EMOTIONAL TEXT TO SPEECH
A DEEP LEARNING APPROACH

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

BY

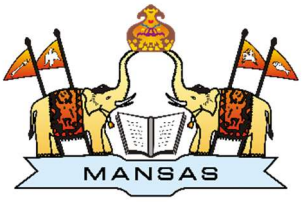
K.S.V Ravi Sastry
(20331A0588)

N.V.S.R Srujan
(20331A05C6)

K. Sarath Kumar
(20331A0584)

I. Deena
(20331A0573)

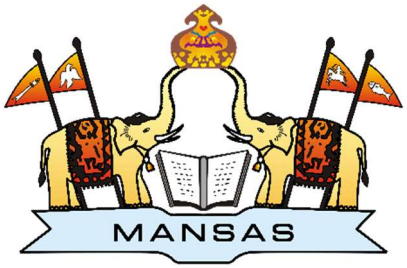
Under the Supervision of
Dr. C. Kalyana Chakravarthy
Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MAHARAJ VIJAYARAM GAJAPATHI RAJ COLLEGE OF ENGINEERING
(Autonomous)
(Approved by AICTE, New Delhi, and permanently affiliated to JNTUGV, Vizianagaram), Listed u/s 2(f) & 12(B) of UGC Act 1956.
Vijayaram Nagar Campus, Chintalavalasa,Vizianagaram-535005, Andhra Pradesh

APRIL, 2024

CERTIFICATE



This is to certify that the project report entitled "**EMOTIONAL TEXT TO SPEECH – A DEEP LEARNING APPROACH**" being submitted by K.S.V Ravi Sastry(20331A0588), N.V.S.R Srujan(20331A05C6), K. Sarath Kumar(20331A0584), I. Deena(20331A0573), in partial fulfillment for the award of the degree of "**Bachelor of Technology" in Computer Science and Engineering** is a record of bonafide work done by them under my supervision during the academic year 2023-2024.

Dr. C. Kalyana Chakravarthy
Professor,
Supervisor,
Department of CSE,
MVGR College of Engineering(A),
Vizianagaram.

Dr. T. Pavan Kumar
Associate Professor,
Head of the Department,
Department of CSE,
MVGR College of Engineering(A),
Vizianagaram.

External Examiner

DECLARATION

We hereby declare that the work done on the dissertation entitled "**EMOTIONAL TEXT TO SPEECH – A DEEP LEARNING APPROACH**" has been carried out by us and submitted in partial fulfillment for the award of credits in Bachelor of Technology in Computer Science and Engineering of MVGR College of Engineering (Autonomous) and affiliated to JNTUGV, Vizianagaram. The various contents incorporated in the dissertation have not been submitted for the award of any degree of any other institution or university.

ACKNOWLEDGEMENTS

We express our sincere gratitude to **Dr. C. Kalyana Chakravarthy** for his invaluable guidance and support as our mentor throughout the project. His unwavering commitment to excellence and constructive feedback motivated us to achieve our project goals. We are greatly indebted to him for his exceptional guidance.

Additionally, we extend our thanks to **Prof. P. S. Sitharama Raju (Director)**, **Prof. Ramakrishnan Ramesh (Principal)**, and **Dr. T. Pavan Kumar (Head of the Department)** for their unwavering support and assistance, which were instrumental in the successful completion of the project. We also acknowledge the dedicated assistance provided by all the staff members in the Department of Computer Science & Engineering. Finally, we appreciate the contributions of all those who directly or indirectly contributed to the successful execution of this endeavor.

K.S.V Ravi Sastry (**20331A0588)**

N.V.S.R Srujan (**20331A05C6)**

K. Sarath Kumar (**20331A0584)**

I. Deena (**20331A0573)**

LAST MILE EXPERIENCE (LME)

PROJECT TITLE

EMOTIONAL TEXT TO SPEECH – A DEEP LEARNING APPROACH

BATCH NUMBER – 7B

BATCH SIZE – 4

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Name: K.S.V Ravi Sastry
Email: ravisastrykolluru2021@gmail.com
Contact Number: 9347914462

Name: N.V.S.R Srujan
Email: srujanraghc@gmail.com
Contact Number: 7032858376

Name: K. Sarath Kumar
Email: sarathramkilli2003@gmail.com
Contact Number: 6304662018

Name: I. Deena
Email: ingilapallideena1@gmail.com
Contact Number: 7416943679

Project Objectives

1. Develop a TTS system that adds emotions to speech for engaging user experiences and effective communication.
2. Improve accessibility in communication tech for users with disabilities through emotionally expressive interaction.
3. Reduce the gap between human and synthetic speech with emotionally intelligent TTS for natural, empathetic interactions.

Project Outcomes

1. Develop a TTS system to convey anger, sleepiness, disgust, and neutral emotions for enhanced communication.
2. Enhance user interaction with emotionally expressive synthetic speech, especially for those with unique communication needs.
3. Extend TTS applications to diverse user requirements, including assistive technology, with personalized and adaptable communication tools.

Domain of Specialisation

The project falls within the domain of Deep Learning, Natural Language Processing (NLP), and Human-Computer Interaction (HCI), with a specific focus on Emotional Speech Synthesis and Assistive Technology.

How your solution helping the domains?

Our solution utilizes deep learning for actionable insights. Emotionally expressive TTS aids communication for those with disabilities. In education, accessible materials enhance learning for all. In customer service, empathetic TTS improves user experience.

List the Program Outcomes (POs) that are being met by doing the project work

PO9: Individual and teamwork	Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.
PO10: Communication	Communicate effectively on complex engineering activities with the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11: Project management and finance	Demonstrate knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work as a member and leader in a team, to manage projects and in multi-disciplinary environments.
PO12: Life-long learning	Recognize the need for, and have the preparation and ability to engage in, independent and life-long learning

End Users of Your Solution

Non-verbal and autistic individuals, those with learning or visual impairments, and students in these categories.

Project Supervisor

Name: Dr.C. Kalyana Chakravarthy
Designation: Professor
Email: kalyan@mvgrce.edu.in
Contact Number: 9848129319

ABSTRACT

The concept of emotional text-to-speech (TTS) aims to create computer-generated voices that convey emotions when reading text. Unlike current approaches relying on emotion detection within the text, this proposed technique directly converts the input text into the desired emotional output specified by the user. This technology holds promise not only in robotics but also for individuals facing communication challenges, such as non-verbal or autistic individuals. It could offer a more expressive and personalized means of communication for those with unique needs, expanding its potential impact beyond technological applications.

CONTENTS

List of Abbreviations

AI	–	Artificial Intelligence
ML	–	Machine Learning
NN	–	Neural Networks
ANN	–	Artificial Neural Networks
CNN	–	Convolutional Neural Networks
RNN	–	Recurrent Neural Networks
SVM	–	Support Vector Machine
KNN	–	K-nearest neighbors
MOS	–	Mean Opinion score

List of Figures

List of Tables

CHAPTER 1
INTRODUCTION

Over the last two decades, the area of computer science has seen massive transformations, resulting in a new era of technical innovation comparable to the industrial revolution. With technology being thoroughly integrated into daily life, the landscape of tasks and interactions has shifted considerably, particularly with the widespread influence of Artificial Intelligence (AI). This growth raises questions about the future of AI, particularly in important sectors like Natural Language Processing (NLP). Among the numerous applications of NLP, we concentrate on text and voice processing, with particular focus on Expressive Text-to-voice (TTS) tasks.

1.1.Problem Identification:

Text-to-speech (TTS) technology is critical in improving access to information and communication for people with learning disabilities, visual impairments, and literacy issues. However, traditional TTS systems often lack emotional expressiveness, which is a barrier to effective communication and engagement. This gap highlights the necessity to address the emotional dimension of speech synthesis in order to improve inclusivity and user experience

1.2. Problem Definition:

The task at hand is to develop a Text-to-Speech program that can not only produce fluent and clear speech but also convey emotion effectively. Current TTS systems frequently produce speech with a neutral and emotionless tone, emphasizing the gap in emotional expressiveness. Overcoming this challenge requires extensive exploration and adaptation of emotional modules within existing speech synthesis models

1.3. Objective:

Our objective is to improve text-to-speech technology by giving it the ability to express emotions effectively. By integrating emotional intelligence into TTS systems, we want to improve user experience and accessibility, bridging the gap between natural and synthetic speech

1.4. Existing Models:

Existing TTS methods mostly use neural networks for speech synthesis. In our research, we will look at the key concepts and prior non-neural network-based technologies before evaluating the composition and efficacy of the latest neural network architectures utilized in

TTS. In addition, we will look at the cutting-edge approaches used in TTS development, with a focus on neural networks and their architectural layouts in achieving high-quality speech synthesis.

CHAPTER 2
LITERATURE SURVEY

Standard Text-to-Speech (TTS) systems often lack the ability to convey emotions effectively, hindering natural and engaging communication. This project aims to bridge this gap by incorporating emotion recognition and expression into speech synthesis. However, the task of converting text to emotional speech is very complicated. Several researchers have proposed various methodologies and technologies for achieving this goal.

Sejnowski and Rosenberg (1987)[1] pioneered parallel networks for English text pronunciation. McCulloch, Bedworth, and Bridle (1987)[2] focused on practical TTS implementation with "NETspeak." Damper (1995)[3] explored self-learning and connectionist approaches to text-to-phoneme conversion. These works collectively advance TTS by leveraging neural networks, optimization, and adaptive learning mechanisms.

Traditional TTS systems employ separate modules for text analysis, acoustic modeling, and speech synthesis, as outlined by Jurafsky and Martin (2020) in "Speech and Language Processing"[4]. Stanford's lecture materials on the "Diphone TTS architecture" [5]further illustrate this modular approach.

Diphone TTS Architecture[5] Stanford (2019) presents the Diphone TTS architecture, offering insights into traditional methods of text-to-speech synthesis and their relevance in modern speech synthesis research.

Speech Synthesis Overview Jurafsky and Martin (2020) [6] provide a comprehensive overview of speech and language processing, laying the foundation for understanding the principles behind speech synthesis.

Speech Synthesis Technologies[7] Yin (2018) offers an overview of speech synthesis technology, outlining its evolution and key methodologies

Style Transfer and Super-Resolution[8] Johnson et al. (2016) propose perceptual losses for real-time style transfer and super-resolution, contributing to the enhancement of speech synthesis quality

Unsupervised Style Modeling[9] Wang et al. (2018) introduce Style Tokens, enabling unsupervised style modeling, control, and transfer in end-to-end speech synthesis systems.

Prosody Transfer[10] Skerry-Ryan et al. (2018) present advancements towards end-to-end prosody transfer for expressive speech synthesis using Tacotron, improving the naturalness and expressiveness of synthesized speech.

Normalization in TTS[11] Pennell and Liu (2012) evaluate the effect of normalizing informal text on TTS output, highlighting the importance of preprocessing steps in text-to-speech systems.

Deep Learning Architectures[12][13] Srivastava et al. (2015) introduce Highway Networks, while He et al. (2015) propose Deep Residual Learning for image recognition, inspiring the development of more complex neural architectures in speech synthesis.

End-to-End Text-to-Speech[14] Yasuda et al. (2020) present an end-to-end text-to-speech system using latent duration based on VQ-VAE, contributing to the simplification and efficiency of speech synthesis pipelines.

Latent Representations for Style Control[15] Zhang et al. (2019) propose learning latent representations for style control and transfer in end-to-end speech synthesis, enabling more flexible and expressive synthesis models.

TACOTRON: TOWARDS END-TO-END SPEECH SYNTHESIS[16] Wang et al. (2017) introduce Tacotron, an end-to-end speech synthesis architecture, advancing the field towards more natural and human-like speech generation.

Flow-Based Generative Networks[17] Prenger et al. (2018) present WaveGlow, a flow-based generative network for speech synthesis, offering high-quality and efficient synthesis capabilities.

Generative Flow with Invertible Convolutions[18] Kingma and Dhariwal (2018) introduce Glow, a generative flow model with invertible 1x1 convolutions, contributing to the advancement of generative modeling techniques in speech synthesis.

Convolutional Sequence Learning[19][20] Ping et al. (2018) propose Deep Voice 3, scaling text-to-speech synthesis with convolutional sequence learning, improving the efficiency and quality of synthesized speech.

WaveNet Architecture[21] van den Oord et al. (2016) introduce WaveNet, a generative model for raw audio, revolutionizing speech synthesis with its ability to generate high-fidelity speech waveforms.

Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions[22] Shen et al. (2017) propose conditioning WaveNet on mel spectrogram predictions, enabling natural and high-quality text-to-speech synthesis with improved expressiveness and realism.

Attention Mechanisms[23] Vaswani et al. (2017) introduce the Transformer architecture, highlighting the effectiveness of attention mechanisms in sequence-to-sequence tasks, including text-to-speech synthesis.

Transformer Network for Speech Synthesis[24] Li et al. (2019) propose a neural speech synthesis model based on the Transformer network, demonstrating the effectiveness of self-attention mechanisms in speech generation.

Emotional Speech Synthesis[25][26] Choi et al. (2019) and Kwon et al. (2019) explore emotional speech synthesis using WaveNet and Tacotron frameworks, enhancing the expressiveness and emotional fidelity of synthesized speech.

Emotional Speech Synthesis Based on Style Embedded Tacotron2 Framework[26] Kwon et al. (2019) propose emotional speech synthesis based on the Style Embedded Tacotron2 framework, enhancing the expressiveness and emotional range of synthesized speech.

EmotionLines: An Emotion Corpus of Multi-Party Conversations[27] Chen et al. (2018) introduce the EmotionLines corpus, which comprises multi-party conversations annotated with emotional labels, facilitating research in emotion recognition and emotional speech synthesis.

LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech[28] Zen et al. (2019) present LibriTTS, a corpus derived from LibriSpeech specifically for text-to-speech synthesis research, providing a diverse and extensive dataset for training and evaluation.

Sequence-to-Sequence with Attention[29] Socher (2018) discusses sequence-to-sequence models with attention mechanisms, which have been widely applied in various natural language processing tasks, including speech synthesis.

Recurrent Neural Networks (RNNs)[30] Amidi and Amidi (2018) provide a cheatsheet on recurrent neural networks (RNNs), which are fundamental architectures used in many speech synthesis systems for sequence modeling.

Text-to-Speech Deep Learning Architectures[31] Machine Learns (2018) offers insights into various deep learning architectures used in text-to-speech synthesis, providing a comprehensive overview of the field's technological landscape.

Audio Feature Extraction[32] Machine learning frameworks like OpenSmile (2021) provide tools for audio feature extraction, enabling feature engineering in speech synthesis systems.

PixelCNN Decoders: van den Oord et al. (2016) propose conditional image generation with PixelCNN decoders, contributing to the development of generative models applicable to speech synthesis tasks.

Increasing Receptive Field with Dilated Convolution[33] Kin (2018) discusses WaveNet's architecture, highlighting its use of dilated convolutions to increase the receptive field, resulting in improved speech synthesis quality.

Emotional Speech Databases:[34][35] Speech databases such as the SEMAINE Database (Kwon et al., 2013), MELD Dataset (Poria et al., 2018), and others provide annotated multimodal records and emotional speech data, enabling research in emotional speech synthesis and recognition.

The LJ Speech Dataset[36] Ito and Johnson (2017) introduce the LJ Speech Dataset, a large-scale dataset for speech synthesis research, facilitating the training and evaluation of speech synthesis models.

Emotion Understanding Models: De Pinto et al. (2020)[37]

propose emotion understanding models using deep neural networks and Mel-Frequency Cepstral Coefficients (MFCCs), contributing to the development of emotionally expressive speech synthesis systems.

CHAPTER 3
THEORETICAL BACKGROUND

3.1. General overview of TTS

The modern task of speech synthesis, also called Text-To-Speech or TTS, is to produce speech (acoustic waveforms) from text input.

Synthesizers are used to conduct dialogues with people and are very important in non-conversational applications that speak to people, such as in devices that read out loud for the blind, or in video games or children's toys. Finally, speech synthesis can be used to speak for sufferers of neurological disorders, such as astrophysicist Steven Hawking who, having lost the use of his voice due to ALS1, speaks by typing to a speech synthesizer and having the synthesizer speak out the words. State-of-the-art systems in speech synthesis can achieve remarkably natural speech for a very wide variety of input situations, although even the best systems still tend to sound wooden and are limited in the voices they use.

In an overview of speech synthesis technology, a generic description of TTS is exposed and it also explains the current structure of TTS with all its variants. When we approach TTS, we can divide it into two essential blocks: NLP2 and DSP3 which is shown in Figure 3.1

The first one is the Natural language processing block; his role is to take the input text and transcribe it into a phonetic representation. The second block is Digital signal processing and its role is to generate the speech sound, this element is generated from the phonetic representation of the NLP block. The NLP block can be considered as the number of steps followed to get the phonetic representation and is composed of these elements:

1. The Pre-Processing where "Regularization" operations are done (convert abbreviations, acronyms turn into full text, etc.)
2. The Morphological Analysis and Contextual Analysis work together, in this step the tokenization is applied. if we have some problem related to some particular language the contextual analysis is performed.
3. The Syntactic analysis and it's main goal is to use the information of the precedent step to predict the prosody.

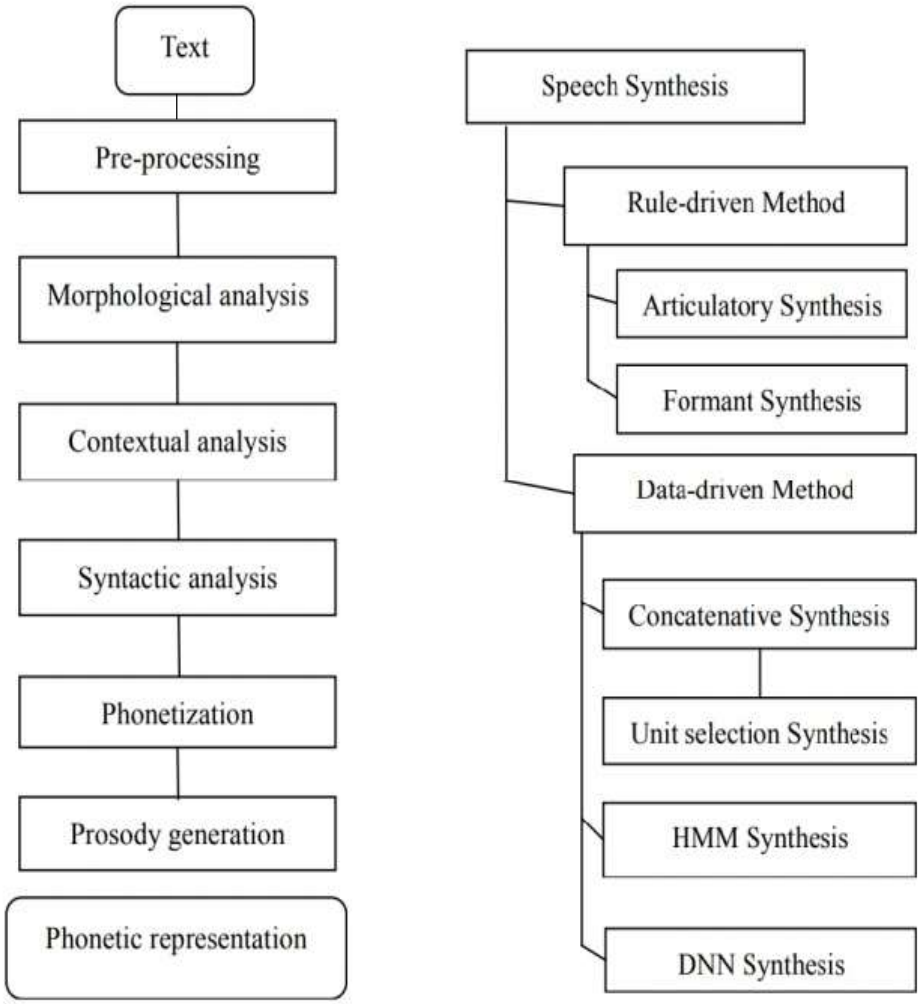


Figure 3.1 NLP and DSP block

4. The phonetization block generates the sequence of phonetic symbols for each word. A pronunciation lexicon should be constructed first, and the phonetic symbols of each word should be produced according to the lexicon.

5. The Prosody generation will generate the prosodic features which include pitch curve, duration and pause information, stress, and rhythm features. This will influence the smoothness of the TTS. Then we have the DSP block, which is composed of two main blocks: the rule-driven methods and data-driven methods. Rule-driven methods simulate the speech voice according to the rules deduced from the acoustic process while the principle of data-driven methods is to generate the speech by recorded speech data or by the statistical parameters obtained from speech data. The two approaches of Rule-driven synthesis methods are articulatory synthesis and formant synthesis while the main approaches of Data-driven

synthesis methods are concatenative synthesis, unit selection synthesis, HMM4 synthesis, and DNN5 synthesis.

3.1.1. Data-driven

The Concatenative system generates the voice connecting small recorded voices, then the utterance is modified by the prosodic model, and the sound quality is very good. there can be some problems with memory because the system is more natural if we have longer units since less concatenation is needed, but if we have shorter units the sound quality will become bad, the disadvantages are the discontinuities of unit
boundaries and can lead to some unstable results. In this case, the Unit Selection can fix the problem.

Unit Selection is similar to Concatenative Synthesis but solves unnaturalness by storing multiple instances of each unit with a different prosodic feature, However, this method also has the same problems of selecting error units, and a huge speech corpus is needed.

Since we can have a problem of database size in Unit Selection (need a big corpus) we can use a parametric synthesis to infer the acoustic parameter from speech data. in this case, less memory is needed to store the parameter (we're not storing anymore all the corpus). HMM is a parametric model and consists of two phases:

1. The training phase is similar to the one used in speech recognition, we extract the feature and model this vector of features depending on the context (grammatic, prosody).
2. Synthesis filter transforms the feature vector in acoustic signal (Synthesis phase). this allows more flexibility concerning Unit Selection but can lead to some limitations on synthesized speech.

In an HMM-based speech synthesis system, some limitations would degrade the accuracy of acoustic models and the quality of synthesized speech. DNN was introduced to solve this problem, thanks to the hidden layer we are trying to learn the more useful feature. DNN could be used to model the features of text and speech sounds and output the vectors of phonetic signals, but on the other side, the power computation is bigger than HMM.

Rule-driven synthesis methods are used less today because the synthesized speech sounds are not good enough. However, their advantages are the low processing costs and high flexibility. Data-driven synthesis methods nowadays are the dominant methods because they can provide a higher quality of synthesized speech. Unit selection synthesis could generate the best speech but it also cost the largest memory and data. Compared with Concatenative synthesis and Unit Selection synthesis, HMM, DNN, and other statistical synthesis techniques could produce

similar natural speech but only need fewer speech data and memory, they achieve the best balance at present.

3.2. Convolutional Neural Network

A CNN is used for image processing. The input of a convolutional neural network is an image in 3 dimensions (width, height, and depth). It is composed of hidden layers and an output layer but the layers have neurons arranged in 3 dimensions.

The typical architecture of a convolutional neural network is shown in Figure

3.2. CNN are typically made of blocks that include:

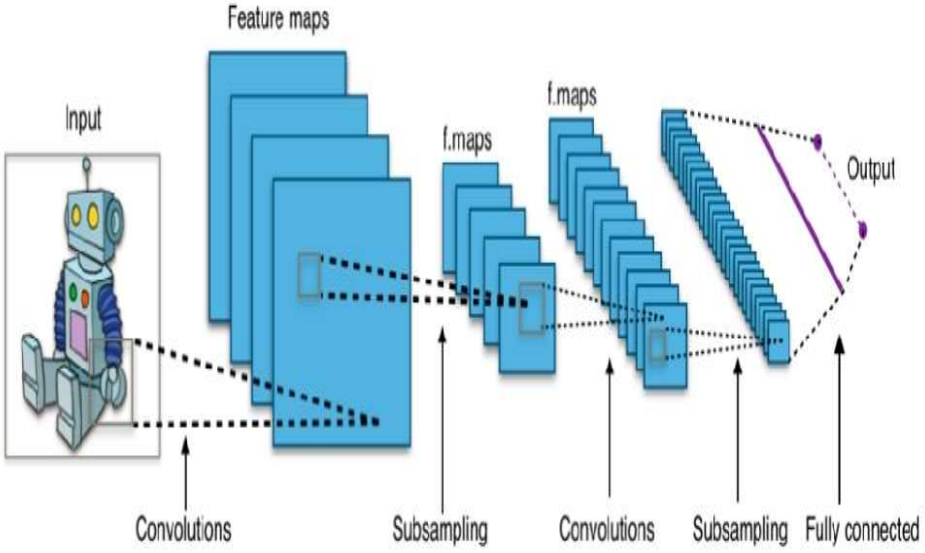


Figure 3.2 CNN Block diagram

• Convolutional Layers

• Nonlinearities (activation function)

• Maxpooling

An image passing through a CNN is transformed into a sequence of volumes, as the deep increases, the height, and width of the volume decrease. The convolutional layers apply the convolution operation, in which a kernel slides over the input feature map, and at each kernel position, the elementwise product is computed between the kernel and the overlapped input subset, then the result is summed up (Figure 3.3). Using different kernels lead to different effect. Therefore, convolutional layers are described by a set of filters, that represent the weights of these linear combinations.

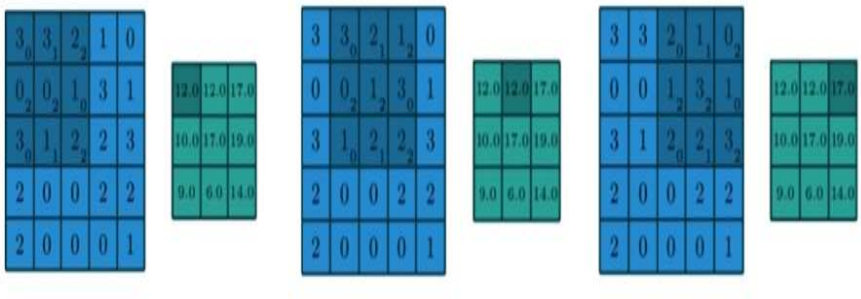


Figure 3.3 Computational Details

The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. A dot product is an element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. Because it results in a single value, the operation is often referred to as the "scalar product".

3.3. Recurrent Neural Network

Classical neural networks can deal only with static datasets. When we want to deal with dynamic data, we can use either memoryless models such as autoregressive or feedforward neural networks or we can use memory-based models such as linear dynamic systems, hidden Markov models, or recurrent neural networks (RNN10). In RNN, memory is introduced through a context vector and output depends on both the current input and previous context. It can help when we need to remember something from the past. RNN is proved to be Turing-complete. The output of an RNN at step t is passed to the network at step t + 1. In such a way, an RNN can be seen as multiple copies of the same network, each one passing information to the network at the next time step. This characteristic is shown in Figure 3.4. Therefore, the output at step t is predicted considering the information at step t −1.

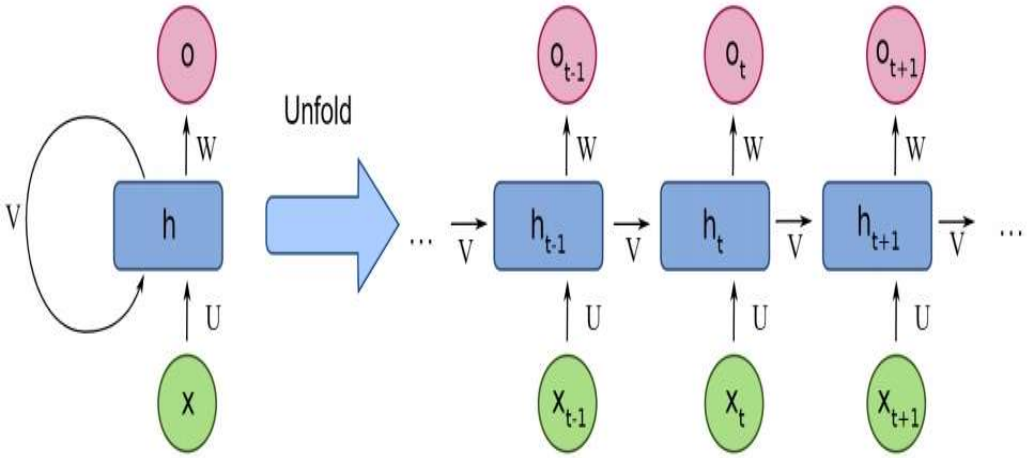


Figure 3.4 Recurrent Neural networks

Taken together, all of these considerations lead to a two-pass algorithm for training the weights in RNNs. In the first pass, we perform forward inference, computing ht, yt, and accumulating the loss at each step in time, saving the value of the hidden layer at each step for use at the next time step. In the second phase, we process the sequence in reverse, computing the required error term gradients as we go, computing and saving the error term for use in the hidden layer for each step backward in time. This general approach is commonly referred to as Backpropagation Through Time. Due to its properties, different types of RNN (Figure 3.5 on the next page) can solve a wide spectrum of problems such as simple image classification, music generation, sentiment classification, machine translation, and many others.

The main advantages of RNN are the possibility to process data of any length, model size is not increased with increasing size of the input, the computation can take historical information into account, and weights are shared across time. However, during long backpropagation, with the classical one-gate model, there appears one problem, which could not be solved for a long time. This problem is a vanishing/exploding gradient. The reason for this is that it is difficult to capture long-term dependencies due to multiplicative gradients which will change a lot in some layers.

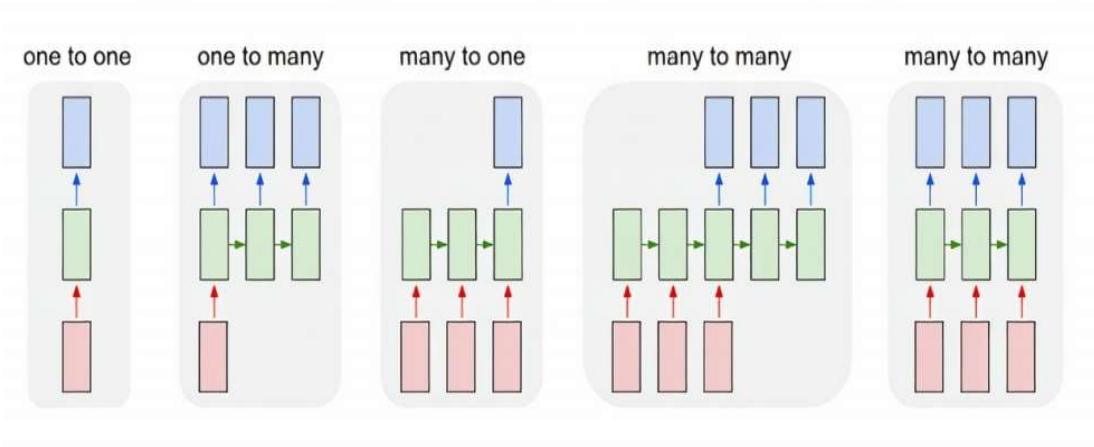


Figure 3.5 RNN Types

3.4. Seq2Seq and Attention Mechanism

Sequence-to-sequence neural network architectures involve two RNNs. We start with the source sentence and after proceeding with word embedding, which is the process of converting textual input into representational vector space, we feed this to the encoder RNN. Encoder RNN produces an encoding of the source sentence and stores this encoding in the final hidden state.

The next step is to pass this encoding to decoder RNN in the initial state. Firstly, we feed the start token to the decoder, and then we can produce the first state of the decoder, as we are using the last hidden state of the encoder as the initial state of the decoder.

Training and testing processes for the encoder part are the same and involve creating a contextualized representation of the input sequence. However, the Decoder part is different for training and testing. During the training part, we feed to the decoder also target sentence as the ground truth. Everything that produces a decoder is used only for computing loss J, where J is weighed sum over each loss of each decoder output. The loss of each output is computed as the negative log probability of the next word. The training process is depicted in Figure 3.6.

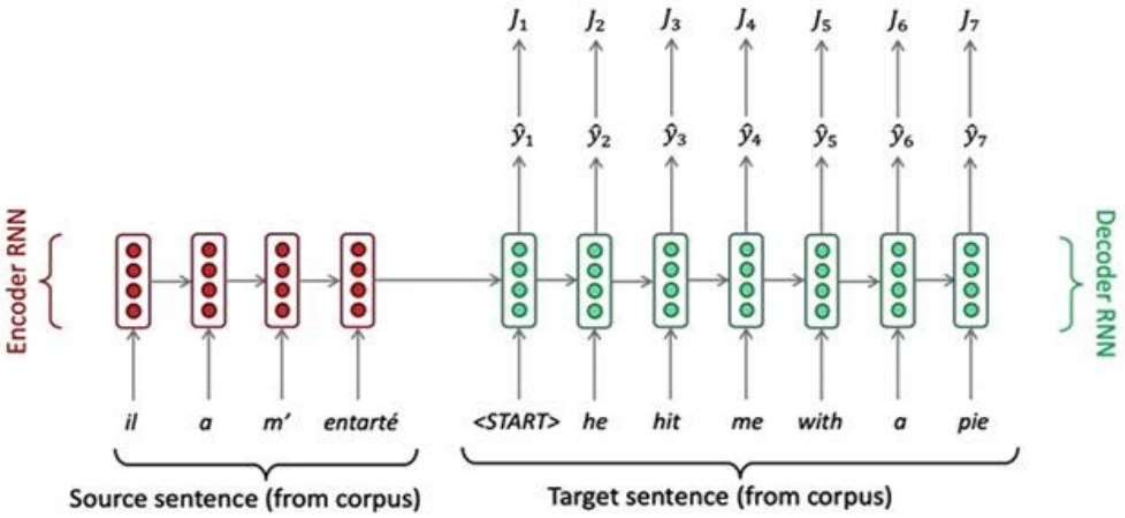


Figure 3.6 seq2seq training

The testing part does not have a target sentence and in every state of the decoder, using a probability distribution, tries to predict the word which can come next using 'argmax' of this distribution. Then we feed the chosen word to the next step of the decoder and repeat this procedure until we get the end token. The choice of the next word is conditioned by the previous state of a decoder and we use a word, which was produced by the previous step, as an input of the current step of the decoder. This is not generally true as in the attention mechanism which will be described later. The test process is depicted below (Figure 3.7 on the following page). One of the problems of the seq2seq model described above is that it chooses words greedily, which means that in long-term games it can produce bad results. To solve this problem, when choosing a word to generate, we choose K's most

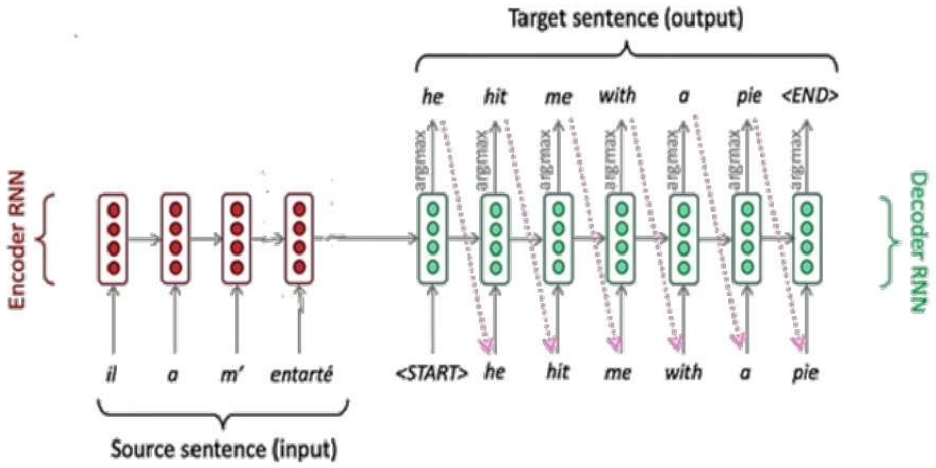


Figure 3.7 seq2seq testing

promising words at each stage of the decoder. It will generate a tree in which we need to choose a branch with a smaller normalized error value. This approach is called Beam search. Both the encoder and decoder could be also vanilla or bidirectional RNN, GRU, or LSTM depending on the problem we are solving.

There is a drawback when we use this architecture and is related to interpretability, it is more difficult to debug and control. Worldwide there are many types of research in this kind of analysis of RNN, and this problem is still not resolved.

One improvement was created and it was so crucial that it is currently considered as new vanilla and it is called Attention. In standard seq2seq all the encoding of the sentence is on the last block of the Encoder RNN and this can lead to a bottleneck (Figure 3.8 on the next page) as this state is overwhelmed with information.

Attention provides a solution to the bottleneck problem. The idea is to use for each step of the encoder a direct connection to the encoder to focus on a particular part of the source sequence (Figure 3.9 on the next page). At the beginning we take the dot product between the first decoder hidden state and the first encoder hidden state, the result is called attention score.

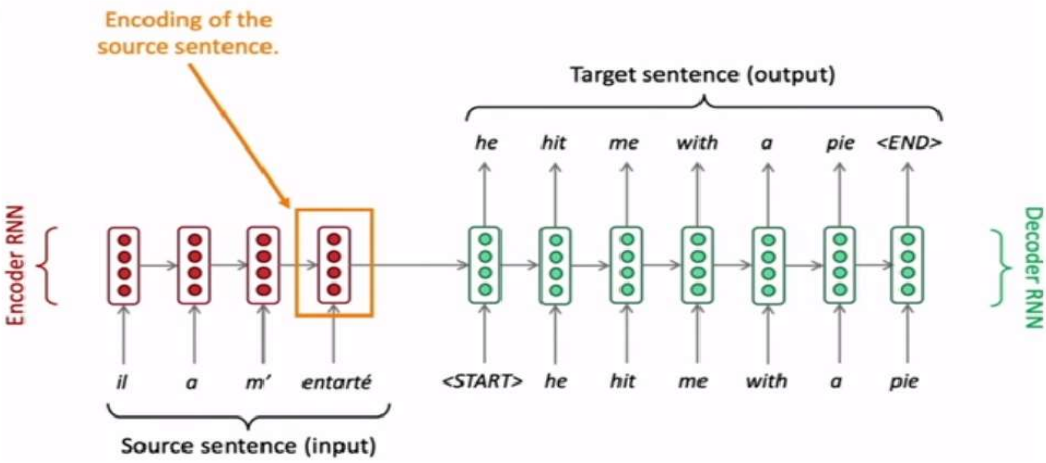


Figure 3.8 seq2seq Bottle neck

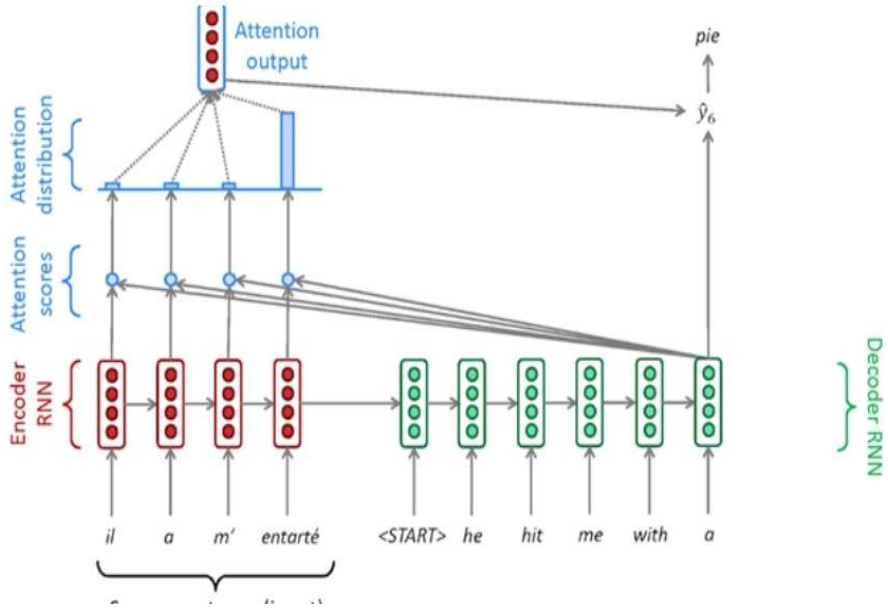


Figure 3.9 seq2seq Attention

The value obtained with all the other encoder hidden states, and we will have one attention score for each hidden state of the source sentence.

Using these attention scores, we apply the softmax function and we get a probability distribution for each attention score, and this will be the attention distribution. This value will be used to produce the attention output It is the weighted sum of the encoder hidden state.

The Attention output which is a single vector is going to contain mostly information from the hidden states that received high attention. Then we use the Attention output to influence the output of the next word, we concatenate the attention output with the decoder's hidden state.

Then for each element of the decoder, we compute the same step as before for getting the output. Notice that in some cases attention output from the previous step is fed into the decoder. General improvements:

- Attention significantly improves the performance.
- Attention solves the bottleneck problem.
- Attention helps with the vanishing gradient problem.
- Attention provides some interpretability by inspecting the attention distribution we can see what the decoder was focusing on.
- Soft alignment is for free.

3.5. Tacotron

Tacotron is an end-to-end character-based architecture for Text-To-Speech synthesis based on seq2seq with an attention mechanism. Word-level neural machine translation models, despite their popularity, have some major weaknesses such as difficulty with modeling out-of-vocabulary words, and with huge dictionaries, its complexity becomes a problem. Character-based models can avoid these problems and they perform better for multilingual translation.
The advantage of end-to-end models is that they require much less labor feature engineering, which may require some heuristics and difficult design choices. Also, it more easily allows for rich conditioning on various attributes, such as speaker or language, or high-level features like sentiment.

3.5.1 Model description

The model of the architecture is shown in Figure 3.10. At a high level, the model takes characters as input and produces spectrogram frames, which are then converted to waveforms.

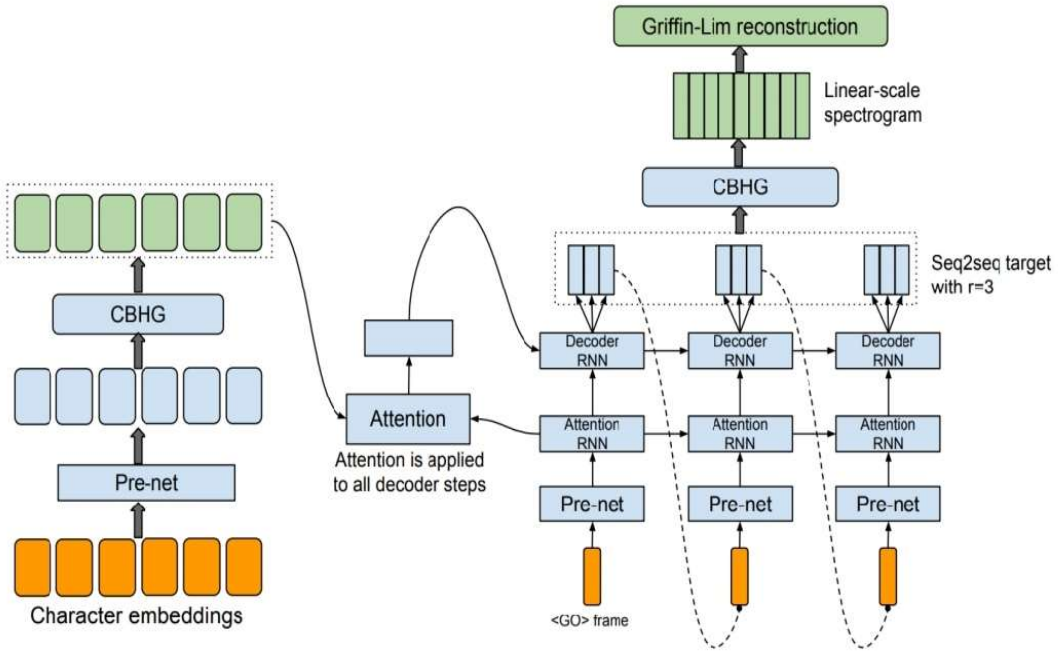


Figure 3.10 Tacatroon Architecture

The left part of the figure represents the encoder which is used to extract contextual representation of the input sequence. The input sequence of characters is firstly embedded into a continuous vector. Then, for each embedding, we apply non-linear transformation using two fully connected layers with dropout. This transformation is called “pre-net” and it is used for improving generalization. Then we pass the output to the CBHG module.

The right part of the figure represents the decoder. More precisely it uses a content-based attention decoder, which was described in the previous chapter. The score is computed with the tanh function. The task is to concatenate the context vector and the attention RNN cell output to form the input to the decoder RNNs.

The target of the decoder is an important component. In the Tacotron case, the compressed target might be sufficient as far as it provides sufficient intelligibility and prosody information for an inversion process. In the work, they use an 80-band Mel-scale spectrogram as the target. One important thing that was done is that the decoder outputs several frames at once. It is done as it speeds up convergence and training and one character is usually represented by more than one frame, so quality does not get worse. The first decoder step is conditioned on an all-zero frame, which represents a frame. In inference, at decoder step t, the last frame of the r predictions is fed as input to the decoder at step t + 1.

The next step is to pass all decoder outputs to CBHG, which in this case is responsible for the post-processing net. Post-processing net learns to predict spectral magnitude sampled on the

linear frequency scale. Also, this net can see the whole decoder output at once and make some adjustments. The last step is a synthesis of the waveform, which was done using the Griffin-Lim algorithm. This algorithm was chosen for simplicity as it is differentiable and can be tuned in any way.

3.5.2 CBHG module

CBHG module takes into input the result of the pre-net transformation. CBHG consists of a bank of 1-D convolutional filters, the input sequence is convolved with K sets of 1-D convolutional filters, where the k th set contains Ck filters of width k. As we can see from Figure 3.11, we have a different filter of a different dimension.

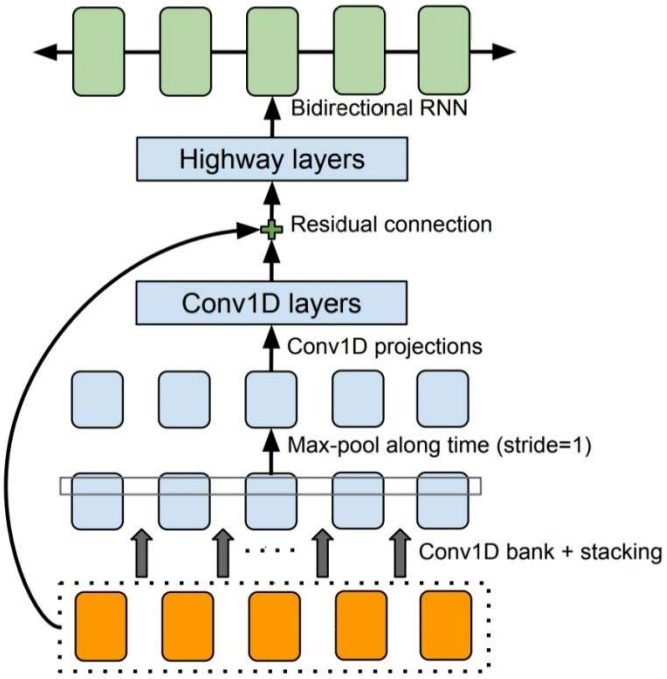


Figure 3.11 CBHG Architecture

The convolutional output is stacked together, and a further max is pooled along time to increase the local invariances.

After these steps, we further pass the processed sequence to a few fixed-width 1-D convolutions whose output is added to the original input sequence. This can be done by using a Residual Connection. Notice that Batch normalization is used for all convolutional layers. The convolution outputs are fed into a multi-layer highway network to extract the high-level feature.

Finally, we stack a bidirectional GRU RNN on top to extract sequential features from both forward and backward contexts. The CBHG module improves the generalization.

3.6 Tacotron 2

During these years, different techniques have dominated the field of TTS. Initially, Concatenative synthesis with unit selection was the state of the art, later some methods based on statistical parameters have solved some known problems of concatenative synthesis, but the audio produced was still muffled and not natural.

Tacotron 2 combines the two best approaches: a sequence-to-sequence Tacotron-style model that generates Mel-spectrograms, followed by a modified WaveNet vocoder.

Vocoders are the neural systems used to generate human voice from mathematical models of the vocal tract or Mel-spectrograms. During the analysis phase, vocoder parameters are extracted from the speech waveform which represents the excitation speech signal and filter transfer function.

WaveNet vocoder (see Section 3.3) was later introduced, a generative model of time-domain waveforms that produces the audio quality that begins to rival that of real human speech, however, it requires significant domain expertise to produce, involving elaborate text-analysis systems as well as a robust lexicon.

The architecture of Tacotron 2 is depicted in Figure 3.12 and is composed of two main components:

• Recurrent sequence-to-sequence feature prediction network with attention.

• Modified version of WaveNet.

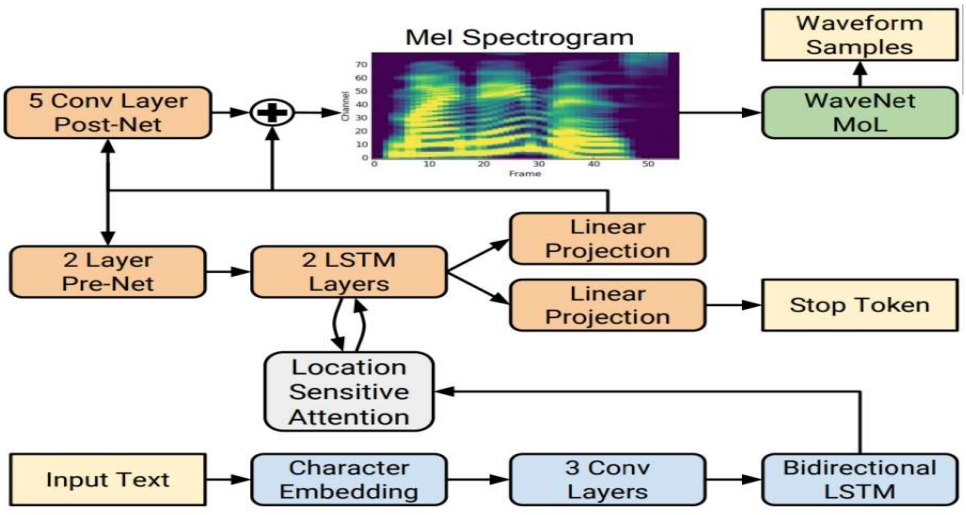


Figure 3.12 Tacotron 2 Model Architecture

3.6.1 Intermediate Feature Representation

Mel frequency spectrograms are chosen as a low-level acoustic representation. With this representation is easier to operate on time-domain waveforms, furthermore is smoother than waveform samples, and easier to train.

The Mel frequency spectrograms can be obtained by applying a nonlinear transform to the frequency of the STFT1. There are properties in high frequency and low frequency that can be summarized as features that have been always used in previous TTS systems.

Linear spectrograms discard this kind of information but algorithms like GriffinLim can estimate this discarded information. Mel-spectrogram discards even more information and can lead to a difficult problem to resolve. However, the Melspectrogram is simple since it contains a lower-level acoustic representation of the audio signals. Since the input of WaveNet is simpler is possible to generate high-quality audio from Mel-spectrograms with the modified WaveNet.

3.6.2 Spectrogram Prediction Network

As in Tacotron, this model uses an encoder and decoder with attention. However, there are several significant differences. Instead of using pre-net and CBHG layers at the embedding stage, we pass input characters through 512- 512-dimensional embedding, and then it passes to 3 convolutional layers, each having 512 filters 5-by-1.

This dimensionality of the filter is useful as it can look at the current character, 2 previous, and two followings. Each convolution is followed by batch normalization and ReLu activation function. Afterward, the output is passed to the single bidirectional LSTM.

The encoder output is passed to a location-sensitive attention network as a context vector, which summarizes the full encoded sentence at each decoder output step. The difference between this attention mechanism with respect to the classical one is that it also considers cumulative attention weights from the previous step of the decoder.

The decoder of Tacotron 2 is an autoregressive recurrent neural network, which learns to predict Mel-spectrogram from the given encoder context.

The prediction from the previous step is firstly passed to pre-net which has the same structure as in the original Tacotron as it helps with learning attention. Then the output of pre-net and attention context is concatenated and passed through 2 LSTM. Then we project the output via linear transform to predict the target spectrogram frame.

Lastly, we use 5 layers of CNN post-net with dropout which predicts a residual and improves overall reconstruction. In parallel to frame prediction, we pass decoder LSTM output and

context of the attention, after projecting to a scalar, through the sigmoid function which helps to determine whether the input sentence is finished via “stop token”.

3.6.3 Wavenet Vocoder

The vocoder is a modified version of the standard WaveNet architecture. As in the original one, there are 30 dilated convolution layers grouped into 3 dilation cycles To work with the 12.5 ms frame hop of the spectrogram frames, only 2 up-sampling layers are used in the conditional stack (original Wavenet used 3 layers).

PixelCNN and Parallel Wavenet are used to generate 16-bit samples at 24 kHz, there is a 10-component mixture of logistic distributions instead of the softmax layer. The logistic mixture is computed by passing through a ReLu activation followed by a linear project to predict parameters for each mixture component. Loss is computed as the negative log-likelihood of the ground truth.

3.7 WaveNet

Wavenet is a generative model for the audio waveform. The main idea is to predict for each time point a bit of depth. However, this is a huge classification problem as for each time point, we need to predict one of the 65536 values in stereo sound since the bit depth is 16. The first measure, that was taken to solve this problem is compounding transformation, which transforms these 65536 values in the range of 256 values, which in the end we will invert to get the original value back.

PixelCNN during the development of Wavenet, had to deal with the training procedure in the following way: during the training in PixelCNN we can train all the pixels in parallel with the assumption, that we can look only at previous pixels in the image, they solved this problem with masked convolutions. In Wavenet they solved this issue via causal convolutions, by shifting output cells on the right, which helps to not look at the future values.

Another innovation in the Wavenet is dilated convolutions. The problem of causal convolutions is that they need to have a large number of layers to increase the receptive field. The bigger receptive field allows us to look for a dependency on data in the long term. A dilated convolution is a convolution where the filter is applied over an area larger than its length by skipping input values with a certain step. If in traditional linear filters the output is the weighted sum of the inputs, with causal convolutions extra non-linear operations are applied after each convolution.

Dilated causal convolutions are implemented in WaveNet by doubling the dilation factor until we will get to the given limit after we start from the first value of the set. In the end, we stack dilated convolution operations one over another as illustrated in Figure 3.13.

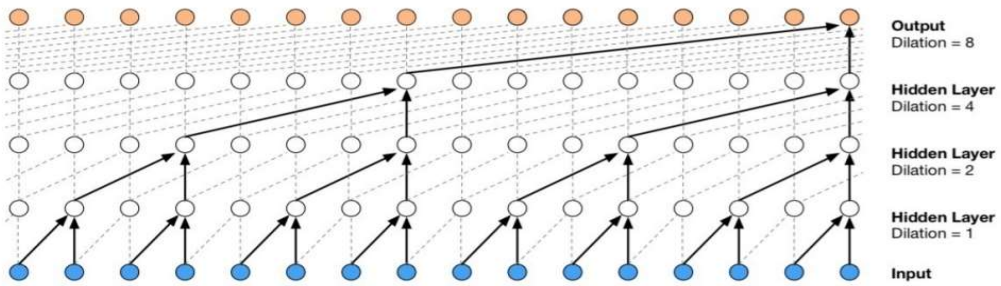


Figure 3.13 Visualization of a stack of dilated casual convolutional layers

3.8 WaveGlow

WaveGlow is a flow-based network capable of generating high-quality speech from Mel-spectrograms. It combines the insights of Glow and WaveNet, the great advantage of this network is in its speed speech generation without losing great quality. Waveglow is formed by a single network trained using only a single cost function, auto-regression was not needed.

In detail, we have a generative model that generates audio by sampling from a distribution. To use a neural network as a generative model, the samples are taken from a simple distribution, a zero-mean spherical Gaussian z with the same number of dimensions as the desired output, and put those samples through a series of layers x that transforms the simple distribution into one which has the desired distribution.

$$\mathbf{z} \sim \mathcal{N}(\mathbf{z}, 0, \mathbf{I})$$
$$x = \boldsymbol{f_0} \cdot \boldsymbol{f_1} \cdot \ldots \boldsymbol{f_k}(z)$$

The training was carried out to minimize the negative log-likelihood. In most cases, it would be very difficult to deal with this type of network with this type of loss, which is why Flow-based networks have been used. We can solve this problem by making reverse mapping possible, the likelihood can be calculated directly using a change of variables:

$$logp_\Theta(x) = logp_\Theta(z) + \sum_{i=1}^{k} log \left| det(\boldsymbol{J}(\boldsymbol{f_i^{-1}}(x))) \right|$$
$$z = \boldsymbol{f_k^{-1}} \cdot \boldsymbol{f_{k-1}^{-1}} \cdot \ldots \boldsymbol{f_0^{-1}}(x)$$

The first term is the log-likelihood of the spherical Gaussian. This term penalizes the L2-norm of the transformed sample. The second term arises from the change of variables, and the J is the Jacobian. The log-determinant of the Jacobian rewards any layer for increasing the volume of the space during the forward pass.

This term also keeps a layer from just multiplying the x terms by zero to optimize the thenL2-norm. The sequence of transformations is also referred to as a normalizing flow. The model is most similar to Glow and is depicted. For the forward pass through the network, groups of 8 audio samples are taken as vectors, and then the 'squeeze' operation is applied, as in Glow. We then process these vectors through several 'steps of flow'. A step of flow here consists of an invertible 1 × 1 convolution followed by an affine coupling layer

3.9 Transformers

For RNN and LSTM networks data should be passed sequentially. We need to have the input of the previous state to perform any operations in the current state. They are not able to use all the power of modern GPU which are designed to make parallel computations. That is why RNN architecture is slow and LSTM even slower due to its high complexity.

To deal with this problem, Transformer Architecture was introduced. This architecture also uses the encoder-decoder model as in RNN but its main feature is that it takes the whole sentence as input and performs operations in parallel relying only on self-attention mechanisms. Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence to compute a representation of the sequence. It means that we can extract some context from a given embedding vector and see relations within this vector.

As most of the Text-To-Speech architectures that use Neural Networks, the Transformers architecture has an encoder-decoder structure. The encoder maps an input sequence of symbol representation to a sequence of continuous representation. Given that sequence, the decoder generates an output sequence of symbols one element at a time. It is autoregressive, which means that at each step the model consumes the previously generated symbols as additional input when generating the next one. The transformers use stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. The overall architecture of the TTS model which is based on transformers is depicted in Figure 3.15

3.9.1. Encoder

The encoder is composed of a stack of N = 6 identical layers and each layer has two sub-layers. The first is the multi-head attention mechanism and the second is a position-wise fully

connected feed-forward network. Residual connections are used around each of the two sub-layers followed by layer normalization.

3.9.2. Decoder

The decoder is also composed of a stack of N = 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer which performs multi-head attention over the output of the encoder stack. The masking ensures that the prediction can depend on the known outputs. In addition to attention sub-layers, each of the layers in the encoder and decoder contains a fully connected feed-forward network, which consists of two linear transformations with a ReLU activation. A learned embedding is used to convert the input tokens and output tokens to vectors.

3.9.3 Text to phoneme Converter

The neural networks are responsible for finding regularities in the training process; however, it is difficult to learn all regularities. A rule-based system is implemented and allows to find all the regularities and convert text-to-phoneme

3.9.4 Encoder Pre-net

In Tacotron2 a 3-layer CNN is applied to the input text embeddings. In the Transformers TTS architecture, the phoneme sequence is the input of the same network called “encoder pre-net". Each phoneme has a trainable embedding of 512 dims, and the output of each convolution layer has 512 channels, followed by batch normalization and ReLU activation, and a dropout layer as well. A final linear projection is applied after the ReLU.

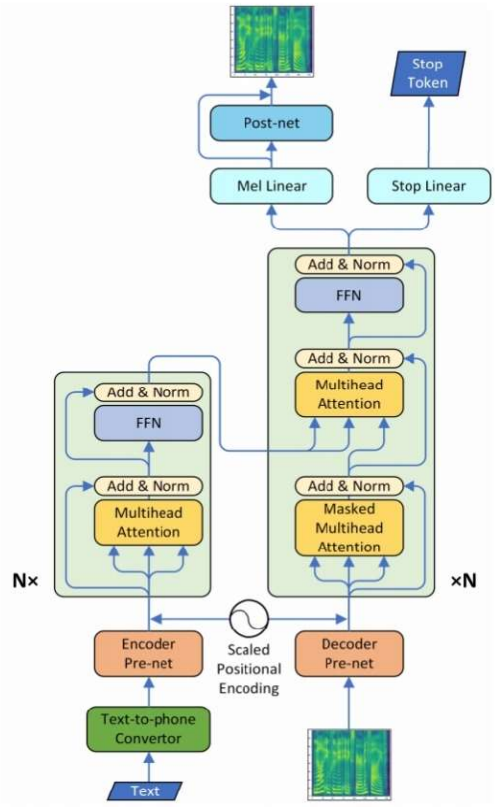


Figure 3.15 Encoder Pre-net

3.9.5. Decoder Pre-net

The Mel-spectrogram is first consumed by a neural network composed of two fully connected layers with ReLU activation. Phonemes have trainable embeddings thus their subspace is adaptive. The decoder "pre-net" is the one responsible for a projection of the Mel-spectrogram into the same subspace as phoneme embeddings to measure the similarity of <phoneme, mel − frame>. An additional linear projection is also added to the encoder pre-net to keep consistency and also to obtain the same dimension as the triangle positional embeddings.

3.9.6. Mel Linear, Stop Linear and Post-net

It is the same technique as in Tacotron 2 in which we use two linear projections to predict the Mel-spectrogram and the stop token. Then produce residuals using 5 layers CNN to refine the reconstruction of the spectrogram. Also, positive weights were imposed on the tail of the positive token as it was imbalanced due to hundreds of negative frames before the stop token.

3.10 Discussion

Concerning spectrogram synthesis models, it is worth mentioning that Tacotron 2 and Transformers are not the only two models that we analyzed during our work. For example, almost at the same time as Tacotron 2 was published, DeepVoice was also introduced.

However, in our opinion, even the demo samples on their official website have a robotic accent, which is incomparably worse than Tacotron2 samples.

Transformers are one of the most trendy and powerful architectures nowadays and they also have shown an excellent quality in terms of neutral speech synthesis.

Also, its big advantage is computation speed as it is six times faster for training compared to Tacotron 2. However, there is one point that has made our decision about the baseline speech synthesis model. The problem with transformers is that by the time of writing this word, there was no strong evidence that they could handle expressive speech. In Seq2Seq models, the idea of bringing emotions into speech is pretty straightforward which could not be said about transformers with their high parallelization.

During inference time, after having trained the spectrogram generation model, we did not choose a particular vocoder as a final one. As Wavenet and Waveglow have their advantages and disadvantages and since we have both pretrained versions we could use them without any problems depending on the particular goal we wanted to achieve. To get the best quality we used WaveNet, and to be faster in testing and more productive in terms of speed we used WaveGlow.

CHAPTER 4
APPROACH DESCRIPTION

4.1. Approach Flow

1. **Data Collection:** The code begins by collecting data related to emotional speech synthesis. This data includes audio recordings paired with textual transcripts, specifically curated for emotional speech synthesis research.
2. **Package Import:** Importing the required Python packages for building and running the classification model. These packages likely include deep learning libraries such as PyTorch, as well as standard data manipulation and visualization libraries like NumPy, Matplotlib, and TensorFlow.
3. **Project Creation and Dataset Loading:** Creating a project environment and loading the dataset. The dataset contains information related to emotional speech, such as textual transcripts and corresponding emotional labels.
4. **Data Cleaning:**
    - Checking for duplicate Audio Files.
    - Overviewing the dataset to understand its structure and contents.
5. **Exploratory Data Analysis (EDA):**
    - Counting the occurrences of different target values (emotions) to understand the distribution of emotional labels.
6. **Training the Model:** Training the classification model using the preprocessed dataset. This involves feeding the input features and corresponding emotional labels into the model and optimizing its parameters to minimize the error.
7. **Result Analysis and Inference Drawing:** Analyzing the model's performance metrics, such as the MOS Score, to conclude its effectiveness in emotional speech.

CHAPTER 5
DATA EXPLORATION

5.1. EmoV-DB dataset

The Emotional Voices Database (EmoV-DB) consists of recordings of emotional speech from multiple speakers in English and French. The dataset aims to facilitate research in emotional speech synthesis and emotion recognition. The emotional speech recordings cover a range of emotion classes, including amusement, anger, sleepiness, disgust, and neutral expressions. Each emotion class provides a variety of emotional contexts for modeling and synthesis.

5.1.1. Understand the Database Content:

- Familiarize yourself with the structure and content of the Emotional Voices Database (EmoV-DB). Note the languages (English and French), gender distribution of speakers, and emotion classes covered (amusement, anger, sleepiness, disgust, and neutral).

5.1.2. Review the Recording Process:

- Understand how the data was recorded, including the recording environment (anechoic chambers), sampling rate (44.1k down sampled to 16k), and file format (16-bit PCM WAV).

5.1.3. Explore Speaker and Emotion Distribution:

- Analyze the distribution of speakers and emotions in the database. Note the number of male and female speakers, the number of utterances per emotion class for each speaker, and any variations in data availability across speakers or emotions.

5.1.4. Review Data Segmentation:

- Understand how the utterances were segmented and annotated. Note any manual segmentation efforts and the completeness of segmentation for each speaker and emotion class.

5.1.5. Explore Existing Databases:

- Review existing emotional speech databases mentioned in the paper (e.g., RAVDESS) and compare their characteristics with EmoV-DB. Note any differences in terms of language, emotion coverage, and suitability for synthesis purposes.

5.1.6. Analyze Training Data for Voice Transformation:

- Examine the amount of training data used for voice transformation experiments, particularly for neutral-to-neutral and neutral-to-angry transformations. Note the distribution of training data across speakers and emotion pairs.

5.1.7. Review Perception Test Results:

- Analyze the results of the Comparative Mean Opinion Score (CMOS) test conducted to evaluate the performance of voice transformation systems. Examine the accuracy of emotion classification and participants' confidence levels in identifying emotional expressions.

CHAPTER 6

DATA ANALYSIS

6.1. Exploratory data analysis (EDA) and its types

Exploratory data analysis (EDA) is a crucial step to analyze datasets and gain insights into the data's important attributes and their relationships. We'll utilize various forms and techniques of data visualization along with statistical summaries.

These summaries are of 2 types:

1. **Numerical:** We'll calculate numerical summaries key attributes in the Emov-DB dataset.

   Ex: Average (Mean), Mode, Median, Summation etc.,

2. **Graphical:** Here the summaries are represented by graphs.

   Ex: histogram, box Plots, Scatter Plots etc.,

6.2. Why do we need Data Analysis?

It is important to analyse the data for the below reasons:

- ✓ To identify the distribution of dataset.
- ✓ Selecting the appropriate machine learning model and algorithm.
- ✓ Best features extraction.
- ✓ Evaluation of the model and presentation of the test results.

6.3. EDA Inferences

6.3.1. Gender and language of recorded sentences of/from each actor/speaker and amount of utterances segmented per speaker and per emotion.

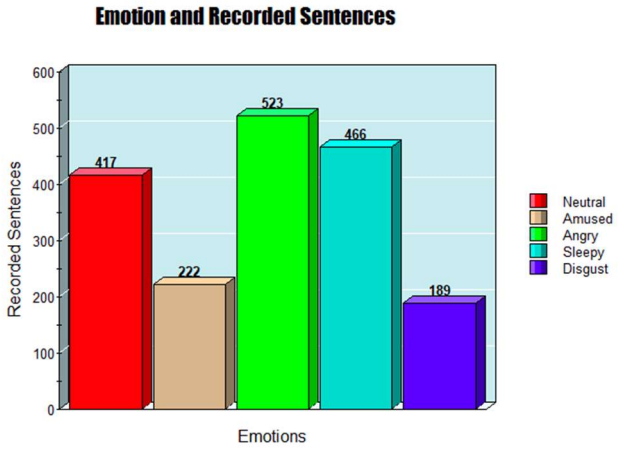


Figure 6.1: Bar Graph for Emotions Vs Recorded Sentences

6.3.2. Amount of data used for training for neutral to neutral and neutral to angry tranformations

Pairs	Spk-Bea	Spk-Sa	Spk-No
Neutral-neutral	355	456	243
Neutral-Angry	296	456	243

Table 6.1: Amount of Data used for Training

6.3.3. Percentage of angry and neutral speech styles being accurately classified

Pairs	Spk-Bea	Spk-Sa	Spk-No
Neutral-neutral	96%	90%	98%
Neutral-Angry	78%	71%	83%

Table 6.2: Percentage of angry and neutral speech

6.3.4. Discussions

The database prioritizes amused speech with speech-laughs due to its perceived amusement. Previous studies highlight its effectiveness over amused smiled speech. Actors were encouraged to use nonverbal expressions to convey other emotions during simulations. This is why we didn't take amused into our research.

CHAPTER 7
MODELLING

Our proposed network architecture aims to synthesize spectrograms from text inputs using a staged synthesis approach, comprising two key components: Text2Mel and Spectrogram Super-resolution Network (SSRN). This approach, inspired by literature [2, 32], offers advantages over direct synthesis methods, particularly in handling high-resolution data.

7.1.Text2Mel: Text to Mel Spectrogram Network

Text2Mel is the cornerstone of our method, focusing on generating a coarse mel spectrogram from input text. This module consists of four interconnected submodules: Text Encoder, Audio Encoder, Attention, and Audio Decoder.

Text Encoder encodes the input text, represented as a sequence of characters, into key (K) and value (V) matrices. These matrices capture the semantic and syntactic information of the input text, facilitating subsequent processing.

$$\mathbf{(K,V)=TextEnc(L),Q=AudioEnc(S_{1:F,1:T})}$$

Audio Encoder processes the coarse mel spectrogram of speech, extracting relevant features and representing them as a query (Q) matrix. This matrix encapsulates the acoustic characteristics of the input speech signal.

Attention mechanism evaluates the relationship between the encoded text and audio features, producing an attention matrix (A) that highlights the correspondence between characters in the text and mel spectrogram frames.

$$\mathbf{A= softmax_{n\text{-}axis(}\ K^{T}Q/\sqrt{D})}$$

Audio Decoder utilizes the attention-guided information to estimate a mel spectrogram from the seed matrix (R). The resulting spectrogram approximates the temporally-shifted ground truth, and the error is evaluated using a combination of L1 loss and binary divergence.

$$\mathbf{R=Att(Q,K,V):=VA}$$

TextEnc and AudioDec leverage fully convolutional networks, incorporating dilated convolutions to capture long contextual information efficiently. These architectures eliminate the need for recurrent units, enhancing computational efficiency without sacrificing performance.

7.1.1. Spectrogram Super-resolution Network (SSRN)

SSRN complements Text2Mel by synthesizing a full spectrogram from the coarse mel spectrogram generated earlier. This network employs deconvolution layers to upsample the temporal axis and increase the resolution of the spectrogram.

7.2. Guided Attention

7.2.1. Guided Attention Loss: Motivation, Method, and Effect

To address the challenges associated with training attention modules, we introduce a guided attention loss mechanism. This method encourages the attention matrix to exhibit a nearly diagonal structure, aligning text positions with audio segments more effectively. By penalizing deviations from this structure, we enhance the training efficiency of the attention module, resulting in improved performance

7.2.2. Forcibly Incremental Attention at the Synthesis Stage

At the synthesis stage, we observe instances where the attention matrix may fail to focus accurately on the correct characters. To mitigate such errors and enhance the robustness of the system, we introduce a heuristic modification to enforce incremental attention adjustments. By dynamically adjusting the attention target based on the current and previous positions, we ensure more reliable character recognition during synthesis.

Through the integration of these components and techniques, our proposed network offers a comprehensive solution for synthesizing high-quality spectrograms from text inputs, with improved efficiency and accuracy.

Based on the provided text, here's a structured modeling of the Tacotron architecture:

7.3 Tacotron Model Architecture:

7.3.1 Encoder:

- **Character Embeddings:** Each character is represented as a one-hot vector and embedded into a continuous vector.
- **Pre-net:** A bottleneck layer with ReLU activations and dropout, applied to each character embedding. This helps convergence and improves generalization.
- **CBHG Module:** Consists of a 1D convolutional bank, a highway network, and a bidirectional GRU. The conv bank models local and contextual information, highway nets extract high-level features, and biGRU captures sequential dependencies.

7.3.2 Decoder:

- **Attention:** Uses content-based tanh attention, where the attention query comes from the decoder RNN output at each timestep.

- **Decoder RNN:** Stack of GRU layers with residual connections across layers, speeding up convergence.
- **Output Layer**: Predicts r nonoverlapping spectrogram frames at each decoder step, e.g. if r=5, it predicts 5 consecutive frames. This reduces number of decoder steps by r.
- **Decoder Input:** First step conditioned on all zeros <GO> frame. Subsequent steps use last predicted frame as input during inference.

7.3.3 Post-Processing Net:

- Another CBHG Module applied on the full decoded mel-spectrogram sequence.
- Converts from the mel-scale spectrogram domain to linear-scale spectrograms suitable for synthesis.
- Having the full decoded sequence allows it to correct individual frame errors using both forward and backward context.

7.3.4 Waveform Synthesis:

- Uses the Griffin-Lim algorithm, an iterative spectrogram inversion technique, to generate waveforms from the predicted linear spectrograms.
- Enhances predicted magnitudes by a power of 1.2 before applying Griffin-Lim to improve harmonics and minimize artifacts.

CHAPTER 8

RESULTS AND CONCLUSIONS

8.1. Results

Below are the results obtained

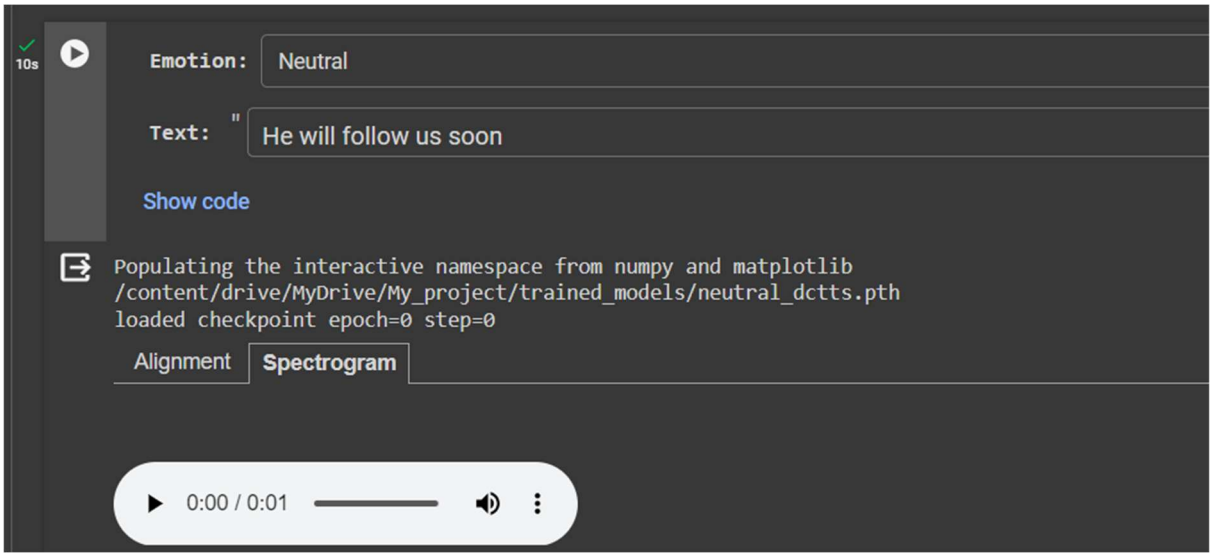


Figure 8.1 Emotional speech output

Here are the mel-spectrograms of various emotional speech types as obtained from our output:

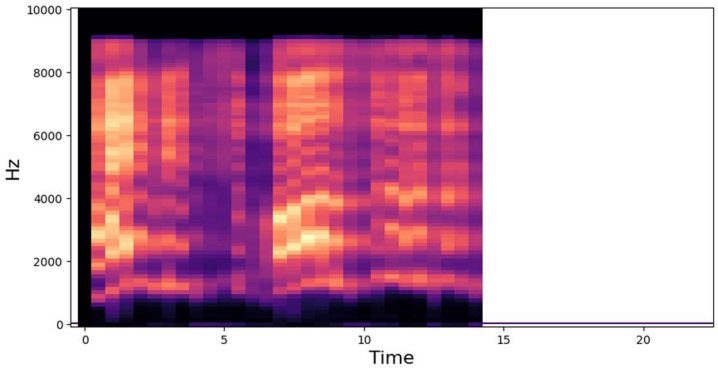


Figure 8.2 Mel-Spectrogram for generated Neutral Voice

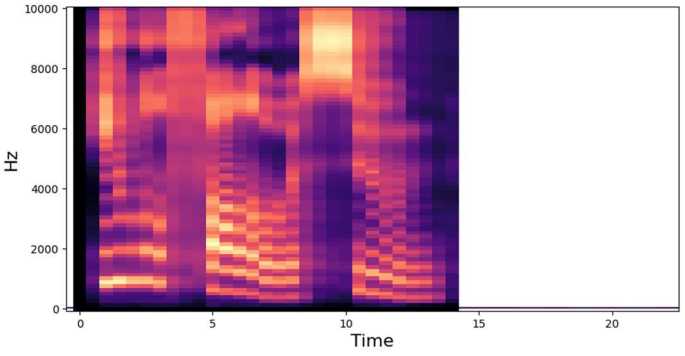


Figure 8.3 Mel-Spectogram for generated Angry Voice

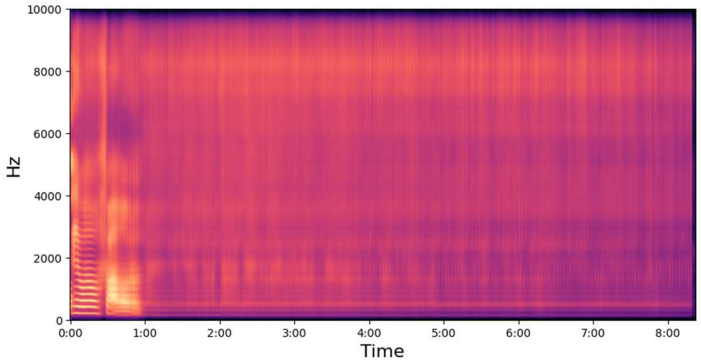


Figure 8.4 Mel-Spectogram for generated Sleepy Voice

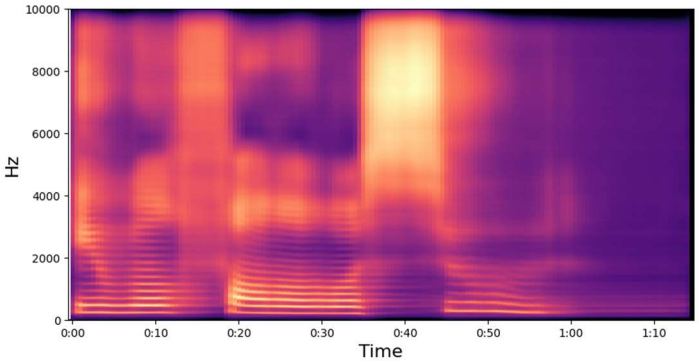


Figure 8.5 Mel-Spectogram for generated Disgust

8.2. Evaluations

1. There might be millions of ideas and hypotheses in the world because it is just an observation that has not been proven to be true in practice.
2. After having enough accumulated evidence which supports a hypothesis, it becomes a theory.
3. As we propose a new idea on expressive TTS synthesis, in order to prove its usefulness, this proposal must be theoretically substantiated, practically feasible, and tested in various ways.
4. We have already provided a theoretical explanation concerning EMO-TTS and shown practical details for the proposed architecture.
5. However, there is still one important piece missing. This piece is a model evaluation in inference time.
6. Initially, we will describe what metrics we have used to evaluate the quality of our Text-To-Speech model and report details of experiments that we have organized.
7. Later, we will show the results obtained regarding the experiment carried out. The experiments aim to evaluate two different aspects.
8. The first aspect is to check the quality of the EMO-TTS audio in comparison with the competing model by comparing the MOS metric.
9. The second objective is to evaluate the ability of the network to produce emotions.

For convenience, we decided to collect data for MOS and emotional accuracy in one survey. We choose to test 4 types of emotions: neutral, angry, disgust, and Sleepiness. To ensure fairness and comparability, we have chosen to collect ratings from a sufficient number of participants for both the original voices and our suggested model.
These are the results obtained:

	Neutral	Angry	Sleepiness	Disgust	Average
Ground Truth	4.5	4.75	4.2	4.6	4.53
Our Model	4.1	4.0	3.6	3.4	3.77

Table 8.1 Mean opinion scores

8.3. Conclusion

In conclusion, the field of computer science has witnessed a profound transformation over the past two decades, reminiscent of an industrial revolution, fueled by advancements in technology, particularly Artificial Intelligence (AI). This evolution has significantly impacted various domains, including Natural Language Processing (NLP). Within NLP, our focus lies on the domain of text and speech processing, with a specific emphasis on Expressive Text-to-Speech (TTS) tasks. The identification of the problem highlights the crucial role of TTS technology in enabling access to information and communication for individuals with diverse needs. However, conventional TTS systems often lack emotional expression, posing a barrier to effective communication. Thus, the problem definition centers on developing TTS tools capable of conveying emotions authentically. Our objective is clear: to enhance TTS technology by infusing it with emotional intelligence, thereby improving user experience and accessibility. This entails exploring and adapting emotional modules within existing speech synthesis models. Existing TTS models primarily rely on neural networks for speech synthesis. Our exploration encompasses understanding earlier non-neural network-based technologies, examining contemporary neural network architectures, and analyzing state-of-the-art methodologies in TTS construction. Overall, our pursuit aims to bridge the gap between human and synthetic speech, ultimately advancing the field of TTS technology to better serve diverse user needs and enhance communication experiences.

REFERENCES

[1] Sejnowski T., and Rosenberg C. (1987), “Parallel networks that learn to pronounce English text”, Complex Systems, Vol. 1, No. 1. 145-168.

[2] McCulloch N., Bedworth M, and Bridle J. (1987), “NETspeak - A re-implementation of NETtalk”, Computer Speech and Language, Vol. 2, 284-301.

[3] Damper R.I. (1995), “Self-learning and connectionist approaches to text-to-phoneme conversion”, in Connectionist Models of Memory and Language, Levy J., Bairaktaris J., Bullinaria J., and Cairns P. (eds.), UCL Press, London, 117-144.

[4] Dan Jurafsky and James H. Martin. Speech and Language Processing. 3rd ed. Stanford, 2020.

[5] Stanford. “Diphone TTS architecture”. In: 2019. URL: https://nlp.stanford.edu/courses/lsa352/lsa352.lec4.6up.pdf

[6] Dan Jurafsky and James H. Martin. Speech and Language Processing. 3rd ed. Stanford, 2020.

[7] Zhigang Yin. “An Overview of Speech Synthesis Technology”. In: 2018. URL: https://www.researchgate.net/publication/340230997_An_Overview_of_Speech_Synthesis_Technology

[8] Justin Johnson, Alexandre Alahi, Li Fei-Fei. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. In: 2016. URL: https://arxiv.org/pdf/1603.08155.pdf

[9] Yuxuan Wang, Daisy Stanton, Yu Zhang, RJ Skerry-Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Fei Ren, Ye Jia, Rif A. Saurous. “Style Tokens: Unsupervised Style Modeling, Control and Transfer in End-to-End Speech Synthesis”. In: 2018. URL: https://arxiv.org/pdf/1803.09017.pdf

[10] RJ Skerry-Ryan, Eric Battenberg, Ying Xiao, Yuxuan Wang, Daisy Stanton, Joel Shor, Ron J. Weiss, Rob Clark, Rif A. Saurous. “Towards End-to-End Prosody Transfer for Expressive Speech Synthesis with Tacotron”. In: 2018. URL: https://arxiv.org/pdf/1803.09047.pdf

[11] Deana L. Pennell, Yang Liu. “Evaluating the effect of normalizing informal text on TTS output”. In: 2012. URL: https://ieeexplore.ieee.org/document/6424271

[12] Rupesh Kumar Srivastava, Klaus Greff, Jürgen Schmidhuber. “Highway Networks”. In: 2015. URL: https://arxiv.org/abs/1505.00387

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. “Deep Residual Learning for Image Recognition”. In: 2015. URL: https://arxiv.org/pdf/1512.03385.pdf

[14] Yusuke Yasuda, Xin Wang†, Junichi Yamagishi. “END-TO-END TEXT-TO-SPEECH USING LATENT DURATION BASED ON VQ-VAE”. In: 2020. URL: https://arxiv.org/pdf/2010.09602.pdf

[15] Ya-Jie Zhang, Shifeng Pan2, Lei He, Zhen-Hua Ling. “LEARNING LATENT REPRESENTATIONS FOR STYLE CONTROL AND TRANSFER IN END-TO-END SPEECH SYNTHESIS”. In: 2019. URL: https://arxiv.org/pdf/1812.04342.pdf

[16] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss†, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio†, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, Rif A. Saurous. “TACOTRON: TOWARDS END-TO-END SPEECH SYNTHESIS”. In: 2017. URL: https://arxiv.org/pdf/1703.10135.pdf

[17] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. WaveGlow: A Flow-based Generative Network for Speech Synthesis. 2018.

[18] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. 2018.

[19] Wei Ping et al. Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning. 2018.

[20] Jonathan Boilard, Philippe Gournay, R. Lefebvre. “A Literature Review of WaveNet: Theory, Application and Optimization”. In: 2019. URL: https://www.researchgate.com/publication/333135603_A_Literature_Review_of_WaveNet_Theory_Application_and_Optimization

[21] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu. “WAVENET: A GENERATIVE MODEL FOR RAW AUDIO”. In: 2016. URL: https://arxiv.org/pdf/1609.03499.pdf

[22] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, Yonghui Wu. “Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions”. In: 2017. URL: https://arxiv.org/abs/1712.05884

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Łukasz Kaiser. “Attention is all you need”. In: 2017. URL: https://arxiv.org/pdf/1706.03762.pdf

[24] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, Ming Liu, Ming Zhou. “Neural Speech Synthesis with Transformer Network”. In: 2019. URL: https://arxiv.org/pdf/1809.08895.pdf

[25] Heejin Choi, Sangjun Park, Jinuk Park, Minsoo Hahn. “Emotional Speech Synthesis for Multi-Speaker Emotional Dataset Using WaveNet Vocoder”. In: 2019. URL: https://ieeexplore.ieee.org/document/8661919

[26] Ohsung Kwon, Inseon Jang, ChungHyun Ahn, Hong-Goo Kang. “Emotional Speech Synthesis Based on Style Embedded Tacotron2 Framework”. In: 2019. URL: https://ieeexplore.ieee.org/abstract/document/8793393

[27] Sheng-Yeh Chen, Chao-Chun Hsu, Chuan-Chun Kuo, Ting-Hao (Kenneth) Huang, Lun-Wei Ku. “EmotionLines: An Emotion Corpus of Multi-Party Conversations”. In: 2018. URL: https://arxiv.org/pdf/1802.08379.pdf

[28] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, Yonghui Wu. “LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech”. In: 2019. URL: https://arxiv.org/abs/1904.02882

[29] Richard Socher. Sequence-to-sequence with attention. 2018. URL: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/lectures/lecture11.pdf

[30] Afshine Amidi and Shervine Amidi. Recurrent Neural Networks cheatsheet. 2018. URL: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[31] Machine Learns. Text to Speech Deep Learning Architectures. 2018. URL: https://erogol.com/text-speech-deep-learning-architectures/

[32] Open Smile. AUDIO FEATURE EXTRACTION. 2021. URL: https://www.audeering.com/opensmile/

[33] Kion Kin. WaveNet: Increasing reception field using dilated convolution. 2018. URL: https://medium.com/@kion.kim/wavenet-a-network-good-to-know-7caaae735435#:~:text=WaveNet%20is%20an%20generative%20model,second%20for%20a%20reasonable%20quality..

[34] Gary Mckeown, Michel Valstar, Roddy Cowie, Maja Pantic. “The SEMAINE Database: Annotated Multimodal Records of Emotionally Colored Conversations between a Person and a Limited Agent”. In: 2013. URL: https://www.researchgate.net/publication/224248863_The_SEMAINE_Database_Annotated_Multimodal_Records_of_Emotionally_Colored_Conversations_between_a_Person_and_a_Limited_Agent

[35] Soujanya Poria et al. “MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversations”. In: CoRR abs/1810.02508 (2018). URL: http://arxiv.org/abs/1810.02508

[36] Keith Ito and Linda Johnson. The LJ Speech Dataset. URL: https://keithito.com/LJ-Speech-Dataset/. 2017

[37] Carlos Busso et al. “IEMOCAP: Interactive emotional dyadic motion capture database”. In: Language Resources and Evaluation 42 (Dec. 2008), pp. 335–359

[38] M. Kathleen Pichora-Fuller and Kate Dupuis. Toronto emotional speech set (TESS). Version DRAFT VERSION. 2020. URL: https://doi.org/10.5683/SP2/E8H2MF

[39] Steven R. Livingstone and Frank A. Russo. “The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English”. In: PLOS ONE 13.5 (May 2018), pp. 1–35. URL: https://doi.org/10.1371/journal.pone.0196391

Appendix: A- Packages, Tools used & Working Process

Python Programming language

Python is a high-level Interpreter based programming language used especially for general-purpose programming. Python features a dynamic type of system and supports automatic memory management.

It supports multiple programming paradigms, including object-oriented, functional and Procedural and also has its a large and comprehensive standard library. Python is of two versions. They are Python 2 and Python 3.

This project uses the latest version of Python, i.e., Python 3. This python language uses different types of memory management techniques such as reference counting and a cycle-detecting garbage collector for memory management. One of its features is late binding (dynamic name resolution), which binds method and variable names during program execution. Python's offers a design that supports some of things that are used for functional programming in the Lisp tradition. It has vast usage of functions for faster results such as filter, map, split, list comprehensions, dictionaries, sets and expressions. The standard library of python language has two modules like itertools and functools that implement functional tools taken from Standard machine learning.

Libraries

NumPy

Numpy is the basic package for scientific calculations and computations used along with Python. NumPy was created in 2005 by Travis Oliphant. It is open source so can be used freely. NumPy stands for Numerical Python. And it is used for working with arrays and mathematical computations.

Using NumPy in Python gives you much more functional behavior comparable to MATLAB because they both are interpreted, and they both allows the users to quickly write fast programs as far as most of the operations work on arrays, matrices instead of scalars. Numpy is a library consisting of array objects and a collection of those routines for processing those arrays.

Numpy has also functions that mostly works upon linear algebra, Fourier transform, arrays and matrices. In general scenario the working of NumPy in the code involves searching, join, split, reshaping etc. operations using NumPy.

The syntax for importing the NumPy package is → import NumPy as np indicates NumPy is imported alias np. We used the version 1.23.0.

TensorFlow (tf):

TensorFlow is an open-source machine learning framework developed by Google. It provides a comprehensive ecosystem of tools, libraries, and community resources for building and training machine learning models. TensorFlow uses data flow graphs to represent computation, allowing for efficient execution on CPUs, GPUs, and TPUs (Tensor Processing Units). It supports a wide range of tasks, including classification, regression, clustering, and natural language processing. We used 2.15.0 version.

docopt:

docopt is a Python library for parsing command-line arguments based on the usage string of the command-line interface (CLI). It allows developers to define command-line interface specifications using a human-readable format, and automatically generates a parser based on those specifications. docopt simplifies the process of creating user-friendly CLI applications by handling argument parsing and validation. We used 0.6.2 version.

PyTorch:

PyTorch is an open-source machine learning library developed by Facebook. It is known for its dynamic computation graph feature, which allows for more flexibility in model construction and debugging compared to static computation graphs. PyTorch provides a rich set of tools and modules for building neural networks, optimization algorithms, and handling data. It has gained popularity for its ease of use, flexibility, and strong support for GPU acceleration. we used 2.2.1 version.

nltk:

NLTK (Natural Language Toolkit) is a leading platform for building Python programs to work with human language data. It provides tools and resources for tasks such as tokenization, stemming, tagging, parsing, and semantic reasoning. NLTK also includes a collection of corpora and lexical resources for training and evaluating natural language processing models. It is widely used in academia and industry for research, education, and development of NLP applications. We used 3.8.1 version.

tacotron_pytorch:

tacotron_pytorch appears to be a custom module or package for Tacotron implemented in PyTorch. Tacotron is a neural network architecture used for text-to-speech synthesis, specifically designed to generate high-quality and natural-sounding speech from input text. The PyTorch implementation of Tacotron likely includes components for text preprocessing, model architecture, training, and inference.

tqdm:

tqdm is a Python library that provides a fast, extensible progress bar for loops and iterables. It enhances the user experience by displaying a progress bar with estimated completion time, percentage completed, and other relevant information. tqdm is easy to integrate into existing codebases and supports various iterable types, making it useful for monitoring the progress of tasks such as data loading, training epochs, and batch processing. We used 4.66.2 version

librosa:

librosa is a Python package for music and audio analysis. It provides tools for loading audio files in various formats, extracting features such as spectrograms and MFCCs (Mel-frequency cepstral coefficients), manipulating audio signals, and performing tasks such as pitch detection, tempo estimation, and beat tracking. librosa is widely used in audio research, music information retrieval, and sound processing applications. We used 0.9.1 version.

tensorboard_logger:

tensorboard_logger is a Python library for logging PyTorch training metrics to TensorBoard. TensorBoard is a visualization tool for deep learning experiments developed by Google as part of the TensorFlow ecosystem. tensorboard_logger allows users to log scalar values, images, audio clips, and other types of data during model training, and visualize them in TensorBoard's web interface. It simplifies the process of monitoring and analyzing model performance and behavior. We used 2.15.2 version.

setuptools:

setuptools is a Python library for packaging and distributing Python projects. It provides utilities for defining project metadata, specifying dependencies, creating distribution packages, and installing packages from source or binary distributions. setuptools simplifies the process of managing project dependencies and deploying Python applications across different environments. It is commonly used in conjunction with tools such as pip and virtualenv for managing Python packages. We used 67.7.2 version.

Pandas

Pandas is used whenever working with matrix data, time series data and mostly on tabular data. Pandas is also open-source library which provides high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

This helps extremely in handling large amounts of data with help of data structures like Series, Data Frames etc. It has inbuilt methods for manipulating data in different formats like csv, html etc.,

Simply we can define pandas is used for data analysis and data manipulation and extremely works with data frames objects in our project, where data frame is dedicated structure for two-dimensional data, and it consists of rows and columns similar to database tables and excel spreadsheets.

In our code we firstly import pandas package alias pd and use pd in order to read the csv file and assign to data frame and in the further steps. We work on the data frames by manipulating them and we perform data cleaning by using functions on the data frames such as df.isna().sum(). So, finally the whole code depends on the data frames which are to be acquired by coordinating with pandas. So, this package plays a key role in our project. We used 1.5.3 version.

Matplotlib

Matplotlib is a library used for plotting in the Python programming language and it is a numerical mathematical extension of NumPy. Matplotlib is most commonly used for visualization and data exploration in a way that statistics can be known clearly using different visual structures by creating the basic graphs like bar plots, scatter plots, histograms etc.

Matplotlib is a foundation for every visualizing library and the library also offers a great flexibility with regards to formatting and styling plots. We can choose freely certain assumptions like ways to display labels, grids, legends etc.

In our code firstly we import the matplotlib.pyplot alias plt, This plt comes into picture in the exploratory data analysis part to analyze and summarize datasets into visual methods, we use plt to add some characteristics to figures such as title, legends, labels on x and y axis as said earlier ,to understand more clearly we can also use different plots. We used 3.7.1 version.

Appendix: B

Sample Source Code with Execution

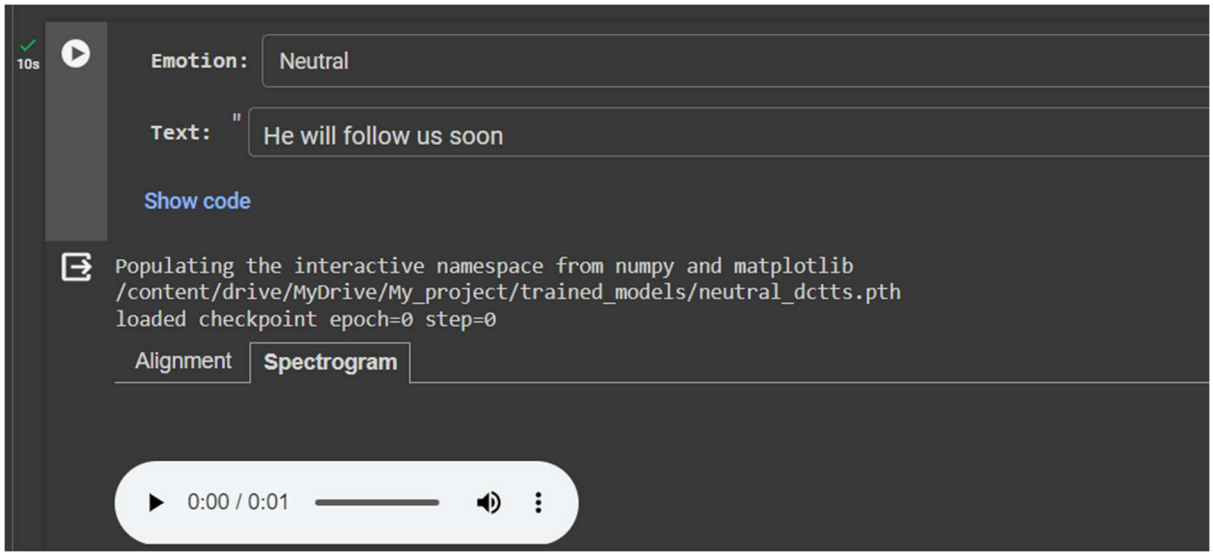
```
import tensorflow as tf
%pylab inline
rcParams["figure.figsize"] = (10,5)

import os
import sys
import numpy as np
import librosa
sys.path.append('/content/drive/MyDrive/My_project/pytorch-dc-tts')
sys.path.append('/content/drive/MyDrive/My_project/pytorch-dc-
tts/models')
sys.path.append("/content/drive/MyDrive/My_project/tacotron_pytorch")
sys.path.append("/content/drive/MyDrive/My_project/tacotron_pytorch/lib
/tacotron")

# For the DC-TTS
import torch
from text2mel import Text2Mel
from ssrn import SSRN
from audio import save_to_wav, spectrogram2wav
from utils import get_last_checkpoint_file_name, load_checkpoint_test,
save_to_png, load_checkpoint
from datasets.emovdb import vocab, get_test_data
import librosa.display
import matplotlib.pyplot as plt
import ipywidgets as widgets

from text import text_to_sequence, symbols
# from util import audio

from tacotron_pytorch import Tacotron
from synthesis import tts as _tts

# For Audio/Display purposes
import librosa.display
import IPython
from IPython.display import Audio
from IPython.display import display
from google.colab import widgets
from google.colab import output
import warnings
warnings.filterwarnings('ignore')
```

```
torch.set_grad_enabled(False)
text2mel = Text2Mel(vocab).eval()

ssrn = SSRN().eval()
load_checkpoint('/content/drive/MyDrive/My_project/trained_models/ssrn.
pth', ssrn, None)

model = Tacotron(n_vocab=len(symbols),
                 embedding_dim=256,
                 mel_dim=80,
                 linear_dim=1025,
                 r=5,
                 padding_idx=None,
                 use_memory_mask=False,
                 )

def visualize(alignment, spectrogram, Emotion, fs=22050,
hop_length=512):
    label_fontsize = 16
    tb = widgets.TabBar(['Alignment', 'Spectrogram'], location='top')

    with tb.output_to('Alignment'):
        plt.imshow(alignment.T, aspect="auto", origin="lower",
interpolation=None)
        plt.xlabel("Decoder timestamp", fontsize=label_fontsize)
        plt.ylabel("Encoder timestamp", fontsize=label_fontsize)

    with tb.output_to('Spectrogram'):
        if Emotion in ['Disgust', 'Amused', 'Sleepiness']:
            librosa.display.specshow(spectrogram.T, sr=fs,
hop_length=hop_length, x_axis="time", y_axis="linear")
        else:
            librosa.display.specshow(spectrogram, sr=fs,
hop_length=hop_length, x_axis="time", y_axis="linear")

        plt.xlabel("Time", fontsize=label_fontsize)
        plt.ylabel("Hz", fontsize=label_fontsize)
        plt.xlim(0, 1)
def tts_dctts(text2mel, ssrn, text):
  sentences = [text]

  max_N = len(text)
  L = torch.from_numpy(get_test_data(sentences, max_N))
  zeros = torch.from_numpy(np.zeros((1, 80, 1), np.float32))
  Y = zeros
  A = None

  for t in range(1000):
```

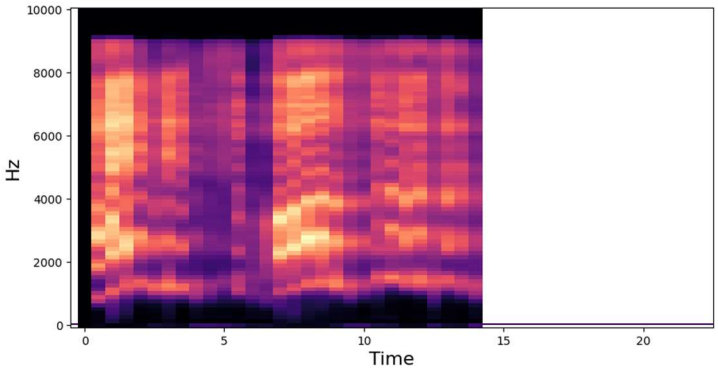
```
      _, Y_t, A = text2mel(L, Y, monotonic_attention=True)
      Y = torch.cat((zeros, Y_t), -1)
      _, attention = torch.max(A[0, :, -1], 0)
      attention = attention.item()
      if L[0, attention] == vocab.index('E'):  # EOS
          break
  _, Z = ssrn(Y)
  Y = Y.cpu().detach().numpy()
  A = A.cpu().detach().numpy()
  Z = Z.cpu().detach().numpy()
  return spectrogram2wav(Z[0, :, :].T), A[0, :, :], Y[0, :, :]

def tts_tacotron(model, text):
    waveform, alignment, spectrogram = _tts(model, text)
    return waveform, alignment, spectrogram

def present(waveform, Emotion, figures=False):
  if figures!=False:
        visualize(figures[0], figures[1], Emotion)
  IPython.display.display(Audio(waveform, rate=fs))
fs = 20500 #20000
hop_length = 1000
model.decoder.max_decoder_steps = 200

%pylab inline
Emotion = "Neutral" #@param ["Neutral", "Angry", "Disgust",
"Sleepiness"]
Text = 'He will follow us soon' #@param {type:"string"}

wav, align, mel = None, None, None

if Emotion == "Neutral":
  load_checkpoint('/content/drive/MyDrive/My_project/trained_models/'+E
motion.lower()+'_dctts.pth', text2mel, None)
  wav, align, mel = tts_dctts(text2mel, ssrn, Text)
elif Emotion == "Angry":
  load_checkpoint_test('/content/drive/MyDrive/My_project/trained_model
s/'+Emotion.lower()+'_dctts.pth', text2mel, None)
  wav, align, mel = tts_dctts(text2mel, ssrn, Text)
  # wav = wav.T
elif Emotion == "Disgust" or Emotion == "Sleepiness":
  checkpoint =
torch.load('/content/drive/MyDrive/My_project/trained_models/'+Emotion.
lower()+'_tacotron.pth', map_location=torch.device('cpu'))
  model.load_state_dict(checkpoint["state_dict"])
  wav, align, mel = tts_tacotron(model, Text)

present(wav, Emotion, (align,mel))
```

Below are the outputs:

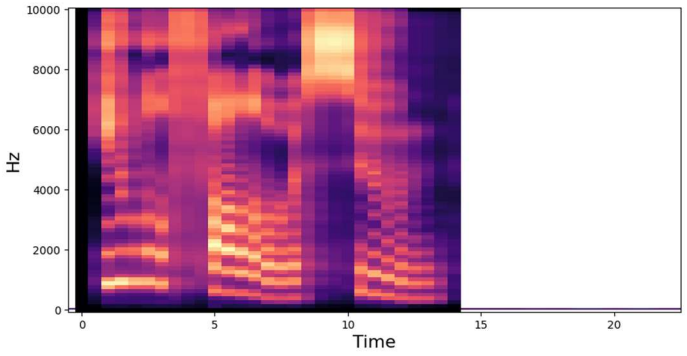
Here are the mel-spectrograms of various emotional speech types as obtained from our output:



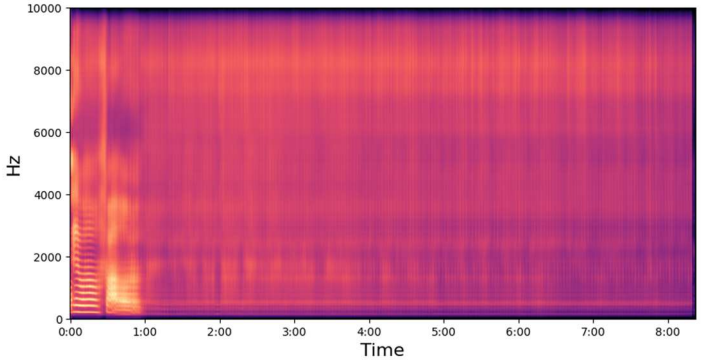
Emotional speech output



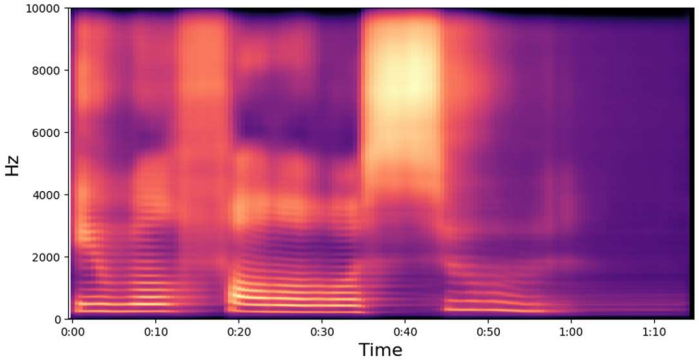
Mel-Spectrogram for generated Neutral Voice



Mel-Spectogram for generated Angry Voice



Mel-Spectogram for generated Sleepy Voice



Mel-Spectogram for generated Disgust Voice