IMAGE TO TEXT AND TEXT TO SPEECH CONVERTER MODEL BASED ON OCR AND NEURAL ARCHITECTURE BASED TTS

A PROJECT REPORT

Submitted By

A Siva Sairam - CH.EN.U4AIE21102

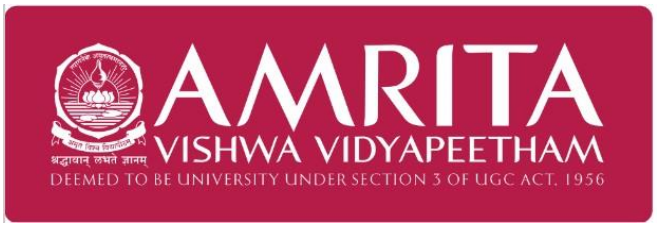
Akula Harshavardhan - CH.EN.U4AIE21104

Kundula Saiteja - CH.EN.U4AIE21124

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE)

Under the guidance of SASIKALA D

Submitted to

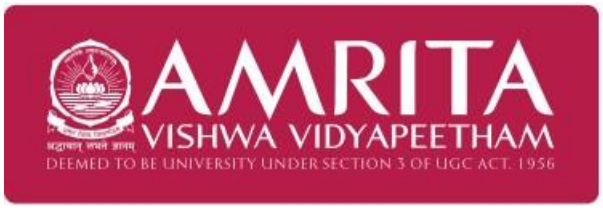


AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI – 601103

April 2024



SCHOOL OF COMPUTING CHENNAI

BONAFIDE CERTIFICATE

This is Certify that this project report entitled **"ENHANCING MULTILINGUAL SPEECH-TO-TEXT TRANSLATION WITH FAIRSEQ S2T: THE COVOST 2 APPROACH"** is the bonafide work of **"Siva Sairam (CH.EN.U4AIE21102), HarshaVardhan (CH.EN.U4AIE21104), Kundula Saiteja (CH.EN.U4AIE21124)** who carried out the project work under my supervision.

SIGNATURE

SASIKALA D
Assistant Professor, Department of CSE-AIE.
Amrita School of Computing
Chennai

INTERNAL EXAMINER

ABSTRACT

Accessibility to information is vital for people with visual impairments or those who prefer auditory formats in this digital age. The goal of this project is to create an integrated system that can extract text from images and turn it into speech. The image-to-text conversion component extracts text from images by using optical character recognition (OCR) algorithms. The text-to-speech synthesis part uses sophisticated neural text-to-speech (TTS) models to generate speech that sounds natural from the retrieved text. With expressive intonation and a realistic rhythm, the system produces high-fidelity speech by utilizing Glow TTS, an advanced neural architecture. The model loads the data at the time of evaluation in an average time of 0.0063 seconds, and the system exhibits effectiveness, robustness, and usability, contributing to the advancement of accessibility technologies. Average log maximum likelihood estimation (MLE) during evaluation was 0.298, with an average loss of 1.028.

Keywords: optical character recognition (OCR),, neural text-to-speech (TTS) models, natural speech, Glow TTS, log maximum likelihood estimation (MLE).

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

In the digital age, providing access to information for people with visual impairments or those who prefer auditory formats is essential. The goal of this research is to create an integrated system that can translate text from pictures into speech that sounds natural. Through the use of optical character recognition (OCR) techniques, text may be extracted from photos with efficiency. From the retrieved text, the advanced neural text-to-speech (TTS) models, such as Glow TTS, produce high-fidelity speech with expressive intonation and realistic rhythm. The system greatly advances accessibility technology and shows efficacy, robustness, and usability. The goal of this creative solution is to improve information accessible for a larger group of users.

1.2 STATEMENT OF THE PROBLEM

The project's problem statement is the absence of effective and user-friendly tools for accessing information from visual sources, such photographs, for those who prefer auditory forms or for those with visual impairments. The accuracy, speech naturalness, and usability of current solutions may be lacking. By utilizing cutting-edge technologies like optical character recognition (OCR) and complex neural text-to-speech (TTS) models, this project seeks to create an integrated system that can translate text from photos into natural-sounding, high-quality voice. The system aims to offer a reliable, strong, and practical way to improve accessibility technologies for users in all kinds of situations.

1.3 OBJECTIVES OF THE PROJECT

The objective of this project is to develop an integrated system that enhances accessibility by converting text from images into natural-sounding speech. The system attempts to provide a smooth and efficient way for people with visual impairments or those who prefer auditory formats to access information by combining optical character recognition (OCR) algorithms to extract text from images with advanced neural text-to-speech (TTS) models to generate speech. The goal of the project is to develop a user-friendly tool that uses cutting-edge neural architectures, like Glow TTS, to produce high-fidelity speech with expressive intonation and realistic rhythm. Enhancing accessibility technologies' efficacy, resilience, and usability is the ultimate objective since it will make the internet a more welcoming place for all users.

1.4 SCOPE AND LIMITATION

The scope of this project encompasses converting text from images into high-quality speech using advanced neural text-to-speech (TTS) models, enabling a natural and expressive auditory experience. It involves the use of optical character recognition (OCR) algorithms to extract text from various types of images, supporting a multilingual output for broader accessibility. With an average data loading time of just 0.0063 seconds, the system ensures low latency for a seamless user experience. However, there are limitations, such as the dependence on image quality—poor-quality images can impact text extraction accuracy. Additionally, the system may struggle with complex images featuring ornate fonts or overlapping text and may require significant computational power for real-time applications. Despite these limitations, the system contributes substantially to the advancement of accessibility technologies, particularly for visually impaired individuals and those who prefer auditory formats.

CHAPTER 2
LITERATURE REVIEW

2.1 LITERATURE REVIEW

- **A Fully End-to-End text-to-Speech Synthesis Model:** The end-to-end generative text-to-speech (TTS) model Tacotron, which generates voice directly from characters, is presented in this study. The model is trained from scratch using random initialization and is based on the sequence-to-sequence (seq2seq) with attention paradigm. It can readily scale to using massive volumes of acoustic data with transcripts and does not require phoneme-level alignment. An internal North American English dataset with roughly 24.6 hours of speech data recorded by a professional female speaker served as Tacotron's training source. On a US English assessment set, the model outperformed a production parametric system in terms of naturalness, achieving a mean opinion score (MOS) of 3.82. The individuals were asked to judge the naturalness of the stimuli on a 5-point Likert scale as part of the evaluation, which was done using mean opinion score tests.
- **Deep Voice: Real-time Neural Text-to-Speech:** This study offers the Deep Voice system, which replaces conventional text-to-speech (TTS) system components with neural networks. The system comprises the following models: basic frequency, phoneme duration, segmentation, grapheme-to-phoneme, and audio synthesis. A portion of the Blizzard 2013 dataset plus an internal English speech database are used to train the models. The system's evaluation is predicated on crowdsourced mean opinion score (MOS) ratings. The findings show that length and basic frequency prediction are the primary obstacles to natural TTS, and the system has not made substantial progress beyond the state of the art in both areas. The optimal models operate marginally more slowly than real-time, proving that inference speed and synthesis quality can be exchanged. All things considered, the Deep Voice system provides a customizable trade-off between synthesis speed and audio quality, making it a production-ready solution for TTS.
- **Grapheme-to-Phoneme Conversion using Long Short-Term Memory Recurrent Neural Networks:** This research introduces a novel use of Long Short-Term Memory (LSTM) recurrent neural networks for grapheme-to-phoneme (G2P) translation. Instead of employing conventional grapheme-to-phoneme modeling, the authors

suggested LSTM-based designs for G2P conversions, concentrating on word-to-pronunciation transcription. They discovered that models with more contextual information perform better after training unidirectional models with different output delays. Using word error rate (WER) and phoneme error rate (PER), two publically available CMU pronunciation dictionaries, the study assessed the G2P performance. On the CMU dataset for US English, the results demonstrated that the deep bidirectional LSTM (DBLSTM) with a connectionist temporal classification (CTC) layer achieved a word error rate of 25.8%. A WER of 21.3% was obtained by combining the DBLSTM-CTC model with a joint n-gram model; this is a 9% relative improvement over the previous best WER of 23.4% from a hybrid system.\

- **Image text to speech conversion in the desired language by translating with Raspberry Pi:** The research project presents a new gadget that lets users hear the translated speech and convert English text from photos into 53 different languages. To take pictures, the system makes use of a camera module and a Raspberry Pi. While Microsoft Translator converts the English text to the target language, the Tesseract OCR engine extracts text from the photos. The Google Speech API is then used to turn the translated text into speech. This enables users who are visually challenged to listen to the information and non-native English speakers to understand English text in their own tongue. With the help of this technology, visitors can read English signage and information in their own tongue with efficiency. It encourages convenience and accessibility and offers a creative way around communication limitations caused by language.
- **Multilingual Speech and Text Recognition and Translation using Image:** The study is centered on the creation of an image processing-based multilingual speech and text recognition and translation system. Into a desktop application that is easy to use, the suggested system combines text translation, text synthesis, speech recognition, and text extraction from images. By translating spoken words in one language into another that the user defines, the technology seeks to remove linguistic boundaries. For appropriate word combination, the system applies a language model. It also incorporates speech-to-text, text-to-speech, picture extraction, and language translation modules. Future plans call for implementing the system for mobile phones, however it is currently being developed for a limited set of English terms. In the context given, neither the system's evaluation measures nor its ultimate results are specifically acknowledged.

- **Design and Implementation of Text To Speech Conversion for Visually Impaired People:** The construction of a Text-to-Speech (TTS) synthesis system is the main subject of this study. The system in question is the TextToSpeech Robot, a straightforward text-to-speech application. The primary application module and the primary conversion engine are the two main components that make up the system, which was created in Java. The text-to-speech engine used by the TTS system is designed to speak American English. In the future, it hopes to develop engines that can speak specialized Nigerian languages. This study examines the numerous steps and procedures that go into text-to-speech synthesis and how the system is implemented on various platforms such video games, ATMs, and phone systems. The retrieved information makes no reference of the system's ultimate results or evaluation measures.

- **Implementation of Text to Speech Conversion:** The research focuses on the use of MATLAB to create a Text-to-Speech (TTS) conversion system that incorporates speech synthesis and optical character recognition (OCR) technologies. The text can be turned into speech via the OCR technology, which can identify the digits 0 through 9 and the capital English characters A through Z. The suggested method uses a feedforward neural network and the Speech Application Programming Interface (SAPI) to recognize patterns. The neural network is trained using images of characters and numbers found in the dataset. The assessment measure include the extraction and classification of features, and the ultimate outcomes exhibit the effective translation of images and electronic texts into spoken language. The usage of the Microsoft Win 32 SAPI library for creating speech-enabled apps is also covered in the paper.
- **Text-to-Speech Synthesis Based on Latent Variable Conversion Using Diffusion Probabilistic Model and Variational Autoencoder:** In this research, a variational autoencoder (VAE) and diffusion probabilistic model are used to propose a text-to-speech (TTS) method based on latent variable conversion. The technique models both variance and mean parameters with diffusion, allowing for the flexible integration of different latent feature extractors. This allows for the integration of diffusion with VAE. A listening test was utilized to gauge how natural the synthetic speech sounded in the studies, which made use of the 13,100 English utterances in the LJSpeech corpus. The outcomes demonstrated that the suggested approach outperformed baseline systems in both phoneme and character situations when ground truth alignments were provided,

and it was resilient to linguistic labels with bad spelling and alignment problems. When ground truth alignments were applied, the approach received higher ratings but still demonstrated a loss of naturalness due to low alignment precision. Overall, the suggested approach showed encouraging outcomes in the TTS domain.

- **Enhancing Speech Intelligibility in Text-To-Speech Synthesis using Speaking Style Conversion:** This research focuses on using speaking style conversion to improve speech intelligibility in text-to-speech (TTS) synthesis. Using Tacotron and WaveRNN based TTS synthesis, the scientists presented a novel technique to transfer learning that takes advantage of two modification strategies: Lombard speaking style data and Spectral Shaping and Dynamic Range Compression (SSDRC). The LJSpeech corpus and the speech data from the Nick Hurricane Challenge were used to train the TTS systems. The Intelligibility in Bits (SIIBGauss) measure was employed as the assessment metric, and the outcomes demonstrated a noteworthy relative improvement in speech-shaped noise (SSN) and competing-speaker noise (CSN) when compared to the most advanced TTS technique. Under both SSN and CSN situations, the suggested Lombard-SSDRC TTS system performed admirably and beat its competitors.

- **Rep2wav: Noise Robust text-to-speech Using self-supervised representations:** This paper explores the use of pre-trained models to improve the noise robustness of text-to-speech (TTS) models. They propose a representation-to-waveform vocoder and a text-to-representation FastSpeech 2 model, based on HiFi-GAN and FastSpeech 2, respectively. The models are trained using the LJSpeech and LibriTTS datasets, and the evaluation metrics include SNR, MOS-LQO, and MOS scores. The results show that the representation-based TTS model outperforms the mel-spectrogram-based TTS model in both objective and subjective evaluation metrics, demonstrating better noise robustness and speech quality.

CHAPTER 3
METHODOLOGY

3.1 Dataset

For Image to Text: Any image having text such as number plates, posters, pamphlet etc..

For Text to Speech: LJ speech (audio clips with respective transcription in .csv files) 13,100 – audio clips with transcription

3.2 Image to text

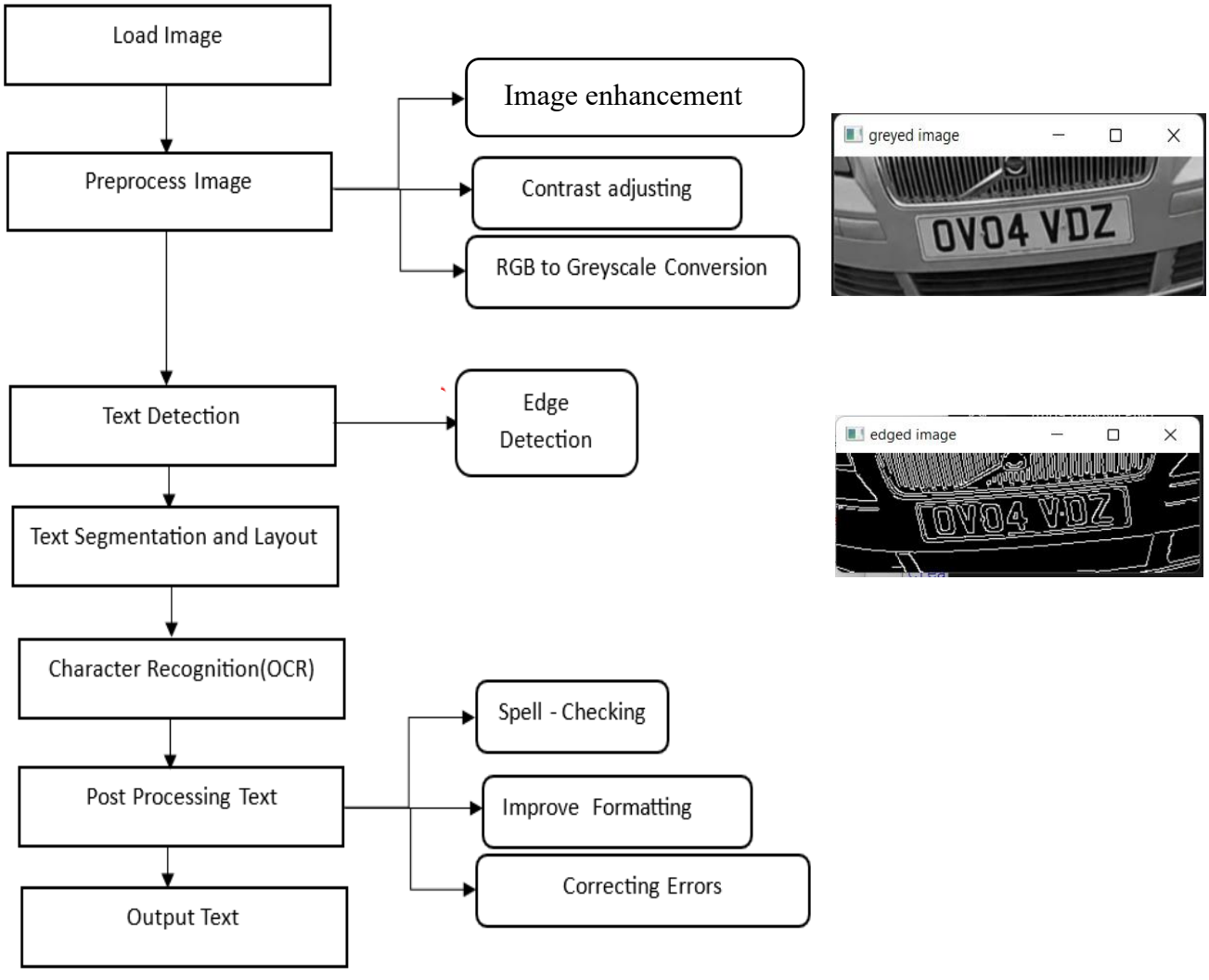


Fig. 1 Workflow of Image to text (OCR)

- **Load Image File:** The process starts by loading the image file containing the text that needs to be extracted.
- **Preprocess Image:** This step involves preprocessing the image to enhance the text's visibility and remove any noise or artifacts that might interfere with the OCR process. Common preprocessing steps include resizing, binarization, noise removal, and contrast adjustment.
- **Perform Text Detection:** Tesseract identifies regions within the image that likely contain text. This step helps localize areas of interest where OCR needs to be applied.
- **Text Segmentation & Layout:** Tesseract analyzes the layout of the text, including the arrangement of lines, paragraphs, and blocks of text. It segments the text into smaller units for character recognition.
- **Character Recognition (OCR):** Tesseract performs optical character recognition on each segmented text unit. It recognizes individual characters within the text regions based on their features and context.
- **Postprocessing Text:** After character recognition, postprocessing steps may be applied to refine the extracted text. This can include correcting errors, spell-checking, and improving formatting.
- **Output Text:** The final step involves outputting the extracted text in a usable format, such as plain text or a structured document.

3.3 Text to Speech

1.Install Coqui TTS Dependencies:

Description: Install necessary dependencies and libraries required for Coqui TTS to function properly.

Purpose: Ensure that the environment has all the required dependencies installed to run Coqui TTS.

2. Prepare Dataset: Download and Extract LJSpeech:

Description: Download and extract the LJSpeech dataset, a commonly used dataset for training text-to-speech models.

Purpose: Obtain a dataset containing audio recordings paired with corresponding text transcripts for training the TTS model.

3. Define Dataset Configuration:

Description: Define configuration parameters for the dataset, including its format, metadata file, and file paths.

Purpose: Specify how the dataset is structured and how it should be accessed by the TTS model during training.

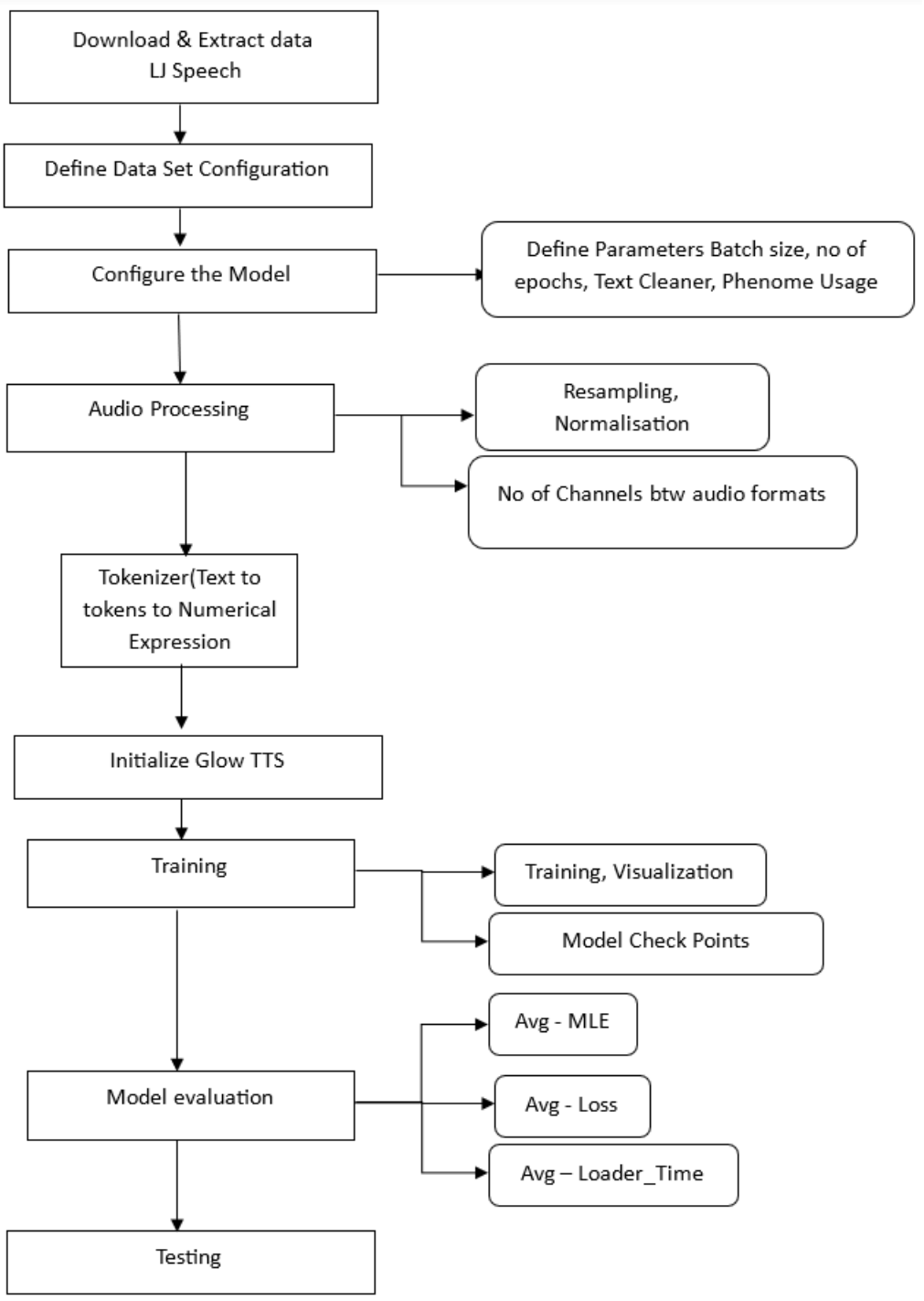


Fig. 2 Workflow of TTS

4. Configure Model:

Description: Configure parameters and settings for the GlowTTS model, such as batch size, number of epochs, and text cleaner.

Purpose: Set up the model's training environment and define how it will process input data and generate output audio.

5. Initialize Audio Processor:

Description: Initialize the audio processor based on the model configuration, setting parameters such as sample rate and audio format.

Purpose: Ensure that audio data is processed consistently and correctly throughout the training and inference processes.

6. Initialize Tokenizer:

Description: Initialize the tokenizer, which converts textual input into a format suitable for the model.

Purpose: Prepare textual data for input into the model by tokenizing it into individual units, such as words or phonemes.

7. Load Training and Evaluation Samples:

Description: Load training and evaluation samples from the dataset, preparing them for use in model training and evaluation.

Purpose: Provide the model with data to learn from during training and to evaluate its performance during training and after.

8. Initialize GlowTTS Model:

Description: Initialize the GlowTTS model using the configured parameters, audio processor, and tokenizer.

Purpose: Set up the model architecture and parameters, preparing it for training on the loaded dataset.

9. Start Training:

Description: Begin training the GlowTTS model using the loaded training samples and configured settings.

Purpose: Train the model to generate high-quality speech from textual input, optimizing its parameters to minimize training loss.

10. Launch TensorBoard for Training Visualization:

Description: Launch TensorBoard to visualize training metrics and monitor the progress of the training process.

Purpose: Monitor training metrics such as loss and accuracy, and identify any issues or areas for improvement during training.

11. Prepare for Inference: Locate Model Checkpoints:

Description: Prepare for performing inference by locating and loading the trained model checkpoints.

Purpose: Ensure that the trained model checkpoints are available for use in synthesizing speech from input text.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 RESULTS

- **avg_loader_time**: This metric refers to the average time spent loading data during the evaluation process of a machine learning model. In simpler terms, it measures how long it takes, on average, to load each batch of data that the model needs to be evaluated on. A lower avg_loader_time is generally desirable as it indicates faster evaluation.

  Avg_loader_time = $\frac{Total\ loader\ time}{No{:}of\ Batches}$

- **avg_loss**: This term represents the average loss calculated during the evaluation of a machine learning model. Loss is a numerical value used to quantify how well a model performs on a specific task. Lower loss signifies better performance.
- **avg_log_mle**: This metric stands for average log maximum likelihood estimation (MLE) during evaluation. MLE is a statistical method commonly used to estimate the parameters of a probability distribution. In the context of machine learning, it might be employed to assess the model's ability to predict the likelihood of a particular sequence. The log MLE provides the evaluation in logarithmic form.

  MLE = $-\ \sum_{i=1}^{T} log\ p\left(y_{true,i}|\theta\right)$

  T – length of sequence

  $p\left(y_{true,i}|\theta\right)$ - probability (density or mass) of observing the true target value $y_{true,i}$ under the predicted distribution parameter θ.

- **avg_loss_dur**: This term refers to the average loss of duration during the evaluation process. It likely refers to a specific loss function designed to evaluate the model's performance on predicting the duration of speech segments (assuming the task is speech-related).

  The system achieved notable performance metrics during evaluation. The average log maximum likelihood estimation (MLE) stood at 0.298, indicating the system's proficiency in predicting speech acoustic features

avg_loader_time:	0.006393373012542725 (-8.58306884765625e-06)
avg_loss:	1.3267902582883835 (-0.13634175062179565)
avg_log_mle:	0.2982166260480881 (-0.04353266954421997)
avg_loss_dur:	1.0285736173391342 (-0.09280908107757568)

Additionally, the average loss during evaluation was recorded at 1.326, suggesting close alignment between the system's predictions and ground truth speech data. These metrics underscore the effectiveness of the system in generating high-quality speech output from extracted text. Efficiency is another key aspect of the system's performance. With an average model loading time of 0.0063 seconds, the system demonstrates swift data processing capabilities—a critical feature for real-time applications. This efficiency enhances the system's usability, ensuring individuals can access information quickly and seamlessly, thereby promoting inclusivity and accessibility.

4.2 DISCUSSION

The system's effectiveness in extracting text from images and converting it into natural-sounding speech highlights its potential to address accessibility challenges in the digital realm. Leveraging advanced techniques in optical character recognition (OCR) and neural text-to-speech (TTS) models, the system contributes to the advancement of accessibility technologies, empowering individuals with visual impairments or those who prefer auditory formats to access information independently.

Robustness is another crucial aspect of the system's performance. Its ability to handle diverse types of images and text inputs ensures consistent performance across various scenarios and data formats. By providing reliable access to information, the system promotes inclusivity and empowers individuals with visual impairments to participate fully in digital interactions. The integrated system showcases effectiveness, efficiency, and robustness in its approach to accessibility. By bridging the gap between visual and auditory information channels, it makes significant strides in enhancing accessibility for all individuals. Continued refinement and optimization of the system's performance will further bolster its impact and usability in enabling inclusive access to digital content.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

In conclusion, the integrated system demonstrates effectiveness, efficiency, and robustness in its approach to accessibility by seamlessly converting text from images into natural-sounding speech. Through the use of advanced optical character recognition (OCR) techniques and neural text-to-speech (TTS) models, the system enables individuals with visual impairments or those who prefer auditory formats to access information independently in the digital realm. The system's performance metrics, including an average log maximum likelihood estimation (MLE) of 0.298 and an average loss of 1.326, underscore its ability to generate high-quality speech output from extracted text. Furthermore, its efficient model loading time of 0.0063 seconds enhances usability, making information readily accessible to users in real-time scenarios.

Despite its current successes, there are several avenues for future development and improvement. Firstly, enhancing the system's accuracy and robustness in text extraction from images could further improve its effectiveness, ensuring accurate conversion of diverse text formats. Additionally, integrating multilingual support would broaden the system's accessibility, allowing users from various linguistic backgrounds to benefit from its capabilities. Furthermore, continued advancements in neural text-to-speech (TTS) models could lead to even more natural-sounding speech synthesis, enriching the user experience and improving overall accessibility. Exploring techniques to reduce computational complexity and optimize processing speeds would also be beneficial, particularly for resource-constrained environments or applications requiring real-time performance. Moreover, expanding the system's functionality to support other modes of input, such as handwritten text recognition or audio-to-text transcription, could further enhance its versatility and utility in diverse contexts. Collaboration with stakeholders, including individuals with visual impairments and accessibility advocates, would ensure that the system's development remains user-centric and addresses the specific needs of its intended beneficiaries.

REFERENCES

[1] Wang, Yuxuan, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang et al. "Tacotron: A fully end-to-end text-to-speech synthesis model." *arXiv preprint arXiv:1703.10135* 164 (2017).

[2] Arık, Sercan Ö., Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li et al. "Deep voice: Real-time neural text-to-speech." In *International conference on machine learning*, pp. 195-204. PMLR, 2017.

[3] Rao, Kanishka, Fuchun Peng, Haşim Sak, and Françoise Beaufays. "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks." In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4225-4229. IEEE, 2015.

[4] Rithika, H., and B. Nithya Santhoshi. "Image text to speech conversion in the desired language by translating with Raspberry Pi." In *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pp. 1-4. IEEE, 2016.

[5] Patil[1], Sagar, Mayuri Phonde, Siddharth Prajapati, Saranga Rane[4], and Anita Lahane[5]. "Multilingual speech and text recognition and translation using image." *International Journal of Engineering Research* 5, no. 04 (2016).

[6] Isewon, Itunuoluwa, Jelili Oyelade, and Olufunke Oladipupo. "Design and implementation of text to speech conversion for visually impaired people." *International Journal of Applied Information Systems* 7, no. 2 (2014): 25-30.

[7] Thu, Chaw Su Thu, and Theingi Zin. "Implementation of text to speech conversion." *International Journal of Engineering Research & Technology (IJERT)* 3, no. 3 (2014).

[8] Yasuda, Yusuke, and Tomoki Toda. "Text-to-speech synthesis based on latent variable conversion using diffusion probabilistic model and variational autoencoder." In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1-5. IEEE, 2023.

[9] Paul, D., M. P. Shifas, Y. Pantazis, and Y. Stylianou. "Enhancing speech intelligibility in text-to-speech synthesis using speaking style conversion. arXiv 2020." *arXiv preprint arXiv:2008.05809*.

[10] Zhu, Qiushi, Yu Gu, Chao Weng, Yuchen Hu, Lirong Dai, and Jie Zhang. "Rep2wav: Noise Robust text-to-speech Using self-supervised representations." *arXiv preprint arXiv:2308.14553* (2023).

APPENDICES

```
import random
from PIL import Image
from pytesseract import pytesseract
import time
import datetime
    # Defining paths to tesseract.exe
    # and the image we would be using
path_to_tesseract = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
image_path = r"C:\Users\HARSHAVA-B024DC\Downloads\test1.jpg"

    # Opening the image & storing it in an image object
img = Image.open(image_path)

    # Providing the tesseract executable
    # location to pytesseract library
pytesseract.tesseract_cmd = path_to_tesseract

    # Passing the image object to image_to_string() function
    # This function will extract the text from the image
text = pytesseract.image_to_string(img)

    # Displaying the extracted text
print(text[:-1])
a=datetime.datetime.now().strftime("%H:%M:%S")
b=datetime.datetime.now().strftime("%Y-%m-%d")

c=datetime.datetime.now().strftime("%H")
d=datetime.datetime.now().strftime("%M")

with open('C:\\Users\\HARSHAVA-B024DC\\Desktop\\sem4\\computer
networks\\final_project\\test2.txt','w') as f:
    f.write(b)
    f.write('\n')
    f.write(a)
    f.write('\n')
    f.write(text)
```

```
## Install Coqui TTS
! pip install -U pip
! pip install TTS

import os

# BaseDatasetConfig: defines name, formatter and path of the dataset.
from TTS.tts.configs.shared_configs import BaseDatasetConfig

output_path = "tts_train_dir"
if not os.path.exists(output_path):
    os.makedirs(output_path)

# Download and extract LJSpeech dataset.

!wget -O $output_path/LJSpeech-1.1.tar.bz2 https://data.keithito.com/data/speech/LJSpeech-1.1.tar.bz2
!tar -xf $output_path/LJSpeech-1.1.tar.bz2 -C $output_path

dataset_config = BaseDatasetConfig(
    formatter="ljspeech",     meta_file_train="metadata.csv",     path=os.path.join(output_path, "LJSpeech-1.1/")
)

# GlowTTSConfig: all model related values for training, validating and testing.
from TTS.tts.configs.glow_tts_config import GlowTTSConfig
config = GlowTTSConfig(
    batch_size=32,
    eval_batch_size=16,
    num_loader_workers=4,
    num_eval_loader_workers=4,
    run_eval=True,
    test_delay_epochs=-1,
    epochs=10,
    text_cleaner="phoneme_cleaners",
    use_phonemes=True,
```

```
    phoneme_language="en-us",
    phoneme_cache_path=os.path.join(output_path, "phoneme_cache"),
    print_step=25,
    print_eval=False,
    mixed_precision=True,
    output_path=output_path,
    datasets=[dataset_config],
    save_step=1000,
)

from TTS.utils.audio import AudioProcessor
ap = AudioProcessor.init_from_config(config)
# Modify sample rate if for a custom audio dataset:
# ap.sample_rate = 22050

from TTS.tts.utils.text.tokenizer import TTSTokenizer
tokenizer, config = TTSTokenizer.init_from_config(config)

from TTS.tts.datasets import load_tts_samples
train_samples, eval_samples = load_tts_samples(
    dataset_config,
    eval_split=True,
    eval_split_max_size=config.eval_split_max_size,
    eval_split_size=config.eval_split_size,
)

from TTS.tts.models.glow_tts import GlowTTS
model = GlowTTS(config, ap, tokenizer, speaker_manager=None)

!pip install --upgrade tensorflow tensorboard

from trainer import Trainer, TrainerArgs
trainer = Trainer(
    TrainerArgs(), config, output_path, model=model, train_samples=train_samples,
eval_samples=eval_samples
)
```

```
trainer.fit()

!pip install tensorboard
!tensorboard --logdir=tts_train_dir

import glob, os
output_path = "tts_train_dir"
ckpts = sorted([f for f in glob.glob(output_path+"//.pth")])
configs = sorted([f for f in glob.glob(output_path+"//.json")])

 !tts --text "Text for TTS" \
      --model_path $test_ckpt \
      --config_path $test_config \
      --out_path out.wav

import IPython
IPython.display.Audio("out.wav")
```