Universitatea Tehnică din Cluj – Napoca

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

How to Convert Text to Speech

(Speech Synthesis using Transformers)

Studenta: *Ioana Giulia HOSSU*

Nume îndrumător:

Facultate: *Electronică, Telecomunicații și Tehnologia Informației*

Specializare: *Tehnologii Multimedia*

An: *I*

Disciplina: *ASRSV*

Data predării proiectului (zi/lună/an): *16/01/2024*

Cuprins

1 Descrierea temei

1.1 Sinteza vocală

Sinteza vocală se referă la formarea artificială a vorbirii umane. Scopul principal a fost ca oamenii orbi să aibă un sistem care să le citească din cărți. Provocările sintezei vocale presupun implementarea inteligibilității și a naturaleții. Astfel, putem analiza procesul de comunicare între doi vorbitori, urmărind Figura 1. Sunt trei tipuri de reprezentari: înțelesul (ceea ce se dorește a fi transmis), mesajul și semnalul, și patru sisteme: generarea (de la înțeles la mesaj), codarea (de la mesaj la semnal), decodarea (de la semnal la mesaj) și înțelegerea (de la mesaj la înțeles) [1].

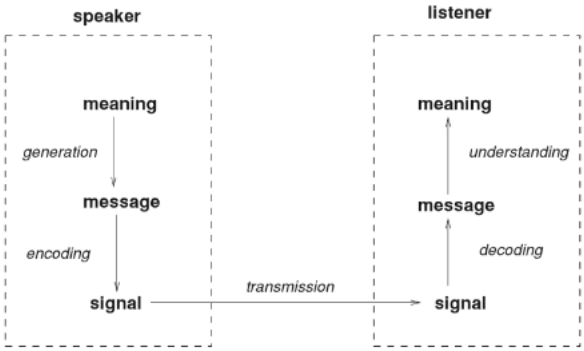


Figura 1 Procesul de comunicare între doi vorbitori [1]

Un sistem text-to-speech (TTS) va converti textul în vorbire. Acesta este compus din două părți. Prima parte va face procesul de normalizare a textului (numită și preprocesare sau tokenizare), adică va converti numerele și abrevierile în echivalentul cuvintelor scrise. Tot în această parte are loc atribuirea de transcripții fonetice fiecărui cuvânt (text-la-fonem), împărțirea și marcarea textului în unități prozodice. A doua parte va transforma reprezentarea lingvistică în sunete [2].

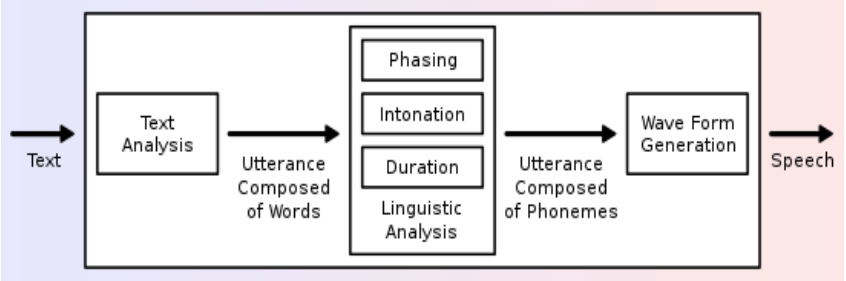


Figura 2 Prezentarea generală a unui sistem TTS [2]

1.2 Transformatori (Transformers)

1.2.1 Aspecte generale

Transformatorii sunt modele de ultimă generație dintr-o bibliotecă. Sunt folosiți pentru procesarea limbajului natural (NLP), procesarea audio și video și viziune computerizată. Pentru ca modelul să recunoască textul, acetsa trebuie să fie „token-izat", adică fiecare secvență să fie împățită în cuvinte sau subcuvinte separate. Acestea, mai apoi sunt convertite în numere.

1.2.2 SpeechT5 - Un Transformer pentru TTS (Text-to-Speech)

Un cadru SpeechT5 este format dintr-o rețea codor-decodor și rețele pre-post modale specifice [3]. În Figura 2 putem observa un model al SpeechT5.

Pre-net (pre-rețelele) convertesc textul X^t într-un spțiu unificat de reprezentări. Astfel, un text este împărțit într-o secvență de caractere $X^t = (x_1^t, x_2^t, ..., x_{N^t}^t)$ ca intrare și ieșire. Acest spațiu unificat are scopul de a salva o reprezentare semantică sau conceptuală coerentă a informației din text. Apoi acesta este introdus într-un codor-decodor pentru conversie.

În cele din urmă, post-net (post-rețelele) generează rezultatul de la ieșirea decodorului. Post-rețelele sunt formate din două module. Primul modul folosește un strat liniar alimentat cu ieșirea decodorului pentru a prezice spectrogramă Mel, $Y^f = (y_1^f, y_2^f, ..., y_{Nf}^f)$, iar al doilea modul este adăugat pentru a putea prezice jetonul de oprire.

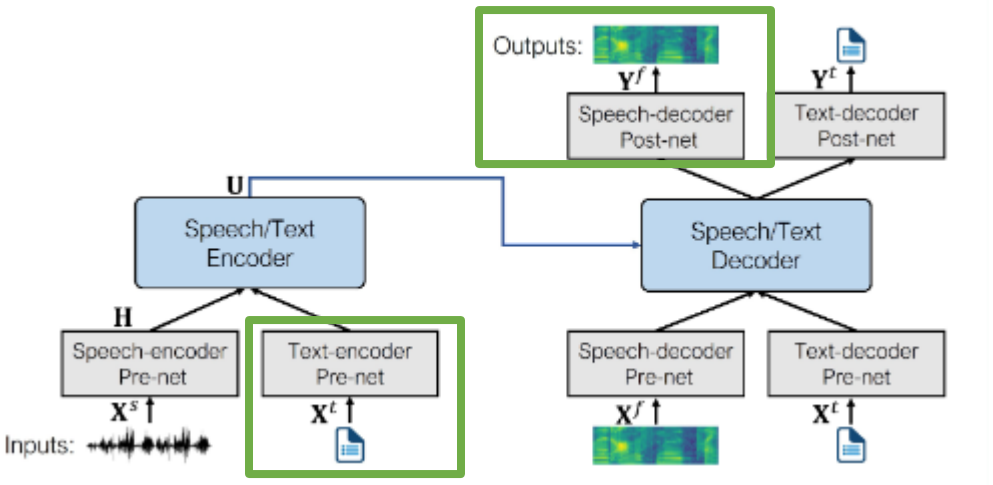


Figura 3 Modelul arhitectural al SpeechT5 [3]

De exemplu, dacă avem un text simplu: "Hello!", reprezentarea sa ca secvență de caractere ar fi X^t = (H,e,l,l,o,!), unde N^t=6 este lungimea totală a secvenței. Modelele de procesare a limbajului natural modifică reprezentarea sub formă de șir în numere pentru a facilita învățarea.

1.2.3 HiFIGAN - Un Transformer pentru Vocoder

Înregistrările audio din lumea reală sunt adesea degradate de factori precum zgomotul, reverberația și distorsiunea egalizării. HiFi-GAN, este o metodă de învățare profundă pentru a transforma vorbirea înregistrată în sunet ca și cum ar fi fost înregistrată într-un studio [4].

Un vocoder este un model care generează semnal audio dintr-o spectrogramă Mel. Este folosit pentru compresia datelor, multiplexarea, criptarea sau transformarea vocii [5]. Studiile arată că omul nu percepe frecvențele la scară liniară, și astfel, s-a propus o unitate de înălțime numită scara Mel. O spectogramă Mel este o spectogramă în care frecvențele sunt convertite la scara Mel [6].

HiFiGAN este un model de rețea adversă generativă (GAN) care generează sunet din spectrogramele Mel. Generatorul folosește convoluții transpuse pentru a supraeșantiona spectrogramele Mel în sunet [7].

Cu scopul de a adapta un model Text to Speech (TTS) pentru a sintetiza o voce personală folosind câteva mostre de vorbire de la difuzorul țintă, clonarea vocii oferă un serviciu TTS specific. O metodă de clonare a vocii bazată pe HiFi-GAN este prezentată în Figura 4, astfel [8]:

1. Vectorul x este utilizat ca vector de încorporare pentru a îmbunătăți capacitatea de reprezentare a caracteristicilor codificatorului difuzorului.
2. Pentru a îmbunătăți performanța vocoderului HiFi-GAN, spectrul Mel de intrare este procesat printr-o strategie competitivă de convoluție multiscale.
3. Convoluția separabilă unidimensională în funcție de adâncime este utilizată pentru a înlocui toate convoluțiile unidimensionale standard, reducând semnificativ parametrii modelului și crescând viteza de inferență.

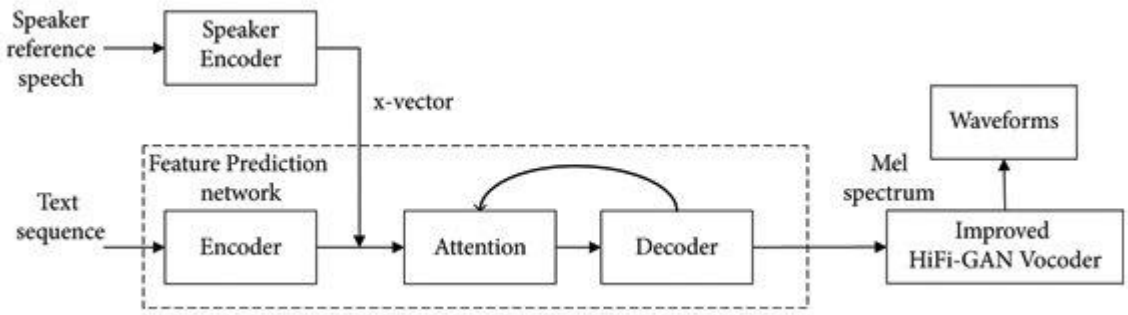


Figura 4 Arhitectura sistemului de clonare a vocii HiFi-GAN [9]

2 Implementare

Implementarea aplicației este realizată în Python.

➔ Importăm modulele necesare pentru utilizarea modelului SpeechT5 și pentru procesarea sunetului.

```
from transformers import SpeechT5Processor, SpeechT5ForTextToSpeech,
SpeechT5HifiGan
from datasets import load_dataset
import torch
import random
import string
import soundfile as sf
```

➔ Pregătim mediul de lucru prin încărcarea resurselor necesare.

Astfel, încărcăm, procesorul asociat modelului SpeechT5. Acest procesor se ocupă de preprocesarea textului într-un format compatibil cu modelul.

Și încărcăm setul de date care conține înglobările (embeddings) ale vorbitorilor, utilizate ulterior pentru obținerea caracteristicilor vocale.

```
# Verificăm disponibilitatea dispozitivului CUDA (GPU) pentru accelerarea
procesării,
# altfel utilizăm CPU.
device = "cuda" if torch.cuda.is_available() else "cpu"

# Încărcăm procesorul (processor) pentru modelul SpeechT5.
processor = SpeechT5Processor.from_pretrained("microsoft/speecht5_tts")

# Încărcăm modelul SpeechT5ForTextToSpeech și îl mutăm pe dispozitivul
specificat.
model =
SpeechT5ForTextToSpeech.from_pretrained("microsoft/speecht5_tts").to(device)

# Încărcăm modelul SpeechT5HifiGan pentru vocoder și îl mutăm pe dispozitivul
specificat.
vocoder =
SpeechT5HifiGan.from_pretrained("microsoft/speecht5_hifigan").to(device)

# Încărcăm setul de date cu înglobări (embeddings) ale vorbitorilor pentru
# a obține caracteristicile vocale.
embeddings_dataset = load_dataset("Matthijs/cmu-arctic-xvectors",
split="validation")
```

➔ În dicționarul, speakers, asociem identificatori cu vorbitorii specifici și cu înglobările lor corespunzătoare.

Acest dicționar este util pentru a specifica vorbitorul dorit atunci când se generează un discurs.

- awb: Bărbat scoțian, identificator de vorbitor cu valoarea înglobării de 0.
- bdl: Bărbat american, identificator de vorbitor cu valoarea înglobării de 1138.
- clb: Femeie americană, identificator de vorbitor cu valoarea înglobării de 2271.
- jmk: Bărbat canadian, identificator de vorbitor cu valoarea înglobării de 3403.
- ksp: Bărbat indian, identificator de vorbitor cu valoarea înglobării de 4535.
- rms: Bărbat american, identificator de vorbitor cu valoarea înglobării de 5667.
- slt: Femeie americană, identificator de vorbitor cu valoarea înglobării de 6799.

```
# Dicționar cu identificatori pentru vorbitorii utilizați.
speakers = {
    'awb': 0,      # Bărbat scoțian
    'bdl': 1138,   # Bărbat american
    'clb': 2271,   # Femeie americană
    'jmk': 3403,   # Bărbat canadian
    'ksp': 4535,   # Bărbat indian
    'rms': 5667,   # Bărbat american
    'slt': 6799    # Femeie americană
}
```

➔ Funcția ***save_text_to_speech*** primește un text și opțional un vorbitor și generează discursul asociat utilizând modelele SpeechT5 și un vocoder.

```
def save_text_to_speech(text, speaker=None):
    # Prelucrăm textul pentru a-l transforma în input pentru modelul
SpeechT5.
    inputs = processor(text=text, return_tensors="pt").to(device)

    # Dacă avem un vorbitor specificat, încărcăm înglobările (embeddings)
corespunzătoare.
    if speaker is not None:
        speaker_embeddings =
torch.tensor(embeddings_dataset[speaker]["xvector"]).unsqueeze(0).to(device)
    else:
        # Dacă nu avem un vorbitor specificat, generăm un vector aleatoriu,
reprezentând o voce aleatoare.
        speaker_embeddings = torch.randn((1, 512)).to(device)

    # Generăm discursul utilizând modelele SpeechT5 și vocoderul.
    speech = model.generate_speech(inputs["input_ids"], speaker_embeddings,
vocoder=vocoder)

    # Construim un nume de fișier în funcție de prezența sau absența unui
vorbitor specificat.
    if speaker is not None:
        output_filename = f"{speaker}-{'-'.join(text.split()[:6])}.mp3"
    else:
        random_str =
''.join(random.sample(string.ascii_letters+string.digits, k=5))
        output_filename = f"{random_str}-{'-'.join(text.split()[:6])}.mp3"

    # Salvăm discursul generat într-un fișier cu o rată de eșantionare de
16KHz.
```

```
    sf.write(output_filename, speech.cpu().numpy(), samplerate=16000)

    # Returnăm numele fișierului pentru referință.
    return output_filename
```

Aplicația dispune și de o interfață grafică.

➔ Importurile și modulele necesare pentru a începe implementarea unei aplicații cu interfață grafică utilizând Tkinter în Python

```
import tkinter as tk
from tkinter import ttk
from tts_transformers import save_text_to_speech, speakers
import pygame
import os
```

➔ Creăm o clasă ***TextToSpeechApp*** care conține toate elementele interfeței grafice și metodele asociate.

```
class TextToSpeechApp:
```

➔ Adăugăm elementele principale ale interfeței grafice

- Etichetă pentru introducerea textului
- Câmp de text pentru introducerea textului
- Etichetă pentru afișarea numărului de caractere rămase
- Actualizarea automată a numărului de caractere rămase
- Etichetă pentru selecția vorbitorului
- Combobox pentru selecția vorbitorului
- Butoane pentru generarea discursului și redarea audio
- Etichetă pentru afișarea rezultatelor

```
def __init__(self, root):
    self.root = root
    self.root.title("Text to Speech Generator")

    ttk.Label(root, text="Enter text (max 200 characters):").pack(pady=10)

    self.text_var = tk.StringVar()
    self.text_text = tk.Text(root, height=5, wrap=tk.WORD, font=('Helvetica',
12))
    self.text_text.pack(pady=10, padx=10)

    self.counter_label = ttk.Label(root, text="")
    self.counter_label.pack(pady=10)

    self.text_text.bind("<<Modified>>", self.update_counter)
    self.schedule_update_counter()

    ttk.Label(root, text="Select speaker:").pack(pady=10)

    speaker_options = ["Scottish Male", "American Male", "American Female",
```

```
"Canadian Male", "Indian Male", "American Male 2", "American Female 2",
"Random"]
    self.speaker_var = tk.StringVar()
    self.speaker_combobox = ttk.Combobox(root, textvariable=self.speaker_var,
values=speaker_options)
    self.speaker_combobox.pack(pady=10)
    self.speaker_combobox.set("Random")

    ttk.Button(root, text="Generate Speech",
command=self.generate_speech).pack(pady=20)
    ttk.Button(root, text="Play", command=self.play_audio).pack(pady=10)

    self.result_label = ttk.Label(root, text="")
    self.result_label.pack(pady=10)

    pygame.init()
```

➔ Adăugăm metoda ***generate_speech*** care este responsabilă de generarea discursului în funcție de textul introdus și de vorbitorul selectat.

```
def generate_speech(self):
    text = self.text_text.get("1.0", tk.END).strip()
    selected_speaker = self.speaker_var.get()

    if selected_speaker == "Random":
        speaker = None
    else:
        speaker_mapping = {
            "Scottish Male": "awb",
            "American Male": "bdl",
            "American Female": "clb",
            "Canadian Male": "jmk",
            "Indian Male": "ksp",
            "American Male 2": "rms",
            "American Female 2": "slt",
            "Random": None
        }
        speaker = speakers[speaker_mapping[selected_speaker]]

    output_filename = save_text_to_speech(text, speaker)
    self.result_label.config(text=f"Speech generated and saved as:
{output_filename}")
    self.audio_file = output_filename
```

➔ Adăugăm metoda ***play_audio***, care se ocupă de redarea audio a discursului generat.

```
def play_audio(self):
    try:
        if hasattr(self, 'audio_file') and os.path.exists(self.audio_file):
            pygame.mixer.music.load(self.audio_file)
            pygame.mixer.music.play()
            self.result_label.config(text="Playing audio...")
        else:
            self.result_label.config(text="No audio file to play.")
```

```
    except Exception as e:
        self.result_label.config(text=f"Error playing audio: {str(e)}")
```

➔ Adăugăm metoda ***update_counter***, care este responsabilă pentru actualizarea numărului de caractere rămase în câmpul de text.

```
def update_counter(self, event=None):
    max_characters = 200
    text_content = self.text_text.get("1.0", tk.END).strip()
    remaining_characters = max_characters - len(text_content)
    self.counter_label.config(text=f"Characters remaining:
{remaining_characters}")
```

➔ Adăugăm metoda ***schedule_update_counter***, care planifică actualizarea periodică a numărului de caractere rămase.

```
def schedule_update_counter(self):
    self.update_counter()
    self.root.after(100, self.schedule_update_counter)
```

➔ Verificăm dacă scriptul este executat direct

- root = tk.Tk(): Se creează o instanță a clasei Tk din modulul tkinter. Această instanță reprezintă fereastra principală a interfeței grafice.
- app = TextToSpeechApp(root): Se creează o instanță a clasei TextToSpeechApp, care a fost definită anterior. Aceasta preia fereastra principală (root) ca argument pentru inițializarea sa.
- root.mainloop(): Această linie pornește bucla principală a interfeței grafice, ceea ce înseamnă că aplicația va rămâne în execuție până când fereastra este închisă de către utilizator. Aceasta este o metodă esențială pentru afișarea și interacțiunea cu interfața grafică creată cu tkinter.

```
if __name__ == "__main__":
    root = tk.Tk()
    app = TextToSpeechApp(root)
    root.mainloop()
```

3 Rezultate experimentale

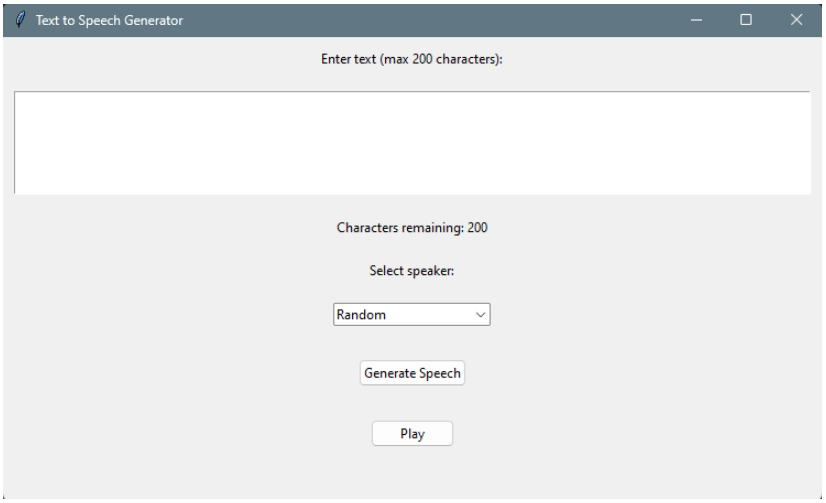
➔ Exemplu default de utilizare a funcției save_text_to_speech pentru a genera discurs cu diferite texte și vorbitori.

```
# Generăm discurs cu voce feminină americană.
save_text_to_speech("Python is my favorite programming language",
speaker=speakers["slt"])

# Generăm discurs cu o voce aleatoare.
save_text_to_speech("Python is my favorite programming language")

# Text de test care să fie generat pentru toți vorbitorii.
text = """In his miracle year, he published four groundbreaking papers.
These outlined the theory of the photoelectric effect, explained Brownian
motion,
introduced special relativity, and demonstrated mass-energy equivalence."""

# Generăm discurs pentru fiecare vorbitor și salvăm în fișiere separate.
for speaker_name, speaker in speakers.items():
    output_filename = save_text_to_speech(text, speaker)
    print(f"Salvat {output_filename}")

# Generăm discurs cu o voce aleatoare.
output_filename = save_text_to_speech(text)
print(f"Salvat {output_filename}")
```

Astfel, în terminal o să apară când fișierele au fost salvate.

```
Salvat 0-In-his-miracle-year,-he-published.mp3
Salvat 1138-In-his-miracle-year,-he-published.mp3
Salvat 2271-In-his-miracle-year,-he-published.mp3
Salvat 3403-In-his-miracle-year,-he-published.mp3
Salvat 4535-In-his-miracle-year,-he-published.mp3
Salvat 5667-In-his-miracle-year,-he-published.mp3
Salvat 6799-In-his-miracle-year,-he-published.mp3
Salvat FsqBf-In-his-miracle-year,-he-published.mp3
```

➔ Pentru exemplificarea în timp real, putem folosi interfața grafică.



4 Concluzie

În concluzie, proiectul prezintă o aplicație Text-to-Speech (TTS), folosind modelele SpeechT5 și SpeechT5HifiGan. Această aplicație oferă utilizatorilor posibilitatea de a transforma textul scris în discurs vocal natural, cu opțiunea de a alege între diferiți vorbitori.

Prin integrarea funcționalităților de procesare a limbajului natural și de sinteză vocală și o interfață grafică simplă și intuitivă, aplicația permite utilizatorilor să transforme în fișier audio diferite texte și folosind mai multe tipuri de vorbitori.

În comparație cu alte aplicații de sinteză vocală sau Text-to-Speech (TTS), proiectul se evidențiază prin utilizarea modelelor de limbaj avansate, precum SpeechT5 și SpeechT5HifiGan. Iată câteva aspecte de luat în considerare în comparație cu alte aplicații TTS existente:

- ➔ Calitatea Discursului:
  - Utilizarea modelelor de limbaj pre-antrenate și de înaltă calitate contribuie la generarea unui discurs vocal natural și clar.
- ➔ Variatia Vorbitorilor:
  - Proiectul permite utilizatorilor să aleagă între diferiți vorbitori, oferind astfel o experiență personalizată.
- ➔ Interfață Grafică:
  - Interfața grafică intuitivă facilitează utilizarea aplicației, permițând utilizatorilor să introducă text și să selecteze opțiuni cu ușurință.
- ➔ Model de Generare a Vocii și Vocoder:
  - Integrarea unui model de generare a vocii (SpeechT5) și a unui vocoder (SpeechT5HifiGan) contribuie la o sinteză vocală de înaltă calitate.
- ➔ Posibilități de Extindere:
  - Proiectul oferă potențial pentru extindere prin adăugarea de funcționalități suplimentare, cum ar fi recunoașterea vocii sau personalizarea vorbitorilor.
- ➔ Utilizare a Tehnologiilor Avansate:
  - Utilizarea bibliotecilor și framework-urilor precum Transformers de la Hugging Face și PyTorch demonstrează integrarea tehnologiilor avansate în dezvoltarea aplicației.

5 Bibliografie

[1] P. Taylor, Text-to-speech synthesis, Prentice Hall Professional Technical Reference, 2009.

[2] "Wikipedia," Speech synthesis, 30 11 2023. [Online]. Available: https://en.wikipedia.org/wiki/Speech_synthesis. [Accessed 05 12 2023].

[3] R. W. L. Z. S. L. S. R. Y. W. T. K. Q. L. Y. Z. Z. W. Y. Q. J. L. F. W. Junyi Ao, "SpeechT5: Unified-Modal Encoder-Decoder Pre-Training for Spoken Language Processing," *Annual Meeting of the Association for Computational Linguistics*, 14 10 2021.

[4] Z. J. A. F. Jiaqi Su, "HiFi-GAN:High-Fidelity Denoising and Dereverberation Based on Speech Deep Features in Adversarial Networks," 2021.

[5] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Vocoder. [Accessed 09 01 2024].

[6] L. Roberts, "Medium," 06 03 2020. [Online]. Available: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53. [Accessed 09 01 2024].

[7] "nvidia," 5 09 2023. [Online]. Available: https://docs.nvidia.com/tao/tao-toolkit/text/tts/vocoder.html. [Accessed 2023 12 12].

[8] Z. &. T. J. &. Z. Y. &. L. J. &. B. X. Qiu, "Voice Cloning Method Based on the Improved HiFi-GAN Model," *Computational Intelligence and Neuroscience*, 2022.

[9] "A Voice Cloning Method Based on the Improved HiFi-GAN Model," ResearchGate, [Online]. Available: https://www.researchgate.net/figure/System-architecture-of-voice-cloning-is-based-on-improved-HiFi-GAN_fig1_364319026. [Accessed 16 01 2024].