).

Rating	Speech Quality	Level of Distortion
5	Excellent	Imperceptible
4	Good	Just perceptible, but not annoying
3	Fair	Perceptible and slightly annoying
2	Poor	Annoying, but not objectionable
1	Bad	Very annoying and objectionable

The approach of doing model validation and verification that we randomly generated 96 audio files for seen and unseen data. 1) Audio Comparison for Seen is we manually listen and compare between the original audio and the generated audios that model has learnt. 2) Audio Comparison for Unseen is the comparison between the original audio and the generated audio that model hasn’t learnt.

We used AI Name Reader application to generate bulk audio (wav) files based on input of student IDs and names from CSV source file and playback option to verify accuracy of audio pronunciation. Inaccurate audio can be flagged for re-recording by human speaker using AI Name Reader and used for further retraining.

The below table shows mean opinion scores that we have tested in our project:

Ethnicity	Count (%)	MOS

Chinese	42 (44%)	4.45
Malay	38 (39%)	4.81
Indian	13 (14%)	4.23
Others	3 (3%)	3.67
Overall	**96 (100%)**	**4.54**

We can see overall MOS score is 4.54 tested on 96 audios from all regions in Singapore. The results show that our TTS model is able to produce high-quality Asian names with Singapore accent. As we can see the MOS score (4.81) from Malay is higher than Chinese with similar number of test audio files, we found that is probably because Malay name pronunciation is more aligned to English phonetics, but the TTS model couldn't pronounce those names with Chinese phonetics, such as Teochew, Hokkien and Cantonese dialects, which haven't been fully learnt by model. The MOS test score from Others is quite low and that is probably due to little training data. The MOS can be improved by adding more test data or introducing objective evaluation call Mel Cepstral Distortion in future work.

7 Finding and Achievement

We faced various challenges encountered during the TTS research and implantation activities on AI Name Reader project. Firstly, there are a lot of complex and advanced concepts we have to learn and analyze in Deep learning TTS models. Extensive research is necessary to bridge learned concepts with advanced models. Secondly, cutting edge model selection became another problem as there are minimal documentation on usage and configuration. There are different setup and configuration issues encountered, such hardware, driver and software/library version dependencies. We choose Coqui TTS library as our model training framework because it supports different cutting-edge models friendly using loosely coupled API interface. However, it is still in early development not very stable. We experienced some crashes during training and had to resume training using saved checkpoints which fortunately is provided at pre-defined intervals. Thirdly, curation and preparation of suitable training data from large datasets is tedious and time consuming. It takes long training time in order to produce suitable models.

We have been able to design and develop text-to-speech models and integrate into real application. The TTS model we trained that achieve a high quality and the Mean Opinion Score has an excellent test result 4.54 which means we can deploy it to real environment for reducing manpower and providing a high-quality consistency Name Reading services. Voice talent can use AI Name

Reader application to generate audio and verify by listening. She can perform human recording directly through the GUI if the generated audio is not suitable.

8 Conclusion

We have implemented cutting-edge TTS and vocoder models that are used in our AI Name Reader system. Both of them have resulted a good performance and produce high-quality audios. The AI Name Reader application can perform well after integrating two core models. We hope the TTS models can be improved a lot after introducing new name recording data such as that Chinese phonetics that cannot be pronounced well when the model has not seen data.

While the AI Name Reader application is only involving two deep learning models as now, which can be evaluated and improved by introducing more other custom models. Learning and improving high quality TTS model for audio will be a very interesting direction for future work. We believe will be beneficial from diving into deeper TTS research and projects in our future works.

9 References

Gölge, E. (2021, March 8). GitHub - coqui-ai/TTS: 🐸💬 - a deep learning toolkit for Text-to-Speech, battle-tested in research and production. Retrieved from https://github.com/coqui-ai/TTS.

Eren Gölge, E. (2021, June 3). Solving Attention Problems of TTS models with Double Decoder Consistency | A Blog From Human-engineer-being. Retrieved from https://erogol.com/solving-attention-problems-of-tts-models-with-double-decoder-consistency.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., … Wu, Y. (2017). Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. arXiv. Retrieved from https://arxiv.org/abs/1712.05884.

Yang, G., Yang, S., Liu, K., Fang, P., Chen, W., & Xie, L. (2020). Multi-band MelGAN: Faster Waveform Generation for High-Quality Text-to-Speech. arXiv. Retrieved from https://arxiv.org/abs/2005.05106.

Valizada, A.; Jafarova, S.; Sultanov, E.; Rustamov, S. Development and Evaluation of Speech Synthesis System Based on Deep Learning Models. Symmetry 2021, 13, 819. https://doi.org/10.3390/sym13050819

National Speech Corpus - Infocomm Media Development Authority. (2020, November 26). Retrieved from http://www.imda.gov.sg/programme-listing/digital-services-lab/national-speech-corpus.

Gölge, E. (2021, June 4). TTS 0.0.15 documentation. Retrieved from https://tts.readthedocs.io/en/latest/.

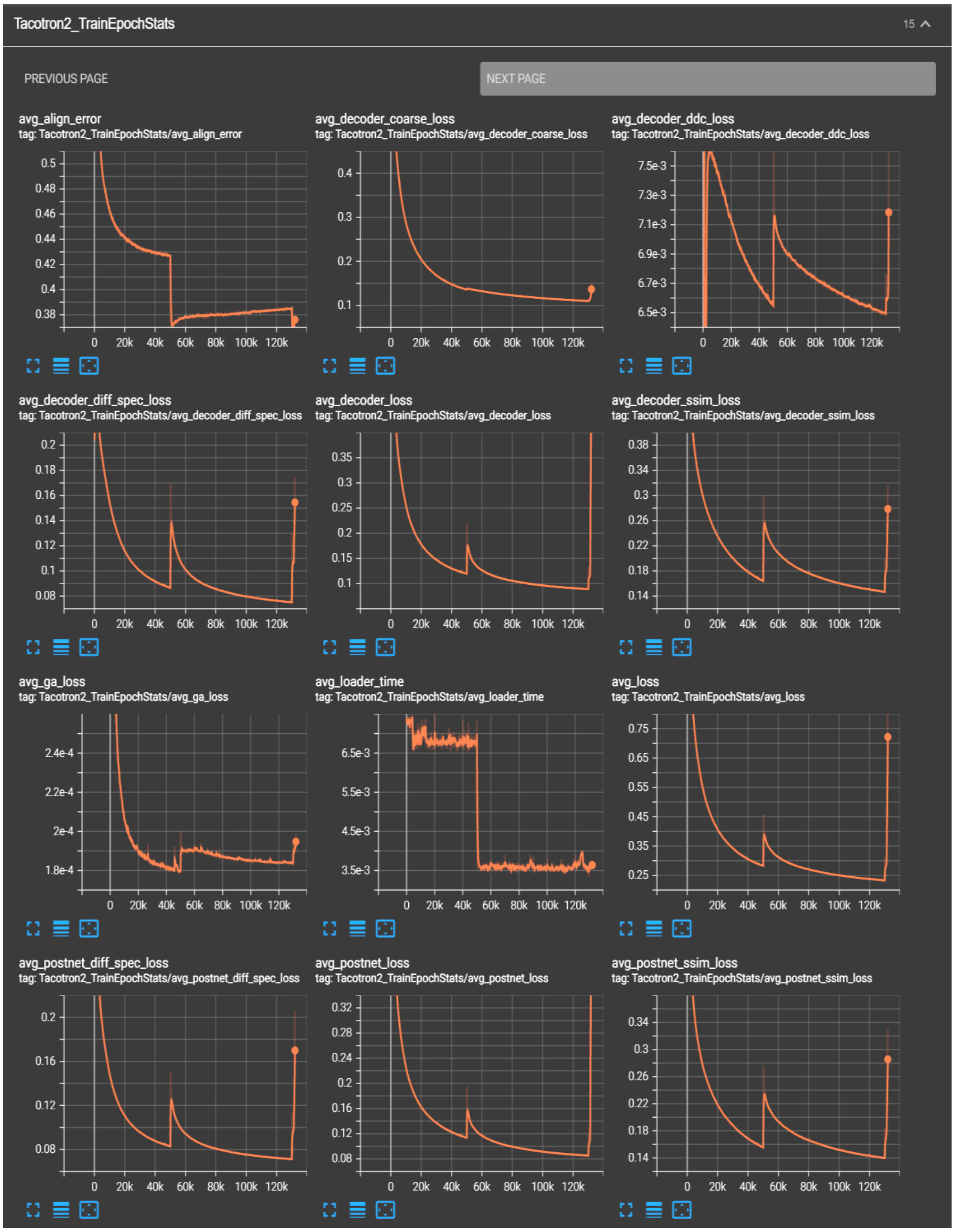
PySimpleGUI documentation. Retrieved from https://pysimplegui.readthedocs.io/en/latest/

Visualize Model Training with TensorBoard documentation. Retrieved from https://pytext.readthedocs.io/en/master/visualize_your_model.html

10 Appendix A – Training Performance

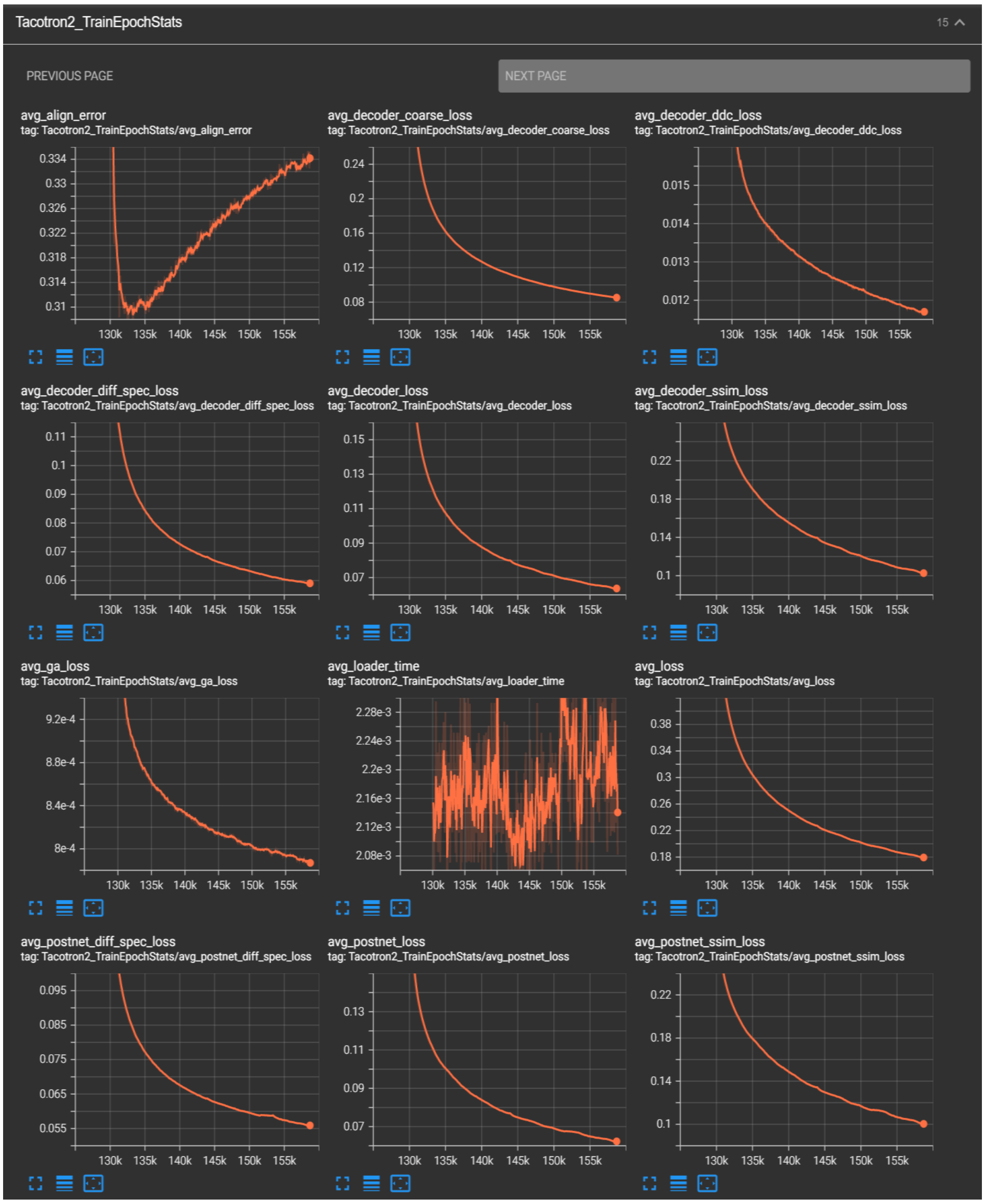
TensorBoard is used as a measurement tool during model training for visualizing the training loss, accuracy and other metrics.

<u>Tacotron2 Baseline Model (NSC data) Training</u>

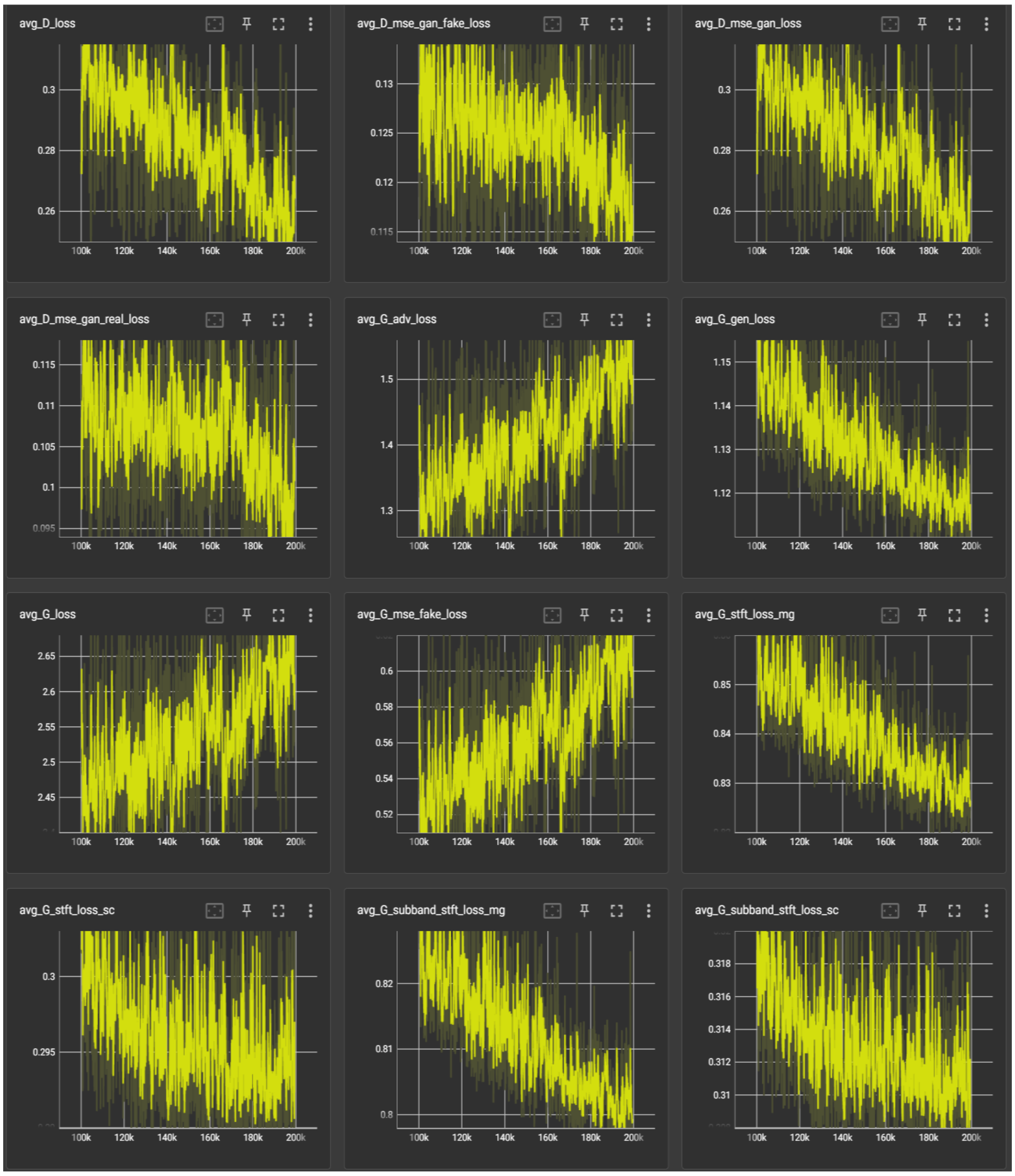


Tacotron2_TrainEpochStats
15
PREVIOUS PAGE
NEXT PAGE
avg_align_error
tag: Tacotron2_TrainEpochStats/avg_align_error
avg_decoder_coarse_loss
tag: Tacotron2_TrainEpochStats/avg_decoder_coarse_loss
avg_decoder_ddc_loss
tag: Tacotron2_TrainEpochStats/avg_decoder_ddc_loss
avg_decoder_diff_spec_loss
tag: Tacotron2_TrainEpochStats/avg_decoder_diff_spec_loss
avg_decoder_loss
tag: Tacotron2_TrainEpochStats/avg_decoder_loss
avg_decoder_ssim_loss
tag: Tacotron2_TrainEpochStats/avg_decoder_ssim_loss
avg_ga_loss
tag: Tacotron2_TrainEpochStats/avg_ga_loss
avg_loader_time
tag: Tacotron2_TrainEpochStats/avg_loader_time
avg_loss
tag: Tacotron2_TrainEpochStats/avg_loss
avg_postnet_diff_spec_loss
tag: Tacotron2_TrainEpochStats/avg_postnet_diff_spec_loss
avg_postnet_loss
tag: Tacotron2_TrainEpochStats/avg_postnet_loss
avg_postnet_ssim_loss
tag: Tacotron2_TrainEpochStats/avg_postnet_ssim_loss
0
20k
40k
60k
80k
100k
120k

Tacotron2 Name Model (RP data) Training



<u>MelGAN Vocoder Model (RP data) Training</u>



11 Appendix B – Installation

The installation is divided into two parts that are training environment and AI Name Reader installation. The training environment installation allow us to do TTS and vocoder model training and those hardware and software prerequisites are described in the preceding section 5.1.2. The AI Name Reader installation is used for audio file generation as well as audio quality verification via playback. Our model environment and AI Name Reader application are only installed and tested on Windows 10 with specific software versions.

Training Environment

1) Python 3.8.8
   Download URL: https://www.python.org/downloads
   - Don't use version 3.9+ as it's not tested, only install python-3.8.8-amd64.exe
   - Opt to install it for all users.
   - Opt to add Python to the PATH.
   - Type command 'python –version' in command prompt to verify python version.

2) CUDA Toolkit 10.1
   Download URL: https://developer.nvidia.com/cuda-10.1-download-archive-base
   - Don't use version 11+ as it's not tested. Only install cuda_10.1.105_418.96_win10.exe.

3) cuDNN v7.6.5
   Download URL: https://developer.nvidia.com/rdp/cudnn-archive (Use version v7.6.5 on 5th November 2019)
   - Extract the file (cudnn-10.1-windows10-x64-v7.6.5.32.zip)
   - Copy what's inside the cuda folder into C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1 (To make sure folder directory is correct if the installation of CUDA Toolkit 10.1 is different)

4) eSpeak NG
   Download URL: https://github.com/espeak-ng/espeak-ng/releases
   - Recommend to install latest 64-bit version of eSpeak NG although no version constraints.

5) Git for Windows
   Download URL: https://git-scm.com/download/win
   - Recommend to install latest 64-bit version of git for windows although no version constraints.
   - Type command 'git –version' in command prompt to verify python version.

6) Microsoft Visual C++ 14.0++

Download URL: https://visualstudio.microsoft.com/visual-cpp-build-tools

- Double click installer and navigate to Workloads → Desktop development with C++, then for Individual Components, select only:
  - ✓ Windows 10 SDK
  - ✓ C++ x64/x86 build tools

7) Open a PowerShell prompt to a folder where you'd like to install Coqui TTS. For example, cd D: \TTS\coqui-ai\
   - To execute the following commands

```
git clone https://github.com/coqui-ai/TTS.git
cd TTS
python -m venv .
.\Scripts\pip install -e .
.\Scripts\pip install torch==1.8.0+cu101 torchvision==0.9.0+cu101
torchaudio===0.8.0 -f https://download.pytorch.org/whl/torch_stable.html
```

8) Create a script called "test_cuda.py" in the TTS folder (e.g., D:\TTS\coqui-ai\TTS) and add the following script into it:

```
import torch
x = torch.rand(5, 3)
print(x)
print(torch.cuda.is_available())
```

9) To verify CUDA for GPU by executing the below secript
   - .\Scripts\python ./test_cuda.py
   - To confirm the output looks like below

```
tensor([[0.2141, 0.7808, 0.9298],
         [0.3107, 0.8569, 0.9562],
         [0.2878, 0.7515, 0.5547],
         [0.5007, 0.6904, 0.4136],
         [0.2443, 0.4158, 0.4245]])
True
```

   The last line of output will be ‘True’ if GPU is available and CUDA is not installed properly.

10) To test built-in pre-trained model by executing the following command line

```
.\Scripts\tts --text "Text for Text Analytics" --model_name
"tts_models/en/ljspeech/tacotron2-DDC" --vocoder_name
"vocoder_models/en/ljspeech/hifigan_v2" --out_path d:/output.wav
```

AI Name Reader GUI

As the Name Reader Automation is developed by Python language with several third-party libraries that requires python to be installed. It requires Python version 3 and later and third-party libraries:

- Application GUI (pip install PySimpleGUI)
- Audio Recording (pip install sounddevice)
- Audio Processing (pip install pydub)
- Audio Playback (pip install simpleaudio)

The ffmpeg executable library is required.

- Click the Microsoft icon download button from the download page of ffmpeg website via https://www.ffmpeg.org/download.html for windows ffmpeg version.
- Unzip the downloaded zip file and save it to file path C:\ffmpeg
- To add ffmpeg bin directory (c:\ffmpeg\bin) to Windows system PATH environment variables as follows:
  Windows Start Search > Type 'View Advance System Settings' > Click 'Environment Variables' > Double click Path row under system variable section > Add file path 'c:\ffmpeg\bin' to PATH environment variables as above figure.

Extract and copy GUI folder to D:\TTS. Launch the AI Name Reader application by using below script:
D:\TTS\coqui-ai\TTS\Scripts\python D:\TTS\GUI\name_recorder_gui.py