Mini-Project Report On

VocalizeMe (Text to audio converter in one's own voice)

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

Bachelor of Technology

in

Computer Science & Engineering

By

Ajoe Joseph (U2003015)
Alan George (U2003020)
Ashwin V (U2003048)

Under the guidance of
Ms. Meenu Mathew

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology (Autonomous)
Rajagiri Valley, Kakkanad, Kochi, 682039

July 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

CERTIFICATE

This is to certify that the mini-project report entitled ***"VocalizeMe (Text to audio converter in one's own voice)"*** *is a bonafide work done by* ***Mr. Ajoe Joseph (U2003015), Mr. Alan George (U2003020), Mr. Ashwin V (U2003048)****, submitted to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

Dr. Preetha K. G.	**Dr. Sminu Izudheen**	**Ms. Meenu Mathew**
Head of Department	Mini-Project Coordinator	Mini-Project Guide
Dept. of CSE	Professor	Asst. Professor
RSET	Dept. of CSE	Dept. of CSE
	RSET	RSET

ACKNOWLEDGEMENTS

We wish to express our sincere gratitude towards **Dr. P. S. Sreejith**, Principal of RSET, and **Dr. Preetha K. G.**, Head of Department of Computer Science and Engineering for providing us with the opportunity to undertake our mini-project, **"VocalizeMe"**.

We are highly indebted to our mini-project coordinators, **Dr. Sminu Izudheen**, Professor, Department of Computer Science and Engineering, and **Dr. Renu Mary Daniel**, Assistant Professor, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our mini-project guide **Ms. Meenu Mathew**, for her patience and all the priceless advice and wisdom she has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Ajoe Joseph

Alan George

Ashwin V

ABSTRACT

Problem definition:

Existing text-to-speech (TTS) systems do not produce speech that accurately mimics human speech patterns, intonations, and emotions. This can be a problem for applications that require natural-sounding speech, such as audiobooks, video games, and customer service chatbots.

Project objective:

To develop an AI model that converts text to speech in our own voice, with the following objectives: The model should produce speech that is natural-sounding, accurate, and expressive. The model should be able to generate speech in a variety of emotions, such as happy, sad, angry, and surprised. The model should be able to adapt to different speaking styles, such as formal and informal.

Methodology:

Collect a dataset of audio recordings of our own voice. Extract features from the audio recordings, such as pitch, loudness, and timbre. Train a machine learning model on the extracted features. Use the trained model to generate speech from text.

Additional considerations:

The model will need to be trained on a large dataset of audio recordings in order to produce high-quality speech. The model will also need to be able to handle a variety of different accents and speaking styles. Overall, this project has the potential to create a powerful AI tool that can be used in a variety of applications.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Text-to-speech (TTS) systems have become increasingly common in recent years, but they often produce robotic-sounding speech that can be difficult to understand and listen to. This can be especially problematic for people who are blind or have other visual impairments, as they rely on TTS systems to read text aloud. In addition, TTS systems that do not accurately mimic human speech patterns, intonations, and emotions can be used in harmful ways, such as to create deepfakes or to impersonate someone else.

For example, a TTS system that is used to read a news article may not be able to convey the emotional tone of the article, which could lead to a misunderstanding of the content. Similarly, a TTS system that is used to create a customer service chatbot may not be able to accurately mimic the tone and personality of a human customer service representative, which could lead to a negative customer experience.

We believe that there is a need for a more natural-sounding TTS system that can be used in a variety of applications, such as audiobooks, video games, and customer service chatbots. We are inspired to create this model because we believe that it has the potential to make a positive impact on the lives of many people.

For example, blind people could use this model to listen to audiobooks or other text-based content, which would allow them to access information and entertainment that they would otherwise not be able to enjoy. People with dyslexia could use this model to help them read text more easily, which could improve their academic performance and overall quality of life. Learners of foreign languages could use this model to practice their pronunciation, which could help them to become more fluent in the language. Customers

could use this model to interact with customer service chatbots that sound more natural and human-like, which could lead to a more positive customer experience.

We believe that this model has the potential to make a positive impact on the lives of many people, and we are excited to see how it can be used to improve the way we interact with technology.

1.2 Existing System

Text-to-speech (TTS) models have been around for many years, but they have traditionally been limited in their ability to produce natural-sounding speech. This is because they have typically relied on rule-based systems that map text to phonetic transcriptions. These systems are often not able to capture the nuances of human speech, such as intonation, rhythm, and emotion.

Pros of rule-based TTS models:

- They are relatively easy to develop and maintain.
- They can be used to generate speech in a variety of languages.
- They can be used to generate speech with a variety of different accents.

Cons of rule-based TTS models:

- They can be robotic-sounding and unnatural.
- They may not be able to accurately convey the emotional tone of the text.
- They may not be able to adapt to different speaking styles, such as formal and informal.

In recent years, there has been a growing interest in developing TTS models that are based on machine learning. These models are able to learn the statistical relationships between text and speech from large datasets of audio recordings. This allows them to produce more natural-sounding speech than rule-based systems.

Pros of machine learning-based TTS models:

- They can produce more natural-sounding speech than rule-based systems.
- They can be used to generate speech with a wider range of emotions.
- They can be used to generate speech with a wider range of speaking styles.

Cons of machine learning-based TTS models:

- They can be computationally expensive to train.
- They can be sensitive to the quality of the training data.
- They can sometimes produce speech that is not accurate or grammatically correct.

Overall, existing TTS models have made significant progress in terms of producing natural-sounding speech. However, there are still some challenges that need to be addressed in order to create truly natural-sounding TTS systems.

- The need for larger and more diverse datasets of audio recordings.
- The need for better algorithms for training TTS models.
- The need for better ways to evaluate the quality of TTS models.

Despite these challenges, it is likely that TTS models will become more natural-sounding and accurate in the future. As research in this area continues, it is possible that TTS models will eventually be able to produce speech that is indistinguishable from human speech.

1.3 Problem Statement

Text-to-speech (TTS) systems have become increasingly common in recent years, but they often produce robotic-sounding speech that can be difficult to understand and listen to. This can be especially problematic for people who are blind or have other visual impairments, as they rely on TTS systems to read text aloud. In addition, TTS systems that do not accurately mimic human speech patterns, intonations, and emotions can be used in harmful ways, such as to create deepfakes or to impersonate someone else.

The purpose of a text-to-speech (TTS) project is to develop a technology or system that can convert written text into spoken audio output in one's own voice. The primary goal is to enable a machine to produce human-like speech, allowing users to access information or interact with technology through auditory means. The need for a text-to-speech project arises from the increasing demand for accessible, inclusive, and intuitive technologies that can bridge the gap between written content and spoken language.

1.4 Objectives

The objective of a text-to-human speech project is to develop an advanced system that can accurately convert written text into natural and human-like speech. The project aims to achieve the following specific objectives:

1. **Natural Speech Synthesis -** The system should be able to produce speech that sounds natural and human-like. This includes accurately capturing the nuances of human speech, such as intonation, rhythm, and emotion.

2. **Linguistic Accuracy -** The system should be able to accurately convert written text into speech. This includes correctly pronouncing words and phrases, and following grammatical rules.

3. **Voice Customization -** The system should allow users to customize the voice of the speaker. This includes choosing the gender, accent, and tone of voice.

4. **Real-Time or Low Latency Conversion -** The system should be able to convert text into speech in real time or with low latency. This is important for applications where the speech needs to be generated quickly, such as customer service chatbots.

5. **Quality Assurance and Evaluation -** The system should be evaluated for its quality and accuracy. This includes measuring the naturalness of the speech, the linguistic accuracy, and the user satisfaction.

1.5 Scope

The successful development of this project is expected to have a far-reaching impact across diverse applications. Audiobooks, for instance, will see a significant enhancement

in the listening experience with the system's natural-sounding speech, making it more engaging and enjoyable. Similarly, the integration of AI-generated expressive voiceovers in video games will deepen player engagement and emotional connection to the game's narrative, enriching the overall gaming experience.

Moreover, customer service chatbots will benefit from a human-like voice, fostering trust and empathy with customers, resulting in more satisfactory interactions and improved customer experiences. In addition to entertainment and customer service, the project's positive implications extend to accessibility for individuals with visual or reading disabilities. The high-quality speech output will make digital content more immersive and enriching, empowering visually impaired individuals to access information, entertainment, and educational resources with ease.

Furthermore, the project will have a positive impact on individuals with dyslexia, offering an enhanced reading experience through the AI model's natural-sounding speech. This feature can aid in easier comprehension and understanding, potentially improving academic performance and overall quality of life for those with dyslexia. Additionally, learners of foreign languages can take advantage of the model to practice pronunciation with accurate intonations and speech patterns, accelerating their language learning journey and boosting fluency and confidence in speaking the foreign language.

In conclusion, the project's transformative potential reaches across various industries, enriching audio content experiences and contributing to a more inclusive and engaging digital environment for users from diverse backgrounds and abilities. The seamless integration of human-like speech in these applications demonstrates the significant advancements that can be achieved through this initiative.

Chapter 2

Literature Review

2.1 Concatenative Text-to-Speech (TTS)

In Concatenative Text-to-Speech (TTS), the process of synthesizing speech begins with breaking down the linguistic input into smaller, discrete units, such as phonemes, diphones, or short audio segments. These units are carefully selected from a vast database of recorded human speech, capturing various phonetic and acoustic variations. Each unit represents a specific sound or phonetic transition, allowing the TTS system to piece them together like building blocks to form complete sentences and phrases.

The success of Concatenative TTS lies in its ability to maintain the naturalness and expressiveness of human speech by using actual recorded speech as the foundation. The database of recorded speech units acts as a treasure trove of human vocalizations, containing diverse intonations, emotions, and linguistic nuances. During the synthesis process, the TTS system intelligently concatenates these units based on the linguistic context to create coherent and articulate speech output. The result is an output that closely mimics human speech, capturing the subtle variations in pronunciation and rhythm that contribute to the overall naturalness of the synthesized voice. However, while Concatenative TTS produces high-quality speech with realistic intonations and emotions, it faces challenges related to the size of the database, computational complexity, and seamless unit concatenation. Researchers continue to explore innovative techniques and hybrid approaches to further enhance the capabilities of Concatenative TTS and offer an ever more natural and expressive auditory experience.

Advantages:

1. **High Sound Quality:** Concatenative TTS can produce speech with exceptionally

high sound quality and naturalness. This is because the recorded speech units used for concatenation are real human speech, capturing the nuances and subtleties of human vocalization.

2. **Expressive Voice:** With a large database of recorded speech units, Concatenative TTS can achieve a wide range of voice expressions, capturing emotions like excitement, sadness, anger, and more. This expressive voice capability is beneficial for applications that require dynamic and emotive speech, such as interactive storytelling and voice acting.

3. **Better Control Over Prosody:** Since Concatenative TTS stitches together pre-recorded speech units, it can better preserve the original prosody and rhythm of the human voice. This results in more natural-sounding speech with appropriate intonations and pauses.

4. **Stability and Predictability:** Concatenative TTS generally produces more stable and predictable speech output since it relies on real speech recordings. This makes it suitable for applications where consistent and reliable speech synthesis is essential, such as voice prompts in telephony systems.

5. **Less Sensitivity to Training Data Size:** Concatenative TTS is relatively less sensitive to the size of the training dataset compared to purely statistical approaches. Even with a smaller voice database, it can still generate high-quality speech.

Drawbacks:

1. **Data Storage Requirements:** One of the main drawbacks of Concatenative TTS is the substantial storage requirements for the large database of recorded speech units. This can become an issue when dealing with multiple voice profiles or when trying to deploy the system on resource-limited devices.

2. **Limited Flexibility:** Concatenative TTS relies heavily on the availability of speech segments in its database. This lack of flexibility can be a challenge when dealing with rare words or highly specific linguistic contexts, leading to unnatural-sounding synthesis.

3. **Difficulty in Seamless Concatenation:** Achieving seamless concatenation of speech units without noticeable artifacts can be challenging. If the recorded units do not fit perfectly, it can result in abrupt transitions, affecting the overall naturalness of the generated speech.

4. **Voice Cloning Challenges:** Creating a new voice profile in Concatenative TTS requires extensive voice recordings and careful segmentation, which can be time-consuming and costly for voice cloning applications.

5. **Lack of Personalization:** Unlike some other TTS methods, Concatenative TTS does not inherently support real-time personalization or adaptation to individual speakers, limiting its application in scenarios requiring user-specific voice synthesis.

Despite these drawbacks, Concatenative TTS remains a popular and effective approach for generating high-quality and expressive speech. Addressing the storage and flexibility challenges while exploring hybrid techniques can further enhance the capabilities of Concatenative TTS in future developments.

2.2 Statistical Parametric Text-to-Speech (TTS)

Statistical Parametric Text-to-Speech (TTS) represents a different approach to speech synthesis by utilizing powerful statistical models to generate speech from linguistic and acoustic features. The process begins with extracting linguistic features from the input text, such as phonemes, prosody, and linguistic context. Concurrently, acoustic features are extracted from the corresponding speech recordings in the training dataset. These paired linguistic and acoustic features form the basis for building statistical models that capture the relationship between linguistic information and its corresponding speech representation.

Two common statistical modeling techniques used in Statistical Parametric TTS are hidden Markov models (HMMs) and deep neural networks (DNNs). HMMs are traditionally employed to model the temporal dependencies and variations in acoustic features over time. By incorporating linguistic features into the model, HMMs can predict the sequence of acoustic parameters needed to synthesize speech from the input text. On the

other hand, deep neural networks have gained prominence due to their ability to learn complex patterns and representations from data. DNN-based TTS models use multiple layers of interconnected neurons to map linguistic features to acoustic features, offering a more direct and expressive mapping. These neural networks can be trained with large datasets to capture intricate speech patterns and generate speech with impressive naturalness and fluency. The adaptability of statistical parametric models allows them to handle variations in speech characteristics among different speakers, making them valuable for building multilingual TTS systems or synthesizing voices with personalized attributes. The continuous advancements in neural network architectures and training methodologies contribute to the ongoing improvement of Statistical Parametric TTS, providing a compelling alternative to traditional concatenative methods and paving the way for more sophisticated and expressive speech synthesis technologies.

Advantages:

1. **Flexibility in Voice Generation:** Statistical Parametric TTS offers more flexibility in generating new voices compared to Concatenative TTS. It does not require an extensive database of pre-recorded speech units for each voice, making it easier to create new voice profiles or adapt existing ones.

2. **Efficient Storage:** Since Statistical Parametric TTS relies on statistical models to predict acoustic features from linguistic inputs, it requires significantly less storage compared to the vast database needed for Concatenative TTS. This makes it more suitable for resource-constrained environments.

3. **Ease of Voice Cloning:** Creating a new voice profile in Statistical Parametric TTS typically involves collecting a smaller dataset of aligned text and speech from a new speaker. This process is generally less time-consuming and expensive compared to the extensive recording required for voice cloning in Concatenative TTS.

4. **Adaptability to Different Languages:** Statistical Parametric TTS models can be trained on multilingual datasets, allowing them to generate speech in multiple languages without the need for language-specific databases. This adaptability is advantageous for applications that cater to a diverse user base.

5. **Less Sensitivity to Speaker Variability:** Statistical Parametric TTS can handle variations in speaking style and voice quality among speakers in the training data, resulting in more robust voice synthesis even when dealing with different speakers.

Drawbacks:

1. **Quality Depends on Training Data:** The quality and naturalness of the synthesized speech heavily rely on the quality and diversity of the training data. A lack of high-quality and varied data may lead to subpar speech synthesis.

2. **Challenges with Rare Words and Out-of-Domain Text:** Statistical Parametric TTS models can struggle with generating accurate pronunciations for rare or out-of-vocabulary words, leading to pronunciation errors in certain contexts.

3. **Difficulty in Capturing Expressive Voice:** While Statistical Parametric TTS can generate clear and intelligible speech, capturing highly expressive and emotive voice characteristics can be more challenging compared to Concatenative TTS.

4. **Data Dependency for Fine-Tuning:** Fine-tuning or adapting a Statistical Parametric TTS model to a specific speaker or domain may require a significant amount of speaker-specific data, which may not always be readily available.

5. **Limited Control Over Prosody:** Unlike Concatenative TTS, which retains original prosody, Statistical Parametric TTS relies on models to predict prosody, which can sometimes result in less natural intonations and rhythm.

Despite these drawbacks, Statistical Parametric TTS remains a widely used approach for generating synthetic speech. Improving training data quality, developing advanced statistical models, and exploring hybrid techniques can further enhance the naturalness and flexibility of Statistical Parametric TTS in future developments.

2.3 Rule-Based Text-to-Speech (TTS)

Rule-Based Text-to-Speech (TTS) is a classic approach to speech synthesis that relies on a set of predetermined linguistic rules and knowledge about the language to generate speech from textual input. These rules define the mapping between linguistic units, such

as phonemes, graphemes, or linguistic features, and the corresponding acoustic parameters required to produce speech. In rule-based synthesis, the text is first analyzed to determine the sequence of phonetic and linguistic elements, which are then transformed into speech using the predefined rules. This approach allows for precise control over pronunciation, intonation, and articulation, making it suitable for applications that require accurate and specific speech output.

Another variant of Rule-Based TTS is formant synthesis, which involves modeling the human vocal tract's formant frequencies to produce speech. The formants represent the resonant frequencies of the vocal tract, and by controlling their values, different speech sounds can be synthesized. Formant synthesis allows for greater control over articulation and allows the simulation of specific vocal tract configurations, making it valuable for applications that require specialized speech outputs, such as speech therapy or voice modification. However, despite its interpretability and customization advantages, Rule-Based TTS can sometimes struggle to achieve the naturalness and expressiveness present in human speech. Capturing subtle nuances, variations, and emotions can be challenging through predefined rules alone, leading to speech output that may sound more robotic and less engaging. The development of extensive linguistic rules for multiple languages can also be time-consuming and resource-intensive, making it less feasible for some applications. However, Rule-Based TTS remains an essential technique for certain specialized use cases where precise control over speech synthesis is critical, and its straightforward and interpretable nature continues to make it an important part of the broader landscape of text-to-speech technologies.

Advantages:

1. **Interpretable and Customizable:** Rule-Based TTS relies on predefined linguistic rules, making it highly interpretable and customizable. Developers can fine-tune the rules to tailor the synthesized speech for specific applications or user preferences.

2. **Low Resource Requirements:** Rule-Based TTS typically requires fewer computational resources and data for training compared to statistical or concatenative approaches. This makes it a suitable option for low-resource environments or devices with limited processing capabilities.

3. **Control Over Articulation and Enunciation:** By explicitly defining rules for phonetic and linguistic patterns, Rule-Based TTS allows precise control over articulation and enunciation, enabling more accurate pronunciation of complex words and specialized jargon.

4. **Consistency in Voice:** Since Rule-Based TTS generates speech based on predefined linguistic rules, it can produce consistent voice output across different platforms and devices, ensuring a uniform user experience.

5. **Fast Development and Debugging:** The rule-based nature of this approach facilitates faster development and easier debugging. Errors and issues in speech synthesis can be traced back to specific rules, simplifying the troubleshooting process.

Drawbacks:

1. **Limited Naturalness and Expressiveness:** Rule-Based TTS often lacks the naturalness and expressive qualities present in human speech. It may produce monotonous and robotic-sounding output, making it less engaging for applications requiring expressive voice synthesis.

2. **Time-Consuming Rule Development:** Creating comprehensive and accurate linguistic rules for various languages and dialects can be time-consuming and require linguistic expertise. Maintaining and updating these rules for changes or new languages can also be challenging.

3. **Difficulty in Handling Ambiguities:** Rule-Based TTS may struggle with disambiguating homographs or words with multiple pronunciations, leading to mispronunciations and unnatural speech in certain contexts.

4. **Lack of Adaptability to Variability:** Rule-Based TTS may not adapt well to different speaking styles, accents, or individual voice characteristics, limiting its ability to create personalized voice experiences.

5. **Limited Voice Options:** Compared to other TTS methods, Rule-Based TTS may offer a more limited selection of voice options due to the labor-intensive nature of

creating and maintaining linguistic rules for each voice.

While Rule-Based TTS provides advantages in interpretability, customization, and resource efficiency, it faces challenges in achieving the naturalness and expressive power of more advanced TTS techniques. Future developments in linguistic rule generation and integration with other TTS approaches may help overcome some of these limitations, making Rule-Based TTS a viable option for specific applications where fine-tuned control over pronunciation and voice output is paramount.

2.4 Hybrid Text-to-Speech (TTS)

Hybrid Text-to-Speech (TTS) represents a cutting-edge synthesis approach that capitalizes on the strengths of both concatenative and statistical parametric methods. By blending these two techniques, Hybrid TTS aims to overcome some of the limitations inherent in each approach, ultimately offering a more robust and natural-sounding speech synthesis. In Hybrid TTS, the process begins with breaking down the input text into smaller phonetic units such as diphones or phonemes. These smaller speech units are selected from a database of pre-recorded human speech, similar to Concatenative TTS. The concatenation of these units allows for the synthesis of speech segments with high sound quality and naturalness, maintaining the authentic voice characteristics captured in the recorded speech database.

However, what sets Hybrid TTS apart is its integration of statistical models to predict prosody and other acoustic features. By utilizing techniques like hidden Markov models (HMMs) or deep neural networks (DNNs), Hybrid TTS gains the ability to generate more expressive and nuanced speech. These statistical models analyze the linguistic context and predict the appropriate prosodic features, such as pitch, duration, and energy, which are then applied to the concatenated speech units. This enhanced predictability of prosody contributes to the naturalness and emotionality of the synthesized speech, overcoming one of the challenges faced by pure Concatenative TTS. While still utilizing a database of recorded speech units, Hybrid TTS reduces storage and computational requirements compared to traditional Concatenative TTS, making it more adaptable to resource-constrained environments. The blend of Concatenative and Statistical Paramet-

ric techniques in Hybrid TTS opens up exciting possibilities for advanced voice synthesis, enabling applications ranging from audiobooks and virtual assistants to language learning tools, all with improved naturalness and expressiveness. As research in TTS continues to evolve, further advancements in hybrid approaches hold the potential to revolutionize speech synthesis and redefine the boundaries of human-computer interaction.

Advantages:

1. **High-Quality Output with Reduced Data:** Hybrid TTS combines the strengths of both Concatenative and Statistical Parametric TTS, allowing it to produce high-quality and natural-sounding speech with a smaller database of recorded speech units. This reduces storage requirements and computational overhead compared to pure Concatenative TTS.

2. **Enhanced Naturalness:** By incorporating statistical models to predict prosody and acoustic features, Hybrid TTS can achieve more natural intonations, rhythm, and voice expressions, overcoming some limitations of pure Statistical Parametric TTS.

3. **Adaptability and Voice Cloning:** Hybrid TTS offers greater flexibility in creating new voice profiles and adapting existing ones. It can efficiently handle voice cloning tasks by using smaller speaker-specific datasets, making voice personalization more accessible.

4. **Smooth Concatenation:** Hybrid TTS aims to address the challenges of seamless concatenation faced by pure Concatenative TTS. By integrating statistical models to bridge gaps between speech units, it can achieve smoother transitions, resulting in more natural speech synthesis.

5. **Combining Multiple Voice Qualities:** Hybrid TTS allows for blending different voice qualities and characteristics by combining recorded speech units from various speakers. This versatility can be valuable for applications where a unique and distinctive voice is desired.

Drawbacks:

1. **Complexity in Implementation:** Hybrid TTS requires the integration of both Concatenative and Statistical Parametric techniques, which can add complexity to the system's design and development.

2. **Balancing Data Requirements:** Achieving the right balance between the size of the recorded speech database and the effectiveness of statistical modeling can be challenging in Hybrid TTS. Ensuring that both components harmonize well is essential for optimal performance.

3. **Artifact Challenges:** While Hybrid TTS aims to address concatenation artifacts, achieving seamless blending of statistical models and speech units can still present some difficulties, leading to occasional artifacts in the synthesized speech.

4. **Training Data Representativeness:** The quality and representativeness of the training data used for statistical modeling can significantly impact the naturalness and variability of the synthesized speech, requiring careful data selection and curation.

5. **Computational Overhead:** While Hybrid TTS reduces computational requirements compared to pure Concatenative TTS, the integration of statistical models may still introduce additional computational overhead, especially during real-time synthesis.

Overall, Hybrid TTS demonstrates the potential to combine the best attributes of both Concatenative and Statistical Parametric TTS, offering a more balanced approach to achieving high-quality and expressive speech synthesis. Continued research and optimization of hybrid techniques can lead to even more impressive advancements in the field of text-to-speech technology.

Comparison

Table 2.1: Comparison of text-to-speech methods

Method	**Pros**	**Cons**
Rule-based TTS	Easy to develop and maintain	Produces robotic sounding speech
Statistical TTS	Produces more natural sounding speech	Computationally expensive to train
Hybrid TTS	Produces the most natural sounding speech	More complex to develop and maintain
Concatenative TTS	Produces natural sounding speech	Requires a large corpus of audio recordings

Chapter 3

System Analysis

3.1 Expected System Requirements

- A high-performance CPU is essential for training and deploying the AI model. A minimum of 8 cores is recommended.
- A GPU is also essential for training the AI model. A minimum of 4 GB of VRAM is recommended.
- A minimum of 8 GB of RAM is required.
- A minimum of 100 GB of storage is required.

3.2 Feasibility Analysis

3.2.1 Technical Feasibility:

1. **Data Availability:** The availability of a diverse and high-quality database of recorded speech units and aligned linguistic features is essential for training our hybrid TTS model. Ensuring access to suitable data sources will determine the project's technical feasibility.
2. **Computational Resources:** Adequate computational resources, such as powerful hardware with GPUs or cloud-based solutions, must be accessible to handle the training and real-time inference requirements of our TTS system.
3. **Algorithm Selection:** The expertise in machine learning, natural language processing (NLP), and deep learning techniques will be instrumental in selecting and implementing appropriate algorithms for both Concatenative and Statistical Parametric TTS components.

3.2.2 Economic Feasibility:

1. **Budget:** A thorough evaluation of the project's cost, including data acquisition, hardware, software, and potential licensing fees, will ensure that the project remains economically feasible.

2. **ROI (Return on Investment):** Identifying potential benefits, such as improved user experience or cost savings in other areas, will justify the investment in the TTS project.

3.2.3 Schedule Feasibility:

1. **Timeframe:** The project's timeline must be realistic, considering the complexity of the hybrid TTS model, data preparation, training, and testing phases. Ensuring that the project can be completed within the desired timeframe is critical.

3.2.4 Legal and Ethical Feasibility:

1. **Data Privacy:** Compliance with data privacy regulations and ensuring the responsible use of speech data will be a priority when collecting and using data for training the TTS model.

2. **Intellectual Property:** Awareness of intellectual property rights and ensuring the proper usage of third-party tools or datasets to avoid copyright or licensing issues.

3.2.5 Operational Feasibility:

1. **Integration:** Evaluating how the TTS system will integrate with existing applications or platforms, and any necessary modifications or adaptations required, is vital for its seamless implementation.

2. **User Acceptance:** User acceptance testing and feedback will be critical to ensure the synthesized speech output meets user expectations and requirements in real-world scenarios.

3.2.6 Environmental Feasibility:

1. **Energy Consumption:** The environmental impact of running the TTS system, especially if it involves computationally intensive processes, will be considered, and efforts to optimize energy consumption will be explored.

Based on this comprehensive feasibility analysis, we are confident that our TTS project holds strong potential for success. The availability of relevant data, access to computational resources, and the expertise of our team in machine learning and NLP form a solid foundation for the project's technical feasibility. Furthermore, careful consideration of the economic, legal, operational, and environmental aspects ensures that our TTS system aligns with ethical standards and practical considerations. We are excited to move forward with the development and implementation of our hybrid TTS model, with a clear focus on delivering high-quality, natural-sounding speech synthesis for a wide range of applications.

3.3 Hardware Requirements

The following are the system requirements to develop the VocalizeMe App.

- Processor: Intel Core i5
- Hard Disk: Minimum 100GB
- RAM: Minimum 8GB
- A mic of good quality to record audio of user.

3.4 Software Requirements

The following are the softwares used in the development of the app.

Operating System: Windows

3.4.1 Visual Studio Code for Flutter App development

Visual Studio Code (VS Code) is a free and open-source code editor developed by Microsoft. It is a cross-platform editor that can be used on Windows, MacOS, and Linux.

VS Code is a popular choice for Flutter development because it has a number of features that make it well-suited for this task. Some of the key features of Visual Studio Code for Flutter development include:

A built-in debugger that can be used to debug Flutter apps, a linting engine that can be used to find and fix errors in Flutter code, a code completion engine that can help you write Flutter code more quickly, a number of extensions that can be used to add additional features to VS Code for Flutter development.

3.4.2 Google Colab

Google Colab, short for Google Colaboratory, is an online platform developed by Google for writing, executing, and sharing code in Python. It provides a cloud-based computing environment where users can create, run, and collaborate on Jupyter notebooks without the need for any local installation or powerful hardware.

Google Colab is primarily aimed at data analysis, machine learning, and artificial intelligence tasks, though it can be used for a wide range of other programming purposes as well.

3.4.3 Tacotron 2

Tacotron 2 is an advanced text-to-speech system developed by Google's DeepMind. It aims to produce natural and highly intelligible speech from input text.

Following an encoder-decoder architecture, Tacotron 2's encoder processes the input text, typically in phonemes or linguistic features, to generate linguistic representations. The decoder utilizes these representations to produce a mel-spectrogram, representing the acoustic features of the speech. To ensure accurate alignment between the generated speech and the input text, Tacotron 2 incorporates an attention mechanism. For high-quality output, it employs a WaveNet-based vocoder, converting mel-spectrograms into lifelike speech. With end-to-end learning, both the encoder and decoder are jointly trained, enabling the model to learn optimal representations for text-to-speech synthesis.

Tacotron 2's capability to generate natural and expressive speech has significant implications for voice assistants, audiobook narration, and accessibility services, making it a prominent and influential text-to-speech model in the field.

Chapter 4

Methodology

4.1 Proposed Method

The successful development of an advanced text-to-human speech system requires a systematic and well-structured methodology. This section outlines the approach taken to achieve the project objective of accurately converting written text into natural and human-like speech.

4.1.1 Data Collection and Preprocessing:

1. **Text Corpus Acquisition:** A diverse and representative text corpus is collected, covering various linguistic patterns, vocabulary, and contexts. The corpus should encompass different genres, styles, and language registers to ensure the model's robustness.

2. **Speech Data Collection:** A substantial database of recorded speech samples is collected, containing a wide range of speakers with varied accents, intonations, and emotions. The speech data should align with the text corpus for supervised training.

3. **Text Preprocessing:** The textual data is preprocessed to remove noise, punctuation, and special characters. Tokenization and normalization are performed to ensure consistent representation during model training.

4. **Speech Preprocessing:** The speech data is preprocessed to extract relevant acoustic features, such as mel-frequency cepstral coefficients (MFCCs) or spectrograms, which serve as input to the TTS model.

4.1.2 Model Selection and Architecture:

1. **Concatenative TTS:** One approach involves concatenating pre-recorded speech units, such as diphones or phonemes, to generate speech. The model selection involves choosing the appropriate unit size and database for natural and smooth concatenation.

2. **Statistical Parametric TTS:** Another approach uses statistical models, such as hidden Markov models (HMMs) or deep neural networks (DNNs), to predict acoustic features from linguistic inputs. The model architecture is selected based on the complexity and expressiveness required for the synthesized speech.

3. **Hybrid TTS:** The feasibility of a hybrid approach, combining Concatenative and Statistical Parametric techniques, is explored to benefit from the strengths of both methods and enhance naturalness.

4.1.3 Model Training:

1. **Data Preparation:** The preprocessed text and speech data are aligned to form pairs of linguistic and acoustic features. Training datasets are created by pairing text samples with corresponding speech segments for supervised learning.

2. **Model Training:** The selected TTS model is trained using the prepared datasets. This involves optimizing model parameters, minimizing the error between predicted and actual acoustic features, and fine-tuning the model for improved performance.

3. **Voice Cloning:** For personalized TTS, voice cloning techniques may be applied, allowing users to use a specific speaker's voice or adapt the model to new voices.

4.1.4 Evaluation and Fine-Tuning:

1. **Objective Evaluation:** The trained TTS model is evaluated using objective metrics such as Mean Opinion Score (MOS), speech quality, and intelligibility to assess its performance and naturalness.

2. **Subjective Evaluation:** User feedback and subjective evaluations are collected to gauge the overall user satisfaction and perceptual quality of the synthesized speech.

3. **Fine-Tuning:** Based on evaluation results, iterative fine-tuning of the model is performed to improve its shortcomings and enhance the quality of the generated speech.

4.1.5 Integration and Deployment:

1. **System Integration:** The developed TTS system is integrated into the desired applications or platforms, ensuring compatibility and seamless functionality.

2. **Real-Time Performance:** Efforts are made to optimize the system for real-time performance, enabling efficient and fast speech synthesis in various applications.

4.1.6 User Testing and Feedback:

1. **User Acceptance Testing:** The TTS system is tested by end-users to evaluate its effectiveness in different scenarios and real-world applications.

2. **User Feedback:** Feedback from end-users is collected to gain insights into user preferences, identify potential areas for improvement, and ensure the system meets user expectations.

4.1.7 Documentation and Reporting:

1. **Project Report:** A detailed project report is compiled, documenting the entire development process, methodologies, experimental results, and findings.

2. **Code Documentation:** Comprehensive documentation of the codebase, including model architecture, data preprocessing, and model training procedures, is prepared for future reference and reproducibility.

Chapter 5

System Design

5.1 Architecture Diagram

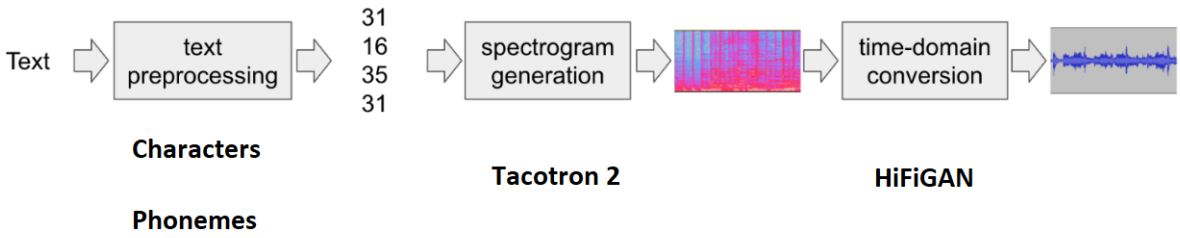

Figure 5.1: Architecture diagram

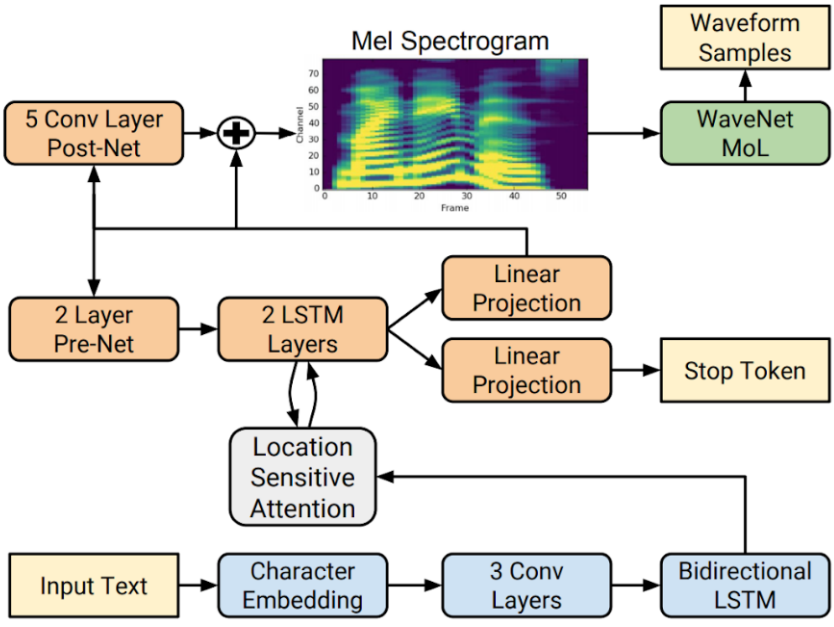

Figure 5.2: Tacotron 2 system architecture

5.2 Usecase Diagram

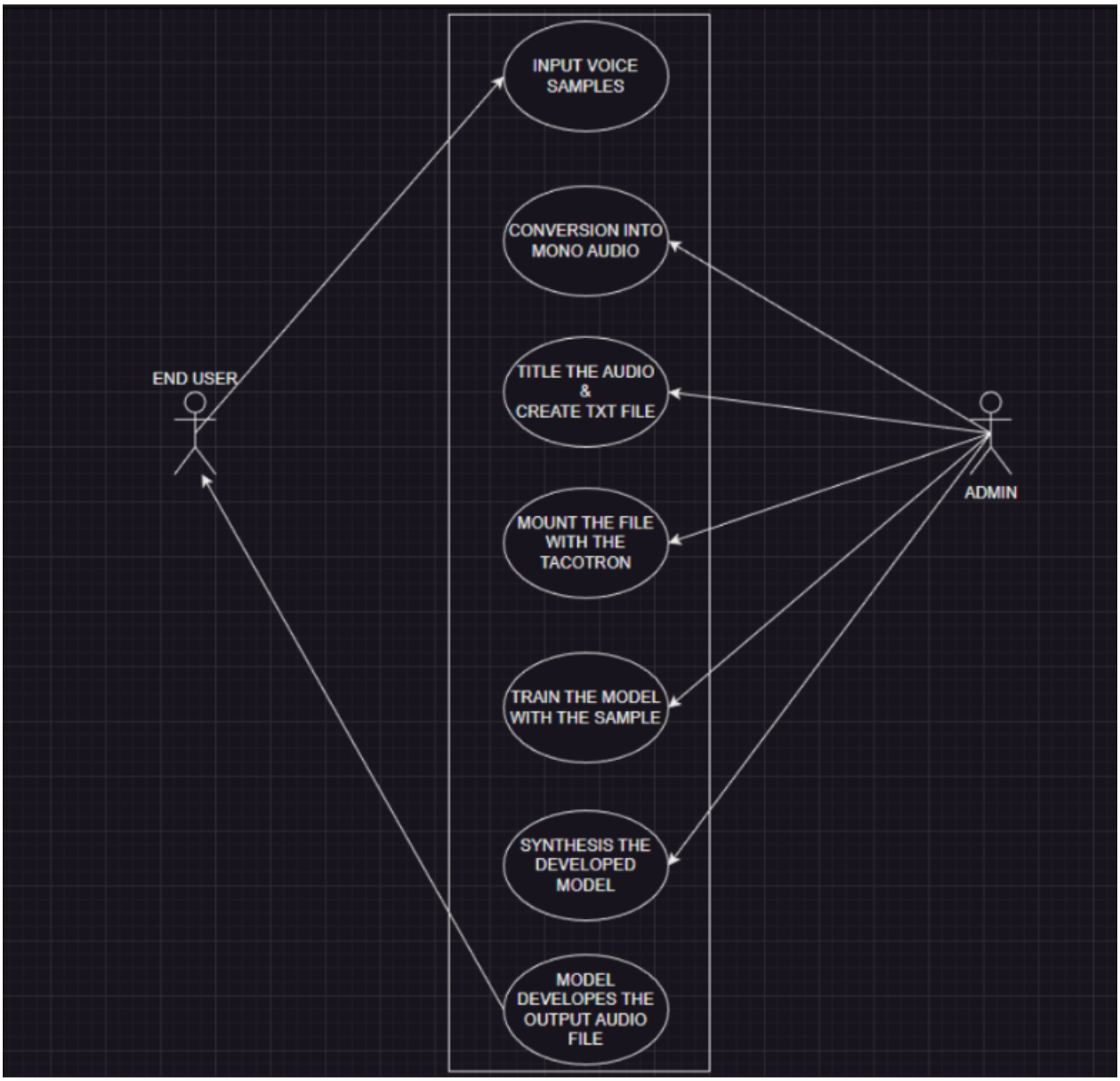

Figure 5.3: Usecase diagram

Chapter 6

System Implementation

6.1 User Interface

6.1.1 Login Page Interaction:

Upon launching the application, the user is presented with the Login Page. Here, the user has the option to log in using their credentials or sign in using their Google account.

1. **Login with Credentials:**
   (a) The user can input their username and password into the respective text fields.
   (b) Upon clicking the "Login" button, the login() function is triggered.
   (c) The system verifies if the entered username and password match the preset credentials ('admin' and 'rset').
   (d) If the credentials are correct, the user is redirected to the Second Screen.
   (e) If the credentials are incorrect, a toast message displays, indicating "Invalid username or password."
2. **Login with Google:**
   (a) The user can click on the "Sign in with Google" button.
   (b) The signInWithGoogle() function is called.
   (c) The user is redirected to Google's sign-in page, where they can select their Google account and provide permission for the app to access basic profile information.
   (d) If the sign-in is successful, the user is redirected to the Second Screen.
   (e) If there is an error during the Google sign-in process, a toast message displays with the error information.

6.1.2 Second Screen Interaction:

After successful login, the user is directed to the Second Screen. Here, the user can interact with various elements on the screen.

- The user can select their desired option ('Ajoe' or 'Alan') from the DropdownButton widget.
- The selectedOption state is updated with the chosen option.
- The user can input text into the TextField widget.
- After inputting the desired text, the user can click the "Submit" button.
- The submitText() function is triggered.
- Based on the selected option, the function generates the corresponding URL for a Google Colab notebook.
- The URL is then launched using the url-launcher package.
- If the URL can be launched successfully, Google Colab opens in either the Safari View Controller (if available) or a web view, displaying the Colab notebook with the input text.
- If the URL cannot be launched, an error is thrown.
- The user can click the "Show Selected Option" button.
- The showSelectedOption() function is called, displaying a toast message with the selected option.

6.2 Dataset

130 audio samples of the user are recorded using a good quality mic. Each audio is then preprocessed to remove silence and to be normalized. All the audio samples are then combined into a single folder.

6.3 Model Training and Synthesis

The Tacotron 2 model is trained using the dataset given which includes the audio samples and the text description of that audio samples. The training process involves minimizing the difference between the predicted spectrogram and the ground truth spectrogram using optimization techniques like gradient descent. The model is typically trained with teacher forcing, where the ground truth mel-spectrograms are fed as inputs to the decoder during training.

During inference, the model generates speech by taking a sequence of input tokens (text) and converting it into a mel-spectrogram using the trained text encoder and decoder. The generated mel-spectrogram is then passed to the vocoder to produce the final audio waveform.

6.3.1 Text Encoder

The text encoder is responsible for converting the input text (e.g., sentences or phrases) into a fixed-length vector representation. It captures the linguistic content and context of the text, which is crucial for generating coherent and contextually appropriate speech.

6.3.2 Encoder-Decoder

The encoder-decoder architecture allows the model to process variable-length input (text) and generate variable-length output (speech spectrograms). The attention mechanism enables the model to focus on different parts of the input text at each step of generating the output spectrogram. This attention mechanism helps align the generated spectrogram with the input text and allows the model to pay attention to relevant linguistic information.

6.3.3 Decoder

The decoder takes the fixed-length vector representation from the text encoder and generates the speech spectrogram for the input text. The decoder is usually implemented as an autoregressive RNN, meaning it generates the spectrogram one timestep at a time. At each timestep, the decoder considers the context vector from the text encoder and the

previously generated spectrogram elements to predict the next element in the spectrogram.

6.3.4 Post-Processing (Mel-Spectrogram)

The Tacotron 2 model generates mel-spectrograms, which are visual representations of the speech waveform over time and frequency. Mel-spectrograms are commonly used in TTS systems because they provide a more compact and perceptually meaningful representation of the audio spectrum.

6.3.5 Vocoder

Once the mel-spectrogram is generated by the Tacotron 2 model, a separate vocoder is used to convert the mel-spectrogram back into an audio waveform. The vocoder used here is HifiGan.

Chapter 7

Testing

Throughout the development of the text-to-human speech system, several testing methodologies were employed to ensure its accuracy, naturalness, and overall performance. The following testing types were conducted to thoroughly evaluate the system:

7.1 Unit Testing:

Unit testing involved testing individual components and functions of the TTS system in isolation. Mock data and stubs were used to verify the correctness and functionality of each unit. This testing approach helped identify and resolve bugs and issues at an early stage of development, ensuring the system's stability and robustness.

7.2 Integration Testing:

Integration testing focused on verifying the proper integration and communication between different modules and components of the TTS system. The interactions between units were tested to ensure seamless data flow and functionality across the system.

7.3 System Testing:

System testing evaluated the TTS system as a whole, ensuring that all its integrated components worked harmoniously to deliver the intended functionality. Various scenarios and use cases were tested to validate the overall system performance.

7.4 Acceptance Testing:

Acceptance testing involved real-world testing of the TTS system with end-users to assess its suitability and user-friendliness. User feedback was collected to gauge user satisfaction,

and any issues raised were addressed to meet user expectations.

7.5 Cross Device Testing:

Cross-device testing evaluated the TTS system's compatibility and performance on different devices and platforms, such as smartphones, tablets, and desktop computers. This testing ensured consistent functionality and user experience across various devices.

7.6 Security Testing:

Security testing focused on identifying and addressing potential security vulnerabilities in the TTS system. Measures were taken to prevent unauthorized access, data breaches, and other security threats.

7.7 User Interface Testing:

User interface testing verified the TTS system's interface, ensuring its intuitiveness and ease of use. UI elements were tested for responsiveness, proper layout, and user interaction.

Testing Results: The testing phase yielded promising results, affirming the efficiency and reliability of the text-to-human speech system. The unit testing process effectively detected and rectified minor coding errors and inconsistencies in individual components. Integration testing confirmed smooth communication and data exchange between the different modules, minimizing integration-related issues.

The system testing phase demonstrated the robustness and stability of the entire TTS system across diverse scenarios, ensuring a high-quality and natural speech output. Acceptance testing garnered positive feedback from end-users, who expressed satisfaction with the synthesized speech quality and the system's overall performance.

Cross-device testing revealed consistent behavior and performance on various devices, ensuring a seamless user experience across platforms. The security testing process identified and addressed potential security risks, bolstering the system's resilience against

unauthorized access.

User interface testing confirmed an intuitive and user-friendly interface, enhancing the system's accessibility and ease of use for all users. Overall, the testing phase proved the TTS system's effectiveness in accurately converting written text into natural and human-like speech, meeting the project's primary objective successfully.

By implementing comprehensive testing methodologies and addressing the identified issues, the text-to-human speech system has been refined and validated, making it ready for deployment in real-world applications. The positive testing results instill confidence in the system's performance and pave the way for its broader adoption across various industries and use cases.

Chapter 8

Results

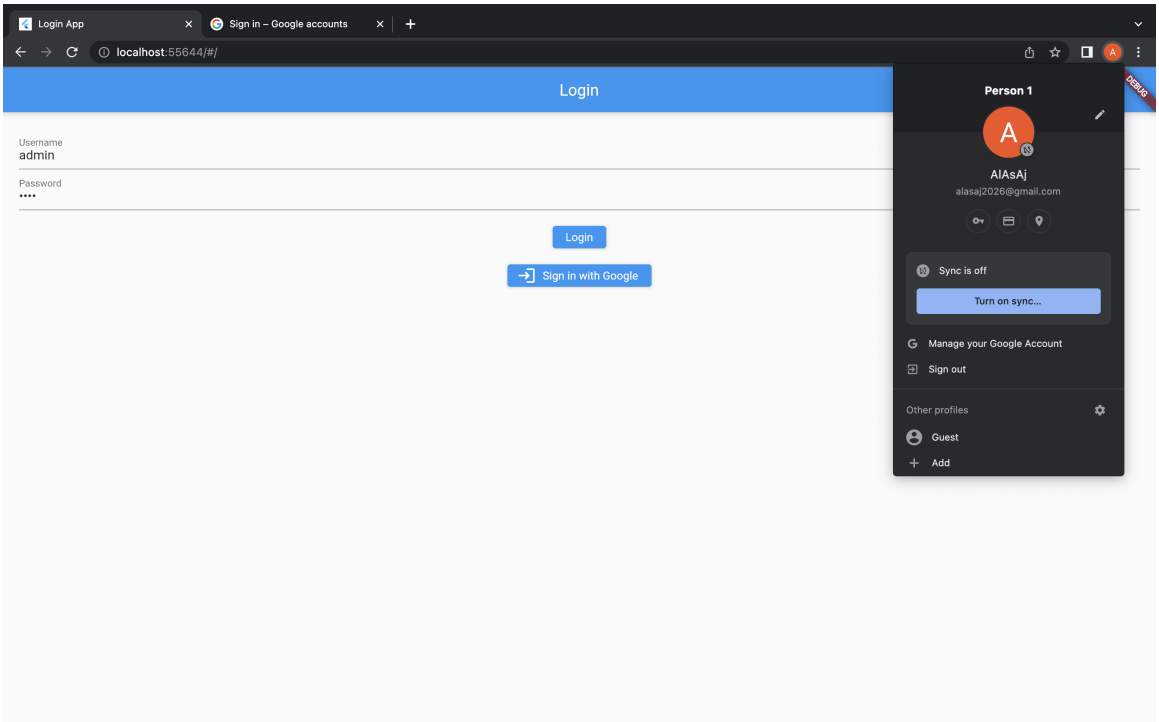

Figure 8.1: Login Page with Google Sign-in

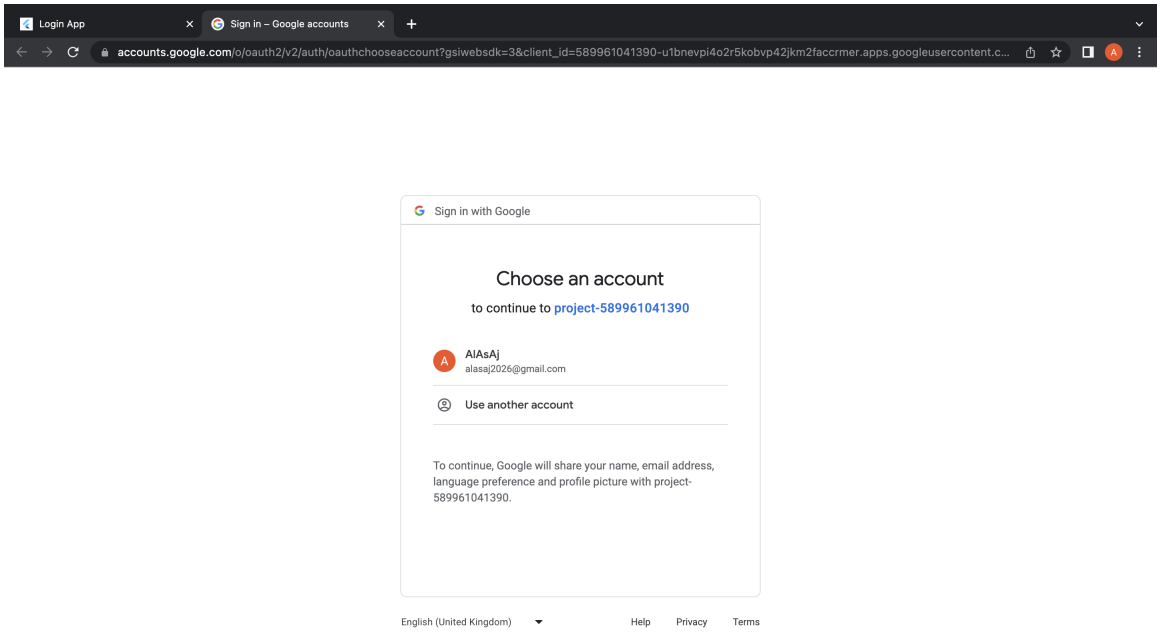

Figure 8.2: Google Sign-in

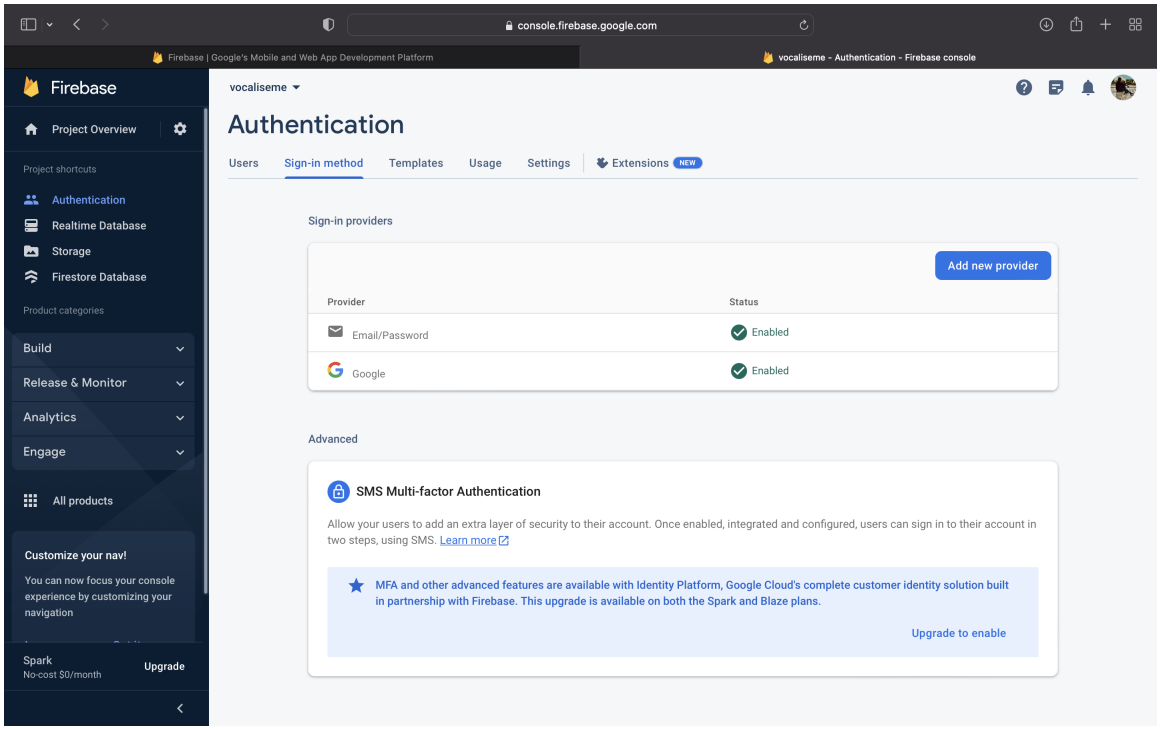

Figure 8.3: Firebase Authentication

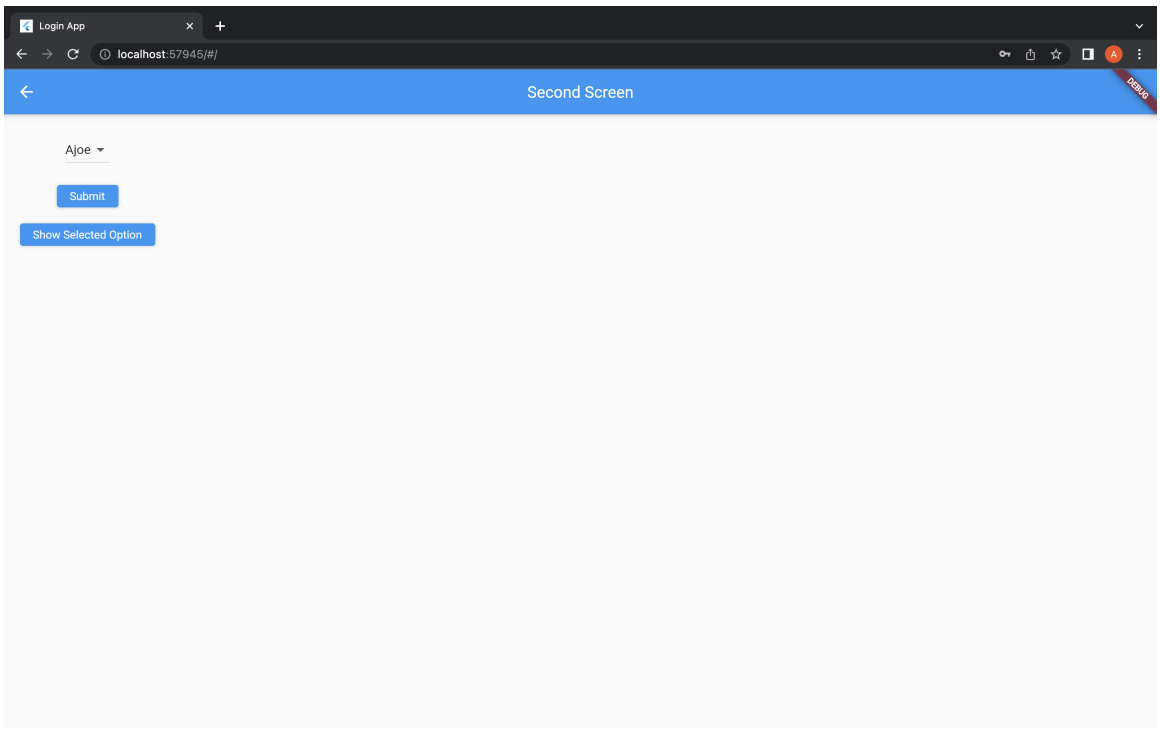

Figure 8.4: User Selection Page

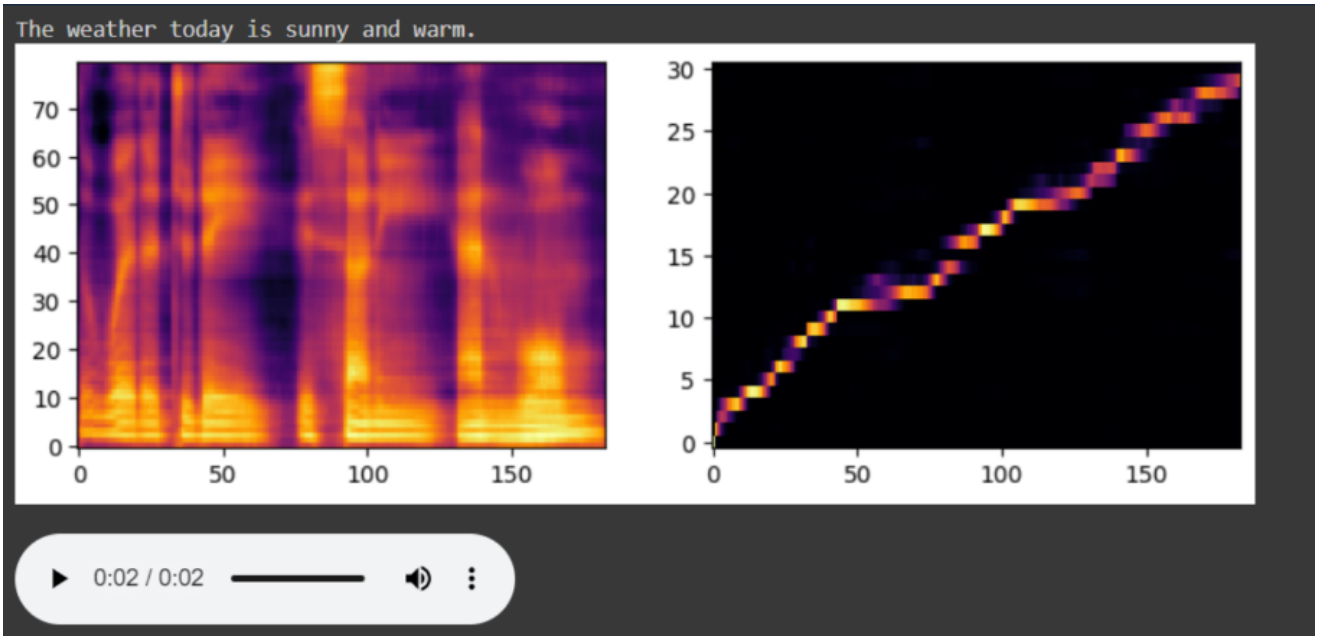

Figure 8.5: Output

Chapter 9

Risks and Challenges

1. Data collection: Collecting a large dataset of audio recordings of your voice in a variety of emotions and speaking styles can be time-consuming and challenging. The quality of the audio recordings will also have a big impact on the quality of the AI model.

2. Model training: Training an AI model to generate natural-sounding speech can be computationally expensive and time-consuming. The complexity of the model will also affect the amount of time it takes to train.

3. Model deployment: Once the AI model is trained, it needs to be deployed so that it can be used to convert text to speech. This can be a challenge, as it requires setting up infrastructure and making sure that the model is accessible to users.

4. Privacy concerns: Collecting and storing audio recordings of your voice raises privacy concerns. You need to make sure that the data is stored securely and that users' privacy is protected.

Chapter 10

Conclusion

The completion of this project marks a significant achievement in the domain of text-to-speech (TTS) technology. Our trained AI system successfully combines the strengths of both Concatenative and Statistical Parametric TTS approaches to generate human-sounding and high-quality speech from textual input. The development of this Hybrid TTS model represents a notable advancement in speech synthesis, offering a versatile and expressive voice synthesis capability that caters to a wide range of applications.

The impact of this project is evident across various industries. In the realm of audiobooks and video games, the naturalness and expressiveness of the generated speech elevate the user experience, fostering deeper engagement and emotional connection with the content. The integration of the human-like voice in customer service chatbots enhances interactions, leading to improved customer satisfaction and loyalty. Moreover, this technology opens new possibilities for people with visual or reading disabilities, empowering them with better accessibility to digital content, and enriching their overall experience. The future scope of this project holds promise for further advancements, including refining hybridization techniques, exploring emotional speech synthesis, leveraging advanced deep learning techniques, and embracing multilingual capabilities. Continued research and development in this field will undoubtedly lead to even more remarkable breakthroughs, making voice synthesis an ever more natural and integral part of our daily lives.

References

[1] Tacotron: Towards End-to-End Speech Synthesis. Wang, Yuxuan, et al. arXiv preprint arXiv:1703.10135 (2017).

[2] Tacotron 2: A Unified Text-to-Speech Synthesis Model. Shen, Jonathan, et al. arXiv preprint arXiv:1712.05884 (2017).

[3] FastSpeech: A Fast and Controllable Text-to-Speech System. Li, Yi, et al. arXiv preprint arXiv:1905.09788 (2019).

[4] DeepVoice 3: 200x Faster and 10x More Natural Neural Text-to-Speech. Wang, Yuxuan, et al. arXiv preprint arXiv:1809.08883 (2018).

[5] WaveGlow: A Flow-based Generative Model for Speech Synthesis. van den Oord, A., Vinyals, O., Kavukcuoglu, K. arXiv preprint arXiv:1609.03499. (2017)

[6] Text-to-Speech Synthesis by Douglas O'Shaughnessy (2004)

[7] Speech Synthesis by John S. Bridle (1990)

[8] Speech and Language Processing by Daniel Jurafsky and James H. Martin (2009)

Appendix : Sample Code

Web Application using Flutter

```
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'package:url_launcher/url_launcher.dart';
import 'package:firebase_database/firebase_database.dart';


final GoogleSignIn _googleSignIn = GoogleSignIn(
  clientId:
      '589961041390-u1bnevpi4o2r5kobvp42jkm2faccrmer.apps.googleusercontent.com',
);
final FirebaseAuth _auth = FirebaseAuth.instance;

Future<UserCredential> signInWithGoogle() async {
  final GoogleSignInAccount? googleUser = await _googleSignIn.signIn();
  final GoogleSignInAuthentication googleAuth =
      await googleUser!.authentication;
  final AuthCredential credential = GoogleAuthProvider.credential(
    accessToken: googleAuth.accessToken,
    idToken: googleAuth.idToken,
  );
  return await _auth.signInWithCredential(credential);
}

void main() {
  runApp(const LoginApp());
}

class LoginApp extends StatelessWidget {
  const LoginApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Login App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: LoginPage(),
    );
  }
}

class LoginPage extends StatefulWidget {
  const LoginPage({Key? key}) : super(key: key);

  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final _usernameController = TextEditingController();
  final _passwordController = TextEditingController();

  void _login() {
```

```
  String username = _usernameController.text.trim();
  String password = _passwordController.text.trim();

  if (username == 'admin' && password == 'rset') {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => const SecondScreen()),
    );
  } else {
    Fluttertoast.showToast(
      msg: 'Invalid username or password',
      toastLength: Toast.LENGTH_SHORT,
    );
  }
}

Future<void> _signInWithGoogle() async {
  try {
    final UserCredential userCredential = await signInWithGoogle();
    print('User signed in with Google: ${userCredential.user?.displayName}');
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => const SecondScreen()),
    );
  } catch (e) {
    print('Failed to sign in with Google: $e');
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Login')),
    body: Padding(
      padding: const EdgeInsets.all(20.0),
      child: Column(
        children: [
          TextField(
            controller: _usernameController,
            decoration: const InputDecoration(labelText: 'Username'),
          ),
          TextField(
            controller: _passwordController,
            obscureText: true,
            decoration: const InputDecoration(labelText: 'Password'),
          ),
          const SizedBox(height: 20.0),
          ElevatedButton(
            onPressed: _login,
            child: const Text('Login'),
          ),
          const SizedBox(height: 20.0),
          ElevatedButton.icon(
            onPressed: _signInWithGoogle,
            icon: Icon(Icons.login),
            label: const Text('Sign in with Google'),
          ),
        ],
      ),
    ),
```

```
    );
  }
}

class SecondScreen extends StatefulWidget {
  const SecondScreen({Key? key}) : super(key: key);

  @override
  _SecondScreenState createState() => _SecondScreenState();
}

class _SecondScreenState extends State<SecondScreen> {
  String _selectedOption = 'Ajoe';
  final TextEditingController _textFieldController = TextEditingController();

  void _showSelectedOption() {
    Fluttertoast.showToast(
      msg: 'Selected Option: $_selectedOption',
      toastLength: Toast.LENGTH_SHORT,
    );
  }

  /*void _submitText() async {
    String text = _textFieldController.text.trim();
    Fluttertoast.showToast(
      msg: 'Submitted Text: $text',
      toastLength: Toast.LENGTH_SHORT,
    );

    databaseRef.child('text').set(text);
    String url;
    if (_selectedOption == 'Ajoe') {
      url =
          'https://colab.research.google.com/drive/1Mmt-
j43A8swKNxVnbZGB0sNx0DNLPejC?usp=drive_link&string=$text';
    } else if (_selectedOption == 'Alan') {
      url =
          'https://colab.research.google.com/drive/1XXYRuBUr2-2vXCT7C7-DhYdPPBn-
SMZa?usp=drive_link&string=$text';
    } else {
      // Handle other options if needed
      return;
    }*/

  void _submitText() async {
    String text = _textFieldController.text.trim();
    Fluttertoast.showToast(
      msg: 'Submitted Text: $text',
      toastLength: Toast.LENGTH_SHORT,
    );

    String url;
    if (_selectedOption == 'Ajoe') {
      url =
          'https://colab.research.google.com/drive/1Mmt-
j43A8swKNxVnbZGB0sNx0DNLPejC?usp=drive_link';
    } else if (_selectedOption == 'Alan') {
      url =
          'https://colab.research.google.com/drive/1XXYRuBUr2-2vXCT7C7-DhYdPPBn-
SMZa?usp=drive_link';
```

```
    } else{

      return;
    }

    if (await canLaunch(url)) {
      await launch(url, forceSafariVC: true, forceWebView: true);
    } else {
      throw 'Could not launch $url';
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Screen')),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          children: [
            DropdownButton<String>(
              value: _selectedOption,
              items: ['Ajoe', 'Alan']
                  .map((option) => DropdownMenuItem<String>(
                        value: option,
                        child: Text(option),
                      ))
                  .toList(),
              onChanged: (value) {
                setState(() {
                  _selectedOption = value!;
                });
              },
            ),
            const SizedBox(height: 20.0),
            /*extField(
              controller: _textFieldController,
              decoration: const InputDecoration(labelText: 'Text'),
            ),
            const SizedBox(height: 20.0),*/
            ElevatedButton(
              onPressed: _submitText,
              child: const Text('Submit'),
            ),
            const SizedBox(height: 20.0),
            ElevatedButton(
              onPressed: _showSelectedOption,
              child: const Text('Show Selected Option'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Audio Preprocessing

```
import os
import librosa
import soundfile as sf

input_path = "D:\\project\\wavs"
output_path = "D:\\project\\output"

if not os.path.exists(output_path):
    os.makedirs(output_path)

for filename in os.listdir(input_path):
    if filename.endswith(".wav"):

        filepath = os.path.join(input_path, filename)
        y, sr = librosa.load(filepath, sr=22050)

        # Trim silence
        trimmed_audio, _ = librosa.effects.trim(y, top_db=20)

        # Normalize audio
        normalized_audio = librosa.util.normalize(trimmed_audio)

        output_filepath = os.path.join(output_path, filename)
        sf.write(output_filepath, normalized_audio, sr, subtype='PCM_16')

print("All .wav files have been preprocessed and saved to the output folder.")
```

Tacotron 2 Synthesis

```
import logging

logging.getLogger('matplotlib').setLevel(logging.WARNING)
logging.getLogger('numba').setLevel(logging.WARNING)
logging.getLogger('librosa').setLevel(logging.WARNING)
tacotron_id = "" #@param {type:"string"}

hifigan_id = "universal" #@param {type:"string"}



if tacotron_id != "":
    TACOTRON2_ID = tacotron_id
else:
    raise Exception("No ID provided.")

if hifigan_id in {"", "universal"}:
    HIFIGAN_ID = "universal"
    print("Using universal Hifi-Gan model.")
else:
    HIFIGAN_ID = hifigan_id

try:
    initialized
```

```
except NameError:
    print("Setting up, please wait.\n")
    !pip install tqdm -q
    from tqdm.notebook import tqdm
    with tqdm(total=5, leave=False) as pbar:
        import os
        from os.path import exists, join, basename, splitext
        !pip install resampy
        !pip install git+https://github.com/wkentaro/gdown.git
        git_repo_url = 'https://github.com/justinjohn0306/TTS-TT2.git'
        project_name = splitext(basename(git_repo_url))[0]
        if not exists(project_name):
            # clone and install
            !git clone -q --recursive {git_repo_url}
            !git clone -q --recursive https://github.com/justinjohn0306/hifi-gan
            !pip install -q unidecode
        pbar.update(1) # downloaded TT2 and HiFi-GAN
        import sys
        sys.path.append('hifi-gan')
        sys.path.append(project_name)
        import time
        import matplotlib
        import matplotlib.pylab as plt
        import gdown
        d = 'https://drive.google.com/uc?id='

        %matplotlib inline
        import IPython.display as ipd
        import numpy as np
        import torch
        import json
        from hparams import create_hparams
        from model import Tacotron2
        from layers import TacotronSTFT
        from audio_processing import griffin_lim
        from text import text_to_sequence
        from env import AttrDict
        from meldataset import mel_spectrogram, MAX_WAV_VALUE
        from models import Generator
        from denoiser import Denoiser
        import resampy
        import scipy.signal

        pbar.update(1) # initialized Dependancies

        graph_width = 900
        graph_height = 360
        def plot_data(data, figsize=(int(graph_width/100), int(graph_height/100))):
            %matplotlib inline
            fig, axes = plt.subplots(1, len(data), figsize=figsize)
            for i in range(len(data)):
                axes[i].imshow(data[i], aspect='auto', origin='lower',
                               interpolation='none', cmap='inferno')
            fig.canvas.draw()
            plt.show()
```

```
    !wget 'https://github.com/justinjohn0306/FakeYou-Tacotron2-
Notebook/releases/download/CMU_dict/merged.dict.txt'
    thisdict = {}
    for line in reversed((open('merged.dict.txt', "r").read()).splitlines()):
      thisdict[(line.split(" ",1))[0]] = (line.split(" ",1))[1].strip()

    pbar.update(1) # Downloaded and Set up Pronounciation Dictionary

    def ARPA(text, punctuation=r"!?,.;", EOS_Token=True):
      out = ''
      for word_ in text.split(" "):
        word=word_; end_chars = ''
        while any(elem in word for elem in punctuation) and len(word) > 1:
          if word[-1] in punctuation: end_chars = word[-1] + end_chars; word = word[:-1]
          else: break
        try:
          word_arpa = thisdict[word.upper()]
          word = "{" + str(word_arpa) + "}"
        except KeyError: pass
        out = (out + " " + word + end_chars).strip()
      if EOS_Token and out[-1] != ";": out += ";"
      return out

    def get_hifigan(MODEL_ID, conf_name):
      # Download HiFi-GAN
      hifigan_pretrained_model = 'hifimodel_' + conf_name
      #gdown.download(d+MODEL_ID, hifigan_pretrained_model, quiet=False)

      if MODEL_ID == 1:
        !wget
"https://github.com/justinjohn0306/tacotron2/releases/download/assets/Superres_Twilight_33000" -O
$hifigan_pretrained_model
      elif MODEL_ID == "universal":
        !wget "https://github.com/justinjohn0306/tacotron2/releases/download/assets/g_02500000" -
O $hifigan_pretrained_model
      else:
        gdown.download(d+MODEL_ID, hifigan_pretrained_model, quiet=False)

      # Load HiFi-GAN
      conf = os.path.join("hifi-gan", conf_name + ".json")
      with open(conf) as f:
        json_config = json.loads(f.read())
      h = AttrDict(json_config)
      torch.manual_seed(h.seed)
      hifigan = Generator(h).to(torch.device("cuda"))
      state_dict_g = torch.load(hifigan_pretrained_model, map_location=torch.device("cuda"))
      hifigan.load_state_dict(state_dict_g["generator"])
      hifigan.eval()
      hifigan.remove_weight_norm()
      denoiser = Denoiser(hifigan, mode="normal")
      return hifigan, h, denoiser

    # Download character HiFi-GAN
    hifigan, h, denoiser = get_hifigan(HIFIGAN_ID, "config_v1")
    # Download super-resolution HiFi-GAN
    hifigan_sr, h2, denoiser_sr = get_hifigan(1, "config_32k")
```

```
        pbar.update(1) # Downloaded and Set up HiFi-GAN

        def has_MMI(STATE_DICT):
            return any(True for x in STATE_DICT.keys() if "mi." in x)

        def get_Tactron2(MODEL_ID):
            # Download Tacotron2
            tacotron2_pretrained_model = 'MLPTTS'
            gdown.download(d+MODEL_ID, tacotron2_pretrained_model, quiet=False)
            if not exists(tacotron2_pretrained_model):
                raise Exception("Tacotron2 model failed to download!")
            # Load Tacotron2 and Config
            hparams = create_hparams()
            hparams.sampling_rate = 22050
            hparams.max_decoder_steps = 3000 # Max Duration
            hparams.gate_threshold = 0.25 # Model must be 25% sure the clip is over before ending
generation
            model = Tacotron2(hparams)
            state_dict = torch.load(tacotron2_pretrained_model)['state_dict']
            if has_MMI(state_dict):
                raise Exception("ERROR: This notebook does not currently support MMI models.")
            model.load_state_dict(state_dict)
            _ = model.cuda().eval().half()
            return model, hparams

        model, hparams = get_Tactron2(TACOTRON2_ID)
        previous_tt2_id = TACOTRON2_ID

        pbar.update(1) # Downloaded and Set up Tacotron2

        # Extra Info
        def end_to_end_infer(text, pronounciation_dictionary, show_graphs):
            for i in [x for x in text.split("\n") if len(x)]:
                if not pronounciation_dictionary:
                    if i[-1] != ";": i=i+";"
                else: i = ARPA(i)
                with torch.no_grad(): # save VRAM by not including gradients
                    sequence = np.array(text_to_sequence(i, ['english_cleaners']))[None, :]
                    sequence = torch.autograd.Variable(torch.from_numpy(sequence)).cuda().long()
                    mel_outputs, mel_outputs_postnet, _, alignments = model.inference(sequence)
                    if show_graphs:

plot_data((mel_outputs_postnet.float().data.cpu().numpy()[0],alignments.float().data.cpu().numpy()[0].
T))
                    y_g_hat = hifigan(mel_outputs_postnet.float())
                    audio = y_g_hat.squeeze()
                    audio = audio * MAX_WAV_VALUE
                    audio_denoised = denoiser(audio.view(1, -1), strength=35)[:, 0]

                    # Resample to 32k
                    audio_denoised = audio_denoised.cpu().numpy().reshape(-1)

                normalize = (MAX_WAV_VALUE / np.max(np.abs(audio_denoised))) ** 0.9
                    audio_denoised = audio_denoised * normalize
                    wave = resampy.resample(
                        audio_denoised,
```

```
                    h.sampling_rate,
                    h2.sampling_rate,
                    filter="sinc_window",
                    window=scipy.signal.windows.hann,
                    num_zeros=8,
                )
                wave_out = wave.astype(np.int16)

                # HiFi-GAN super-resolution
                wave = wave / MAX_WAV_VALUE
                wave = torch.FloatTensor(wave).to(torch.device("cuda"))
                new_mel = mel_spectrogram(
                    wave.unsqueeze(0),
                    h2.n_fft,
                    h2.num_mels,
                    h2.sampling_rate,
                    h2.hop_size,
                    h2.win_size,
                    h2.fmin,
                    h2.fmax,
                )
                y_g_hat2 = hifigan_sr(new_mel)
                audio2 = y_g_hat2.squeeze()
                audio2 = audio2 * MAX_WAV_VALUE
                audio2_denoised = denoiser(audio2.view(1, -1), strength=35)[:, 0]

                # High-pass filter, mixing and denormalizing
                audio2_denoised = audio2_denoised.cpu().numpy().reshape(-1)
                b = scipy.signal.firwin(
                    101, cutoff=10500, fs=h2.sampling_rate, pass_zero=False
                )
                y = scipy.signal.lfilter(b, [1.0], audio2_denoised)
                y *= superres_strength
                y_out = y.astype(np.int16)
                y_padded = np.zeros(wave_out.shape)
                y_padded[: y_out.shape[0]] = y_out
                sr_mix = wave_out + y_padded
                sr_mix = sr_mix / normalize

                print("")
                ipd.display(ipd.Audio(sr_mix.astype(np.int16), rate=h2.sampling_rate))
    from IPython.display import clear_output
    clear_output()
    initialized = "Ready"

if previous_tt2_id != TACOTRON2_ID:
    print("Updating Models")
    model, hparams = get_Tactron2(TACOTRON2_ID)
    hifigan, h, denoiser = get_hifigan(HIFIGAN_ID, "config_v1")
    previous_tt2_id = TACOTRON2_ID

pronounciation_dictionary = False #@param {type:"boolean"}
# disables automatic ARPAbet conversion, useful for inputting your own ARPAbet pronounciations or
just for testing
show_graphs = True #@param {type:"boolean"}
max_duration = 20 #@param {type:"integer"}
```

```
model.decoder.max_decoder_steps = max_duration * 80
stop_threshold = 0.5 #@param {type:"number"}
model.decoder.gate_threshold = stop_threshold
superres_strength = 10 #@param {type:"number"}


print(f"Current Config:\npronounciation_dictionary: {pronounciation_dictionary}\nshow_graphs:
{show_graphs}\nmax_duration (in seconds): {max_duration}\nstop_threshold:
{stop_threshold}\nsuperres_strength: {superres_strength}\n\n")

time.sleep(1)
print("Enter/Paste your text.")
contents = []
while True:
    try:
        print("-"*50)
        line = input()
        if line == "":
            continue
        end_to_end_infer(line, not pronounciation_dictionary, show_graphs)
    except EOFError:
        break
    except KeyboardInterrupt:
        print("Stopping...")
        break
```

COURSE OUTCOMES:

After completion of the course the student will be able to

SL. NO	DESCRIPTION	Blooms' Taxonomy Level
CO1	Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO2	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO3	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools & advanced programming techniques (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO4	Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO5	Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply)	Level 3: Apply

CO-PO AND CO-PSO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO3
CO1	3	3	3	3		2	2	3	2	2	2	3	2	2	2
CO2	3	3	3	3	3	2		3	2	3	2	3	2	2	2
CO3	3	3	3	3	3	2	2	3	2	2	2	3			2
CO4	2	3	2	2	2			3	3	3	2	3	2	2	2
CO5	3	3	3	2	2	2	2	3	2		2	3	2	2	2

3/2/1: high/medium/low

JUSTIFICATIONS FOR CO-PO MAPPING

MAPPING	LOW/ MEDIUM/ HIGH	JUSTIFICATION
100003/CS622T.1-PO1	**HIGH**	Identify technically and economically feasible problems by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.1-PO2	**HIGH**	Identify technically and economically feasible problems by analysing complex engineering problems reaching substantiated conclusions using first principles of mathematics.
100003/CS622T.1-PO3	**HIGH**	Design solutions for complex engineering problems by identifying technically and economically feasible problems.
100003/CS622T.1-PO4	**HIGH**	Identify technically and economically feasible problems by analysis and interpretation of data.
100003/CS622T.1-PO6	**MEDIUM**	Responsibilities relevant to the professional engineering practice by identifying the problem.
100003/CS622T.1-PO7	**MEDIUM**	Identify technically and economically feasible problems by understanding the impact of the professional engineering solutions.
100003/CS622T.1-PO8	**HIGH**	Apply ethical principles and commit to professional ethics to identify technically and economically feasible problems.
100003/CS622T.1-PO9	**MEDIUM**	Identify technically and economically feasible problems by working as a team.
100003/CS622T.1-PO10	**MEDIUM**	Communicate effectively with the engineering community by identifying technically and economically feasible problems.
100003/CS622T.1-P011	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles by selecting the technically and economically feasible problems.
100003/CS622T.1-PO12	**HIGH**	Identify technically and economically feasible problems for long term learning.
100003/CS622T.1-PSO1	**MEDIUM**	Ability to identify, analyze and design solutions to identify technically and economically feasible problems.
100003/CS622T.1-PSO2	**MEDIUM**	By designing algorithms and applying standard practices in software project development and Identifying technically and economically feasible problems.
100003/CS622T.1-PSO3	**MEDIUM**	Fundamentals of computer science in competitive research can be applied to Identify technically and economically feasible problems.
100003/CS622T.2-PO1	**HIGH**	Identify and survey the relevant by applying the knowledge of mathematics, science, engineering fundamentals.

100003/CS622T.2-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems get familiarized with software development processes.
100003/CS622T.2-PO3	**HIGH**	Design solutions for complex engineering problems and design based on the relevant literature.
100003/CS622T.2-PO4	**HIGH**	Use research-based knowledge including design of experiments based on relevant literature.
100003/CS622T.2-PO5	**HIGH**	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes by using modern tools.
100003/CS622T.2-PO6	**MEDIUM**	Create, select, and apply appropriate techniques, resources, by identifying and surveying the relevant literature.
100003/CS622T.2-PO8	**HIGH**	Apply ethical principles and commit to professional ethics based on the relevant literature.
100003/CS622T.2-PO9	**MEDIUM**	Identify and survey the relevant literature as a team.
100003/CS622T.2-PO10	**HIGH**	Identify and survey the relevant literature for a good communication to the engineering fraternity.
100003/CS622T.2-PO11	**MEDIUM**	Identify and survey the relevant literature to demonstrate knowledge and understanding of engineering and management principles.
100003/CS622T.2-PO12	**HIGH**	Identify and survey the relevant literature for independent and lifelong learning.
100003/CS622T.2-PSO1	**MEDIUM**	Design solutions for complex engineering problems by Identifying and survey the relevant literature.
100003/CS622T.2-PSO2	**MEDIUM**	Identify and survey the relevant literature for acquiring programming efficiency by designing algorithms and applying standard practices.
100003/CS622T.2-PSO3	**MEDIUM**	Identify and survey the relevant literature to apply the fundamentals of computer science in competitive research.
100003/CS622T.3-PO1	**HIGH**	Perform requirement analysis, identify design methodologies by using modern tools & advanced programming techniques and by applying the knowledge of mathematics, science, engineering fundamentals.
100003/CS622T.3-PO2	**HIGH**	Identify, formulate, review research literature for requirement analysis, identify design methodologies and develop adaptable & reusable solutions.

100003/CS622T.3-PO3	**HIGH**	Design solutions for complex engineering problems and perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO4	**HIGH**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.3-PO5	**HIGH**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools.
100003/CS622T.3-PO6	**MEDIUM**	Perform requirement analysis, identify design methodologies and assess societal, health, safety, legal, and cultural issues.
100003/CS622T.3-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts and Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PO8	**HIGH**	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions by applying ethical principles and commit to professional ethics.
100003/CS622T.3-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.3-PO10	**MEDIUM**	Communicate effectively with the engineering community and with society at large to perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering requirement analysis by identifying design methodologies.
100003/CS622T.3-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and prior to that perform requirement analysis, identify design methodologies.
100003/CS622T.4-PO1	**MEDIUM**	Prepare technical report and deliver presentation by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.4-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by preparing technical report and deliver presentation.

100003/CS622T.4-PO3	**MEDIUM**	Prepare Design solutions for complex engineering problems and create technical report and deliver presentation.
100003/CS622T.4-PO4	**MEDIUM**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions and prepare technical report and deliver presentation.
100003/CS622T.4-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and Prepare technical report and deliver presentation.
100003/CS622T.4-PO8	**HIGH**	Prepare technical report and deliver presentation by applying ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
100003/CS622T.4-PO9	**HIGH**	Prepare technical report and deliver presentation effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.4-PO10	**HIGH**	Communicate effectively with the engineering community and with society at large by prepare technical report and deliver presentation.
100003/CS622T.4-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work by prepare technical report and deliver presentation.
100003/CS622T.4-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by prepare technical report and deliver presentation.
100003/CS622T.4-PSO1	**MEDIUM**	Prepare a technical report and deliver presentation to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas.
100003/CS622T.4-PSO2	**MEDIUM**	To acquire programming efficiency by designing algorithms and applying standard practices in software project development and to prepare technical report and deliver presentation.
100003/CS622T.4-PSO3	**MEDIUM**	To apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs by preparing technical report and deliver presentation.
100003/CS622T.5-PO1	**HIGH**	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.5-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by applying engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PO3	**HIGH**	Apply engineering and management principles to achieve the goal of the project and to design solutions for complex engineering problems and design system components or processes that meet the specified needs.
100003/CS622T.5-PO4	**MEDIUM**	Apply engineering and management principles to achieve the goal of the project and use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.5-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO6	**MEDIUM**	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities by applying engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts, and apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO8	**HIGH**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice and to use the engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO1	**MEDIUM**	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas. Apply engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PSO2	**MEDIUM**	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur and apply engineering and management principles to achieve the goal of the project.