



# UNDERGRADUATE PROJECT REPORT

<b>Project Title:</b>	<b>Web API For Voice Synthesis</b>
<b>Surname:</b>	<b>Luo</b>
<b>First Name:</b>	<b>Wei Ming</b>
<b>Student Number:</b>	<b>201918020412</b>
<b>Supervisor Name:</b>	<b>Dr. Joojo Walker</b>
<b>Module Code:</b>	<b>CHC 6096</b>
<b>Module Name:</b>	<b>Project</b>
<b>Date Submitted:</b>	<b>May 5, 2023</b>

Chengdu University of Technology Oxford Brookes College

Chengdu University of Technology

**BSc (Single Honours) Degree Project**

Programme Name: **Software Engineering**

Module No.: **CHC 6096**

Surname: Luo

First Name: WeiMing

Project Title: **Web API For Voice Synthesis**

Student No.: 201918020412

Supervisor: Dr. Joojo Walker

2<sup>ND</sup> Supervisor (if applicable): **Not Applicable**

Date submitted: **May 5, 2023**

*A report submitted as part of the requirements for the degree of BSc (Hons) in Software Engineering*

*At*

**Chengdu University of Technology Oxford Brookes College**

## Declaration

### Student Conduct Regulations:

Please ensure you are familiar with the regulations in relation to Academic Integrity. The University takes this issue very seriously and students have been expelled or had their degrees withheld for cheating in assessment. It is important that students having difficulties with their work should seek help from their tutors rather than be tempted to use unfair means to gain marks. Students should not risk losing their degree and undermining all the work they have done towards it. You are expected to have familiarised yourself with these regulations.

<https://www.brookes.ac.uk/regulations/current/appeals-complaints-and-conduct/c1-1/>

Guidance on the correct use of references can be found on [www.brookes.ac.uk/services/library](http://www.brookes.ac.uk/services/library), and also in a handout in the Library.

The full regulations may be accessed online at <https://www.brookes.ac.uk/students/sirt/student-conduct/>

If you do not understand what any of these terms mean, you should ask your Project Supervisor to clarify them for you.

**I declare that I have read and understood Regulations C1.1.4 of the Regulations governing Academic Misconduct, and that the work I submit is fully in accordance with them.**

Signature




Date: 2023/5/2

REGULATIONS GOVERNING THE DEPOSIT AND USE OF OXFORD BROOKES UNIVERSITY MODULAR PROGRAMME PROJECTS AND DISSERTATIONS

Copies of projects/dissertations, submitted in fulfilment of Modular Programme requirements and achieving marks of 60% or above, shall normally be kept by the Oxford Brookes University Library.

**I agree that this dissertation may be available for reading and photocopying in accordance with the Regulations governing the use of the Oxford Brookes University Library.**

Signature



Date: 2023/5/2

## **Acknowledgment**

In no particular order, I would like to thank my supervisor, Joojo walker, for his dedication and guidance in providing me with good ideas for my writing. I would like to thank Cjang, luoyi and rcell for their help with the VITS research, which helped me with many training and deployment issues. I would like to thank my parents for their moral support during this time.

## Table of Contents

<b>Declaration</b> .....	ii
<b>Acknowledgment</b> .....	iii
<b>Table of Contents</b> .....	iv
<b>Abstract</b> .....	vi
<b>Abbreviations</b> .....	vii
<b>Glossary</b> .....	viii
<b>Chapter 1 Introduction</b> .....	1
<b>1.1 Background</b> .....	1
<b>1.2 Aim</b> .....	2
<b>1.3 Objectives</b> .....	2
<b>1.4 Project Overview</b> .....	2
<b>1.4.1 Scope</b> .....	2
<b>1.4.2 Audience</b> .....	3
<b>Chapter 2 Background Review</b> .....	4
<b>2.2 Existing Approaches</b> .....	5
<b>2.3 For TTS Models</b> .....	6
<b>Chapter 3 Methodology</b> .....	7
<b>3.1 Approach</b> .....	7
<b>3.1.1 Model For Original VITS</b> .....	7
<b>3.1.2 Emotion Training For VITS</b> .....	11
<b>3.1.3 User Interface Model:</b> .....	12
<b>3.2 Technology</b> .....	13
<b>3.2.2 For Back-end Language</b> .....	13
<b>3.2.3 For support language</b> .....	14
<b>3.2.4 For Data Set</b> .....	14
<b>3.3 Project Version Management</b> .....	16
<b>Chapter 4 Results</b> .....	18
<b>Chapter 5 Professional Issues</b> .....	31
<b>5.1 Project Management</b> .....	31
<b>5.1.1 Activities</b> .....	31
<b>5.1.2 Schedule</b> .....	33

<b>5.1.3</b>	<b>Project Data Management</b> .....	34
<b>5.1.4</b>	<b>Project Deliverables</b> .....	35
<b>5.2</b>	<b>Risk Analysis</b> .....	35
<b>5.3</b>	<b>Professional Issues</b> .....	37
<b>5.3.1</b>	<b>Technology Issue</b> .....	37
<b>Chapter 6</b>	<b>Conclusion</b> .....	41
<b>References</b>	.....	43

## **Abstract**

In the field of speech synthesis, audio datasets are usually completed by hiring a voice actor, but many voice actors are not professional enough to cause the audio to fluctuate in mood enough and the training results are not vivid enough. In this project, the emotive voices from the game are extracted and trained based on the VITS model. At the same time, emotion embedding is added to the training, so that the model can better reproduce the emotional fluctuations of the characters and allow the model to produce a wider range of voices. A web UI was designed for this and the model was deployed on the huggingFace server using Gradio.

**Keywords: Voice synthesis, NLP, VITS, Gradio, Web UI, Emotional training, Emotion embedding, Game dataset, VAE, GAN.**

## Abbreviations

<b>ABBREVIATIONS</b>	<b>FULL NAME</b>
<b>VITS</b>	Variational Inference Text-to-Speech
<b>AI</b>	Artificial Intelligence
<b>NLP</b>	Neuro-Linguistic Programming
<b>GPU</b>	Graphics Processing Unit
<b>CPU</b>	Central Processing Unit
<b>TTS</b>	Text-To-Speech
<b>API</b>	Application Programming Interface
<b>GAN</b>	Generative Adversarial Network
<b>VAE</b>	Variational Auto Encoder
<b>UI</b>	User Interface
<b>RNN</b>	Recursive Neuro Network



## Glossary

<b>Words</b>	<b>Explanation</b>
<b>VAE</b>	A kind of generative model with encoder and decoder
<b>TTS</b>	A work that input text then gets voice speech
<b>NLP</b>	A kind of Linguistic in deep learning
<b>VITS</b>	A VAE-based voice synthesis model
<b>Tacotron</b>	A voice synthesis TTS model made by Micorsoft
<b>Gradio</b>	A Python API to make a portable web UI
<b>Emotion embedding</b>	A tensor with 129 dimensions to record emotion in dataset

# Chapter 1 Introduction

## 1.1 Background

Text-to-speech (TTS) synthesis is an important technology that has broad applications in various fields, such as human-computer interaction, entertainment and education. Recently, there has been a growing interest in developing TTS systems that can generate high-quality speech with fast speed. One promising approach to achieve this goal is to use Variational Autoencoder (VAE) based TTS systems.

VAE-based TTS systems have shown great potential in generating high-quality speech with fast speed. In a VAE-based TTS system, the input text is first encoded into a latent representation, which is then decoded to generate speech. The VAE-based TTS system can be trained in an end-to-end manner, which makes it easy to optimize and fine-tune the system.

One of the most recent and promising VAE-based TTS systems is the Very Fast and High-Quality Speech Synthesis System (VITS) [1]. VITS is a novel TTS system that uses a VAE-based architecture to generate speech with high-quality and fast speed. Unlike traditional VAE-based TTS systems, VITS uses a WaveNet[2]-like decoder to generate speech, which allows for faster and more efficient generation of speech.

The VITS system has several advantages over traditional TTS systems. First, it can generate high-quality speech with fast speed, which makes it suitable for real-time applications. Second, it can be trained in an end-to-end manner, which makes it easy to optimize and fine-tune the system. Finally, it can be used to generate speech in different languages and accents, making it suitable for a wide range of applications.

In this report, I propose a website TTS system based on the VITS architecture. I evaluate the performance of our system on a dataset of speech samples and compare it with other state-of-the-art TTS systems. Our results show that the VITS-based TTS system outperforms other TTS systems in terms of both speech quality and speed, demonstrating the effectiveness of the VITS architecture for TTS synthesis.

## 1.2 Aim

The goal of this project is to develop a web-based API that allows users to utilize a web User interface (UI) to synthesize voice they like from the voice of game characters they love.

## 1.3 Objectives

The objectives of this project are as follows:

- 1) Thoroughly review representative works of literature in the domain of text-to-speech analysis
- 2) Select and collect voice datasets from games.
- 3) Implement the inference of models. Use the weights of the reference models to complete the synthesis task.
- 4) Train some good models to support the synthesis
- 5) Design a web application that allows users to synthesize voices they like.

## 1.4 Project Overview

This project is a voice synthesis web application. It can let the user input text and output the voice video file. The reference model paper states that this is a text-to-speech (TTS) task. When users select a character which is supported by the web application and input text, the web app will find the corresponding model, then load it to read the weights of the inference the models. Finally, generate the voice.

This project is for the user who wants to use voice synthesis for special characters' voice. Like using the characters from games users' friend loved to say happy birthday to his/her friends or some people want someone to read some books to them.

### 1.4.1 Scope

Firstly, the aim of this project is to demonstrate the usefulness of a home-made speech dataset, contributing to the field of audio datasets with text annotation in Neuro-Linguistic Programming (NLP). In addition, this project demonstrates the feasibility of adding emotional embeddings as annotations to models in the field of speech synthesis, and shows excellent results

## **1.4.2Audience**

Researchers in the field of natural speech can complete their training with the dataset of this project, who is more realistic and emotionally richer than the unusual dataset. Game operators can use this product for creation and promotion, which can reduce the cost of hiring voice actors. At the same time, game players can use this project for secondary creation, which can create storylines that are not in the game, enrich the characters and fill in the regrets in the storyline.

## Chapter 2 Background Review

### 2.1 Review of Related Works

Text-to-speech (TTS) synthesis has been an active research area in recent years, with numerous models proposed for generating high-quality speech from text input. One of the most widely used models is the Tacotron 2 [3] model, which uses a sequence-to-sequence architecture with attention to generate speech. Tacotron 2 has been shown to produce high-quality speech, but it can be slow and computationally expensive due to its autoregressive nature.

To address the issues of speed and efficiency, several non-autoregressive TTS models have been proposed. These models can generate speech in parallel, which makes them faster and more efficient than autoregressive models like Tacotron 2. One such model is FastSpeech, which uses a feed-forward transformer to generate speech in a non-autoregressive manner. FastSpeech can generate speech at up to 150 times faster than Tacotron 2, but its speech quality is lower than that of Tacotron 2.

Another non-autoregressive TTS model is the MelGAN-based WaveRNN model, which uses a generative adversarial network (GAN) to generate mel-spectrograms of speech. WaveRNN uses a WaveNet-like architecture to generate speech from the mel-spectrograms, and it can generate speech at a faster speed than Tacotron 2. However, WaveRNN has some limitations in terms of speech quality and the ability to handle long sentences.

Recently, a new non-autoregressive TTS model called Very Fast and High-Quality Speech Synthesis System (VITS) has been proposed. VITS uses a variational autoencoder (VAE) to generate a latent representation of the input text, which is then used to generate speech using a WaveNet-like decoder. VITS can generate speech in parallel, making it faster than Tacotron 2, but its speech quality is comparable to or better than that of Tacotron 2. VITS can also generate speech in different languages and accents, making it suitable for a wide range of applications.

Compared to other non-autoregressive TTS models like FastSpeech and WaveRNN, VITS has several advantages. First, VITS can generate speech at a faster speed while maintaining high speech quality. Second, VITS is capable of handling long sentences and can generate speech in different languages and accents. Third, VITS is easy to train and can be fine-tuned for specific tasks. Fourth, VITS can be used in various applications, such as virtual assistants, audiobooks, and language learning.

In summary, VITS is a promising non-autoregressive TTS model that can generate high-quality speech with fast speed. Compared to other TTS models like Tacotron 2, FastSpeech, and WaveRNN, VITS has several advantages that make it suitable for various applications.

## 2.2 Existing Approaches

Comparison of Existing Approaches:

	Online	Emtion	Nature/human like/fluency	Real time	Free
VITS	√	√	√	×	√
MoeTTS[4]	×	√	√	√	√
BuGu bird	√	√	×	√	×

Table 1 Existing Approaches

The voice synthesis of MeoTTS is an application for voice synthesis, it adds some noise point in voice to achieve voice changing. Though BuGu bird can use emotional voice, it is not free, besides, the synthesis of this application is not natural enough, easily distinguish that is AI synthesis.

Exist DL models:

	Model complex	Inferenc e speed	Real time	quality	Multiple languag	Emotion embedd	Trained without

					e	ing	text
VITS	complex	slow	x	high	√	√	x
soVITS [5]	complex	slow	√	standard	√	x	√
Tacotron2[3]	simple	fast	x	low	x	x	x

Table 2 DL models:

## 2.3 For TTS Models

VITS have the best quality at result, though it not as convenient as Tacotron2 and soVITS, it can give the best experience to users.

For TTS models choose, firstly chose Tacotron2[3] because its model is simple and training faster (200 epoch per hour in RTX2060), but after training, according Turing test, even after 500 epoch training it still have some electric voice, and some tone is not standard enough to restore the original character voice. Reverse, VITS models is more complex than Tacotron2. It shows high degree of reduction for characters in short sentence (maybe less than 10 seconds), but used long time to train. Single speaker models spend 1 whole days for 500 epochs. Multiple speaker models with 12 characters 1 hour only trained 20 epoches.

# Chapter 3 Methodology

## 3.1 Approach

### 3.1.1 Model For Original VITS

For develop model, the waterfall model will be used. Because the Deep Learning(DL) results only can know after whole training. And after models be deployed. Then fine-tune the hyper-parameter to test the result. According Turing test to test the models are good training or not. Dataset is from creaked game and self-made.

For the DL models VITS, this is a VAE [4] based models. However, it is a conditional VAE that it added the normalization flows [6] in the encoder of VAE, add normalization flows in Gauss distribution can make it more complex to simulation the voice which make the result of generative more vivid. Then used GAN [5] to train the advertise network, it is also a conditional GAN [3]. The reason that VITS can support single speaker and multiple speakers, the dim of that tensor will be changed as a hyper-parameter.

The structure of VITS is followed:

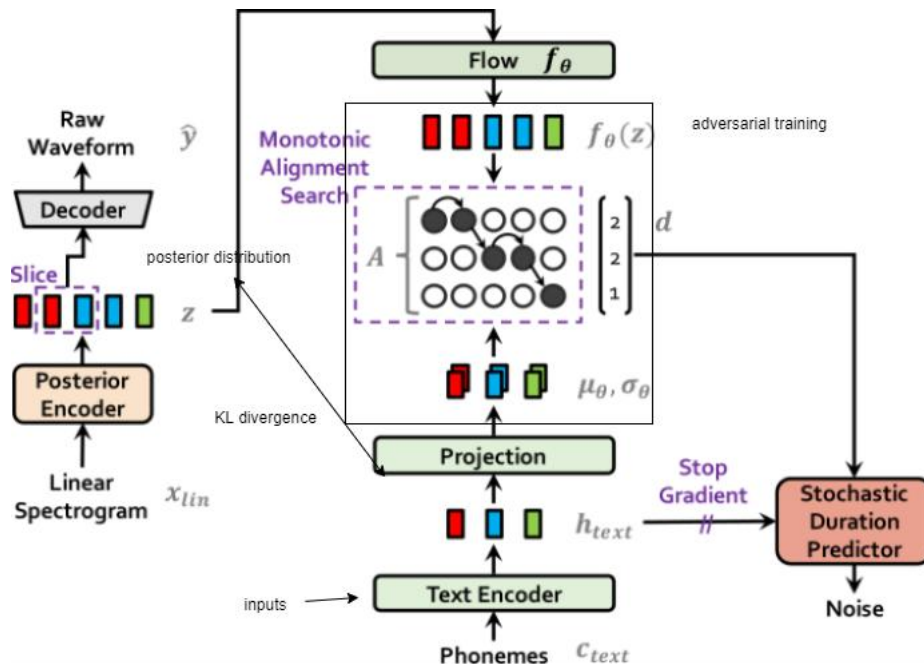


Figure 1 Approach of VITS training



This part is as the encoder of the VAE, it can see that it added flows into normal distribution to make distribution more complex. In VITS training, a transformer [7] based encoder is first used to represent independent text into context sensitive features, and then the output is put into a prior encoder to obtain a prior distribution based on the speaker and text conditions to obtain a latent variable.

In the left section, the waveform is used as input, and the waveform is converted into a linear spectrum through Fourier transform without parameters as a posterior distribution. Then, the prior and posterior distributions are calculated using KL divergence to calculate loss.

$$L_{kl} = \log q\varphi(z|x_{lin}) - \log p\theta(z|c_{text}, A),$$

$$z \sim q\varphi(z|x_{lin}) = N(z; \mu\varphi(x_{lin}), \sigma\varphi(x_{lin})) \quad (1)$$

In this process, the dimension of the text of the prior distribution is expanded to the same dimension as the spectrum. The implementation of this expansion is to dynamically plan during the training process to find the maximum likelihood of the posterior distribution satisfying the prior distribution. This also finds the corresponding time length for each phoneme. At this point, the prior distribution is expanded according to the phoneme to expand the posterior distribution, so that the KL divergence can be calculated when the dimensions are consistent.

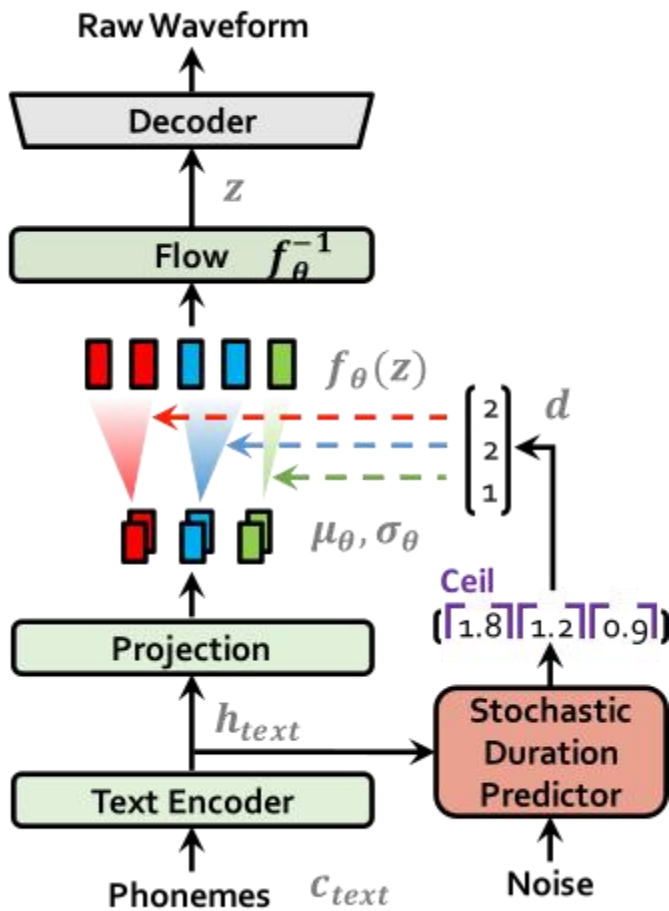


Figure 2 Inference of VITS

In inference process, after get word embedding, it also added flows to generate voice.

However, there is no posterior distribution to verify during reasoning, so the results of dynamic planning will be represented by a time length predictor during training, so that the output of the text encoder can be directly passed into the time length predictor during reasoning, which can directly extend the prior distribution without the posterior distribution.

During reasoning, the text encoder is obtained through text input, and the output is put into a prior encoder and a time length predictor respectively. In the prior encoder, resampling is performed. In the time length predictor, the noise obtained through sampling is used for reverse flow to calculate the duration of each phoneme. The resampling result is then expanded to obtain the spectrum, and then the waveform is obtained through upsampling in the waveform generator.

The model loss includes the KL divergence of a prior encoder and a posterior encoder, the mel spectrum generated by the posterior distribution, and the originally generated calculation of an reconstruction loss, the KL divergence loss of the time length predictor, and the discriminator loss of the adversarial[8] training.

For reconstruction loss:

The mel-spectrogram is used to replace the original waveform as the target data in the reconstruction loss. And samples the potential variable  $z$  onto the waveform, transforming the domain  $y$  Plot into the mel spectral domain  $x$  Plot mel through a decoder.  $L_1$  is then used to reconstruct losses in the prediction and target mel-spectrogram

$$L_{recon} = \|x_{mel} - \hat{x}_{mel}\|_1 \quad (2)$$

For time generate loss  $L_{dur}$ :

The random duration predictor is flow-based generating models, usually trained through maximum likelihood estimation. Variational dequantization and variational data expansion are used to solve these problems. Flow-based can easily build mapping relationship between training distribution and target data set distribution.

$$\text{Log} p_{\theta}(x|c) \geq \mathbb{E} q_{\phi}(z|x) [h \text{log} p_{\theta}(x|z) - \text{log} \frac{q_{\phi}(z|x)}{p_{\theta}(z|c)}] \quad (3)$$

For Adversarial Training:

For training the generator, the adversarial training is used in training. The discriminator  $D$  is used to distinguish between the output generated by the decoder  $G$  and the ground live waveform  $y$ . Moreover, adversarial training uses a least squares loss function and additional feature matching losses for the training generator for speech synthesis.

$$\begin{aligned} L_{adv}(D) &= \mathbb{E}_{(y,z)} [(D(y) - 1)^2 + (D(G(z)))^2] \\ L_{adv}(G) &= \mathbb{E}_z [(D(G(z)) - 1)^2] \\ L_{fm}(G) &= \mathbb{E}_{(y,z)} [\sum^T \frac{1}{N_l} \|D^l(y) - D^l(G(z))\|_1] \end{aligned} \quad (4)$$

The total loss is equaling to

$$L_{vae} = L_{recon} + L_{kl} + L_{dur} + L_{adv}(G) + L_{fm}(G) \quad (5)$$

### 3.1.2 Emotion Training For VITS

The github user innnky[9] have improve this model which can use emotional embedding to label the data set that inference can load the emotional embedding of the dataset to generate audio with appointment emotion.

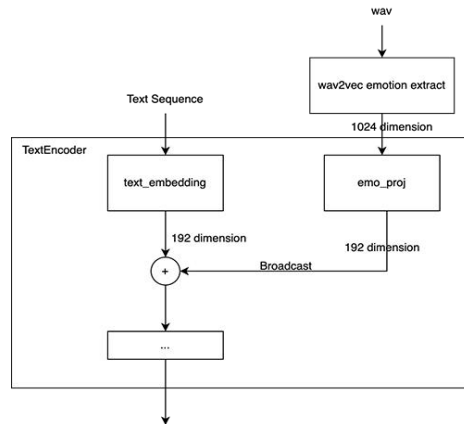


Figure 3 Emotional Embedding

The audio wavs file will be operated by wav2vec pre-training model[10] into 1024 dimensions, then used transformer layer to compress into 192 dimensions and write out a .npy file for per data in dataset to storage embedding of emotion, which can used for emotion inference. Those 192 dimensions emotional embedding will be marge with VITS text encoder in training.

### 3.13 User Interface Model:

This work supports an interface for user to experience this VITS model. The model graphic is below:

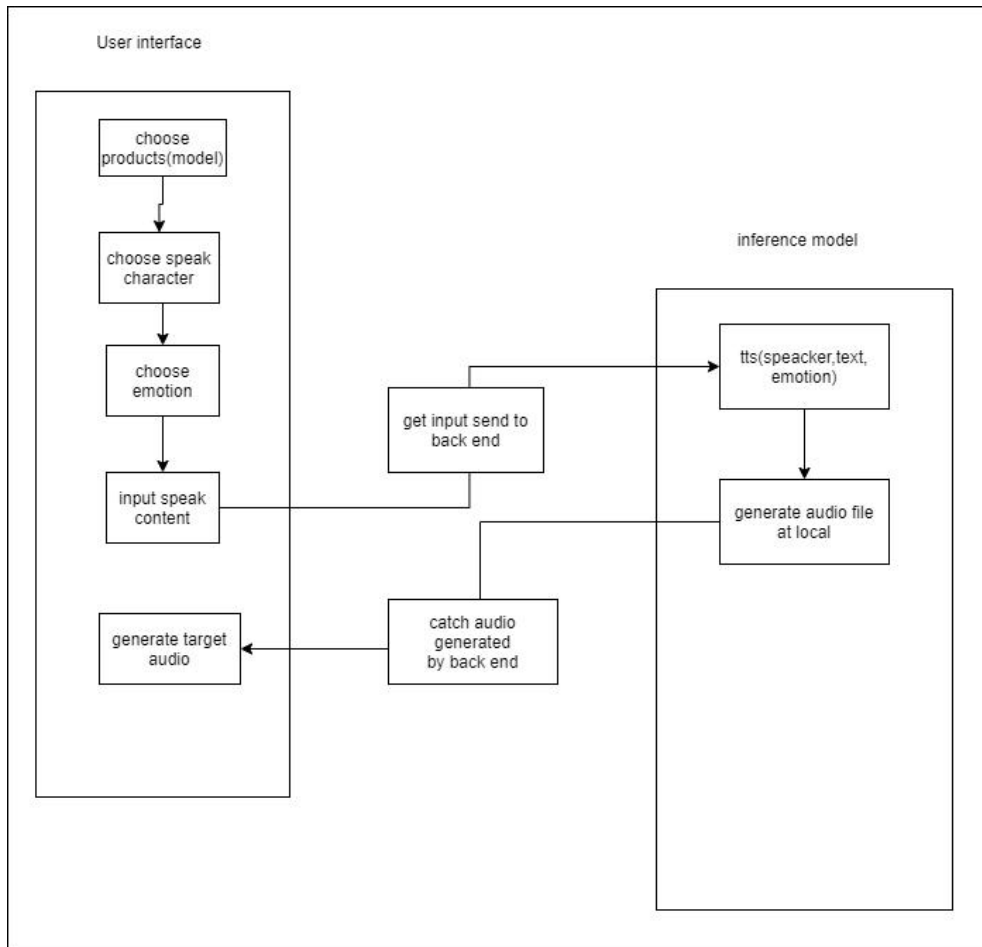


Figure 4 Web interface model

User can use this interface via select the different model have been trained, then select characters which appointed model support. Then select emotion of character and input speak content. After that, interface will send a request to back end to input parameters and call the inference function "TTS". Back end will choose appointment configure file, training inference checkpoint and emotion list according to character chosen by user. Next, back end will generate audio file and the name of the audio file, to make audio file name uniquely, all the audio video

file is named based by time. Because file generate have time sequence, though there are some of audio files generate in one time. Then front end will get file name send by back end to catch audio file and show on web page. User can here it and download it. Or generate it again.

## 3.2 Technology

### 3.2.1 List

The technologies used for the implementation of this project are as follows:

Software	Jetbrains Pycharm
Front end language	Html5, Css, Javascript
Front end frame work	Vue, Gradio
Back-end language	Python
Back-end frame work	Flask, pytorch
Testing methods	The Turing test,MOS
TTS models	VITS
Supporting language	Japanese(most), Chinese,English

Table 3 Develop tools

### 3.2.2 For Back-end Language

For back-end language, python may have some problems that it not effective as Java or PHP and so on. Even though I had learned bootspring for Java, laravel for PHP. The reason why I choose python are that. Firstly, it can easily import the frame work for pytorch. The VITS project is a python project with pytorch frame works, it have a demo inference in file “inference.ipynb”. This file will be rewritten the inference processes in project. Besides, the target aim of this web is multiple languages input and synthesis. A Japanese phonemic package OpenJtalk only

python have, though it had C++ .dll file, it is too old to use, java mix-compile package jni can not import it.

### **3.2.3 For support language**

Original VITS models only support English training. But it is an end to end models, A github user CjangCjengh[11] rewrite the text cleaner and symbols, he use symbols to mark per phoneme and tone, per phoneme will be changed into international phonetic alphabet that, if used other languages text cleaner in text clean file it can input and speak other languages. Because of my data set is from Japanese game. Thus, the model is training into Japanese model.

### **3.2.4 For Data Set**

This data set is got from Japanese word loving adventure game, which have a lot of text word with appointment characters' voice. Loving game have abundant emotion reactive among characters they have sad, happy, surprise, and so many different emotion with vivid voice. That's the reason why those games are chosen to be the data set.

In game's scenario folder, it stores the files which record characters per sentence with specific audio file name and characters' name. This means we only need to write a simple python script to extract the voice data set out with text and characters labels. Then using regular expression to extract them out in to one ".txt" file with special format which fit training style of VITS. The labels per line have two parameters (multiple speaker training have three parameters. Per line for single speaker is made by file path and text label. For multiple speakers training labels are made by file path, index of speaker and text mark labels. The sample are followed.



Figure 5 sample of single speaker data set labels

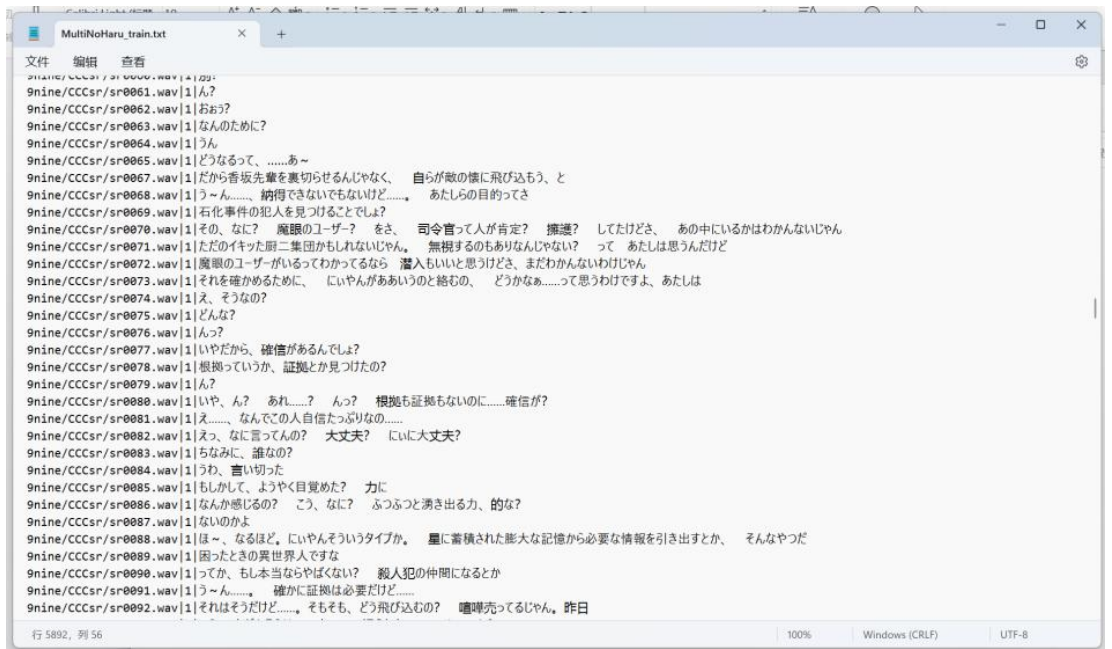


Figure 6 sample of multiple speaker data set labels



Furthermore, the data set have been cleaned to kick out useless data. In this progress, I first using regular expression to kick out the empty sentence which only have symbols, because in game like “!?” expression can show the surprise at that time, but it is no meaning for training. What’s more, I read the data set label again, also drop out the short sentence less than one second or bad quality data like have long pant in one sentence, such as after clime hill too tired. I realize that it will have bad influence in training. However, those data still be storage into another file. After training some epochs, if model work great, those data will join the training. Because it still some kinds of emotion of characters I would try my best to restore whole voice of them. This operation can reduce effective of those bad quality audio files.

For The audio files, to satisfy the format of VITS training, VITS only support .wav format files with 44.100khz and only one channel (two channels training will get an error message. The original file is .ogg format file with 48.000khz sampling rate, 705kbps bits rate use python ffmpeg tool package to transform into “.wav” file and down sample into 44.100khz sampling rate with 352kbps.

### **3.3 Project Version Management**

Time management is a challenge as too much time can be spent on certain tasks leaving too little time for completion of the later tasks. Proper tools and time management techniques can be used to alleviate these challenges.

The waterfall methodology is a sequential liner time process in which each part of a project is handled in order. This gives a clear structure to the project allowing for tasks to be completed in the right order. Gantt charts are typically used alongside this methodology as they allow for the clear structure of each objective in relation to the time frame for the task to be seen visually and assigned to individuals.

Applying these methods to this project allows a time structure to be assigned to all tasks at the beginning of the project. These tasks can then be refined and the progress of the tasks can be tracked throughout the development lifecycle to ensure timely development.

This project will use github and hugging face to do version control.

Github will storage the whole file of the project, the data set, weekly report, front end code and application code. The URL of it is: <https://github.com/AriaMei/9nineEmoTTS>

The hugging face is a github based website, which is especially for deep learning program it can support an interface for python code. Therefore, huggingface is used for showing the demo of program and store relative code for inference and interface. The URL of it is: <https://huggingface.co/spaces/AriaMei/TTSdemo>

For dataset I will shared one of my self-made dataset which stored at Baidu cloud. The URL is: <https://pan.baidu.com/s/136jE4NWhjrdBppURmfpMiw?pwd=8ucq> with password code 8ucq

For VUE Web UI, the source code link is: [https://github.com/AriaMei/VIts\\_UI](https://github.com/AriaMei/VIts_UI)

## Chapter 4 Results

### 4.1 Experimental Setting

#### 4.1.1 Dataset

For data sets, there are three data set in training. First is a 3000 sentences single speaker data set training for normal VITS, second is 4000 sentence single speaker data set for emotional VITS training. Moreover, the second data set is divided into two part, 3000 normal sentences over two second, another part is 1000 sentences which do not have very high quality, for example some sentence less than 2 second or spend long time in pant or sounds too small. After training some epochs those data should add into 3000 sentence part. This is method which can use all data effectively and keep model speak clearly.

The third data set is a 17000 sentence multiple speakers data set, which include the second data set speakers. All the data sets are self-made, methods of making were mentioned in part 3 about data set. Both first two data set done well in training. However, the multiple speakers data set meet some problems. When training after 200 epochs the model still could not speak. The reason I guessed that, there is a characters with shiny disposition, her speaking is usually stutter. It engage 3000 data in data set. Therefore, to valid this guessing, I drop out those 3000 data for this character in data set. Fortunately, this operation is correct, after training for 100 epochs, the multiple speaker models can speak with characters' voice. Finally, the multiple speaker data set become 14000 sentence data.

#### 4.1.2 Training environment

First data set is training for normal VITS models. That time I rent a GPU server at [www.autodl.com](http://www.autodl.com) with two RTX 2080Ti 11GB graphic memory. And training 500 epochs for 2 whole days and half.

When training second data set for emotional VITS, I bought a laptop with RTX 3080 16GB memory. The second data firstly trained 500 epochs for 2 days with 3000 sentences. Then trained anther 500 epochs for 2 days with 4000 sentences data.

The third data set is training on RTX 3080 300 epochs for 3 days with 14000 sentence data set.

## 4.1.2 Evaluation Metrics

### 4.1.2.1 Loss Graph

For training this model, there three mentioned data set in training here are the loss graph for training. As mentioned in part 3, there 4 local parts loss and a total loss in model. The result exhibition is for total loss.

For data set one(reng), 3000 sentences data set for normal VITS training have been training for 2 days.

This data set training for 2 whole days. At 500<sup>th</sup> epoch the loss is from 0.0002 become 0.000188.

For data set two(sora), there are 3000 sentence training for first 500 epochs, then 4000 sentences for next 500 epochs.

This data set have been training for 5 days in 1000 epochs. At 500<sup>th</sup> epoch, the loss is 0.0001867, at 1000<sup>th</sup> epoch, the loss is 0.0001765.

For data set three(multiple), 5 speakers with14000 sentence data set have been trained 300 epochs for 3 days and half.The final loss for 300<sup>th</sup> epochs is 0.0001924

We can see that in first 300 epochs, data set three loss is lower than data set reng and sora, loss of reng and sora is nearly close. At 500<sup>th</sup> epochs, loss of sora is a little lower than reng. For this situation I guess is because of quantity of data set, we can see that, high quantity data set may have high quality.

The table results of three models are followed:

Data set	times	Loss of 300 epochs	Loss of 500 epochs	Loss of 1000 epochs	Final loss	model	Learning rate
reng	2 days	0.0001927	0.000188	/	0.000188	VITS	2e-4

sora	5days	0.0001926	0.0001879	0.0001765	0.0001765	Emo-VITS	2e-4
mutiple	3days	0.0001924	/	/	0.0001924	Emo-VITS	2e-4

Table 4 Training loss

The reason why not change learning rate is that, in original VITS github, it offer an example of config file, the learning rate is using 2e-4 to training. I tried first data set with this learning rate found that it took good results for me. Therefore, learning rate is not changed in my training, because of high training cost of time and money.

#### 4.1.2.2 Mean Option Score (MOS).

Because it is a generate model, the loss only can show the training process, it can't use to judge the result. For testing this model, the only way is let human listen to the systhesis files and judge whether it restore characters' voice like a Turing test. I orally ask 5 people who have known these characters and 5 people who haven't known these characters on QQ to do statistic.

I conducted a crowdsourced MOS test to assess quality. The raters listened to randomly selected audio samples and rated their naturalness on a scale from 1 to 5. The raters were allowed to evaluate each audio sample once. We normalized all audio clips to avoid this effect differences in the magnitude of the scores. The quality of all of this work were evaluated in this way.

Model	MOS
VITS reng data set	4.17(+0.06)
VITS(emotional) sora data set	4.33(+0.05)
VITS(emotional) mutiple speaker 1	4.23(+0.07)
VITS(emotional) mutiple speaker 2	4.30(+0.05)
VITS(emotional) mutiple speaker 3	4.28(+0.06)
VITS(emotional) mutiple speaker 4	4.15(+0.3)

VITS(emotional) mutiple speaker 5	4.26(+/-0.05)
-----------------------------------	---------------

Table 5 MOS of VITS

For reng data set. Listener mostly consider that, over 200 epochs can clearly distinguish the tone of character, when training 500 epochs, 4 of 5 person who haven't here this character voice thinks this voice is human being.

For sora data set, mostly have same result with reng data set. However, it can simulate speaker's tone and emotion more similar. At 500<sup>th</sup> epoch, all of person who have know this character's voice thinks that this model restores the original character vivid. And they compare 500 epochs and 1000 epochs model, think voice of 1000 epochs is a little better than 500 epochs.

For multiple data set over 100 epochs all the speakers in models can restore tone of characters. At 300<sup>th</sup> epoch, it worked as well as dataset two. Moreover, we compared same characters sora in this model, which is same character in data set sora. Three person thinks that 1000 epochs sora data set is better, two persons think both of them are same level.

### 4.1.3 Baseline Model

This project will compare the emotional-VITS with the original VITS and Tacotron2 models. The comparison will be made in terms of training time, inference time and inference results. The performance of the emotional-VITS model will be demonstrated to prove how merit it is.

<b>TACOTRON2</b>	Tacotron2 is a neural network-based TTS synthesis model, which is characterized by adopting an end-to-end generation approach without the need for intermediate phonemes or acoustic features
------------------	---

## 4.1.4 Parameters Setting

### 4.1.4.1 Training Environment

First data set is training for normal VITS models. That time I rent a GPU server at [www.autodl.com](http://www.autodl.com) with two RTX 2080Ti 11GB graphic memory. And training 500 epochs for 2 whole days and half.

When training second data set for emotional VITS, I bought a laptop with RTX 3080 16GB memory. The second data firstly trained 500 epochs for 2 days with 3000 sentences. Then trained another 500 epochs for 2 days with 4000 sentences data.

The third data set is training on RTX 3080 300 epochs for 3 days with 14000 sentence data set.

### 4.1.4.2 Hyper parameters

Due to the large number of parameters set in the original model, time constraints prevented all changes from being made for comparison, and only some of the hyperparameters that would affect the training efficiency were changed. The content table is followed:

	Reng data set	Sora data set	Multiple speakers data set
Learning rate	0.0002	0.0002	0.0002
Epochs	500	1000	300
Batch size	8	16	16
FTP 16	Ture	False	False
Speakers	1	1	5

Table 6 Training parameters

For the choice of learning rate and epochs has been mentioned in the previous article, for the choice of batch size, the rented server's 2080ti only has 12g of video memory, but he can support half-precision (ftp-16) calculation, which can improve the training efficiency of the model

to some extent. However, the last two datasets were trained with my own laptop, which has 16G xian'cun video memory, so I could use a larger batch size, but it does not support ftp 16 computation, so I had to turn off this option for training.

## 4.2 Performance Comparison

In order to compare the efficiency of the models, I trained the Tacotron 2 TTS model on both the reng and sora datasets, keeping the same training environment and settings as the VITS model to control the variables. The Tacotron 2 model was trained much faster than VITS, completing 500 epochs of the dataset in less than one day. I also used the MOS evaluation method to evaluate the training results, and the table below shows the comparison between VITS and Tacotron 2.

Model	MOS
Tacotron 2 reng data set	4.03(+/-0.04)
Tacotron 2 sora data set	4.18(+/-0.03)
VITS reng data set	4.17(+/-0.06)
VITS sora data set	4.33(+/-0.05)

Table 7 Compare VITS and Tacotron2

The Tacotron2 model is slightly less effective than the VITS model in terms of training results, and when evaluated, the Tacotron2 timbre reproduction is similar to the VITS model, but the naturalness of intonation is inferior to that of the VITS model. In addition, Tacotron2 is faster than VITS in training, but slower than VITS in inference.

## 4.3 Ablation study

For this part, emotional embedding did ablation study, For the loss of model which mentioned in part 4.31, mostly they are the same. Using emotional training didn't affect having experience a lot. Both of two models can generate voice like human being, even some people can't detect them. However, it will affect user experience when generate. When user give a lone sentence for normal VITS without mark, the intonation of this sentence will be very strange. To improve it, user should input some symbols and space to mark them like “祭↓りに 行↑った だ↓よね、 知↑らない 女 の 子 と 一 緒 に い て” to help model make pause in reading. However emotional VITS



only need an emotional embedding, default random choose an embedding file in embedding folder, or manually input emotional embedding root to generate audio file. It more portable than normal VITS models.

## 4.4 Model Deployment

### 4.4.1 Web User Interface

For deployment, this project supports a web user interface to help user use this model portable. The web UI designing is below:

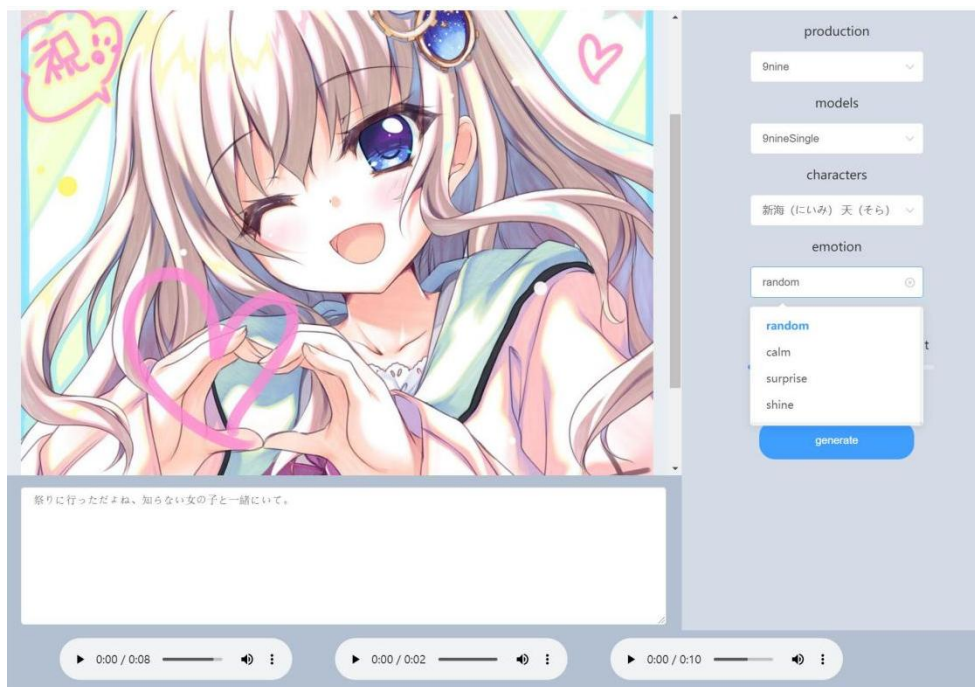


Figure 7 Inference result

Here are four selections for user and one input text bar for user to generate models. For production only support 9nine this game's characters. Then select models, like 9 nine models, there single speaker models and multiple speakers model supported for 9nine production. What's more different models support different characters. For example, single speaker only support speaker "sora", multiple speakers models can support 5 characters, it can easily see that, choices in different models are different.

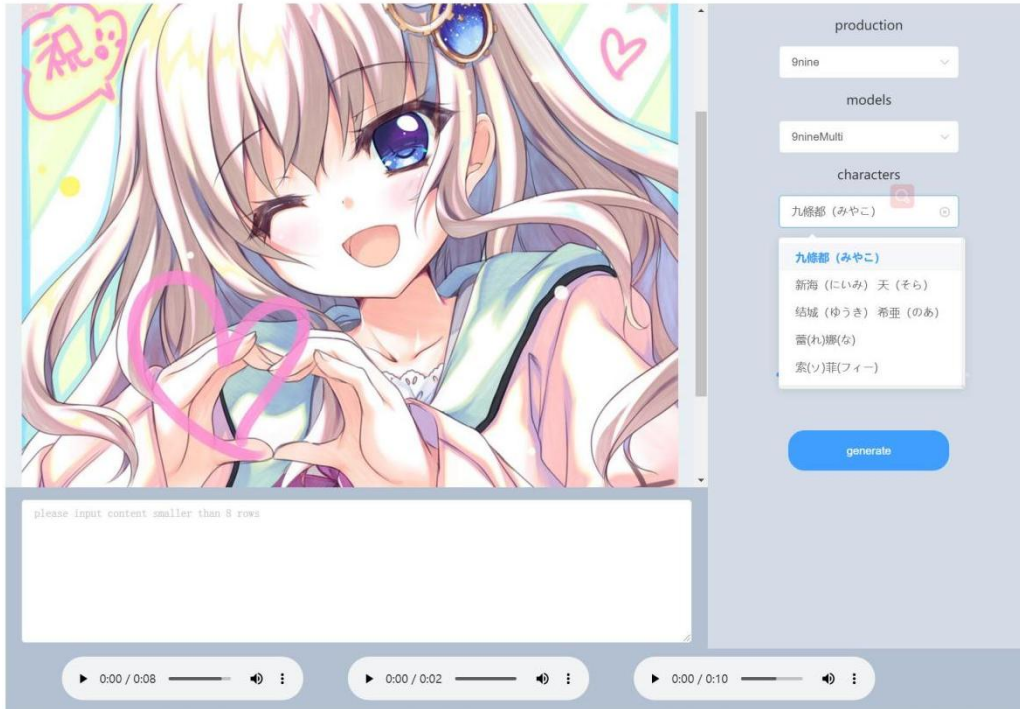


Figure 8 Multiple speaker's options

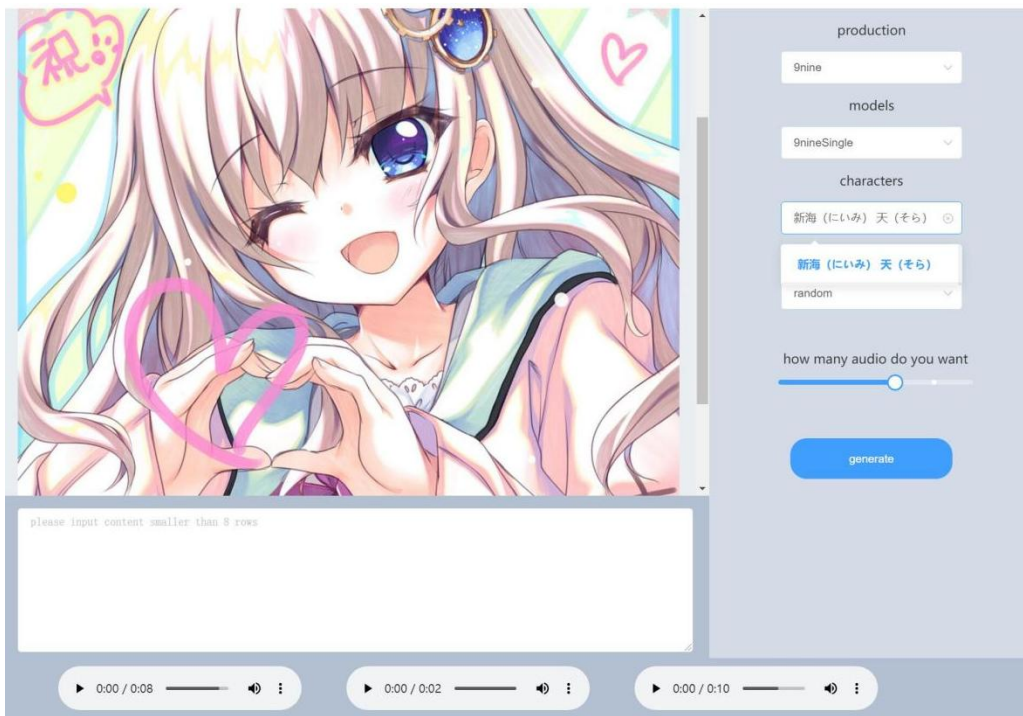


Figure 9 Single speaker options

For the next is choose emotion as result figure, I screen out some great emotion embedding in data set as inference dependence for user to do voice synthesis, then select how much audio files user want, the range is from one to six, they can screen out a best one which is the most fit their requirements.

However, rent a server can support VITS inference is too expensive to rent. Therefore, this web UI have not been deployed yet. Fortunately, huggingFace [11] support a python API package Gradio [12] to provide python program a web UI, the most advantage of it is that, it is free to deploy on server with 2 cores and 16G memory CPU. This server is fit my requirements for voice synthesis. Therefore, there additional web UI designed by Gradio:

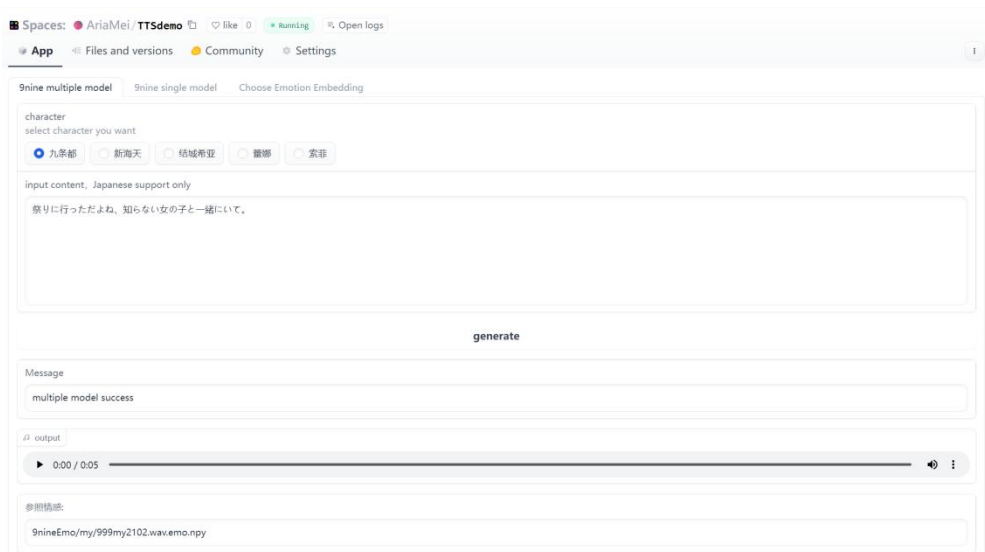


Figure 10 UI example for Gradio

Although this UI is much simpler than Vue UI designing, it still implementing all the functions. What's more I add an output emotion embedding (the last line in this figure) that, this embedding can be used in to third part "Choose Emotion Embedding"

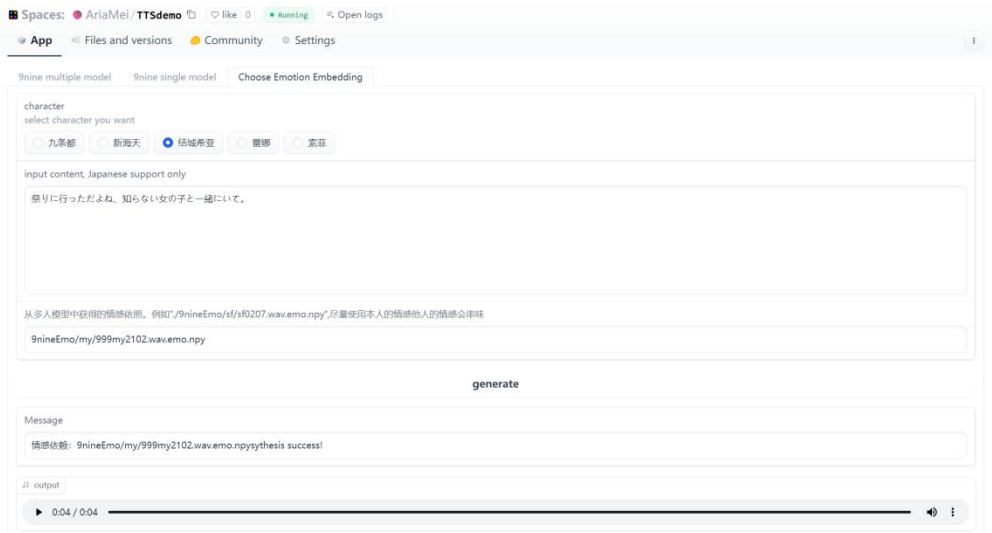


Figure 11 Choose embedding

The first tab is using random emotion in data set, if user get some great emotion, they can input this emotion in third tab, to record their interest emotion. Moreover, although I suggest that user only user speakers' own emotion to inference, there still some good result that user other speakers' emotion. Like example in figure, using “my” (short of miya) emotion in “结城希亚” speaking, sometimes it will happen good result, sometimes the voice is hardly to declare the tone is below to emotion owner or voice owner. This is interesting part for user to experience.

The deployed website demo is <https://huggingface.co/spaces/AriaMei/TTSdemo>

#### 4.4.2 Web Test Case

For web UI, I did a functional test to make sure all the function can be used with right output.

Test Type	Functional Area	Test case	Test Steps	Except Results
-----------	-----------------	-----------	------------	----------------

<b>Web inference API</b>	Single speaker tab	Randomly generate an audio file and get appoint emotion embedding	<ol style="list-style-type: none"> <li>1. Select the single speaker model.</li> <li>2. click supported Character</li> <li>3. Input generate content</li> <li>4. Click generate button</li> </ol>	Return a success message and an audio file have right pronunciation with its emotional embedding.
	Multiple speaker tab	Choose one of supported character and randomly generate an audio file and get appoint emotion embedding	<ol style="list-style-type: none"> <li>1.Select the single speaker model.</li> <li>2.Select one character supported by model.</li> <li>3.Input generate content.</li> <li>4.Click generate button.</li> </ol>	Return a success message and an audio file have right pronunciation in right character voice. And get its emotional embedding.
	Assign emotion generate tab	Choose one of supported character and input emotion embedding user want.	<ol style="list-style-type: none"> <li>1.Select the single speaker model.</li> <li>2.Select one character supported by model.</li> </ol>	Return a success message and an audio file have right pronunciation in right

			3.input special emotion embedding get from last two tabs 4.Input generate content. 5.Click generate button.	character voice and similar emotion with emotion embedding chosen
--	--	--	---	---

## Chapter 5 Professional Issues

### 5.1 Project Management

#### 5.1.1 Activities

Task	Start date	End date	Estimated Time	Complete status
search of similar research work.	2022/10/21	2022/11/3	14	complete
comparison of different models.	2022/11/4	2022/11/16	13	complete
comparison of implement of models	2022/11/17	2022/11/28	11	complete
found target game and get game data	2022/10/21	2022/11/8	18	complete
used regulation expression to make normal audio text file in format	2022/11/9	2022/11/13	5	complete
Cleaned out useless data	2022/11/14	2022/11/15	2	complete
Rent a GPU server at <a href="http://www.autodl.com">www.autodl.com</a>	2022/11/25	2022/11/25	1	complete



Installed relevant dependence from requirement.txt file	2022/11/26	2022/12/3	8	complete
Trained reng data set with normal VITS models for 500 epochs	2022/12/4	2022/12/30	25	complete
Trained sora data set with emotional VITS for 1000 epochs	2023/1/3	2023/1/11	9	complete
Trained multiple speakers data set with emotional VITS for 300 epochs	2023/2/3	2023/3/10	36	complete
Through Turing test, let tester judge the result of generation.	2023/11/25	2023/3/12	107	complete
Rewrote and packaged inference method interface to fit the interface of front end	2023/3/20	2023/3/23	4	complete

designed and coded a web interface based on Vue framework	2023/3/24	2023/3/27	4	complete
Used Gradio package to write a web interface to deploy	2023/3/24	2023/3/27	4	complete
Test web UI function on huggingface	2023/3/29	2023/4/2	4	complete
Disposal graphs of result and models	2023/4/1	2023/4/7	7	complete
Made a PPT and an introduction video for presentation	2023/4/8	2023/4/10	3	complete
Create Poster	2023/4/15	2023/4/23	9	complete

Table 8 Activities

### 5.1.2 Schedule

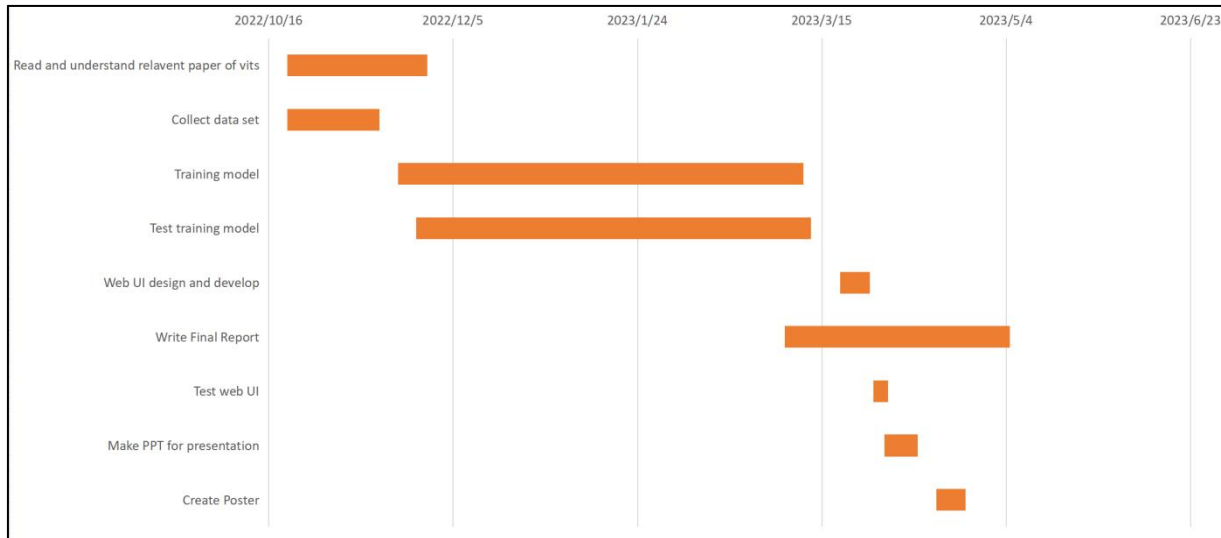


Table 9 Gantt diagram

### 5.1.3 Project Data Management

Data management of the software and code used throughout the project is done by creating a Github and huggingface repositories that were used to store all code associated with the project. Updates are made after each major step of development so that code can be quickly and easily detected and restored in the event of any problems or data loss during the project.

For project storage, project data is divided in Four part:

1. Data set, data set will restore at local, and backup at another hard drive.
2. Model and training result. Because of some data trained at cloud server, those data was stored at cloud server.
3. Program for web UI, it was restored at <https://huggingface.co/spaces/AriaMei/TTSdemo>, and also restore the model code and final result of training.
4. Project report and relevant graphs and text files. This part will backup to github:

Except data set, other files will upload at github URL:

In different branches.

## 5.1.4 Project Deliverables

Those files will be submitted on project folder:

- 1) Proposal.
- 2) Progress report.
- 3) Weekly report.
- 4) Final report.
- 5) Interface code.
- 6) Link of deployed model.

## 5.2 Risk Analysis

Risk ID	Potential Risk	Cause ID	Potential Causes	Severity	Likelihood	Risk	Mitigation ID	Mitigation
R1.1	Missed deadline	C1.1.1	illness	1	3	3	M1.1.1	Take some medical care
		c1.1.2	Poor time management	4	1	10	M1.1.2	work overtime, follow gantt graphic
R1.2	Feature creep	c1.2.1	Over-ambitious project spec.	4	2	12	M1.2.1	change final aim and discuss with supervisor
R1.3	webpage bug	c1.3.1	differnt dependency package version	3	2	4	M1.3.1	update python dependency or de-update some dependency
		c1.3.2	Poor test plan	3	1	7	M1.3.2	develop with testing, try to make test plan clearly and enrich it
R1.4	Loss data	c1.4.1	poor version control	4	4	15	M1.4.1	update project version on time
		c1.4.2	computer or device broken	2	2	15	M1.4.2	Update project to multiple cloud platform

Figure 12 Risk analysis

Missing deadline sometimes will happen during the development time, it may make by illness or poor time management. My solution of it is follow the time table of gantt graph, at gantt graph I also leave some much more time for ensure time enough.

Over ambitious may make feature creep, my project mostly will happened is want to design a gorgeous API UI and deploy it on server. I told my supervisor that, first make a local host API, if time enough, I can deploy it on server.

Different dependence package and poor testing will cause webpage bug. My solution of it is try my best to do a brain storm to guess which bug may happen and use software like selenium IDE test it systematically.

Loss data is a heavy problem between development. My solution of it is solve my data on could like Github, Baidu cloud both. What's more,, upload per update in time to ensure all data have been save.

These risks have been categorised into severity based on the likelihood and impact each risk poses to the project. This is shown in the table below where:

Key	Meaning	Description
Green	Low/Little risk	These risks will have little impact on the project and recovery from these risks will be simple.
Yellow	Moderate Risk	These risks will have a noticeable impact on the project development and will take time to recover from.
Red	High/Great risk	These risks will have a catastrophic impact on the project and could cause the final deadline to be missed or the project to fail. These risks must be constantly considered during the project development.

Impact -> Likelihood	Low impact				High impact
Low likelihood			C1.3.2	C1.1.2	
		C1.3.1		C 1.2.1	C 1.4.2
	C1.1.1				
					C 1.4.1
High likelihood					

Table 10 Risk likelihood and Impact

### 5.3 Professional Issues

#### 5.3.1 Technology Issue

This project is belonging to Artificial Intelligence (AI) and deep learning. It only can be run with GPU rather than CPU. When running this project, it will occupy a lot of sources of GPU server. If lots of users using it at same time, the GPU may out of memories. To clarify, this design sample is not suitable for the large commercial using. It intends to personal or few of users to use.

#### 5.3.2 Ethical Issue

Ethical issues are critical in the development and use of TTS systems, as they are closely related to the protection of human subjects and the integrity of research. The following are some possible ethical issues of a TTS system:

Protection of Human Subjects: The use of TTS systems may involve the collection and processing of personal data, such as voice samples, speech patterns, and linguistic

preferences. Therefore, it is essential to protect the privacy and confidentiality of human subjects, and obtain informed consent before collecting their data. Additionally, researchers should ensure that the data collection process is non-invasive and does not cause any harm or distress to human subjects.

**Fair Use of Data:** The use of TTS systems may require large amounts of data, such as text corpora, audio recordings, and metadata. Therefore, it is necessary to ensure that the data used in TTS systems are obtained legally, without violating any copyright, patent, or trade secret laws. Additionally, researchers should ensure that the data are used fairly and appropriately, without bias or discrimination against any individual or group.

**Accountability and Transparency:** The use of TTS systems may create new forms of accountability and transparency, as users may rely on the accuracy and reliability of the system to perform various tasks. Therefore, researchers should ensure that the TTS system is transparent and accountable, by providing clear and concise information about its performance, limitations, and risks.

### **5.3.3 Legal issue**

In the development and use of TTS systems, legal issues are an important professional issue. Among them, intellectual property, data protection and privacy, and compliance with international standards are potential legal issues for TTS systems.

Firstly, intellectual property is one of the legal issues that must be considered in the development and use of TTS systems. TTS system development may involve patents, copyrights, and trademarks, such as the core technology of TTS systems, speech synthesis algorithms, and audio files used. Therefore, when developing and using TTS systems, it is essential to ensure that no intellectual property rights of others are infringed and to obtain appropriate licenses or permissions before using any copyrighted materials.

Secondly, the collection and handling of personal data may be involved when using TTS systems, such as biometric data, location data, and behavioral data. Therefore, protecting users' data protection and privacy rights is an important legal issue. Researchers should ensure that TTS systems comply with relevant data protection and privacy laws, such as the General Data Protection Regulation (GDPR), and obtain informed consent before collecting user data.

In the development and use of TTS systems, compliance with international standards and regulations, such as ISO standards for speech technology and guidelines for assistive technology for disabled users, may be required. Therefore, it is essential to ensure that TTS systems comply with or exceed relevant standards and regulations and undergo appropriate certification or accreditation before being released to the market to ensure their compliance and reliability.

#### **5.3.4 Social Issues**

Social issues are also significant in the development and use of TTS systems, as they are closely related to the social and cultural impact of the technology. The following are some possible social issues of a TTS system:

**Cultural Diversity and Inclusion:** The use of TTS systems may involve the representation and recognition of different languages, dialects, and accents. Therefore, researchers should ensure that the TTS system is culturally sensitive and inclusive, and avoids any stereotyping or discrimination against any language or culture.

**Social Responsibility and Accountability:** The use of TTS systems may have significant social and economic impact, as it may affect the employment, education, and communication of millions of people worldwide. Therefore, researchers should ensure that the TTS system is socially responsible and accountable, by promoting ethical and sustainable development, and avoiding any negative impact on society and the environment.



Public Awareness and Engagement: The use of TTS systems may require public awareness and engagement, as users may need to understand the benefits and risks of the technology to make informed decisions. Therefore, researchers should ensure that the TTS system is transparent and accessible, by providing clear and concise information about its purpose, functionality, and security.

### **5.3.5 Environmental Issues**

Environmental issues are also relevant in the development and use of TTS systems, as they are closely related to the environmental impact of the technology. The following are some possible environmental issues of a TTS system:

Energy Efficiency and Carbon Footprint: The use of TTS systems may consume significant amounts of energy and generate carbon emissions, especially if the system is used for high-volume or long-duration tasks. Therefore, researchers should ensure that the TTS system is energy-efficient and has a low carbon footprint, by using renewable energy sources, optimizing algorithms and hardware, and reducing waste and emissions.

Sustainable and Responsible Supply Chain: The use of TTS systems may involve the use of various raw materials, such as metals, plastics, and chemicals, which may have adverse environmental and social impacts. Therefore, researchers should ensure that the TTS system is sourced from sustainable and responsible suppliers, who comply with relevant environmental and social standards, and that the supply chain is transparent and traceable.

## Chapter 6 Conclusion

### 6.1 Reflection And Conclusion

The most challenging part of this project is the acquisition of the self-voiced dataset. Many of the existing datasets are a bit stiff and do not sound emotional enough, and it is difficult to obtain speech that is both clear and emotional. So we came up with the idea of using the voiceover from the game, which has both of these advantages along with a large amount of text annotation, which makes the training of the project much easier

The training in this project showed that the clarity of the dataset and the fluency of the speaker's speech were important in the training. In the multi-person training model, the entire multi-person model could not be vocalized because one person did not spit out the words clearly, even though it was only a small part of the entire dataset.

In addition, I found that in the inference of the original VITS model, the reduction of short sentences is very high, but the long sentences are often not reduced enough, and need to be controlled by annotation of pronunciation intonation at the time of input, but the sentiment model solves this problem to some extent. As long as the corresponding sentiment dependency is selected at the time of input, a character's voice can be better restored. With a large number of training texts, a large number of sentiment dependencies are also obtained to restore the character's voice, but this also has some drawbacks, which will be mentioned in the next section.

This project provides a web-based GUI for users to experience. Users can select different characters for speech synthesis in the multiplayer model or single-player model by themselves, and can also speak strong sentiment dependencies in two tabs for specified sentiment inference in the third tab. However, the free server provided by huggingface only has 2 cores and 16G of CPU for inference, so the generation speed is slow, and it takes about 20 seconds on average to infer sentences of 6 seconds in length.

### 6.2 Future Work

The main defect of this project is in the emotion inference, from the model itself, the emotion inference relies on manual selection, the model itself does not know what is "calm", "cheerful"

these emotions, need human labeling, but the training of more than 14000 sentences data set, has an equal amount of emotion files, want to filter is difficult, the future may add a clustering algorithm, the emotion embedded clustering, the clustered embedding will be much easier to label, the user can also be more accurate in the use of the specified emotion.

In terms of web deployment, the free server can only use CPU for inference, which is inefficient for a deep model. Also, the user interface of this free server uses gradiod's GUI, which is less customizable and less beautiful. The web pages designed by ourselves cannot be deployed. In the future, we will deploy a graphics server with good inference effect to deploy the web pages designed by ourselves, and we can also increase the inference speed of speech synthesis significantly, which can increase the user experience.

## References

- [1] J. Kim, J. Kong, and J. Son, “Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech,” Jun. 2021, [Online]. Available: <http://arxiv.org/abs/2106.06103>
- [2] S. Schneider, A. Baevski, R. Collobert, and M. Auli, “wav2vec: Unsupervised Pre-training for Speech Recognition,” Apr. 2019, [Online]. Available: <http://arxiv.org/abs/1904.05862>
- [3] J. Shen *et al.*, “Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions,” Dec. 2017, [Online]. Available: <http://arxiv.org/abs/1712.05884>
- [4] luoyi, “luoyily/MoeTTS: Speech synthesis model /inference GUI repo for galgame characters based on Tacotron2, Hifigan and VITS.” <https://github.com/luoyily/MoeTTS> (accessed Nov. 09, 2022).
- [5] Francis-Komizu, “SoVits.”
- [6] D. J. Rezende and S. Mohamed, “Variational Inference with Normalizing Flows.”
- [7] A. Vaswani *et al.*, “Attention Is All You Need.”
- [8] I. Goodfellow *et al.*, “Generative adversarial networks,” *Commun ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020, doi: 10.1145/3422622.
- [9] “emotional-vits/out.png at main · innnky/emotional-vits.” <https://github.com/innnky/emotional-vits/blob/main/resources/out.png> (accessed Mar. 26, 2023).
- [10] S. Schneider, A. Baevski, R. Collobert, and M. Auli, “wav2vec: Unsupervised Pre-training for Speech Recognition,” Apr. 2019, [Online]. Available: <http://arxiv.org/abs/1904.05862>
- [11] “Hugging Face – The AI community building the future.” <https://huggingface.co/> (accessed Apr. 02, 2023).
- [12] “Gradio.” <https://gradio.app/> (accessed Apr. 02, 2023).