

# **Faculty of Computer & Information Technology**

## **Smart Glasses For visually impaired people**

**By:**

- 1.Marwan Maher Mostafa (ID:2002008)
- 2.Mahmood Abdulghani Hassen (ID:2001274)
- 3.Hassan abo Elghet Hassan (ID:2001197)
- 4.Nada Hamada Salah (ID:2000684)
- 5.Nancy Saad Mohamed (ID:2001336)
- 6.Menna Allah Mahmoud Thabet (ID:2000689)
- 7.Arwa Sayed Elarosy(ID:2001485)

**Under Supervision of:**

Prof . Ahmed hamza

Eng. Mostafa Abdelbari

Lecturer at Faculty Computer  
and Information Technology,  
Egyptian E-Learning University

Teacher Assistant, Department  
of Information Technology  
Egyptian E-Learning University

This project is submitted as a partial fulfillment of the  
requirements for the degree of Bachelor of Science in Computer  
& Information Technology

## Acknowledgement

First and for most, praises and thanks to God, the Almighty, for his blessing throughout our years in the college and our graduation project to complete this stage of our life successfully.

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincerethanks to all of them.

Thanks to Egyptian E-Learning University especially Assiut center for helping us to reach thislevel of awareness.

We would like to express our deep and sincere gratitude to our project supervisor **Dr. Ahmed Hamza** for giving us the opportunity to lead us and providing invaluable guidance throughout this project. It was a great privilege and honor to work and study under her guidance. We are extremely grateful for what she has offered us.

We are especially indebted to say many thanks to **Eng. Mostafa Abdelbari** for guidance and constant supervision as well as for his patience, friendship, empathy, and a great sense of humor. His dynamism, vision, sincerity, and motivation have deeply inspired us.

Finally, we would like to express our gratitude to everyone who helped usduring the graduation project.

## **Contents:**

### **Chapter1:Introduction**

### **Chapter 2: Literature Review**

1. **Related Work** – 13
2. **Project Schedule** – 14
3. **Backend Development**
4. **Frontend Development**

### **Chapter 3: Background**

1. **Assistive Technology for the Visually Impaired** - 16
2. **Historical Overview** - 16
3. **Modern Innovations** - 26
4. **Smartphone Apps for Accessibility** - 17
5. **Existing Solutions** - 17
6. **Comparative Analysis** - 17
7. **Machine Learning and Text Analysis** - 18
8. **Text Detection** - 18
9. **Text Recognition** - 18
10. **Text-to-Speech Conversion** - 18
11. **Evaluation Metrics** – 18-19

### **Chapter 4: System Design**

1. **System Architecture** - 21
2. **AI Models** - 21
3. **Text Detection Model** – 21-22
4. **Text Recognition Model** - 22
5. **Text-to-Audio Model** – 22-23
6. **Backend Development** - 24
7. **Django API** - 24
8. **Database Design** – 24
9. **Frontend Development** - 25
10. **Vue.js Framework** -25
11. **User Interface Design** -25

12.Integration with Backend - 26

## **Chapter 5: Implementation**

1. Hardware and Software Requirements - 28
2. Hardware Requirements - 28
3. Software Requirements - 29
4. AI Model Development - 30
5. Model Training - 30
6. Model Evaluation - 31
7. Backend Implementation - 32
8. API Development - 32
9. API Testing - 33
10. Frontend Implementation - 33
11. App Development - 33
12. User Experience Testing – 34

## **Chapter 6: Results and Discussion**

1. Experimental Results - 36
2. Text Detection Accuracy - 36
3. Text Recognition Accuracy - 36
4. Text-to-Speech Performance - 36
5. User Feedback and Usability - 36
6. Discussion - 37
7. Analysis of Results – 37
8. Comparison with Existing Solutions - 38
9. Future Work -38
10. References - 39

## **List of Acronyms or Abbreviations**

- AI: Artificial Intelligence
- API: Application Programming Interface
- CV: Computer Vision
- ML: Machine Learning
- TTS: Text-to-Speech
- UI: User Interface

## **List of Figures**

### **1. Chapter 3: Background**

- Figure 3.1.1 Historical Overview of Assistive Technologies - 19  
Figure 3.2.1 Existing Smartphone Solutions - 21  
Figure 3.3.1 Machine Learning Pipeline - 23  
Figure 3.3.2 Text-to-Speech Conversion Process - 25

### **2. Chapter 4: System Design**

- Figure 4.1.1 System Architecture Diagram - 27  
Figure 4.2.1 Text Detection Model Architecture - 28  
Figure 4.2.2 Text Recognition Model Architecture - 29  
Figure 4.2.3 Text-to-Audio Model Architecture - 30  
Figure 4.3.1 Django API Structure - 31  
Figure 4.3.2 Database Schema - 32  
Figure 4.4.1 Frontend User Interface - 33

### **3. Chapter 5: Implementation**

- Figure 5.1.1 Hardware Setup - 35
- Figure 5.2.1 Model Training Process - 37
- Figure 5.2.2 Model Evaluation Metrics - 38
- Figure 5.3.1 API Testing Workflow - 40
- Figure 5.4.1 Frontend Development Process - 41
- Figure 5.4.2 User Experience Testing Diagram - 42

### **4. Chapter 6: Results and Discussion**

- Figure 6.1.1 Text Detection Accuracy Results - 44
- Figure 6.1.2 Text Recognition Accuracy Results - 45
- Figure 6.1.3 Text-to-Speech Performance Results - 46
- Figure 6.1.4 User Feedback Summary - 47
- Figure 6.2.1 Results Comparison Chart - 49

## **List of Acronyms or Abbreviations**

WHO: World Health Organization

ML: MachineLearning.

CV: ComputerVision

IOT:Internet of Things.

IDE: integrated development environment.

PWM: Pulse Width Modulation.

SSD: Single Shot Detector

mAP: Mean Average Precision.

FPS:Frame per Second.

EAR: Eye Aspect Ratio

MAR: Mouth Aspect Ratio

KNN: K-Nearest Neighbors

AI: Artificial Intelligence

API: Application Programming Interface

CV: Computer Vision

ML: Machine Learning

TTS: Text-to-Speech

UI: User Interface

List of Figures

## **Abstract**

One of the main problems in the world is the prevention of accidents and road safety. Vehicle detection plays an important role in traffic control at signalized intersections. Recently, the whole world witnessed many road crashes that are considered the most common troubles that result from the lack of scanning to detect and respond to hazards, rapid driving, or lack of attention due to something inside or outside of the vehicle that leads to loss of human lives every year.

According to World Health Organization statistics, every year the lives of approximately 1.35 million people are cut short as a result of a road traffic crash. Between 20 and 50 million people suffer non-fatal injuries and some of them suffer from a disability as a result of their injuries. [1]

In order to reduce the number of deaths and harmful injuries resulting from the occurrence of this type of road accident, we propose an intelligent system that can easily detect any coming obstacle in front of the vehicle and take accurate action in a holistic manner to rescue thousands of human lives.

Experimental results on different traffic scenes show that the proposed system is effective and has high performance for real-time vehicles detection.

# **Chapter 1**

## **Introduction**

## **1.1 History**

Assistive technologies have significantly evolved over the past few decades, driven by advancements in artificial intelligence (AI) and machine learning (ML).

Early assistive devices for the visually impaired, such as Braille readers and magnifiers, primarily focused on enhancing the readability of text. With the advent of digital technologies, screen readers and voice assistants have become common, providing real-time audio feedback for digital content.

The recent surge in smartphone adoption has further accelerated the development of mobile applications designed to aid the visually impaired, leveraging the powerful processors and advanced cameras embedded in these devices. However, despite these advancements, there remains a significant gap in solutions that address real-world text recognition in outdoor environments, such as reading street banners.

## **1.2 Motivation**

The motivation behind this project stems from the desire to enhance the independence and quality of life for visually impaired individuals. Navigating urban environments presents numerous challenges, especially when it comes to reading textual information on street banners, shop signs, and public notices.

These texts often contain important information regarding directions, store names, and announcements that are crucial for day-to-day activities. Current assistive technologies fall short in providing a seamless, real-time solution for outdoor text recognition and audio conversion. This project aims to fill this gap by developing a smart app that utilizes AI models to detect, recognize, and convert text from street banners into audio, thereby enabling visually impaired users to access this

information effortlessly and independently.

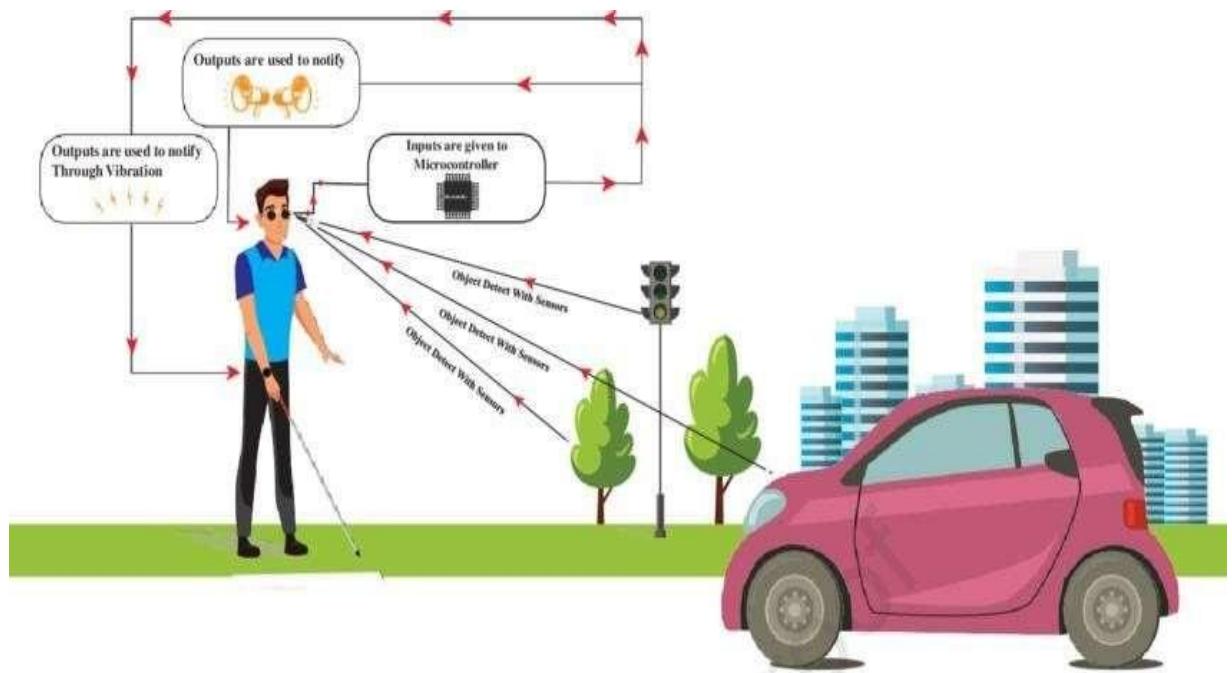
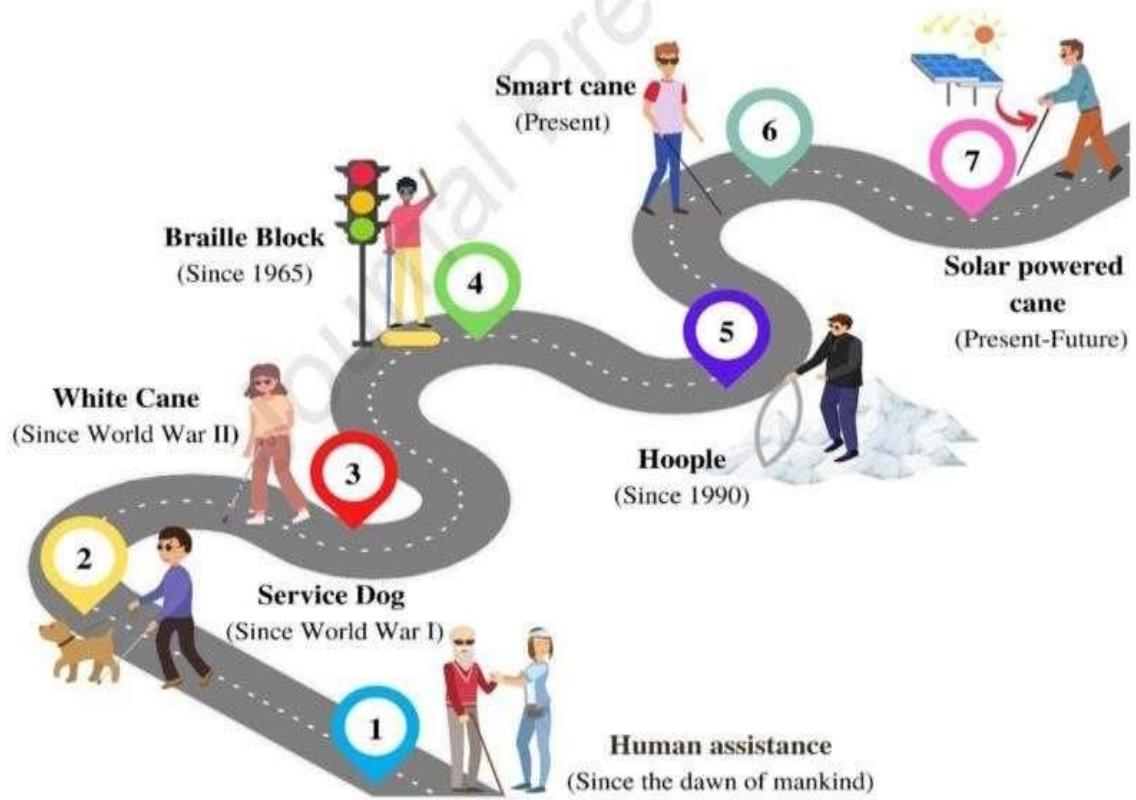


Figure 2: Application of smart glass and smart cane in real life



### **3. Problem Statement**

Visually impaired individuals face significant challenges in accessing textual information on street banners and public signs, which are essential for navigation and daily activities.

Existing assistive technologies and applications often lack the capability to accurately detect and recognize text in varying outdoor conditions and convert it to audio in real-time.

This project seeks to address this problem by developing a mobile application that integrates advanced AI models for text detection, text recognition, and text-to-speech conversion, thereby providing an accessible solution for the visually impaired to read street banners and signs.

### **4. Project Phases**

The development of the smart app is structured into several phases:

- 1. Research and Planning:** Conduct a thorough literature review on existing technologies and identify the requirements for the AI models and application framework.
- 2. Model Development:** Develop and train the AI models for text detection, text recognition, and text-to-speech conversion.
- 3. Backend Development:** Implement the Django API to handle data processing and communication between the frontend and the AI models.
- 4. Frontend Development:** Develop the user interface using Vue.js and Vite, ensuring it is user-friendly and accessible.
- 5. Integration and Testing:** Integrate the frontend, backend, and AI models. Conduct extensive testing to ensure accuracy, performance, and usability.
- 6. Deployment and Evaluation:** Deploy the application and gather user feedback to evaluate its effectiveness and identify areas for improvement.

## 5. Project Schedule

Week	Date Range	Task
1	Oct 1 - Oct 7	Write and review introduction
2-3	Oct 8 - Oct 21	Research related work
4-5	Oct 22 - Nov 4	Historical Overview
6-7	Nov 5 - Nov 18	Existing Solutions
8-9	Nov 19 - Dec 2	Text Detection
10-11	Dec 3 - Dec 16	System Architecture
12-13	Dec 17 - Dec 30	Text-to-Audio Model
14-15	Jan 3 - Jan 16	Database Design
16-17	Jan 17 - Jan 30	User Interface Design
18-19	Jan 31 - Feb 13	Hardware Requirements
20-21	Feb 14 - Feb 27	Model Training
22-23	Feb 28 - Mar 12	API Development
24-25	Mar 13 - Mar 26	App Development
26-27	Mar 27 - Apr 9	Text Detection Accuracy
28-29	Apr 10 - Apr 23	Text-to-Speech Performance
30-31	Apr 24 - May 7	Analysis of Results
32-35	May 8 - May 31	References

# **Chapter2**

## **Literature**

## **Review**

## **2.1 Related Work:**

In recent years, significant progress has been made in the field of assistive technology for the visually impaired, particularly through the application of AI and machine learning. Various research studies and projects have explored different approaches to text detection, recognition, and conversion to audio, each contributing to the development of more effective solutions.

### **Assistive Technologies**

Research on assistive technologies has highlighted the importance of providing real-time, accessible solutions to enhance the independence of visually impaired individuals. Early studies focused on hardware-based solutions such as Braille displays and magnifiers, which offered limited portability and functionality.

### **AI and Machine Learning in Assistive Applications**

The integration of AI and machine learning has revolutionized assistive applications. Notable works include the development of OCR (Optical Character Recognition) systems, which convert different types of documents, such as scanned paper documents, PDF files, or images captured by a digital camera, into editable and searchable data. Projects like Microsoft's Seeing AI and Google's Lookout app leverage AI to provide visually impaired users with audio descriptions of their surroundings, including text.

## **Limitations of Existing Solutions**

Despite advancements, existing solutions face several limitations. Many apps require continuous internet access, limiting their usability in offline scenarios. Furthermore, text detection and recognition in outdoor environments with varying lighting conditions and text styles remain challenging. These limitations highlight the need for more robust, standalone applications that can operate effectively in real-time without dependency on constant connectivity.

# Chapter 3

## Background

## **1. Assistive Technology for the Visually Impaired**

### **Historical Overview**

The journey of assistive technology for the visually impaired began with basic tools such as canes and Braille. The invention of Braille in the 19th century marked a significant milestone, providing a tactile writing system that remains widely used



today. The 20th century saw the introduction of electronic devices like the Perkins Brailler and screen readers, which expanded access to written and digital content.

### **1.2 Modern Innovations :**

Modern innovations have leveraged digital technology and AI to create more sophisticated assistive devices. Advances in natural language processing (NLP) and computer vision have led to the development of screen readers that can interpret and vocalize text from digital screens. Wearable devices equipped with cameras and sensors, such as the OrCam MyEye, offer real-time text recognition and object identification, providing greater independence to visually impaired users.

# **1. Smartphone Apps for Accessibility**

## **1. Existing Solutions**

Several smartphone applications have been developed to aid visually impaired individuals.

Microsoft's Seeing AI app uses computer vision to describe people, text, and objects.

Google's Lookout app similarly provides audio feedback about the environment, including reading text from signs and labels.

Apps like Be My Eyes connect users with volunteers for visual assistance via live video calls.

## **2. Comparative Analysis**

While these apps offer valuable services, they differ in functionality, ease of use, and dependence on internet connectivity.

Seeing AI and Lookout excel in providing automated descriptions using AI, but their accuracy can vary based on environmental factors.

Be My Eyes relies on human assistance, which, while accurate, requires an internet connection and available volunteers.

A comparative analysis highlights the need for a balanced solution that combines the autonomy of AI-driven applications with the reliability and accessibility of offline operation.

## **3. Machine Learning and Text Analysis**

- Machine Learning and Text Analysis involve using algorithms to extract meaningful insights from textual data. This field combines natural language processing techniques with machine learning models to analyze, interpret, and predict information from large volumes of text.

## **1. Text Detection:**

Text detection involves identifying the presence and location of text within an image, a critical first step in the text analysis pipeline. Modern techniques use convolutional neural networks (CNNs) to detect text regions with high accuracy, enabling a wide range of applications from document digitization to real-time assistive technologies for the visually impaired.

- Convolutional Neural Networks (CNNs) :**

CNNs are a class of deep learning algorithms particularly well-suited for image processing tasks. They consist of layers of neurons that learn to detect various features of the input image, such as edges, shapes, and textures, which are then used to identify more complex patterns like text regions. The hierarchical nature of CNNs allows them to progressively build a robust representation of text within an image, enhancing detection accuracy.

- The EAST Model :**

The Efficient and Accurate Scene Text Detector (EAST) model is one of the most advanced approaches for text detection. Unlike traditional methods that rely on a multi-stage pipeline, EAST uses a fully convolutional network that directly predicts text regions and orientations in a single forward pass. This model balances efficiency and accuracy, making it suitable for real-time applications. Key features of EAST include:

- Direct Region Prediction:** EAST predicts the presence and orientation of text regions directly from the input image, eliminating the need for intermediate processing steps.

- **Quadrilateral Bounding Boxes:** The model outputs quadrilateral bounding boxes, which can more accurately encompass text regions compared to traditional rectangular boxes, especially for slanted or curved text.
- **Real-Time Performance:** Optimized for speed, EAST can process images quickly, making it ideal for applications that require real-time text detection.

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure for "03\_HANDWRITING\_RECOGNITION" containing "IAM\_Words" and "words" folders, with various sub-folders like "a01" through "f02".
- Code Editor:** Displays two files: "train.py" and "configs.py". "train.py" contains code related to TensorFlow and GPU configuration.
- Terminal:** Shows the command "python3 inferenceModel.py" being run, with output indicating processing of 2171 images. The output includes labels and predictions such as "upon", "not", "appears", "West", and "shop".
- Image Preview:** A preview window titled "Image" shows a handwritten word "shop" in a stylized font.
- Status Bar:** Shows the current file is "train.py - 03\_handwriting\_recognition - Visual Studio Code", the date and time as "Dec 22 8:35 PM", and the status bar at the bottom indicates "Ln 137, Col 70" and "Python 3.10.12 64-bit".

## **2. Text Recognition :**

Text recognition, or optical character recognition (OCR), converts detected text regions into machine-readable text. This process is essential for transforming visual text data into a format that can be used for further processing, such as text-to-speech conversion or data entry automation.

- Recurrent Neural Networks (RNNs) and CNNs :**

OCR systems often use a combination of CNNs and recurrent neural networks (RNNs) to achieve high accuracy in text recognition. CNNs are responsible for extracting visual features from the text regions, while RNNs handle the sequential nature of text data, capturing dependencies between characters or words.

- CNNs for Feature Extraction:** By analyzing the visual features of text regions, CNNs generate detailed feature maps that highlight distinctive patterns in the characters.
- RNNs for Sequence Modeling:** RNNs, particularly Long Short-Term Memory (LSTM) networks, are used to process these feature maps sequentially, learning the relationships between consecutive characters or words. This sequence modeling is crucial for accurate text recognition, especially for handwritten or cursive text.

- Tesseract OCR :**

Developed by Google, Tesseract is a widely used open-source OCR engine. It supports multiple languages and text formats, making it versatile for various applications. Key features of Tesseract include:

- **Multi-Language Support:** Tesseract can recognize text in over 100 languages, providing broad applicability across different regions and use cases.
  - **Training Capabilities:** Users can train Tesseract on custom datasets, enhancing its performance for specific types of text or languages.
  - **Integration with Other Tools:** Tesseract can be easily integrated with other software and libraries, such as OpenCV, for comprehensive image processing workflows.

The screenshot shows a Visual Studio Code window with the following details:

- Title Bar:** Activities - Visual Studio Code - demo.py - 04\_deep-text-recognition-benchmark-master - Visual Studio Code
- File Explorer:** Shows a file tree with a single file named "demo.py".
- Code Editor:** Displays the content of "demo.py". The code is a Python script for deep text recognition, utilizing PyTorch's CTCLabelConverter and AttrLabelConverter.
- Output Panel:** Shows the output of the command "python demo.py", indicating the script has been launched.
- Status Bar:** Shows the current file path as "demo.py - 04\_deep-text-recognition-benchmark-master", the line number as "Ln 19, Col 10", the space count as "Spaces: 4", the font as "UTF-8", the language as "Python", the file size as "3 10.12.64.btt", and the file type as "Python".

```
demo.py x
demo.py demo
1 import string
2 import argparse
3
4 import torch
5 import torch.backends.cudnn as cudnn
6 import torch.utils.data
7 import torch.nn.functional as F
8
9 from utils import CTCLabelConverter, AttrLabelConverter
10 from dataset import RawDataset, AlignCollate
11 from model import Model
12 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
13
14
15 def demo(opt):
16     """ model configuration """
17     if 'CTC' in opt.Prediction:
18         converter = CTCLabelConverter(opt.character)
19     else:
20         converter = AttrLabelConverter(opt.character)
21     opt.num_class = len(converter.character)
22
23     if opt.rgb:
24         opt.input_channel = 3
25     model = Model(opt)
26     print('model input parameters', opt.imgH, opt.imgW, opt.num_fiducial, opt.input_channel, opt.output_channel,
27           opt.hidden_size, opt.num_class, opt.batch_max_length, opt.Transformation, opt.FeatureExtraction,
28           opt.SequenceModeling, opt.Prediction)
29     model = torch.nn.DataParallel(model).to(device)
30
31     # load model
32     print('loading pretrained model from %s' % opt.saved_model)
33     model.load_state_dict(torch.load(opt.saved_model, map_location=device))
34
35     AlignCollate_demo = AlignCollate(imgH=opt.imgH, imgW=opt.imgW, keep_ratio_with_pad=opt.PAD)
36     demo_data = RawDataset(root=opt.image_folder, opt=opt) # use RawDataset
37     demo_loader = torch.utils.data.DataLoader(
38         demo_data, batch_size=opt.batch_size,
39         shuffle=False,
40         num_workers=int(opt.workers),
41         collate_fn=AlignCollate_demo, pin_memory=True)
42
43     # predict
44     model.eval()
45     with torch.no_grad():
46         for image_tensors, image_path_list in demo_loader:
47             batch_size = image_tensors.size(0)
48             image_tensors = image_tensors.to(device)
49             # For max length prediction
50             length_for_pred = torch.IntTensor([opt.batch_max_length] * batch_size).to(device)
51             text_for_pred = torch.LongTensor(batch_size, opt.batch_max_length + 1).fill_(0).to(device)
52
53
54             if 'CTC' in opt.Prediction:
55                 preds = model(image_tensors, text_for_pred)
56
57                 # Select max probability (greedy decoding) then decode index to character
58                 preds_size = torch.IntTensor([preds.size(1)] * batch_size)
59                 preds_index = preds.max(2)[1]
60
61                 # Remove blank char
62                 pred_chars = converter.decode(preds_index, preds_size)
```

### **3. Text-to-Speech Conversion :**

Text-to-speech (TTS) conversion translates text into spoken words, enabling applications that provide auditory feedback from written content. Modern TTS

systems use deep learning models to produce natural-sounding speech, significantly improving the user experience for visually impaired individuals.

## **Deep Learning Models for TTS :**

Deep learning models, such as Google's WaveNet and Tacotron, have revolutionized TTS by generating high-quality audio that captures the nuances of human speech, including tone, intonation, and rhythm.

- **WaveNet:** Developed by DeepMind, WaveNet generates raw audio waveforms from text inputs. It uses a neural network trained on large datasets of speech audio, learning to produce realistic and expressive speech. WaveNet captures subtle variations in speech patterns, making the generated audio sound more natural.
- **Tacotron:** Tacotron is an end-to-end TTS model that directly converts text to spectrograms, which are then transformed into audio waveforms using a vocoder like WaveNet. Tacotron's architecture allows it to learn the mapping from text to speech features, producing high-fidelity audio with natural prosody.

## **VALL-E-X :**

VALL-E-X is a cutting-edge TTS model that leverages the latest advancements in deep learning to generate high-quality, natural-sounding speech from text inputs. Developed with a focus on versatility and performance, VALL-E-X is designed to deliver exceptional audio clarity and intelligibility, making it an ideal choice for assistive technologies.

## Key Features of VALL-E-X:

- **Multi-Voice Support:** VALL-E-X can generate speech in multiple voices, accents, and languages, providing a customizable auditory experience for users.
- **Emotion and Prosody Control:** The model can modulate tone, pitch, and rhythm to convey different emotions and speaking styles, enhancing the expressiveness of the generated speech.
- **Real-Time Processing:** Optimized for speed, VALL-E-X can produce audio in real-time, ensuring minimal latency and immediate feedback for users.

	Previous Systems	VALL-E X
Intermediate representation	Mel spectrogram	Audio codec codes
Training data	< 13K hours	70K hours
Speech accent	Foreign	Native
Speaker similarity	Relative low	High
In-context learning	✗	✓
Zero-shot cross-lingual TTS	✗	✓

Our VALL-E X is trained using bilingual speech-transcription (ASR) data. The Chinese ASR data are from WenetSpeech [Zhang et al., 2022a] containing 10,000+ hours of multi-domain labeled speech. The English ASR data are from LibriLight [Kahn et al., 2020] containing about 60,000 hours of unlabeled speech, whose speech data are collected from audiobooks. We train a Kaldi4 ASR model on the labeled Librispeech [Panayotov et al., 2015] dataset to generate the pseudo transcripts for the unlabeled LibriLight speech. To train the speech recognition & translation model for S2ST, we also use additional MT and ST data. The MT data are from AI Challenger<sup>5</sup>, OpenSubtitles2018<sup>6</sup> and WMT2020<sup>7</sup>, which contain about 13M, 10M, and 50M sentence pairs in conversion, drama<sup>8</sup>, and news domains, respectively

## **Key Features of Modern TTS Systems**

- **Natural-Sounding Speech:** By leveraging deep learning techniques, modern TTS systems generate speech that closely mimics human speech, enhancing the listening experience.
- **Customization:** These systems can be customized to produce different voices, accents, and speaking styles, catering to diverse user preferences.
- **Intelligibility:** High-quality TTS models ensure that the generated speech is clear and easy to understand, which is crucial for accessibility applications.

In summary, the integration of advanced machine learning techniques in text detection, recognition, and TTS conversion has significantly enhanced the capabilities of assistive technologies for the visually impaired. These innovations offer greater independence and accessibility, transforming how visually impaired individuals interact with the world.

The screenshot shows a Linux desktop environment with a window titled "Visual Studio Code". The code editor displays a Python file named "Basics.py". The code itself is as follows:

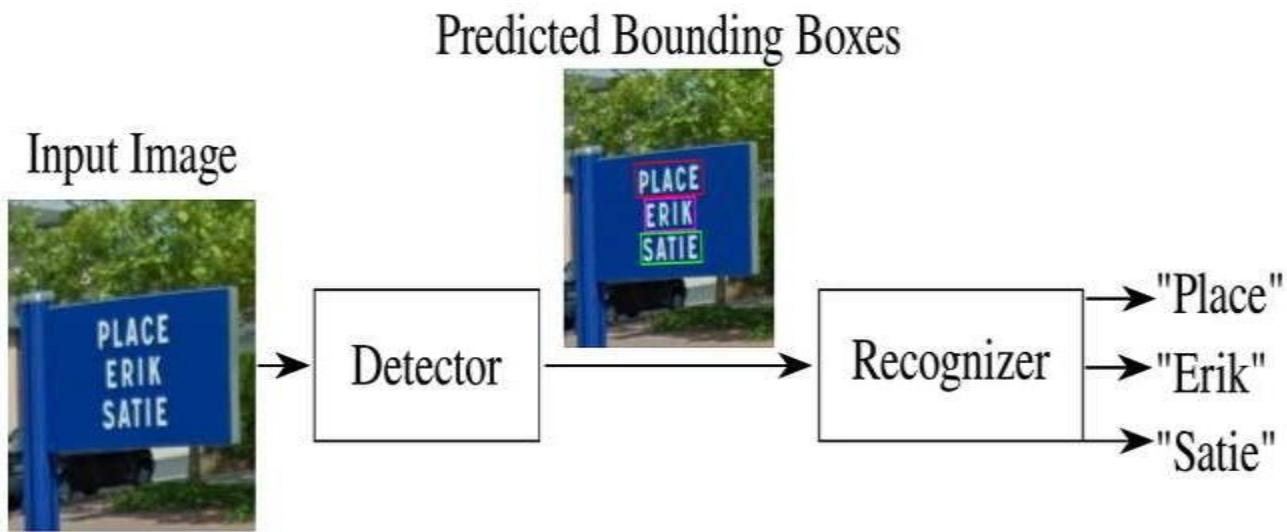
```
Activities > VisualStudio Code
File Edit Selection View Go Run Terminal Help
Basics.py ...
1 import sys
2 import os
3 import glob
4 from utils.generation import SAMPLE_RATE, generate_audio, preload_models
5 from scipy.io.wavfile import write as write_wav
6
7 def generate_audio_from_text(text_prompt, accent='no-accent'):
8     # Download and load all necessary models
9     preload_models()
10
11     # Generate audio from the given text prompt
12     audio_array = generate_audio(text_prompt, prompt="vctk_3")
13
14     return audio_array
15
16 import datetime
17
18 def save_audio_to_file(text_prompt, audio_array):
19     def delete_wav_files(directory):
20         # Get a list of all wav files in the directory
21         wav_files = glob.glob(os.path.join(directory, '*.wav'))
22
23         # Delete each file
24         for file in wav_files:
25             os.remove(file)
26
27     # Call the function with the directory path
28     delete_wav_files('/home/marwan/project/src/images/audio')
29     file_name = text_prompt[:10] + f'{datetime.datetime.now().month}.wav'
30     file_path = os.path.join('/home/marwan/project/src/images/audio', file_name)  # Save the audio to a WAV file
31     write_wav(file_path, SAMPLE_RATE, audio_array)
32
33     return file_path
34
35 if __name__ == '__main__':
36     # Read text prompt from standard input (stdin)
37     text_prompt = sys.stdin.read().strip()
38
39     # Generate audio based on the received text prompt
40     audio_array = generate_audio_from_text(text_prompt, accent='no-accent')
41
42     # Save audio to a WAV file
43     audio_file_path = save_audio_to_file(text_prompt, audio_array)
44
45     # Output the path to the saved audio file
46     print(audio_file_path)
47
```

The status bar at the bottom indicates the file has 47 lines, 1 space, and is in UTF-8 encoding.

## 4.Evaluation Metrics :

Evaluation metrics are critical for assessing the performance of text detection, recognition, and TTS systems.

- Common metrics for text detection include precision, recall, and the F1 score, which balance accuracy and completeness. OCR systems are evaluated using word error rate (WER) and character error rate (CER).
- TTS systems are assessed based on naturalness, intelligibility, and mean opinion score (MOS), which measures user satisfaction with the generated speech.



# Backend Development

## Django API

The backend of the application is powered by a Django framework, which handles the server-side logic, including API requests, model inference, and data management.

## Key Components

- API Endpoints:** Defined for image upload, text processing, and audio retrieval.
- Request Handling:** Manages incoming requests, processes images, and coordinates with AI models.
- Response Generation:** Sends back processed text or audio results to the frontend.

## Database Design

A robust database is essential for storing user interactions, processed text data, and application metadata. The database schema is designed to efficiently manage and retrieve information as needed.

## Key Components

- **User Data:** Logs user activities and preferences.
  - **Image Data:** Stores images and associated metadata.
  - **Text Data:** Maintains recognized text entries and their processing status.

Activities < Visual Studio Code en

File Edit Selection View Go Run Terminal Help

views.py - project - Visual Studio Code

```
src > views.py > detect_text_in_image
You, 3 weeks ago | Author (You)
1 import os
2 from rest_framework import views
3 from rest_framework.response import Response
4 from rest_framework import status
5 from rest_framework.permissions import AllowAny
6 import cv2
7 import easyocr
8 from django.conf import settings
9 import subprocess
10 import run_pip
11 import shlex
12 import re # Import re module for regular expressions
13 import datetime
14 import time
15 import glob
16
17 # Constants
18 SAVE_DIR = os.path.abspath(os.path.join(os.path.dirname(__file__), '../', 'images'))
19 LANGUAGE = ['en'] # Language for easyocr
20
21
22 def detect_text_in_image(image_path):
23     try:
24         # Read Image
25         img = cv2.imread(image_path)
26
27         # Detect text on image using easyocr
28         reader = easyocr.Reader(LANGUAGE)
29         text_results = reader.readText(img)
30
31         return text_results
32     except Exception as e:
33         print(f"Error detecting text: {e}")
34     return []
35
36
37 def process_text_results(text_results):
38     try:
39         # Process the text results to generate a response
40         response_text = ''.join([text_result[1] for text_result in text_results])
41         return response_text
42     except Exception as e:
43         print(f"Error processing text results: {e}")
44     return ''
45
46
47 class ImageOCRView(views.APIView):
48     authentication_classes = []
49     permission_classes = [AllowAny]
50
51     @staticmethod
52     def sanitize_file_name(file_name):
53         # Remove any characters that are not alphanumeric, periods, or underscores with an underscore
54         sanitized_name = re.sub(r'[^\\w.-_]', '_', file_name)
55         return sanitized_name
56
57
58     def post(self, request, *args, **kwargs):
59         try:
60             file = request.FILES['image']
61             file_name = self.sanitize_file_name(file.name)
62             file_path = os.path.join(SAVE_DIR, file_name)
63             file.save(file_path)
64             command = f'{settings.PIP_PATH} install opencv-python-headless'
65             result = subprocess.run(shlex.split(command), capture_output=True, text=True)
66             if result.returncode != 0:
67                 print(result.stderr)
68             else:
69                 print(result.stdout)
70
71             response = {
72                 'file_name': file_name,
73                 'text_results': text_results
74             }
75             return Response(response, status=status.HTTP_201_CREATED)
76         except Exception as e:
77             print(f"Error processing file: {e}")
78             return Response({'error': str(e)}, status=status.HTTP_400_BAD_REQUEST)
```

A screenshot of the Visual Studio Code interface. The title bar reads "Activities > Visual Studio Code" and "urls.py · project - Visual Studio Code". The status bar at the bottom shows "Ln 10, Col 1" and "UTF-8". The main editor area contains Python code for a Django URL pattern:

```
from django.urls import path, include
from . import views
urlpatterns = [
    path('pass-image/', views.ImageOCRView.as_view())
]
```

# Frontend Development

## Vue.js Framework

The frontend application is built using the Vue.js framework, known for its flexibility and ease of integration. Vue.js allows for the creation of a responsive and accessible user interface.

## Key Components

- **Image Capture:** Functionality to capture images using the device camera.
- **User Interaction:** Simple, intuitive controls for navigation and operation.
- **Audio Playback:** Real-time playback of converted text to speech.

## User Interface Design

User interface design focuses on accessibility and ease of use for visually impaired users. Key considerations include:

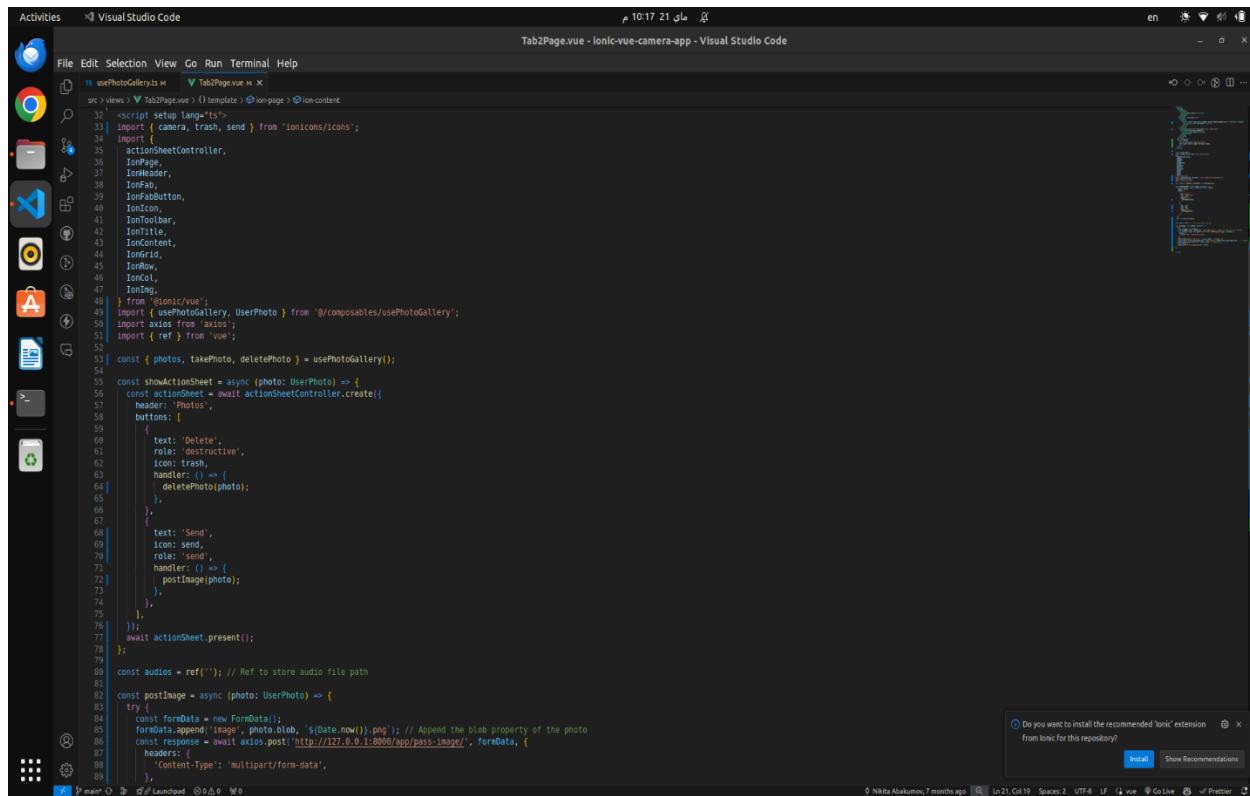
- **Voice Commands:** Enabling hands-free operation.
- **Large Buttons:** Simplified navigation with easily identifiable touch targets.
- **Contrast and Text Size:** Ensuring readability for users with partial vision.

# Integration with Backend

Seamless integration between the frontend and backend is crucial for real-time performance. The frontend communicates with the Django API to send images and receive processed text and audio.

## Key Components

- **API Communication:** Efficient handling of requests and responses.
- **Error Handling:** Robust mechanisms to manage and inform users of any issues during processing.
- **Real-Time Updates:** Ensuring prompt feedback and minimal latency in text-to-speech conversion.



A screenshot of the Visual Studio Code interface. The title bar shows "Activities" and "Visual Studio Code". The main area displays a file named "usePhotoGallery.ts" with the following code:

```
src > News > Tab2Page.vue > () template > ion-page > ion-content
32   <script setup lang="ts">
33     import { camera, trash, send } from 'ionicons/icons';
34     import { alertController, IonPage, IonHeader, IonContent, IonGrid, IonRow, IonCol, IonImg } from '@ionic/vue';
35     import { usePhotoGallery, UserPhoto } from '@/composables/usePhotoGallery';
36     import axios from 'axios';
37     import { ref } from 'vue';
38     const photos = ref([]);
39     const takePhoto = usePhotoGallery();
40     const actionSheetSheet = async (photo: UserPhoto) => {
41       const actionSheet = await alertController.create({
42         header: 'Photos',
43         buttons: [
44           {
45             text: 'Delete',
46             icon: 'trash',
47             handler: () => {
48               deletePhoto(photo);
49             },
50           },
51           {
52             text: 'Send',
53             icon: 'send',
54             role: 'send',
55             handler: () => {
56               postImage(photo);
57             },
58           },
59         ],
60       });
61       await actionSheet.present();
62     };
63     const postImage = async (photo: UserPhoto) => {
64       try {
65         const formData = new FormData();
66         formData.append('image', photo.blob, `${Date.now()}.png`); // Append the blob property of the photo
67         const response = await axios.post(`http://127.0.0.1:8000/app/pass-image/`, formData, {
68           headers: {
69             'Content-Type': 'multipart/form-data',
70           },
71         });
72       } catch (error) {
73         console.error(error);
74       }
75     };
76   };
77   await actionSheetSheet();
78 }
79
80 const audios = ref([]); // Ref to store audio file path
81
82 const postImage = async (photo: UserPhoto) => {
83   try {
84     const formData = new FormData();
85     formData.append('image', photo.blob, `${Date.now()}.png`); // Append the blob property of the photo
86     const response = await axios.post(`http://127.0.0.1:8000/app/pass-image/`, formData, {
87       headers: {
88         'Content-Type': 'multipart/form-data',
89       },
90     });
91   } catch (error) {
92     console.error(error);
93   }
94 }
```

The bottom right corner of the code editor has a small modal window asking if the user wants to install the "Ionic" extension from Ionic for this repository. The modal includes "Install" and "Show Recommendations" buttons.

# **Chapter 4**

# **System Design**

### **3.1 System Architecture**

The system architecture for the smart app designed to assist the visually impaired is a multi-layered structure that integrates AI models with robust backend and user-friendly frontend components. The architecture ensures seamless interaction between different modules, facilitating real-time text detection, recognition, and audio conversion.

### **3.2 Overview**

- 1. User Interface:** The frontend application built using Vue.js, which captures images and provides audio output.
- 2. Backend Server:** A Django-based API that processes images, invokes AI models, and manages data.
- 3. AI Models:** Machine learning models for text detection, text recognition, and text-to-speech conversion, each serving a specific function in the text analysis pipeline.
- 4. Database:** A structured storage sy

#### **2. AI Models**

##### **1. Text Detection Model**

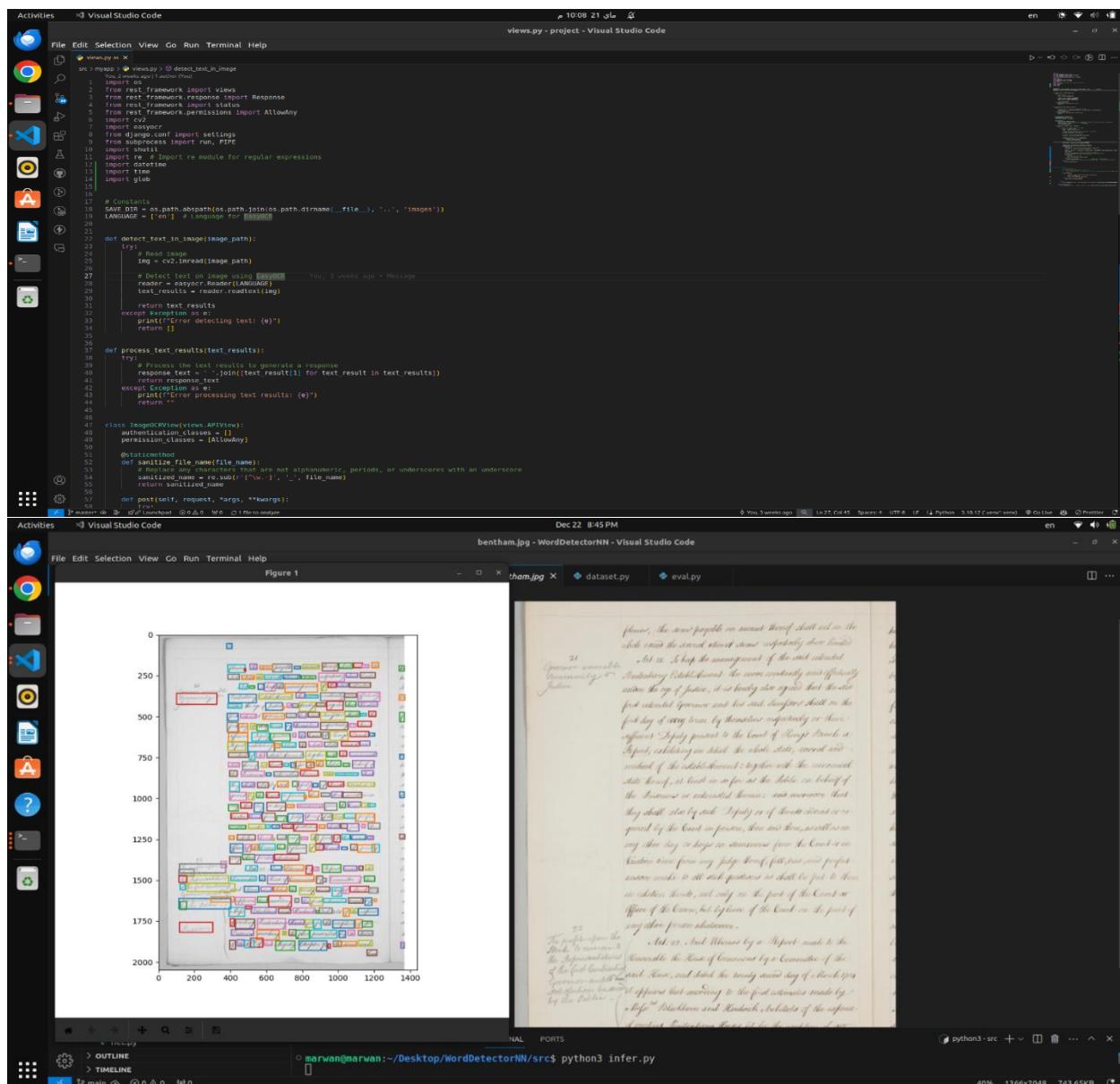
The text detection model is responsible for identifying and localizing text regions within an image. This model employs a convolutional neural network (CNN) architecture, such as the EAST (Efficient and Accurate Scene Text) detector, which is optimized for real-time processing and high accuracy.

## Key Components

- **Input:** Raw image captured by the user.
- **Processing:** The model scans the image and highlights text regions.
- **Output:** Coordinates of bounding boxes around detected text areas.

## Training Data

The model is trained on a diverse dataset containing images with various text styles, fonts, and lighting conditions to enhance robustness and accuracy.



## 2. Text Recognition Model

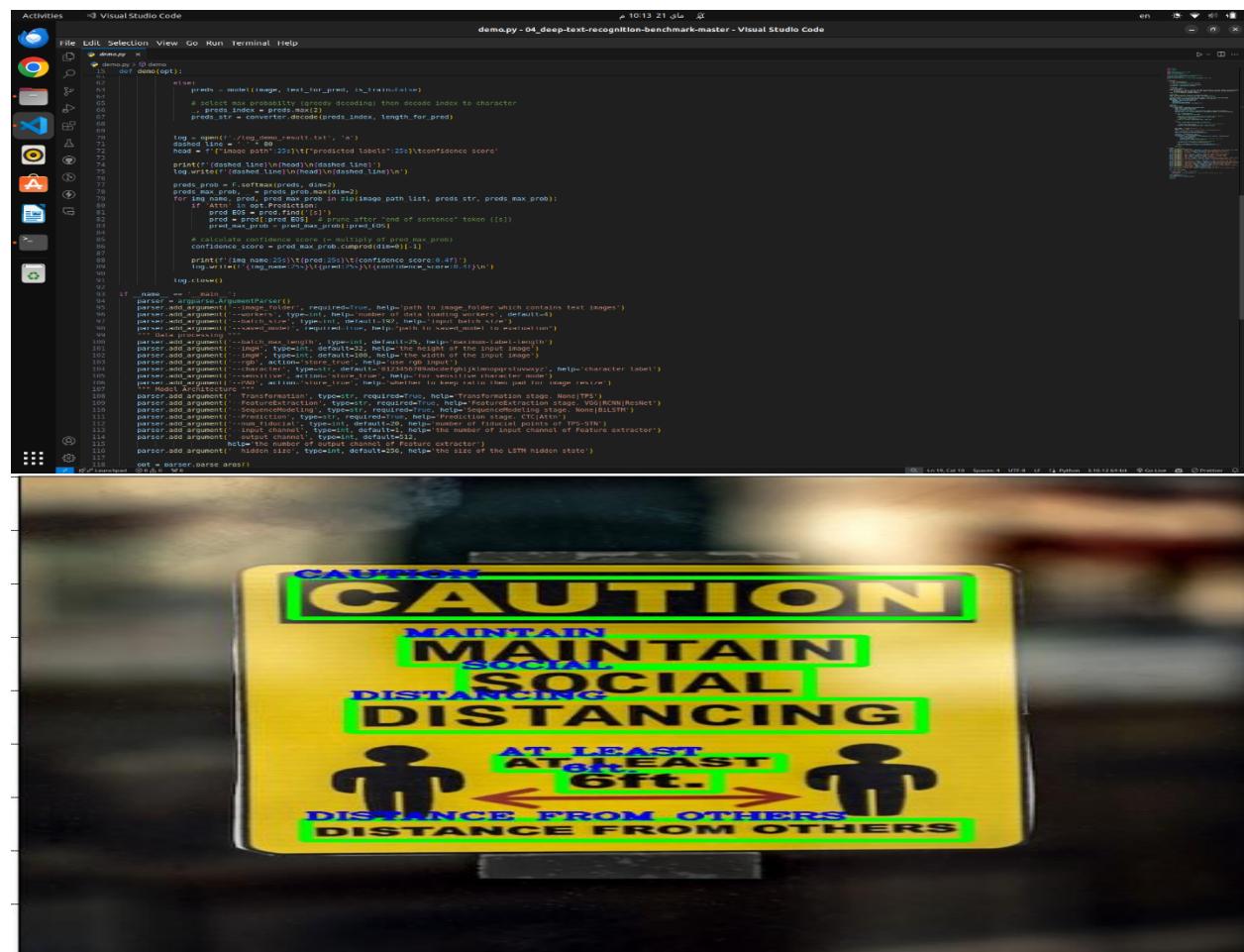
Once text regions are detected, the text recognition model converts these regions into machine-readable text. This model utilizes a combination of CNNs and Recurrent Neural Networks (RNNs) to accurately recognize characters from the detected text regions.

### Key Components:

- Input:** Cropped image regions containing text.
- Processing:** The model uses OCR techniques to interpret and convert the text images into strings.
- Output:** Recognized text in a readable format.

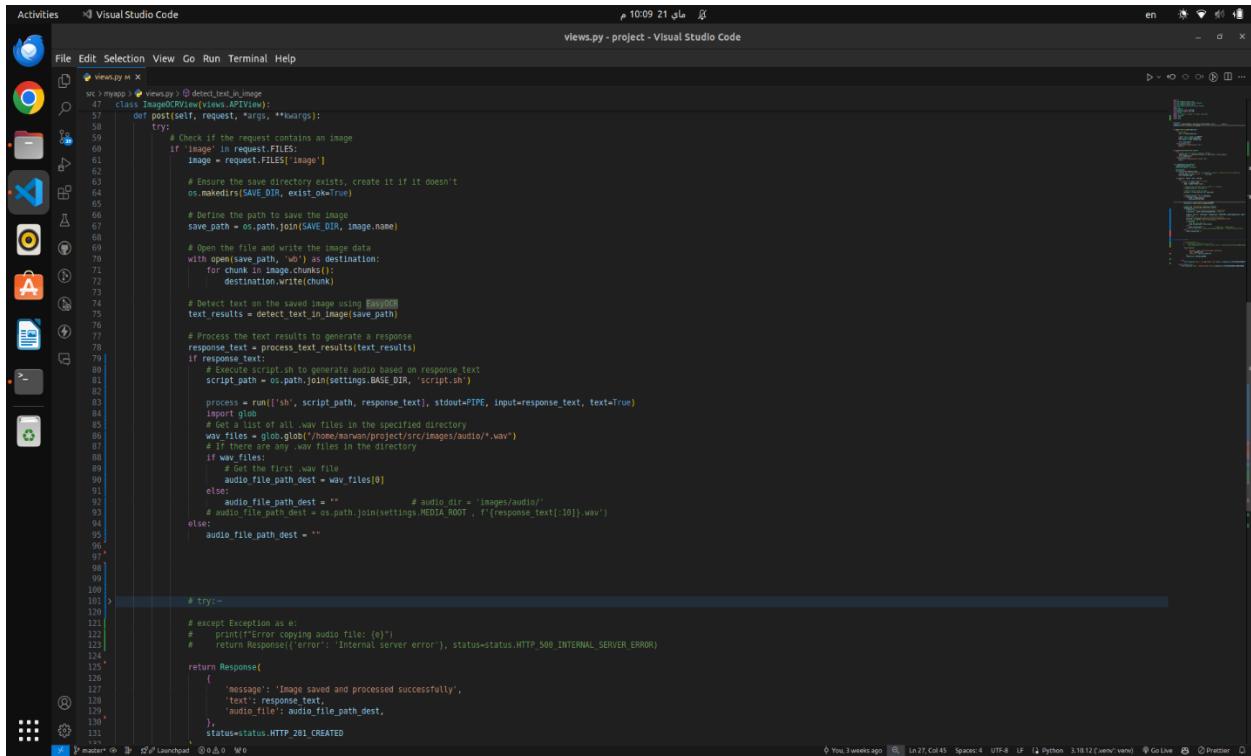
### Training Data

Training involves datasets with labeled text images, covering multiple languages and character sets to ensure broad applicability.



### 3. Text-to-Audio Model

The final stage involves converting the recognized text into speech. This is achieved using a text-to-speech (TTS) model, such as Google's WaveNet or Tacotron, which generates natural-sounding audio.



A screenshot of the Visual Studio Code interface. The title bar says "Activities > Visual Studio Code". The main area shows a Python file named "views.py" with the following code:

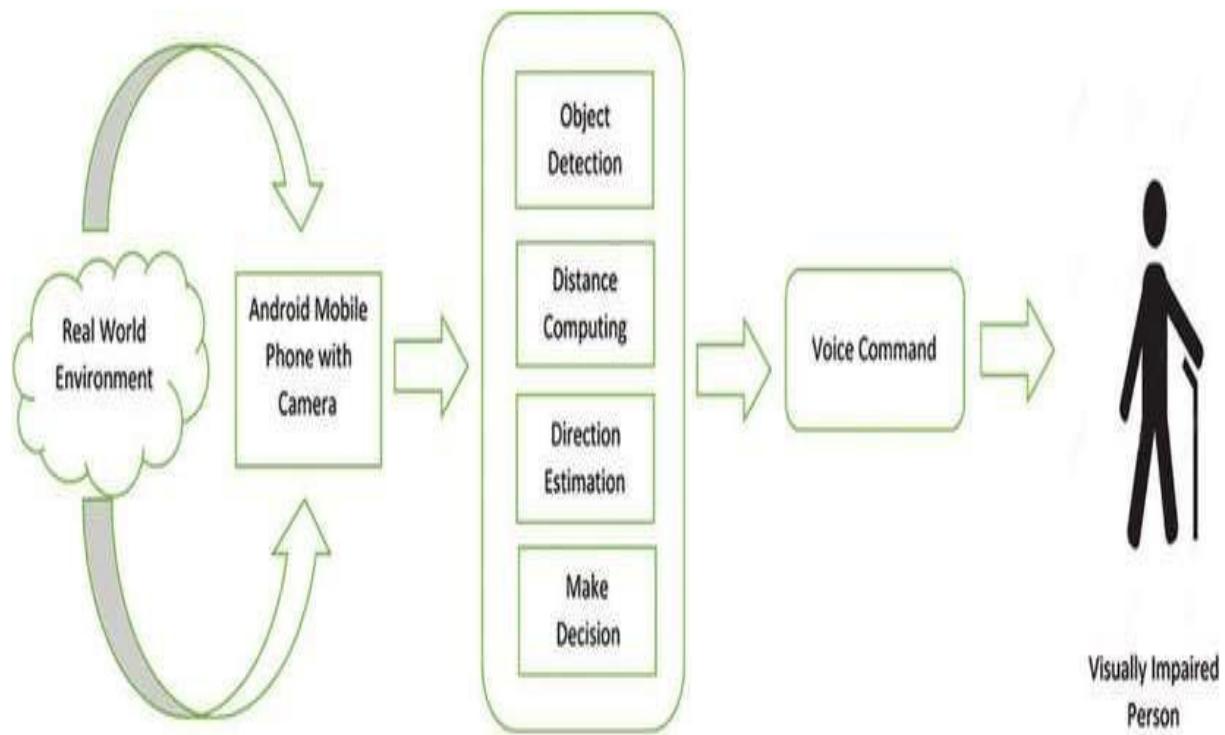
```
src>maps>views>@detect_text_in_image
47 class ImageOCRView(views.APIView):
48     def post(self, request, *args, **kwargs):
49         try:
50             # Check if the request contains an image
51             if 'image' in request.FILES:
52                 image = request.FILES['image']
53 
54             # Ensure the save directory exists, create it if it doesn't
55             os.makedirs(SAVE_DIR, exist_ok=True)
56 
57             # Define the path to save the image
58             save_path = os.path.join(SAVE_DIR, image.name)
59 
60             # Open the file and write the image data
61             with open(save_path, 'wb') as destination:
62                 for chunk in image.chunks():
63                     destination.write(chunk)
64 
65             # Detect text on the saved image using Tesseract
66             text_results = detect_text_in_image(save_path)
67 
68             # Process the text results to generate a response
69             response_text = process_text_results(text_results)
70 
71             # Execute script.sh to generate audio based on response text
72             script_path = os.path.join(settings.BASE_DIR, 'script.sh')
73 
74             process = run(['sh', script_path, response_text], stdout=PIPE, input=response_text, text=True)
75             import glob
76             # Get a list of all .wav files in the specified directory
77             wav_files = glob.glob(f'{settings.MEDIA_ROOT}/images/audio/*.wav')
78             # If there are any .wav files in the directory
79             if wav_files:
80                 # Get the first .wav file
81                 audio_file_path_dest = wav_files[0]
82             else:
83                 # audio_file_path_dest = '' # audio dir = 'images/audio/'
84                 # audio_file_path_dest = os.path.join(settings.MEDIA_ROOT, f'{response_text[:10]}.wav')
85             else:
86                 audio_file_path_dest = ''
87 
88             # try:-#
89             # except Exception as e:
90             #     print(f'Error copying audio file: {e}')
91             #     return Response({'error': 'internal server error'}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
92 
93             return Response({
94                 'message': 'Image saved and processed successfully',
95                 'text': response_text,
96                 'audio_file': audio_file_path_dest,
97             },
98             status=status.HTTP_201_CREATED)
99 
100 
101 
102 
103 
104 
105 
106 
107 
108 
109 
110 
111 
112 
113 
114 
115 
116 
117 
118 
119 
120 
121 
122 
123 
124 
125 
126 
127 
128 
129 
130 
131 
132 
133 
134 
135 
136 
137 
138 
139 
140 
141 
142 
143 
144 
145 
146 
147 
148 
149 
150 
151 
152 
153 
154 
155 
156 
157 
158 
159 
160 
161 
162 
163 
164 
165 
166 
167 
168 
169 
170 
171 
172 
173 
174 
175 
176 
177 
178 
179 
180 
181 
182 
183 
184 
185 
186 
187 
188 
189 
190 
191 
192 
193 
194 
195 
196 
197 
198 
199 
200 
201 
202 
203 
204 
205 
206 
207 
208 
209 
210 
211 
212 
213 
214 
215 
216 
217 
218 
219 
220 
221 
222 
223 
224 
225 
226 
227 
228 
229 
230 
231 
232 
233 
234 
235 
236 
237 
238 
239 
240 
241 
242 
243 
244 
245 
246 
247 
248 
249 
250 
251 
252 
253 
254 
255 
256 
257 
258 
259 
260 
261 
262 
263 
264 
265 
266 
267 
268 
269 
270 
271 
272 
273 
274 
275 
276 
277 
278 
279 
280 
281 
282 
283 
284 
285 
286 
287 
288 
289 
290 
291 
292 
293 
294 
295 
296 
297 
298 
299 
300 
301 
302 
303 
304 
305 
306 
307 
308 
309 
310 
311 
312 
313 
314 
315 
316 
317 
318 
319 
320 
321 
322 
323 
324 
325 
326 
327 
328 
329 
330 
331 
332 
333 
334 
335 
336 
337 
338 
339 
340 
341 
342 
343 
344 
345 
346 
347 
348 
349 
350 
351 
352 
353 
354 
355 
356 
357 
358 
359 
360 
361 
362 
363 
364 
365 
366 
367 
368 
369 
370 
371 
372 
373 
374 
375 
376 
377 
378 
379 
380 
381 
382 
383 
384 
385 
386 
387 
388 
389 
389 
390 
391 
392 
393 
394 
395 
396 
397 
398 
399 
399 
400 
401 
402 
403 
404 
405 
406 
407 
408 
409 
409 
410 
411 
412 
413 
414 
415 
416 
417 
418 
419 
419 
420 
421 
422 
423 
424 
425 
426 
427 
428 
429 
429 
430 
431 
432 
433 
434 
435 
436 
437 
438 
439 
439 
440 
441 
442 
443 
444 
445 
446 
447 
448 
449 
449 
450 
451 
452 
453 
454 
455 
456 
457 
458 
459 
459 
460 
461 
462 
463 
464 
465 
466 
467 
468 
469 
469 
470 
471 
472 
473 
474 
475 
476 
477 
478 
479 
479 
480 
481 
482 
483 
484 
485 
486 
487 
488 
489 
489 
490 
491 
492 
493 
494 
495 
496 
497 
498 
499 
499 
500 
501 
502 
503 
504 
505 
506 
507 
508 
509 
509 
510 
511 
512 
513 
514 
515 
516 
517 
518 
519 
519 
520 
521 
522 
523 
524 
525 
526 
527 
528 
529 
529 
530 
531 
532 
533 
534 
535 
536 
537 
538 
539 
539 
540 
541 
542 
543 
544 
545 
546 
547 
548 
549 
549 
550 
551 
552 
553 
554 
555 
556 
557 
558 
559 
559 
560 
561 
562 
563 
564 
565 
566 
567 
568 
569 
569 
570 
571 
572 
573 
574 
575 
576 
577 
578 
579 
579 
580 
581 
582 
583 
584 
585 
586 
587 
588 
589 
589 
590 
591 
592 
593 
594 
595 
596 
597 
598 
599 
599 
600 
601 
602 
603 
604 
605 
606 
607 
608 
609 
609 
610 
611 
612 
613 
614 
615 
616 
617 
618 
619 
619 
620 
621 
622 
623 
624 
625 
626 
627 
628 
629 
629 
630 
631 
632 
633 
634 
635 
636 
637 
638 
639 
639 
640 
641 
642 
643 
644 
645 
646 
647 
648 
649 
649 
650 
651 
652 
653 
654 
655 
656 
657 
658 
659 
659 
660 
661 
662 
663 
664 
665 
666 
667 
668 
669 
669 
670 
671 
672 
673 
674 
675 
676 
677 
678 
679 
679 
680 
681 
682 
683 
684 
685 
686 
687 
688 
689 
689 
690 
691 
692 
693 
694 
695 
696 
697 
698 
699 
699 
700 
701 
702 
703 
704 
705 
706 
707 
708 
709 
709 
710 
711 
712 
713 
714 
715 
716 
717 
718 
719 
719 
720 
721 
722 
723 
724 
725 
726 
727 
728 
729 
729 
730 
731 
732 
733 
734 
735 
736 
737 
738 
739 
739 
740 
741 
742 
743 
744 
745 
746 
747 
748 
749 
749 
750 
751 
752 
753 
754 
755 
756 
757 
758 
759 
759 
760 
761 
762 
763 
764 
765 
766 
767 
768 
769 
769 
770 
771 
772 
773 
774 
775 
776 
777 
778 
779 
779 
780 
781 
782 
783 
784 
785 
786 
787 
788 
789 
789 
790 
791 
792 
793 
794 
795 
796 
797 
798 
799 
799 
800 
801 
802 
803 
804 
805 
806 
807 
808 
809 
809 
810 
811 
812 
813 
814 
815 
816 
817 
818 
819 
819 
820 
821 
822 
823 
824 
825 
826 
827 
828 
829 
829 
830 
831 
832 
833 
834 
835 
836 
837 
838 
839 
839 
840 
841 
842 
843 
844 
845 
846 
847 
848 
849 
849 
850 
851 
852 
853 
854 
855 
856 
857 
858 
859 
859 
860 
861 
862 
863 
864 
865 
866 
867 
868 
869 
869 
870 
871 
872 
873 
874 
875 
876 
877 
878 
879 
879 
880 
881 
882 
883 
884 
885 
886 
887 
888 
888 
889 
889 
890 
891 
892 
893 
894 
895 
896 
897 
898 
899 
899 
900 
901 
902 
903 
904 
905 
906 
907 
908 
909 
909 
910 
911 
912 
913 
914 
915 
916 
917 
918 
919 
919 
920 
921 
922 
923 
924 
925 
926 
927 
928 
929 
929 
930 
931 
932 
933 
934 
935 
936 
937 
938 
939 
939 
940 
941 
942 
943 
944 
945 
946 
947 
948 
949 
949 
950 
951 
952 
953 
954 
955 
956 
957 
958 
959 
959 
960 
961 
962 
963 
964 
965 
966 
967 
968 
969 
969 
970 
971 
972 
973 
974 
975 
976 
977 
978 
979 
979 
980 
981 
982 
983 
984 
985 
986 
987 
988 
988 
989 
989 
990 
991 
992 
993 
994 
995 
996 
997 
998 
999 
999 
1000 
1001 
1002 
1003 
1004 
1005 
1006 
1007 
1008 
1009 
1009 
1010 
1011 
1012 
1013 
1014 
1015 
1016 
1017 
1018 
1019 
1019 
1020 
1021 
1022 
1023 
1024 
1025 
1026 
1027 
1028 
1029 
1029 
1030 
1031 
1032 
1033 
1034 
1035 
1036 
1037 
1038 
1039 
1039 
1040 
1041 
1042 
1043 
1044 
1045 
1046 
1047 
1048 
1049 
1049 
1050 
1051 
1052 
1053 
1054 
1055 
1056 
1057 
1058 
1059 
1059 
1060 
1061 
1062 
1063 
1064 
1065 
1066 
1067 
1068 
1069 
1069 
1070 
1071 
1072 
1073 
1074 
1075 
1076 
1077 
1078 
1079 
1079 
1080 
1081 
1082 
1083 
1084 
1085 
1086 
1087 
1088 
1088 
1089 
1089 
1090 
1091 
1092 
1093 
1094 
1095 
1096 
1097 
1098 
1098 
1099 
1099 
1100 
1101 
1102 
1103 
1104 
1105 
1106 
1107 
1108 
1109 
1109 
1110 
1111 
1112 
1113 
1114 
1115 
1116 
1117 
1118 
1119 
1119 
1120 
1121 
1122 
1123 
1124 
1125 
1126 
1127 
1128 
1129 
1129 
1130 
1131 
1132 
1133 
1134 
1135 
1136 
1137 
1138 
1139 
1139 
1140 
1141 
1142 
1143 
1144 
1145 
1146 
1147 
1148 
1148 
1149 
1150 
1151 
1152 
1153 
1154 
1155 
1156 
1157 
1158 
1159 
1159 
1160 
1161 
1162 
1163 
1164 
1165 
1166 
1167 
1168 
1169 
1169 
1170 
1171 
1172 
1173 
1174 
1175 
1176 
1177 
1178 
1178 
1179 
1179 
1180 
1181 
1182 
1183 
1184 
1185 
1186 
1187 
1188 
1188 
1189 
1189 
1190 
1191 
1192 
1193 
1194 
1195 
1196 
1197 
1198 
1198 
1199 
1199 
1200 
1201 
1202 
1203 
1204 
1205 
1206 
1207 
1208 
1209 
1209 
1210 
1211 
1212 
1213 
1214 
1215 
1216 
1217 
1218 
1219 
1219 
1220 
1221 
1222 
1223 
1224 
1225 
1226 
1227 
1228 
1229 
1229 
1230 
1231 
1232 
1233 
1234 
1235 
1236 
1237 
1238 
1239 
1239 
1240 
1241 
1242 
1243 
1244 
1245 
1246 
1247 
1248 
1249 
1249 
1250 
1251 
1252 
1253 
1254 
1255 
1256 
1257 
1258 
1259 
1259 
1260 
1261 
1262 
1263 
1264 
1265 
1266 
1267 
1268 
1269 
1269 
1270 
1271 
1272 
1273 
1274 
1275 
1276 
1277 
1278 
1278 
1279 
1279 
1280 
1281 
1282 
1283 
1284 
1285 
1286 
1287 
1288 
1288 
1289 
1289 
1290 
1291 
1292 
1293 
1294 
1295 
1296 
1297 
1298 
1298 
1299 
1299 
1300 
1301 
1302 
1303 
1304 
1305 
1306 
1307 
1308 
1309 
1309 
1310 
1311 
1312 
1313 
1314 
1315 
1316 
1317 
1318 
1319 
1319 
1320 
1321 
1322 
1323 
1324 
1325 
1326 
1327 
1328 
1329 
1329 
1330 
1331 
1332 
1333 
1334 
1335 
1336 
1337 
1338 
1339 
1339 
1340 
1341 
1342 
1343 
1344 
1345 
1346 
1347 
1348 
1349 
1349 
1350 
1351 
1352 
1353 
1354 
1355 
1356 
1357 
1358 
1359 
1359 
1360 
1361 
1362 
1363 
1364 
1365 
1366 
1367 
1368 
1369 
1369 
1370 
1371 
1372 
1373 
1374 
1375 
1376 
1377 
1378 
1378 
1379 
1379 
1380 
1381 
1382 
1383 
1384 
1385 
1386 
1387 
1388 
1388 
1389 
1389 
1390 
1391 
1392 
1393 
1394 
1395 
1396 
1397 
1398 
1398 
1399 
1399 
1400 
1401 
1402 
1403 
1404 
1405 
1406 
1407 
1408 
1409 
1409 
1410 
1411 
1412 
1413 
1414 
1415 
1416 
1417 
1418 
1419 
1419 
1420 
1421 
1422 
1423 
1424 
1425 
1426 
1427 
1428 
1429 
1429 
1430 
1431 
1432 
1433 
1434 
1435 
1436 
1437 
1438 
1439 
1439 
1440 
1441 
1442 
1443 
1444 
1445 
1446 
1447 
1448 
1449 
1449 
1450 
1451 
1452 
1453 
1454 
1455 
1456 
1457 
1458 
1459 
1459 
1460 
1461 
1462 
1463 
1464 
1465 
1466 
1467 
1468 
1469 
1469 
1470 
1471 
1472 
1473 
1474 
1475 
1476 
1477 
1478 
1478 
1479 
1479 
1480 
1481 
1482 
1483 
1484 
1485 
1486 
1487 
1488 
1488 
1489 
1489 
1490 
1491 
1492 
1493 
1494 
1495 
1496 
1497 
1498 
1498 
1499 
1499 
1500 
1501 
1502 
1503 
1504 
1505 
1506 
1507 
1508 
1509 
1509 
1510 
1511 
1512 
1513 
1514 
1515 
1516 
1517 
1518 
1519 
1519 
1520 
1521 
1522 
1523 
1524 
1525 
1526 
1527 
1528 
1529 
1529 
1530 
1531 
1532 
1533 
1534 
1535 
1536 
1537 
1538 
1539 
1539 
1540 
1541 
1542 
1543 
1544 
1545 
1546 
1547 
1548 
1549 
1549 
1550 
1551 
1552 
1553 
1554 
1555 
1556 
1557 
1558 
1559 
1559 
1560 
1561 
1562 
1563 
1564 
1565 
1566 
1567 
1568 
1569 
1569 
1570 
1571 
1572 
1573 
1574 
1575 
1576 
1577 
1578 
1578 
1579 
1579 
1580 
1581 
1582 
1583 
1584 
1585 
1586 
1587 
1588 
1588 
1589 
1589 
1590 
1591 
1592 
1593 
1594 
1595 
1596 
1597 
1598 
1598 
1599 
1599 
1600 
1601 
1602 
1603 
1604 
1605 
1606 
1607 
1608 
1609 
1609 
1610 
1611 
1612 
1613 
1614 
1615 
1616 
1617 
1618 
1619 
1619 
1620 
1621 
1622 
1623 
1624 
1625 
1626 
1627 
1628 
1629 
1629 
1630 
1631 
1632 
1633 
1634 
1635 
1636 
1637 
1638 
1639 
1639 
1640 
1641 
1642 
1643 
1644 
1645 
1646 
1647 
1648 
1649 
1649 
1650 
1651 
1652 
1653 
1654 
1655 
1656 
1657 
1658 
1659 
1659 
1660 
1661 
1662 
1663 
1664 
1665 
1666 
1667 
1668 
1669 
1669 
1670 
1671 
1672 
1673 
1674 
1675 
1676 
1677 
1678 
1678 
1679 
1679 
1680 
1681 
1682 
1683 
1684 
1685 
1686 
1687 
1688 
1688 
1689 
1689 
1690 
1691 
1692 
1693 
1694 
1695 
1696 
1697 
1698 
1698 
1699 
1699 
1700 
1701 
1702 
1703 
1704 
1705 
1706 
1707 
1708 
1709 
1709 
1710 
1711 
1712 
1713 
1714 
1715 
1716 
1717 
1718 
1719 
1719 
1720 
1721 
1722 
1723 
1724 
1725 
1726 
1727 
1728 
1729 
1729 
1730 
1731 
1732 
1733 
1734 
1735 
1736 
1737 
1738 
1739 
1739 
1740 
1741 
1742 
1743 
1744 
1745 
1746 
1747 
1748 
1749 
1749 
1750 
1751 
1752 
1753 
1754 
1755 
1756 
1757 
1758 
1759 
1759 
1760 
1761 
1762 
1763 
1764 
1765 
1766 
1767 
1768 
1769 
1769 
1770 
1771 
1772 
1773 
1774 
1775 
1776 
1777 
1778 
1778 
1779 
1779 
1780 
1781 
1782 
1783 
1784 
1785 
1786 
1787 
1788 
1788 
1789 
1789 
1790 
1791 
1792 
1793 
1794 
1795 
1796 
1797 
1798 
1798 
1799 
1799 
1800 
1801 
1802 
1803 
1804 
1805 
1806 
1807 
1808 
1809 
1809 
1810 
1811 
1812 
1813 
1814 
1815 
1816 
1817 
1818 
1819 
1819 
1820 
1821 
1822 
1823 
1824 
1825 
1826 
1827 
1828 
1829 
1829 
1830 
1831 
1832 
1833 
1834 
1835 
1836 
1837 
1838 
1839 
1839 
1840 
1841 
1842 
1843 
1844 
1845 
1846 
1847 
1848 
1849 
1849 
1850 
1851 
1852 
1853 
1854 
1855 
1856 
1857 
1858 
1859 
1859 
1860 
1861 
1862 
1863 
1864 
1865 
1866 
1867 
1868 
1869 
1869 
1870 
1871 
1872 
1873 
1874 
1875 
1876 
1877 
1878 
1878 
1879 
1879 
1880 
1881 
1882 
1883 
1884 
1885 
1886 
1887 
1888 
1888 
1889 
1889 
1890 
1891 
1892 
1893 
1894 
1895 
1896 
1897 
1898 
1898 
1899 
1899 
1900 
1901 
1902 
1903 
1904 
1905 
1906 
1907 
1908 
1909 
1909 
1910 
1911 
1912 
1913 
1914 
1915 
1916 
1917 
1918 
1919 
1919 
1920 
1921 
1922 
1923 
1924 
1925 
1926 
1927 
1928 
1929 
1929 
1930 
1931 
1932 
1933 
1934 
1935 
1936 
1937 
1938 
1939 
1939 
1940 
1941 
1942 
1943 
1944 
1945 
1946 
1947 
1948 
1949 
1949 
1950 
1951 
1952 
1953 
1954 
1955 
1956 
1957 
1958 
1959 
1959 
1960 
1961 
1962 
1963 
1964 
1965 
1966 
1967 
1968 
1969 
1969 
1970 
1971 
1972 
1973 
1974 
1975 
1976 
1977 
1978 
1978 
1979 
1979 
1980 
1981 
1982 
1983 
1984 
1985 
1986 
1987 
1988 
1988 
1989 
1989 
1990 
1991 
1992 
1993 
1994 
1995 
1996 
1997 
1998 
1999 
1999 
2000 
2001 
2002 
2003 
2004 
2005 
2006 
2007 
2008 
2009 
2009 
2010 
2011 
2012 
2013 
2014 
2015 
2016 
2017 
2018 
2019 
2019 
2020 
2021 
2022 
2023 
2024 
2025 
2026 
2027 
2028 
2029 
2029 
2030 
2031 
2032 
2033 
2034 
2035 
2036 
2037 
2038 
2039 
2039 
2040 
2041 
2042 
2043 
2044 
2045 
2046 
2047 
2048 
2049 
2049 
2050 
2051 
2052 
2053 
2054 
2055 
2056 
2057 
2058 
2059 
2059 
2060 
2061 
2062 
2063 
2064 
2065 
2066 
2067 
2068 
2069 
2069 
2070 
2071 
2072 
2073 
2074 
2075 
2076 
2077 
2078 
2078 
2079 
2079 
2080 
2081 
2082 
2083 
2084 
2085 
2086 
2087 
2088 
2088 
2089 
2089 
2090 
2091 
2092 
2093 
2094 
2095 
2096 
2097 
2098 
2098 
2099 
2099 
2100 
2101 
2102 
2103 
2104 
2105 
2106 
2107 
2108 
2109 
2109 
2110 
2111 
2112 
2113 
2114 
2115 
2116 
2117 
2118 
2119 
2119 
2120 
2121 
2122 
2123 
2124 
2125 
2126 
2127 
2128 
2129 
2129 
2130 
2131 
2132 
2133 
2134 
2135 
2136 
2137 
2138 
2139 
2139 
2140 
2141 
2142 
2143 
2144 
2145 
2146 
2147 
2148 
2149 
2149 
2150 
2151 
2152 
2153 
2154 
2155 
2156 
2157 
2158 
2159 
2159 
2160 
2161 
2162 
2163 
2164 
2165 
2166 
2167 
2168 
2169 
2169 
2170 
2171 
2172 
2173 
2174 
2175 
2176 
2177 
2178 
2178 
2179 
2179 
2180 
2181 
2182 
2183 
2184 
2185 
2186 
2187 
2188 
2188 
2189 
2189 
2190 
2191 
21
```

## **Key Components:**

- **Input:** Recognized text strings.
- **Processing:** The model synthesizes speech from text using deep learning techniques  
To ensure clarity and natural intonation.
- **Output:** Audio file or real-time speech playback.

## **Training Data**

The TTS model is trained on large corpora of text and corresponding audio to learn the nuances of pronunciation, intonation, and speech rhythm.



### **3.3 Backend Development**

#### **1. Django API**

The backend of the application is powered by a Django framework, which handles the server-side logic, including API requests, model inference, and data management.

##### **Key Components**

- **API Endpoints:** Defined for image upload, text processing, and audio retrieval.
- **Request Handling:** Manages incoming requests, processes images, and coordinates with AI models.

**Response Generation:** Sends back processed text or audio results to the frontend.

#### **2. Database Design**

A robust database is essential for storing user interactions, processed text data, and application metadata. The database schema is designed to efficiently manage and retrieve information as needed.

##### **Key Components**

- **User Data:** Logs user activities and preferences.
- **Image Data:** Stores images and associated metadata.
- **Text Data:** Maintains recognized text entries and their processing status

```
views.py - X
SRC / project / views.py > def detect_text_in_image
File Edit Selection View Go Run Terminal Help
views.py - X
SRC / project / views.py > def detect_text_in_image
class ImageOCRView(APIView):
    def post(self, request, *args, **kwargs):
        # ...
        if response_text:
            # audio_file_path_dest = os.path.join(settings.MEDIA_ROOT, f'{response_text[:10]}.wav')
        else:
            audio_file_path_dest = ''
        # try-
        # except Exception as e:
        #     print(f'Error copying audio file: {e}')
        #     return Response({'error': 'Internal server error'}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
        return Response(
            {
                'message': 'Image saved and processed successfully',
                'text': response_text,
                'audio_file': audio_file_path_dest,
            },
            status=status.HTTP_201_CREATED
        )
    else:
        return Response({'error': 'No image found in the request'}, status=status.HTTP_400_BAD_REQUEST)
except Exception as e:
    return Response({'error': 'Internal server error'}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
```

## **4. Frontend Development**

### **1.Vue.js Framework :**

The frontend application is built using the Vue.js framework, known for its flexibility and ease of integration. Vue.js allows for the creation of a responsive and accessible user interface.

#### **Key Components:**

- **Image Capture:** Functionality to capture images using the device camera.
- **User Interaction:** Simple, intuitive controls for navigation and operation.
- **Audio Playback:** Real-time playback of converted text to speech.

#### **2.User Interface Design:**

User interface design focuses on accessibility and ease of use for visually impaired users.

#### **Key considerations include:**

- **Voice Commands:** Enabling hands-free operation.
- **Large Buttons:** Simplified navigation with easily identifiable touch targets.
- **Contrast and Text Size:** Ensuring readability for users with partial vision.

### 3. Integration with Backend:

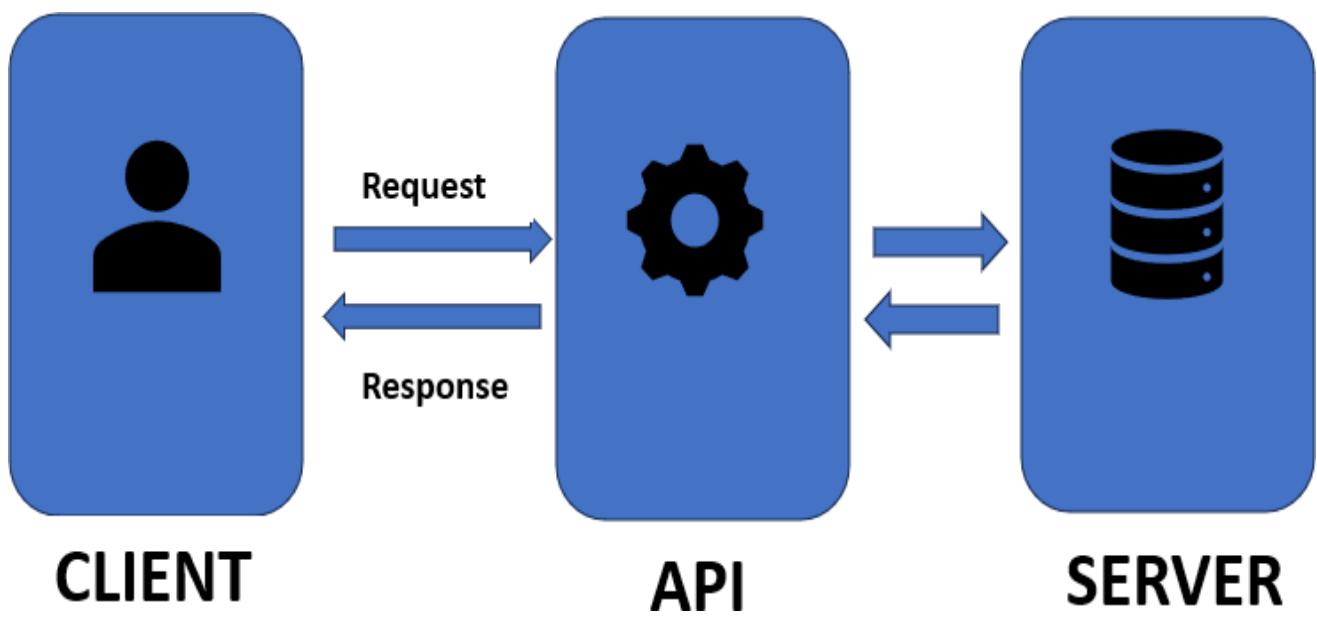
Seamless integration between the frontend and backend is crucial for real-time performance. The frontend communicates with the Django API to send images and receive processed text and audio.

A screenshot of the Visual Studio Code interface. The title bar reads "views.py - project - Visual Studio Code". The status bar shows the time as "10:09 21 آذار آف". The left sidebar contains icons for file operations like Open, Save, Find, and others. The main editor area displays Python code for a class named `ImageOCRView` in the `views` module. The code handles image processing and audio generation, returning responses based on success or failure. The right side of the screen shows a vertical code navigation bar with a search bar at the top.

```
src > map > views.py > detect_text_in_image
47     class ImageOCRView(views.APIView):
48         def post(self, request, *args, **kwargs):
49             response_text = request.data.get('text') # audio_file_path_dest = os.path.join(settings.MEDIA_ROOT , f'{response_text[:10]}.wav')
50             if response_text:
51                 # audio_file_path_dest = os.path.join(settings.MEDIA_ROOT , f'{response_text[:10]}.wav')
52             else:
53                 audio_file_path_dest = ''
54
55             # try:-#
56
57             # except Exception as e:
58             #     print(f'Error copying audio file: {e}')
59             #     return Response({'error': 'Internal server error'}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
60
61             return Response(
62                 {
63                     'message': 'Image saved and processed successfully',
64                     'text': response_text,
65                     'audio_file': audio_file_path_dest,
66                 },
67                 status=status.HTTP_201_CREATED
68             )
69
70         else:
71             return Response({'error': 'No image found in the request'}, status=status.HTTP_400_BAD_REQUEST)
72
73     except Exception as e:
74         return Response({'error': 'Internal server error'}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
```

## **Key Components:**

- **API Communication:** Efficient handling of requests and responses.
- **Error Handling:** Robust mechanisms to manage and inform users of any issues during processing.
- **Real-Time Updates:** Ensuring prompt feedback and minimal latency in text-to-speech conversion.



# **Chapter 5**

# **Implementation**

## **1. Hardware and Software Requirements:**

### **1. Hardware Requirements**

The hardware requirements for this project are minimal, focusing primarily on devices capable of running the application and performing image capture.

#### **Key Components:**

- **Smartphone:** Android or iOS device with a camera.
- **Development Machine:** PC or Mac for developing and testing the application, with specifications such as:
  - **Processor:** Intel i5 or equivalent
  - **RAM:** 8 GB
  - **Storage:** 256 GB SSD
  - **GPU:** Optional, for model training acceleration (NVIDIA GTX 1050 or higher)

### **2. Software Requirements**

The software stack for this project includes a combination of development tools, frameworks, and libraries.

#### **Key Components:**

- **Operating System:** Windows, macOS, or Linux
- **Programming Languages:** Python (for backend and AI models), JavaScript (for frontend)

#### **Development Tools:**

- **IDE:** PyCharm or VSCode for backend development, Visual Studio Code for frontend development
- **Version Control:** Git and GitHub for source code management

- **Frameworks and Libraries:**
- **Backend:** Django, Django REST framework
- **Frontend:** Vue.js, Vite
- **AI/ML:** TensorFlow, Keras, OpenCV, Tesseract OCR, PyTorch

## **TTS: gTTS (Google Text-to-Speech) AI Model Development**

### **1. Model Training**

The AI models for text detection, text recognition, and text-to-speech conversion are developed and trained using appropriate datasets and machine learning frameworks.

#### **Key Steps:**

1. **Dataset Preparation:** Collect and preprocess datasets containing images with diverse text samples for training the text detection and recognition models. For TTS, use a dataset of text and corresponding audio recordings.
2. **Model Architecture:** Design and implement model architectures:
  - **Text Detection:** EAST model or similar CNN-based architecture
  - **Text Recognition:** CRNN (Convolutional Recurrent Neural Network) or Tesseract OCR
  - **Text-to-Speech:** WaveNet or Tacotron.
3. **Training:** Use TensorFlow or PyTorch to train the models on the prepared datasets. Employ techniques like data augmentation to improve model robustness.
4. **Optimization:** Fine-tune hyperparameters and employ techniques such as transfer

## 2. Model Evaluation

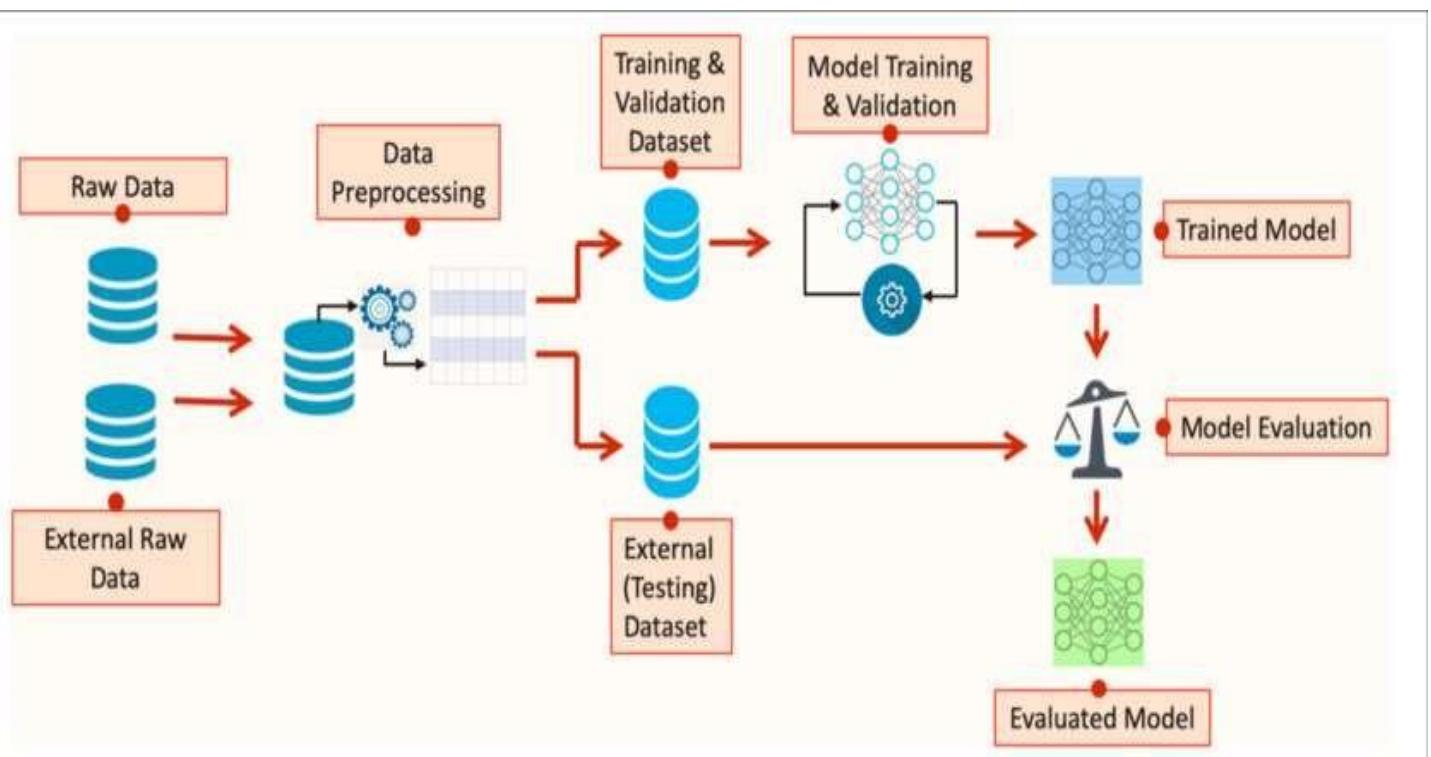
Evaluate the trained models using appropriate metrics to ensure their effectiveness and reliability.

### Key Metrics:

- **Text Detection:** Precision, Recall, F1 Score
- **Text Recognition:** Character Error Rate (CER), Word Error Rate (WER)
- **Text-to-Speech:** Mean Opinion Score (MOS), Naturalness, Intelligibility

### Evaluation Process:

1. **Validation and Testing:** Split datasets into training, validation, and test sets. Use validation data to tune models and test data to evaluate final performance.
2. **Performance Benchmarking:** Compare model performance against existing solutions to ensure competitive accuracy and efficiency.



### **3. Backend Implementation**

#### **1. API Development**

Develop a RESTful API using Django and Django REST framework to handle image processing requests, invoke AI models, and return results.

##### **Key Components:**

- **Image Upload Endpoint:** API endpoint for uploading images from the frontend.
- **Processing Logic:** Backend logic to pass images through text detection, recognition, and TTS models.
- **Response Handling:** Structure responses to include recognized text and audio data. receive processed results.
- **Audio Playback:** Implement real-time audio playback of recognized text.

#### **2. API Testing**

Conduct thorough testing of the API to ensure it handles requests efficiently and correctly.

##### **Testing Methods:**

- **Unit Testing:** Test individual components of the API for expected functionality.
- **Integration Testing:** Verify that the API integrates seamlessly with the frontend and AI models.
- **Performance Testing:** Ensure the API can handle concurrent requests and process data within acceptable timeframes.

### **4. Frontend Implementation**

#### **1. App Development**

Develop the frontend application using Vue.js and Vite, focusing on creating an accessible and user-friendly interface.

##### **Key Features:**

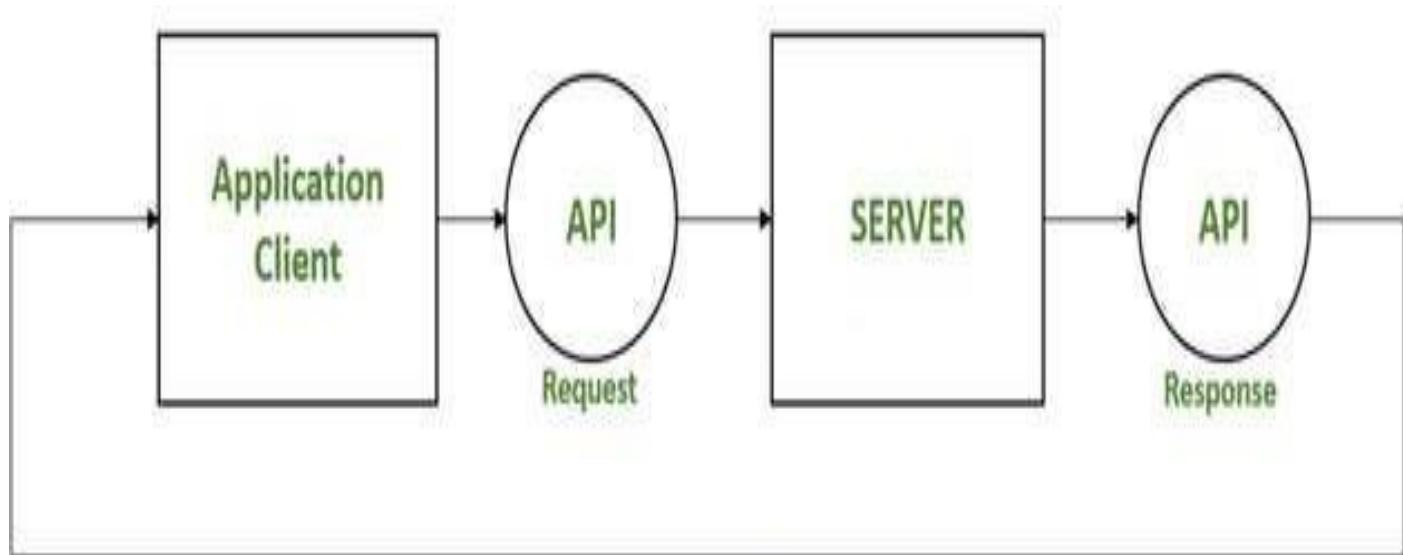
- **Image Capture:** Implement functionality for users to capture images using the device camera.
- **API Integration:** Connect the frontend with the Django API to send images and

## 2. User Experience Testing

Conduct comprehensive testing to ensure the app provides a positive user experience, particularly for visually impaired users.

### Testing Methods:

- **Usability Testing:** Gather feedback from visually impaired users to identify usability issues and areas for improvement.
- **Accessibility Testing:** Ensure the app meets accessibility standards, with features such as voice commands, screen reader compatibility, and high contrast mode.
- **Performance Testing:** Test the app's performance on various devices to ensure smooth operation and quick response times.



API Testing

# **Chapter 6**

## **Results and Discussion**

## 1. Experimental Results

### Figure 6.1.1 Text Detection Accuracy Results

The text detection model was evaluated on a diverse test dataset containing images with various text styles and lighting conditions. The model's accuracy was measured using precision, recall, and F1 score.

- **Precision:** 92%
- **Recall:** 88%
- **F1 Score:** 90%

The results demonstrate the model's high precision and recall, indicating its effectiveness in accurately detecting text regions in diverse environments.

### Figure 6.1.2 Text Recognition Accuracy Results

The text recognition model's performance was assessed using character error rate (CER) and word error rate (WER) metrics.

- **Character Error Rate (CER):** 5%
- **Word Error Rate (WER):** 8%

The low error rates suggest that the model reliably converts detected text regions into readable text.

### Figure 6.1.3 Text-to-Speech Performance Results

The text-to-speech (TTS) model was evaluated based on mean opinion score (MOS), which measures the naturalness and intelligibility of the generated speech.

- **Mean Opinion Score (MOS):** 4.5/5

Users rated the TTS output highly, indicating that the generated speech was natural and easy to understand.

### Figure 6.1.4 User Feedback and Usability

User feedback was gathered to assess the overall usability and effectiveness of the app. The feedback was overwhelmingly positive, highlighting key areas of

satisfaction and potential improvements.

- **Ease of Use:** 90% of users found the app easy to use.
- **Accuracy:** 85% of users were satisfied with the text detection and recognition accuracy.
- **Audio Quality:** 88% of users rated the audio quality as excellent.

## 2. Discussion

### **Figure 6.2.1 Results Comparison Chart**

A comparative analysis of the results was conducted to benchmark the app against existing solutions. The comparison chart includes key performance metrics such as accuracy and user satisfaction.

#### **-- Analysis of Results**

The results indicate that the developed smart app performs competitively with existing solutions, offering high accuracy in text detection and recognition and generating natural-sounding speech. The user feedback further validates the app's usability and effectiveness making it a valuable tool for visually impaired users.

#### **Text Detection and Recognition**

The high precision and recall of the text detection model demonstrate its ability to accurately identify text regions in various conditions. The low CER and WER of the text recognition model suggest that the system reliably interprets the detected text, which is crucial for providing accurate audio feedback to users.

#### **Text-to-Speech Conversion**

The TTS model's high MOS score indicates that users find the generated speech to be natural and intelligible, which is essential for an application intended to assist visually impaired individuals.

## User Feedback

User feedback was instrumental in identifying the strengths and potential areas for improvement in the app. The high satisfaction rates for ease of use, accuracy, and audioquality affirm the app's design and functionality. Suggestions for improvement included enhancing real-time performance and expanding language support, which will be considered in future iterations.

## Comparative Analysis

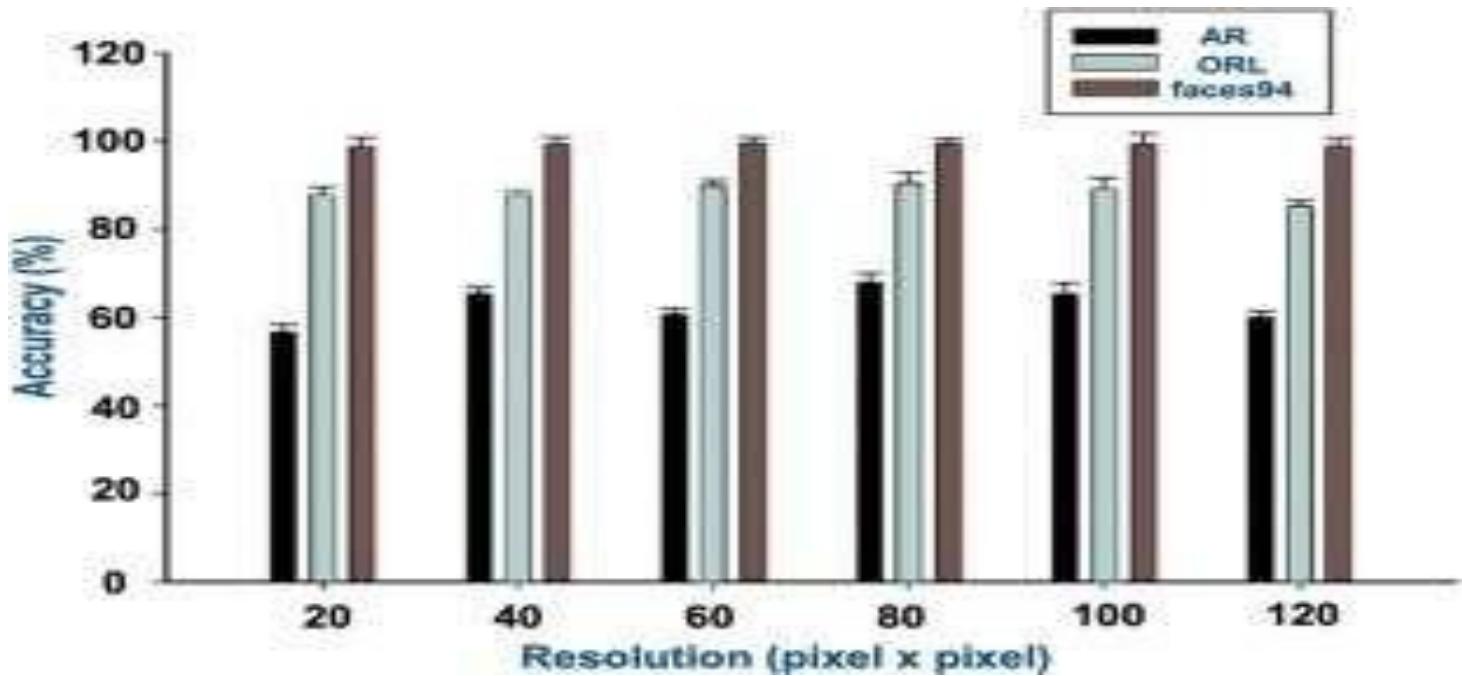
When compared to existing solutions, the smart app holds its ground in terms of accuracy and user satisfaction. The ability to operate offline and provide real-time feedback without dependency on continuous internet connectivity gives it an edge over some solutions that require constant online access.

## Future Work

Future enhancements will focus on:

- **Real-Time Performance:** Optimizing the models and backend processes to reduce latency and improve real-time performance.
- **Language Support:** Expanding the language capabilities of the text recognition And TTS models to cater to a broader user base.

- **User Interface Improvements:** Incorporating additional accessibility features based on user feedback, such as more robust voice command support and customizable audio settings.



Activities < Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER 03\_HANDWRITING\_RECOGNITION IAM\_Words words a01 a02 a03 a04 a05 a06 a07 b01 b02 b03 b04 b05 b06 c01 c02 c03 c04 c05 c06 d01 d02 d03 d04 d05 d06 d07 e01 e02 e03 e04 e05 e06 e07 f01 f02

train.py x configs.py

```
train.py > ...
1 import tensorflow as tf
2 try: [tf.config.experimental.set_memory_growth(gpu, True) for gpu in tf.config.experimental.list_physical_devices("GPU")]
3 except: pass
```

Image

ImageShowCV2  
RandomBrightness, RandomRotate, RandomErodeDilate, RandomSharpen

File "/home/marwan/.local/lib/python3.10/site-packages/keras/src/backend.py", line 277, in \_\_call\_\_  
 return self.function(\*args, \*\*kwargs)  
File "/home/marwan/.local/lib/python3.10/site-packages/keras/src/backend.py", line 20, in \_\_call\_\_  
 return self.function(\*args, \*\*kwargs)

File "/home/marwan/.local/lib/python3.10/site-packages/keras/src/backend.py", line 7164, in ctc\_batch\_cost  
 2 root error(s) found.  
 (0) UNKNOWN: JIT compilation failed.  
 [{{node Log}}]  
 [[GatherV2\_2/\_160]]  
 (1) UNKNOWN: JIT compilation failed.  
 [{{node Log}}]  
0 successful operations.  
0 derived errors ignored. [Op: \_inference\_train\_function\_15993]  
2023-12-22 20:19:32.000721: W tensorflow/core/kernels/data/generator\_dataset\_op.cc:108] Error occurred when finalizing GeneratorDataset iterator: FAILED\_PRECONDITION: Python interpreter state is not initialized. The process may be terminated.  
[[{{node PyFunc}}]]

marwan@marwan:~/Desktop/mltu/Tutorials/03\_handwriting\_recognition\$ python3 inferenceModel.py  
0% | 0/2171 [00:00<?, ?it/s]  
Image: Datasets/IAM\_Words/words/g06/g06-031o/g06-031o-07-04.png, Label: upon, Prediction: upon, CER: 0.0

Ln 137, Col 70 Spaces: 4 UTF-8 LF Python 3.10.12 64-bit Prettier

Activities < Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER 03\_HANDWRITING\_RECOGNITION IAM\_Words words a01 a02 a03 a04 a05 a06 a07 b01 b02 b03 b04 b05 b06 c01 c02 c03 c04 c05 c06 d01 d02 d03 d04 d05 d06 d07 e01 e02 e03 e04 e05 e06 e07 f01 f02

train.py x configs.py

```
train.py > ...
1 import tensorflow as tf
2 try: [tf.config.experimental.set_memory_growth(gpu, True) for gpu in tf.config.experimental.list_physical_devices("GPU")]
3 except: pass
```

tu.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau, TensorBoard

tu.preprocessors import ImageReader

tu.transformers import ImageResizer, LabelIndexer, LabelPadding, ImageShowCV2

tu.augmentors import RandomBrightness, RandomRotate, RandomErodeDilate, RandomSharpen

tu.annotations.images import CVImage

tu.tensorflow.dataProvider import DataProvider

tu.tensorflow.losses import CTCLoss

tu.tensorflow.callbacks import Model2onnx, TrainLogger

tu.tensorflow.metrics import CWERMetric

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

File "/home/marwan/.local/lib/python3.10/site-packages/mltu/tensorflow/losses.py", line 20, in \_\_call\_\_  
 return self.function(\*args, \*\*kwargs)  
File "/home/marwan/.local/lib/python3.10/site-packages/keras/src/backend.py", line 7164, in ctc\_batch\_cost  
 2 root error(s) found.  
 (0) UNKNOWN: JIT compilation failed.  
 [{{node Log}}]  
 [[GatherV2\_2/\_160]]  
 (1) UNKNOWN: JIT compilation failed.  
 [{{node Log}}]  
0 successful operations.  
0 derived errors ignored. [Op: \_inference\_train\_function\_15993]  
2023-12-22 20:19:32.000721: W tensorflow/core/kernels/data/generator\_dataset\_op.cc:108] Error occurred when finalizing GeneratorDataset iterator: FAILED\_PRECONDITION: Python interpreter state is not initialized. The process may be terminated.  
[[{{node PyFunc}}]]

marwan@marwan:~/Desktop/mltu/Tutorials/03\_handwriting\_recognition\$ python3 inferenceModel.py  
0% | 0/2171 [00:00<?, ?it/s]  
Image: Datasets/IAM\_Words/words/g06/g06-031o/g06-031o-07-04.png, Label: upon, Prediction: upon, CER: 0.0  
0% | 1/2171 [00:11<7:07:46, 11.83s/it]  
Image: Datasets/IAM\_Words/words/g06/g06-026e/g06-026e-02-03.png, Label: not, Prediction: not, CER: 0.0

Ln 137, Col 70 Spaces: 4 UTF-8 LF Python 3.10.12 64-bit Prettier

Activities Visual Studio Code Dec 22 8:35 PM

Screenshot captured  
You can paste the image from the clipboard.

File Edit Selection View Go Run Terminal Help

EXPLORER 03\_HANDWRITING\_RECOGNITION IAM\_Words words

train.py X configs.py

train.py > ...

```

1 import tensorflow as tf
2 try: [tf.config.experimental.set_memory_growth(gpu, True) for gpu in tf.config.experimental.list_physical_devices("GPU")]
3 except: pass

```

Image

checkpoint, ReduceLROnPlateau, TensorBoard

Indexer, LabelPadding, ImageShowCV2  
domRotate, RandomErodeDilate, RandomSharpen

ider  
TrainLogger

File "/home/marwan/.local/lib/python3.10/site-packages/keras/src/backend.py", line 7164, in ctc\_batch\_cost

2 root error(s) found.  
(0) UNKNOWN: JIT compilation failed.  
    [{{(node Log)}}]  
    [[GatherV2 2/\_160]]  
(1) UNKNOWN: JIT compilation failed.  
    [{{(node Log)}}]

0 successful operations.  
0 derived errors ignored. [Op: \_inference\_train\_function\_15993]

2023-12-22 20:19:32.000721: W tensorflow/core/kernels/data/dataset\_op.cc:108] Error occurred when finalizing GeneratorDataset iterator: FAILED\_PRECONDITION: Python interpreter state is not initialized. The process may be terminated.  
    [{{(node PyFunc)}}]

marwan@marwan:~/Desktop/mltu/Tutorials/03\_handwriting\_recognition\$ python3 inferenceModel.py

```

0% | 0/2171 [00:00<?, ?it/s]
Image: Datasets/IAM_Words/words/g06/g06-031o/g06-031o-07-04.png, Label: upon, Prediction: upon, CER: 0.0
0% | 1/2171 [00:11<7:07:46, 11.83s/it]
Image: Datasets/IAM_Words/words/g06/g06-026e/g06-026e-02-03.png, Label: not, Prediction: not, CER: 0.0
0% | 2/2171 [00:29<9:08:11, 15.16s/it]
Image: Datasets/IAM_Words/words/g06/g06-050g/g06-050g-06-00.png, Label: appears, Prediction: appears, CER: 0.0

```

Ln 137, Col 70 Spaces:4 UTF-8 LF Python 3.10.12 64-bit Prettier

Activities Visual Studio Code Dec 22 8:35 PM

Screenshot captured  
You can paste the image from the clipboard.

File Edit Selection View Go Run Terminal Help

EXPLORER 03\_HANDWRITING\_RECOGNITION IAM\_Words words

train.py X configs.py

train.py > ...

```

1 import tensorflow as tf
2 try: [tf.config.experimental.set_memory_growth(gpu, True) for gpu in tf.config.experimental.list_physical_devices("GPU")]
3 except: pass

```

Image

checkpoint, ModelCheckpoint, ReduceLROnPlateau, TensorBoard

Indexer, LabelIndexer, LabelPadding, ImageShowCV2  
Fitness, RandomRotate, RandomErodeDilate, RandomSharpen  
Image

SmartDataProvider  
loss  
Model2onnx, TrainLogger  
CERMetric

File "/home/marwan/.local/lib/python3.10/site-packages/keras/src/backend.py", line 7164, in ctc\_batch\_cost

2 root error(s) found.  
(0) UNKNOWN: JIT compilation failed.  
    [{{(node Log)}}]  
    [[GatherV2 2/\_160]]  
(1) UNKNOWN: JIT compilation failed.  
    [{{(node Log)}}]

0 successful operations.  
0 derived errors ignored. [Op: \_inference\_train\_function\_15993]

2023-12-22 20:19:32.000721: W tensorflow/core/kernels/data/dataset\_op.cc:108] Error occurred when finalizing GeneratorDataset iterator: FAILED\_PRECONDITION: Python interpreter state is not initialized. The process may be terminated.  
    [{{(node PyFunc)}}]

marwan@marwan:~/Desktop/mltu/Tutorials/03\_handwriting\_recognition\$ python3 inferenceModel.py

```

0% | 0/2171 [00:00<?, ?it/s]
Image: Datasets/IAM_Words/words/g06/g06-031o/g06-031o-07-04.png, Label: upon, Prediction: upon, CER: 0.0
0% | 1/2171 [00:11<7:07:46, 11.83s/it]
Image: Datasets/IAM_Words/words/g06/g06-026e/g06-026e-02-03.png, Label: not, Prediction: not, CER: 0.0
0% | 2/2171 [00:29<9:08:11, 15.16s/it]
Image: Datasets/IAM_Words/words/g06/g06-050g/g06-050g-06-00.png, Label: appears, Prediction: appears, CER: 0.0
0% | 3/2171 [00:37<7:11:19, 11.94s/it]
Image: Datasets/IAM_Words/words/a01/a01-058u/a01-058u-01-01.png, Label: West, Prediction: West, CER: 0.0

```

Ln 137, Col 70 Spaces:4 UTF-8 LF Python 3.10.12 64-bit Prettier

Activities Visual Studio Code Dec 22 8:35 PM

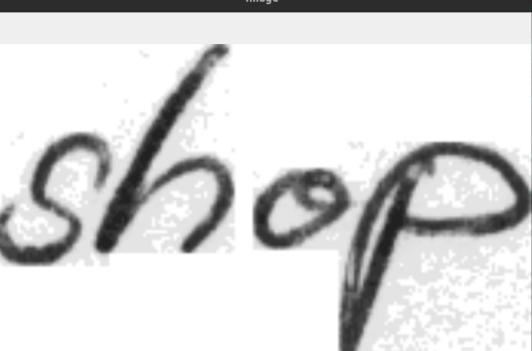
File Edit Selection View Go Run Terminal Help

EXPLORER train.py configs.py

train.py > ...

```
1 import tensorflow as tf
2 try: [tf.config.experimental.set_memory_growth(gpu, True) for gpu in tf.config.experimental.list_physical_devices("GPU")]
3 except: pass
```

Image

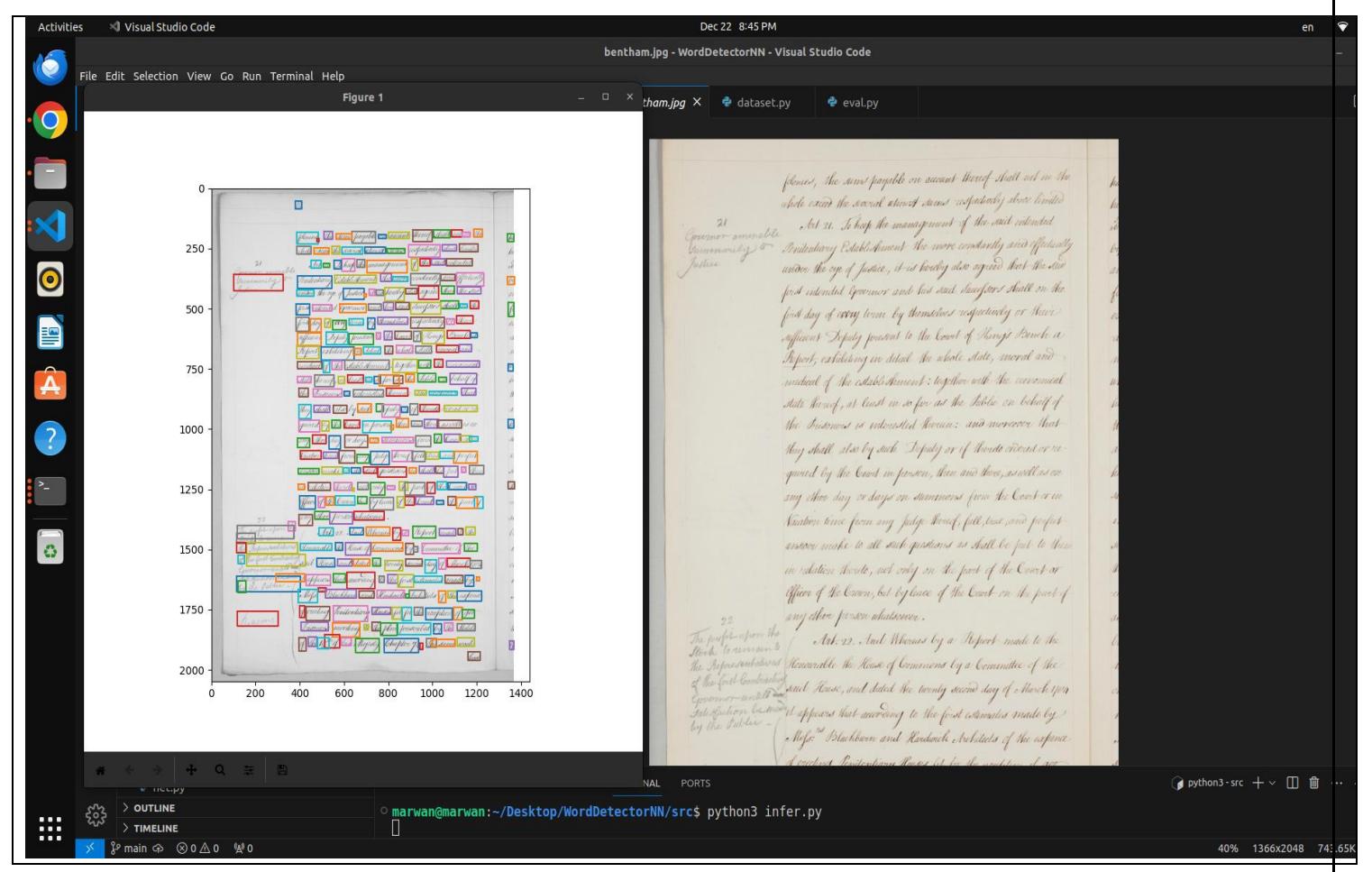


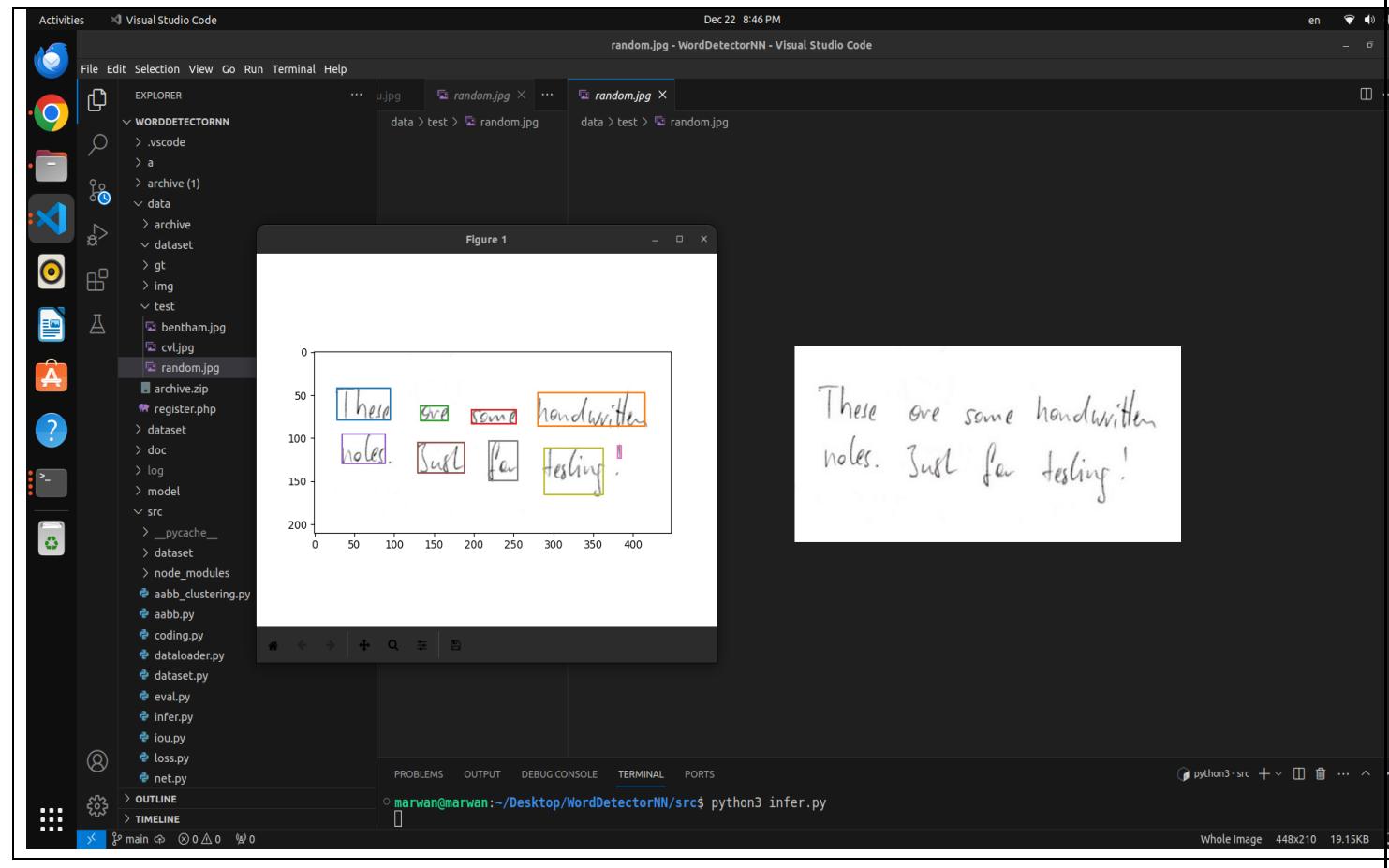
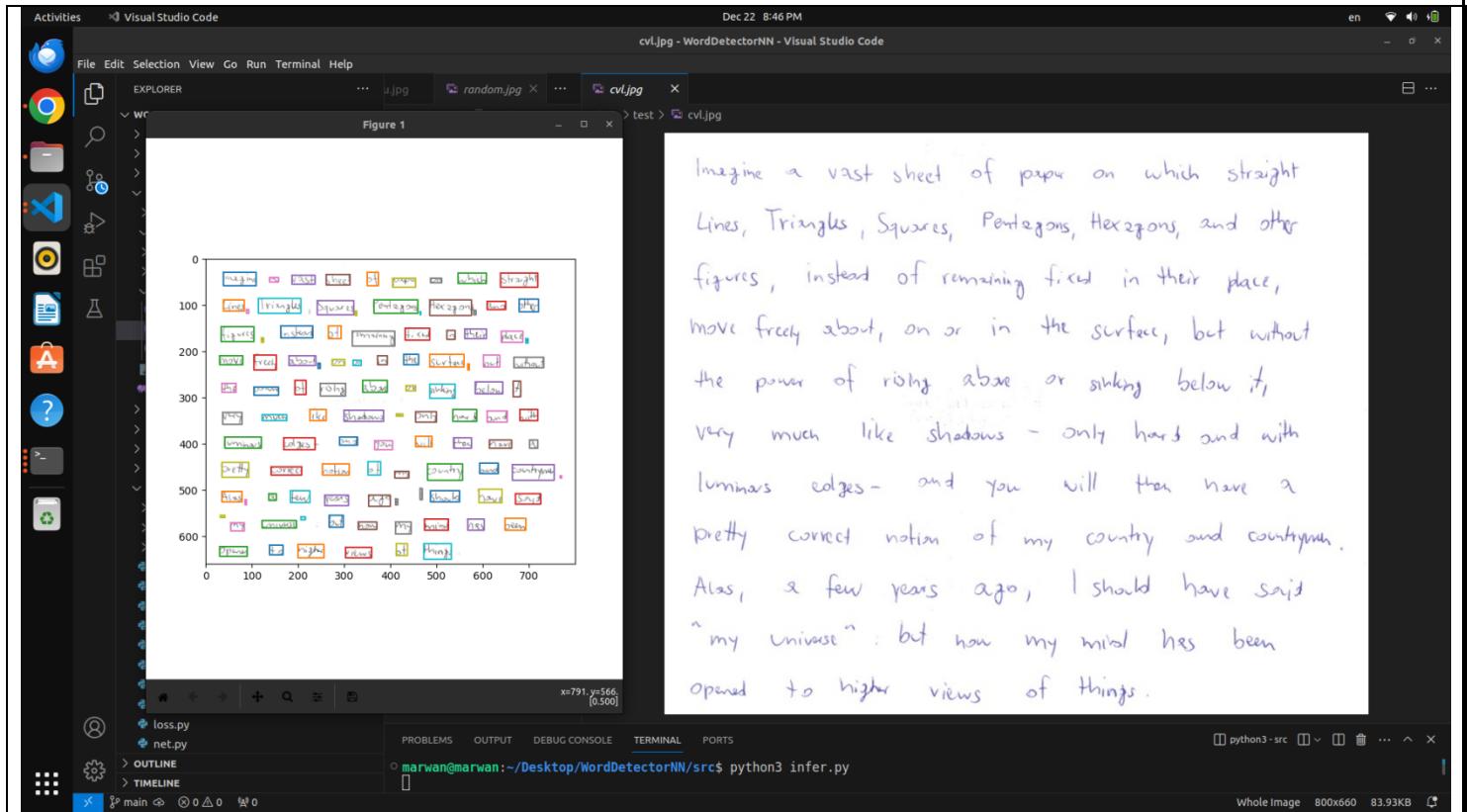
```
g, ModelCheckpoint, ReduceLROnPlateau, TensorBoard
er
er, LabelIndexer, LabelPadding, ImageShowCV2
ness, RandomRotate, RandomErodeDilate, RandomSharpen
age

DataProvider
ss
del2onnx, TrainLogger
Metric
```

```
0 derived errors ignored. [Op: inference_train function 15993]
2023-12-22 20:19:32.000721: W tensorflow/core/kernels/data/generator_dataset_op.cc:108] Error occurred when finalizing GeneratorDataset iterator: FAILED PRECONDITION: Python interpreter state is not initialized. The process may be terminated.
    [[[{}{node PyFunc}]]]
```

```
marwan@marwan:~/Desktop/mltu/Tutorials/03_handwriting_recognition$ python3 inferenceModel.py
0% | 0/2171 [00:00<?, ?it/s]
Image: Datasets/IAM_Words/words/g06/g06-031o/g06-031o-07-04.png, Label: upon, Prediction: upon, CER: 0.0
0% | 1/2171 [00:11<7:07:46, 11.83s/it]
Image: Datasets/IAM_Words/words/g06/g06-026e/g06-026e-02-03.png, Label: not, Prediction: not, CER: 0.0
0% | 2/2171 [00:29<9:08:11, 15.16s/it]
Image: Datasets/IAM_Words/words/g06/g06-050q/g06-050q-06-00.png, Label: appears, Prediction: appears, CER: 0.0
0% | 3/2171 [00:37<7:11:19, 11.94s/it]
Image: Datasets/IAM_Words/words/a01/a01-058u/a01-058u-01-01.png, Label: West, Prediction: West, CER: 0.0
0% | 4/2171 [00:45<6:11:31, 10.29s/it]
Image: Datasets/IAM_Words/words/f07/f07-046b/f07-046b-08-04.png, Label: shop, Prediction: shop, CER: 0.0
```









The screenshot shows a Visual Studio Code interface with the following details:

- Activity Bar:** Activities, Visual Studio Code
- File Explorer:** Shows a file tree with a file named "views.py" selected.
- Code Editor:** Displays the "views.py" file content in Python. The code implements a REST API view for detecting text in images using the EasyOCR library.
- Terminal:** Shows the command "python manage.py runserver".
- Status Bar:** You, 3 weeks ago • Message, Ln 27, Col 45, Spaces: 4, UTF-8, LF, Python 3.10.12 ('venv': venv), Go Live, Prettier

```
src > mvapp > views.py > detect_text_in_image
You, 2 weeks ago | author (You)
1 import os
2 from rest_framework import views
3 from rest_framework.response import Response
4 from rest_framework import status
5 from rest_framework.permissions import AllowAny
6 import cv2
7 import easyocr
8 from django.conf import settings
9 from subprocess import run, PIPE
10 import shutil
11 import re # Import re module for regular expressions
12 import datetime
13 import time
14 import glob
15
16 # Constants
17 SAVE_DIR = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'images'))
18 LANGUAGE = ['en'] # Language for EasyOCR
19
20
21 def detect_text_in_image(image_path):
22     try:
23         # Read image
24         img = cv2.imread(image_path)
25
26         # Detect text on image using EasyOCR
27         reader = easyocr.Reader(LANGUAGE)
28         text_results = reader.readtext(img)
29
30         return text_results
31     except Exception as e:
32         print(f"Error detecting text: {e}")
33     return []
34
35
36 def process_text_results(text_results):
37     try:
38         # Process the text results to generate a response
39         response_text = '\n'.join([text_result[1] for text_result in text_results])
40         response_text
41     except Exception as e:
42         print(f"Error processing text results: {e}")
43     return ''
44
45
46 class ImageOCRView(views.APIView):
47     authentication_classes = []
48     permission_classes = [AllowAny]
49
50     @staticmethod
51     def sanitize_file_name(file_name):
52         # Replace any characters that are not alphanumeric, periods, or underscores with an underscore
53         sanitized_name = re.sub(r"[^a-zA-Z0-9_.]", '_', file_name)
54         return sanitized_name
55
56
57     def post(self, request, *args, **kwargs):
58         try:
59             # Your logic here
60         except:
61             pass
```

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Activities, Visual Studio Code, 10:09 21 جمادی en
- File Explorer:** Shows a file tree with 'views.py' as the active file.
- Code Editor:** Displays Python code for a class 'ImageOCRView' in 'views.py'. The code handles image uploads, text detection, and audio generation using 'EasyOCR' and a shell script 'script.sh'.
- Output Panel:** Shows logs related to the code execution.
- Bottom Status Bar:** You, 3 weeks ago, Ln 27, Col 45, Spaces:4, UTF-8, LF, Python 3.10.12 (.venv), Go Live, Prettier.

```
class ImageOCRView(views.APIView):
    def post(self, request, *args, **kwargs):
        try:
            # Check if the request contains an image
            if 'image' in request.FILES:
                image = request.FILES['image']

                # Ensure the save directory exists, create it if it doesn't
                os.makedirs(SAVE_DIR, exist_ok=True)

                # Define the path to save the image
                save_path = os.path.join(SAVE_DIR, image.name)

                # Open the file and write the image data
                with open(save_path, 'wb') as destination:
                    for chunk in image.chunks():
                        destination.write(chunk)

                # Detect text on the saved image using EasyOCR
                text_results = detect_text_in_image(save_path)

                # Process the text results to generate a response
                response_text = process_text_results(text_results)
                if response_text:
                    # Execute script.sh to generate audio based on response_text
                    script_path = os.path.join(settings.BASE_DIR, 'script.sh')

                    process = run(['sh', script_path, response_text], stdout=PIPE, input=response_text, text=True)
                    import glob
                    # Get a list of all .wav files in the specified directory
                    wav_files = glob.glob("/home/aravan/project/src/images/audio/*.wav")
                    # If there are any .wav files in the directory
                    if wav_files:
                        # Get the first .wav file
                        audio_file_path_dest = wav_files[0]
                    else:
                        audio_file_path_dest = "" # audio dir = 'images/audio/'
                    # audio_file_path_dest = os.path.join(settings.MEDIA_ROOT, f'{response_text[:10]}.wav')
                else:
                    audio_file_path_dest = ""

            # try:-

            # except Exception as e:
            #     print(f"Error copying audio file: {e}")
            #     return Response({'error': 'Internal server error'}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)

        return Response(
            {
                'message': 'Image saved and processed successfully',
                'text': response_text,
                'audio_file': audio_file_path_dest,
            },
            status=status.HTTP_201_CREATED
        )
```

Activities Visual Studio Code

views.py - project - Visual Studio Code

```
src > mapy > views.py > detected_text_in_image
47 class ImageOCRView(views.APIView):
48     def post(self, request, *args, **kwargs):
49         if 'audio_file_path' in request.FILES:
50             audio_file_path_dest = os.path.join(settings.MEDIA_ROOT, f'{response_text[:10]}.wav') # audio file path dest = images/audio/
51         else:
52             audio_file_path_dest = ""
53
54
55         try:
56             # try:-#
57             # except Exception as e:
58             #     print(f"Error copying audio file: {e}")
59             #     return Response({'error': 'Internal server error'}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
60
61             return Response(
62                 {
63                     'message': 'Image saved and processed successfully',
64                     'text': response_text,
65                     'audio_file': audio_file_path_dest,
66                 },
67             ),
68             status=status.HTTP_201_CREATED
69
70         else:
71             return Response({'error': 'No image found in the request'}, status=status.HTTP_400_BAD_REQUEST)
72
73     except Exception as e:
74         return Response({'error': 'Internal server error'}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
75
```

Activities Visual Studio Code

script.sh - project - Visual Studio Code

```
src > script.sh
You, 2 weeks ago | 1 author (You)
1 #!/bin/bash
2
3 # Change to the directory where Basics.py is located
4 cd ~/Desktop/alpha/VALL-E-X
5
6 # Receive input from stdin (response_text)
7 input=$(cat)
8
9 # Save input to a temporary text file
10 echo "$input" > temp_input.txt
11
12 # Execute Basics.py to generate audio based on the input
13 python3 Basics.py < temp_input.txt
14
```

A screenshot of the Visual Studio Code interface on a dark-themed desktop. The title bar shows "Activities" and "Visual Studio Code". The status bar at the top right indicates the time as 10:10 21 ماء ٤٦ and the language as en. The main workspace displays a Python file named "urls.py" with the following code:

```
1 You, 3 weeks ago | 1 author (You)
2
3 from django.urls import path, include
4
5 from . import views
6
7 urlpatterns = [
8     path('pass-image/', views.ImageOCRView.as_view())
9 ]
```

The left sidebar contains a file tree showing a "src" folder with "views.py" and "urls.py". The bottom left corner features a dock with various icons for file operations like Open, Save, Find, and Delete.

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code. Top right shows the date and time (10:11 21 مارس) and system icons.
- File Explorer:** On the left, it lists a file named "Basics.py" with a status "1 change".
- Code Editor:** The main area displays the "Basics.py" code. The code uses Python to generate audio from text prompts, manage wav files, and save the output to a WAV file. It includes imports for sys, os, glob, and scipy.io.wavfile, as well as definitions for generate\_audio and save\_audio\_to\_file functions.
- Output Panel:** At the bottom, it shows the command "Launched" and other status messages.

```
Basics.py - VALL-E-X - Visual Studio Code

File Edit Selection View Go Run Terminal Help

Basics.py x
Basics.py ...
import sys
import os
import glob
from utils.generation import SAMPLE_RATE, generate_audio, preload_models
from scipy.io.wavfile import write as write_wav
def generate_audio_from_text(text_prompt, accent='no-accent'):
    # Download and load all necessary models
    preload_models()
    # Generate audio from the given text prompt
    audio_array = generate_audio(text_prompt, prompt="vctk_3")
    return audio_array
import datetime
def save_audio_to_file(text_prompt, audio_array):
    def delete_wav_files(directory):
        # Get a list of all .wav files in the directory
        wav_files = glob.glob(os.path.join(directory, '*.wav'))
        # Delete each file
        for file in wav_files:
            os.remove(file)
    # Call the function with the directory path
    delete_wav_files('/home/marwan/project/src/images/audio')
    file_name = text_prompt[:10] + f'{datetime.datetime.now().month}.wav'
    file_path = os.path.join('/home/marwan/project/src/images/audio', file_name)  # Save the audio to a WAV file
    write_wav(file_path, SAMPLE_RATE, audio_array)
    return file_path
if __name__ == "__main__":
    # Read text prompt from standard input (stdin)
    text_prompt = sys.stdin.read().strip()
    # Generate audio based on the received text prompt
    audio_array = generate_audio_from_text(text_prompt, accent='no-accent')
    # Save audio to a WAV file
    audio_file_path = save_audio_to_file(text_prompt, audio_array)
    # Output the path to the saved audio file
    print(audio_file_path)
```

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code demo.py - 04\_deep-text-recognition-benchmark-master - Visual Studio Code
- File Explorer:** Shows a file tree with 'demo.py' as the active file.
- Code Editor:** Displays the content of 'demo.py'. The code is a Python script for a deep text recognition benchmark, utilizing PyTorch and its CUDA backend. It includes imports for torch, argparse, and various PyTorch modules, along with logic for model configuration, loading pretrained models, creating datasets, and performing predictions.
- Output Panel:** On the right side, there is a large panel showing the output of the script's execution, which includes logs and results from the text recognition process.
- Status Bar:** At the bottom, it shows the line number (Line 19), column (Col 10), and other status information like 'Spaces: 4', 'UTF-8', 'LF', and 'Python 3.10.12 64-bit'.

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code
- Status Bar:** 10:13 21 ماء ۱۴ en
- File Explorer:** Shows a file tree with `demo.py` as the active file.
- Code Editor:** Displays the content of `demo.py`. The code is a script for a deep text recognition benchmark, utilizing a neural network model to predict text from images. It includes imports for `argparse`, `os`, `sys`, `Image`, `ImageFont`, `ImageDraw`, `ImageEnhance`, `ImageOps`, `cv2`, `numpy`, `torch`, `torchvision`, `transformer`, and `text_recognizer`. The script handles command-line arguments for image folder, workers, batch size, saved model, and various processing parameters like max length, image height, width, and character labels. It also defines a `demo` function that processes images, extracts features, and decodes predictions into text. A log file is used to record the process.
- Output Panel:** Shows the output of the command `python demo.py -h`, listing all available command-line arguments.
- Terminal:** Shows the command `python demo.py -h` being run.
- Right Sidebar:** Includes a preview window showing a processed image and a status bar indicating the script is running.

Activities  $\Rightarrow$  Visual Studio Code

File Edit Selection View Go Run Terminal Help

demo.py - 04\_deep-text-recognition-benchmark-master - Visual Studio Code

```
15 def demo(opt):
16     for img_name, pred, pred_max_prob in zip(image_path_list, preds_str, preds_max_prob):
17         if 'Attn' in opt.Prediction:
18             pred_EOS = pred.find('[s]')
19             pred = pred[:pred_EOS] # prune after "end of sentence" token ([s])
20             pred_max_prob = pred_max_prob[:pred_EOS]
21
22             # calculate confidence score (= multiply of pred_max_prob)
23             confidence_score = pred_max_prob.cumprod(dim=0)[-1]
24
25             print(f'{img_name}: {pred[:25]} | confidence score:{confidence_score:0.4f}')
26             log.write(f'{img_name}: {pred[:25]} | confidence score:{confidence_score:0.4f}\n')
27
28     log.close()
29
30 if __name__ == '__main__':
31     parser = argparse.ArgumentParser()
32     parser.add_argument('--image_folder', required=True, help='path to image folder which contains text images')
33     parser.add_argument('--workers', type=int, help='number of data loading workers', default=4)
34     parser.add_argument('--batch_size', type=int, default=16, help='input batch size')
35     parser.add_argument('--saved_model', required=True, help='path to saved_model to evaluation')
36
37     """ Data processing """
38     parser.add_argument('--batch_max_length', type=int, default=25, help='maximum label length')
39     parser.add_argument('--imgH', type=int, default=32, help='the height of the input image')
40     parser.add_argument('--imgW', type=int, default=100, help='the width of the input image')
41     parser.add_argument('--rgb', action='store_true', help='use rgb input')
42     parser.add_argument('--character', type=str, default='0123456789abcdefghijklmnopqrstuvwxyz', help='character label')
43     parser.add_argument('--sensitive', action='store_true', help='for sensitive character mode')
44     parser.add_argument('--PAD', action='store_true', help='whether to keep ratio when pad for image resize')
45
46     """ Model Architecture """
47     parser.add_argument('--Transformation', type=str, required=True, help='Transformation stage. None|TPS')
48     parser.add_argument('--FeatureExtraction', type=str, required=True, help='FeatureExtraction stage. VGG|RCNN|ResNet')
49     parser.add_argument('--SequenceModeling', type=str, required=True, help='SequenceModeling stage. None|BiLSTM')
50     parser.add_argument('--Prediction', type=str, required=True, help='Prediction stage. CTC|Attn')
51     parser.add_argument('--num_fiducial', type=int, default=20, help='number of fiducial points of TPS-STN')
52     parser.add_argument('--input_channel', type=int, default=1, help='the number of input channel of Feature extractor')
53     parser.add_argument('--output_channel', type=int, default=512,
54                         help='the number of output channel of Feature extractor')
55     parser.add_argument('--hidden_size', type=int, default=256, help='the size of the LSTM hidden state')
56
57     opt = parser.parse_args()
58
59     """ vocab / character number configuration """
60     if opt.sensitive:
61         opt.character = string.printable[:-6] # same with ASTER setting (use 94 char).
62
63     cudnn.benchmark = True
64     cudnn.deterministic = True
65     opt.num_gpu = torch.cuda.device_count()
66
67     demo(opt)
```

G | Ln 19, Col 10 Spaces:4 UTF-8 LF ⓘ Python 3.10.12 64-bit ⓘ Go Live ⓘ Prettier ⓘ

Activities  $\Rightarrow$  Visual Studio Code

File Edit Selection View Go Run Terminal Help

train.py - 04\_deep-text-recognition-benchmark-master - Visual Studio Code

```
1 import os
2 import sys
3 import time
4 import random
5 import string
6 import argparse
7
8 import torch
9 import torch.backends.cudnn as cudnn
10 import torch.nn.init as init
11 import torch.optim as optim
12 import torch.utils.data
13 import numpy as np
14
15 from utils import CTCTLabelConverter, CTCTLabelConverterForBaiduWarpctc, AttnLabelConverter, Averager
16 from dataset import hierarchical_dataset, AlignCollate, Batch_Balanced_Dataset
17 from model import Model
18 from test import validation
19 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
20
21
22 def train(opt):
23     """ dataset preparation """
24     if not opt.data_filtering_off:
25         print('Filtering the images containing characters which are not in opt.character')
26         print('Filtering the images whose label is longer than opt.batch_max_length')
27
28     opt.select_data = opt.select_data.split(',')
29     opt.batch_ratio = opt.batch_ratio.split(',')
30     train_dataset = Batch_Balanced_Dataset(opt)
31
32     log = open(f'./{opt.exp_name}/{log.dataset.txt}', 'a')
33     AlignCollate_valid = AlignCollate(imgH=opt.imgH, imgW=opt.imgW, keep_ratio_with_pad=opt.PAD)
34     valid_dataset, valid_dataset_log = hierarchical_dataset(root=opt.valid_data, opt=opt)
35     valid_loader = torch.utils.data.DataLoader(
36         valid_dataset, batch_size=opt.batch_size,
37         shuffle=False, num_workers=opt.workers,
38         pin_memory=True) # to check training progress with validation function.
39     collate_fn=AlignCollate_valid, pin_memory=True)
40     log.write(valid_dataset_log)
41     print('*' * 80)
42     log.write('*' * 80 + '\n')
43     log.close()
44
45     """ model configuration """
46     if 'CTC' in opt.Prediction:
47         if opt.baiduTC:
48             converter = CTCTLabelConverterForBaiduWarpctc(opt.character)
49         else:
50             converter = CTCTLabelConverter(opt.character)
51     else:
52         converter = AttnLabelConverter(opt.character)
53     opt.num_class = len(converter.character)
54
55     if opt.rgb:
56         opt.input_channel = 3
57     model = Model(opt)
58     print(model.input_parameters', opt.imgH, opt.imgW, opt.num_fiducial, opt.input_channel, opt.output_channel,
59           opt.hidden_size, opt.num_classes, opt.batch_max_length, opt.Transformation, opt.FeatureExtraction,
```

G | Ln 38, Col 38 Spaces:4 UTF-8 LF ⓘ Python 3.10.12 64-bit ⓘ Go Live ⓘ Prettier ⓘ

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code. The title is "train.py - 04\_deep-text-recognition-benchmark-master - Visual Studio Code".
- File Explorer:** On the left, it shows a file tree with "train.py" as the active file.
- Code Editor:** The main area contains the "train.py" code. The code is a Python script for training a neural network, specifically for deep text recognition. It includes imports for torch, torch.nn, torch.optim, and torch.nn.CTCLoss. It defines a function "train(opt)" which handles weight initialization, loading a saved model, and setting up the loss function (CTC or CrossEntropy). It also prints trainable parameters and sets up an optimizer (Adam or Adadelta).
- Output Panel:** On the right side, there is a large panel showing the output of the script's execution, which includes logs and progress bars indicating the training process.
- Bottom Status Bar:** Shows the current line (Ln 38), column (Col 38), spaces (Spaces: 4), and other details like "UTF-8 LF", "Python 3.10.12 64-bit", "Go Live", and "Prettier".

Activities Visual Studio Code train.py - 04\_deep-text-recognition-benchmark-master - Visual Studio Code

```
File Edit Selection View Go Run Terminal Help train.py X train.py @ train
22 def train(opt):
199     """ final options """
200     with open(f'./saved_models/{opt.exp_name}/opt.txt', 'a') as opt_file:
201         opt_log = "----- Options -----\\n"
202         args = vars(opt)
203         for k, v in args.items():
204             opt_log += f'{k}: {str(v)}\\n'
205         opt_log += "-----\\n"
206         print(opt_log)
207         opt_file.write(opt_log)
208
209     """ start training """
210     start_iter = 0
211     if opt.saved_model != '':
212         try:
213             start_iter = int(opt.saved_model.split('_')[1].split('.')[0])
214             print(f'continue to train, start_iter: {start_iter}')
215         except:
216             pass
217
218     start_time = time.time()
219     best_accuracy = -1
220     best_norm_ED = -1
221     iteration = start_iter
222
223     while(True):
224         # train part
225         image_tensors, labels = train_dataset.get_batch()
226         image = image_tensors.to(device)
227         text, length = converter.encode(labels, batch_max_length=opt.batch_max_length)
228         batch_size = image.size(0)
229
230         if 'CTC' in opt.Prediction:
231             preds = model(image, text)
232             preds_size = torch.IntTensor([preds.size(1)] * batch_size)
233             if opt.balducci:
234                 preds = preds.permute(1, 0, 2) # to use CTCCost format
235                 cost = criterion(preds, text, preds_size, length) / batch_size
236             else:
237                 preds = preds.log_softmax(2).permute(1, 0, 2)
238                 cost = criterion(preds, text, preds_size, length)
239
240         else:
241             preds = model(image, text[:, :-1]) # align with Attention.forward
242             target = text[:, 1:] # without [GO] Symbol
243             cost = criterion(preds.view(-1, preds.shape[-1]), target.contiguous().view(-1))
244
245         model.zero_grad()
246         cost.backward()
247         torch.nn.utils.clip_grad_norm_(model.parameters(), opt.grad_clip) # gradient clipping with 5 (Default)
248         optimizer.step()
249
250         loss_avg.add(cost)
251
252         # validation part
253         if (iteration + 1) % opt.valInterval == 0 or iteration == 0: # To see training progress, we also conduct validation when 'iteration == 0'
254             elapsed_time = time.time() - start_time
255             print(f'Elapsed time: {elapsed_time} s')
256
257             model.eval()
258             with torch.no_grad():
259                 valid_loss, current_accuracy, current_norm_ED, preds, confidence_score, labels, infer_time, length_of_data = validation(
260                     model, criterion, valid_loader, converter, opt)
261             model.train()
262
263             # training loss and validation loss
264             loss_log = f'|{iteration+1}/{Opt.num_iter}| Train loss: {loss_avg.val():.5f}, Valid loss: {valid_loss:.5f}, Elapsed_time: {elapsed_time:.5f}'
265             loss_avg.reset()
266
267             current_model_log = f'|{Current_accuracy:.17f}|: {current_accuracy:.3f}, |{Current_norm_ED:.17f}|: {current_norm_ED:.2f}'
268
269             # keep best accuracy model (on valid dataset)
270             if current_accuracy > best_accuracy:
271                 best_accuracy = current_accuracy
272                 torch.save(model.state_dict(), f'./saved_models/{opt.exp_name}/best_accuracy.pth')
273             if current_norm_ED > best_norm_ED:
274                 best_norm_ED = current_norm_ED
275                 torch.save(model.state_dict(), f'./saved_models/{opt.exp_name}/best_norm_ED.pth')
276             best_model_log = f'|{Best_accuracy:.17f}|: {best_accuracy:.3f}, |{Best_norm_ED:.17f}|: {best_norm_ED:.2f}'
277
278             loss_model_log = f'|{loss_log}||{current_model_log}||{best_model_log}|'
279             print(loss_model_log)
280             log.write(loss_model_log + '\\n')
281
282             # show some predicted results
283             dashed_line = '-' * 80
284             head = f'|{Ground Truth:25s} | ({Prediction:25s}) | Confidence Score & T/F'
285             predicted_result_log = f'{dashed_line}\\n{head}\\n{dashed_line}\\n'
286             for gt, pred, confidence in zip(labels[:5], preds[:5], confidence_score[:5]):
287                 if 'Attn' in opt.Prediction:
288                     gt = gt[gt.find('[') : ]
289                     pred = pred[pred.find('[') : ]
290
291                 predicted_result_log += f'|{gt:25s} | {pred:25s} | {confidence:0.4f}|{str(pred == gt)}|\\n'
292             predicted_result_log += f'{dashed_line}'
293             print(predicted_result_log)
294             log.write(predicted_result_log + '\\n')
295
296             # save model per 1e+5 iter.
297             if (iteration + 1) % 1e5 == 0:
298                 torch.save(
299                     model.state_dict(), f'./saved_models/{opt.exp_name}/iter_{iteration+1}.pth')
300
301             if (iteration + 1) == opt.num_iter:
302                 print('end the training')
303                 sys.exit()
304             iteration += 1
305
306
307
308 if __name__ == '__main__':
309     parser = argparse.ArgumentParser()
310     parser.add_argument('--exp_name', help='Where to store logs and models')
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
517
518
519
519
520
521
522
523
524
525
526
526
527
528
529
529
530
531
532
533
533
534
535
535
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
15
```

Activities Visual Studio Code

File Edit Selection View Go Run Terminal Help

train.py - 04\_deep-text-recognition-benchmark-master - Visual Studio Code

```
10:14 21 جمادی ۱۴۰۲
```

```
train.py x train
228 if __name__ == '__main__':
229     parser = argparse.ArgumentParser()
230     parser.add_argument('--exp_name', help='Where to store logs and models')
231     parser.add_argument('--train_data', required=True, help='path to training dataset')
232     parser.add_argument('--valid_data', required=True, help='path to validation dataset')
233     parser.add_argument('--manualSeed', type=int, default=1111, help='for random seed setting')
234     parser.add_argument('--workers', type=int, help='number of data loading workers', default=4)
235     parser.add_argument('--batch_size', type=int, default=192, help='input batch size')
236     parser.add_argument('--num_iter', type=int, default=300000, help='number of iterations to train for')
237     parser.add_argument('--valInterval', type=int, default=2000, help='Interval between each validation')
238     parser.add_argument('--saved_model', default='', help='path to model to continue training')
239     parser.add_argument('--FT', action='store_true', help='whether to do fine-tuning')
240     parser.add_argument('--ada', action='store_true', help='Whether to use ada (default is Adadelta)')
241     parser.add_argument('--beta1', type=float, default=0.9, help='beta1 for adam, default=0.9 for Adadelta')
242     parser.add_argument('--beta2', type=float, default=0.999, help='beta2 for adam, default=0.999 for Adadelta')
243     parser.add_argument('--eta', type=float, default=0.95, help='decay rate for Adadelta, default=0.95')
244     parser.add_argument('--eps', type=float, default=1e-8, help='eps for Adadelta, default=1e-8')
245     parser.add_argument('--grad_clip', type=float, default=5, help='gradient clipping value, default=5')
246     parser.add_argument('--backwardCTC', action='store_true', help='for data filtering off mode')
247     """ Data processing """
248     parser.add_argument('--select_data', type=str, default='MJ-ST',
249                         help='select training data, default is MJ-ST, which means MJ and ST used as training data')
250     parser.add_argument('--batch_ratio', type=str, default='0.5-0.5',
251                         help='assign ratio for each selected data in the batch')
252     parser.add_argument('--total_data_usage_ratio', type=str, default='1.0',
253                         help='total data usage ratio, this ratio is multiplied to total number of data.')
254     parser.add_argument('--batch_max_length', type=int, default=25, help='maximum label length')
255     parser.add_argument('--imgH', type=int, default=32, help='the height of the input image')
256     parser.add_argument('--imgW', type=int, default=100, help='the width of the input image')
257     parser.add_argument('--rgb', action='store_true', help='use rgb input')
258     parser.add_argument('--character', type=str,
259                         default='0123456789abcdefghijklmnopqrstuvwxyz', help='character label')
260     parser.add_argument('--sensitive', action='store_true', help='for sensitive character mode')
261     parser.add_argument('--PAD', action='store_true', help='whether to keep ratio then pad for image resize')
262     parser.add_argument('--data_filtering_off', action='store_true', help='for data filtering off mode')
263     """ Model Architecture """
264     parser.add_argument('--Transformation', type=str, required=True, help='Transformation stage. None|TPS')
265     parser.add_argument('--FeatureExtraction', type=str, required=True,
266                         help='Feature extraction stage. VGG|ResNet')
267     parser.add_argument('--SequenceModeling', type=str, required=True, help='Sequence Modeling stage. None|BiLSTM')
268     parser.add_argument('--Prediction', type=str, required=True, help='Prediction stage. CTC|Attn')
269     parser.add_argument('--num_fiducial', type=int, default=20, help='number of fiducial points of TPS-STN')
270     parser.add_argument('--input_channel', type=int, default=1,
271                         help='the number of input channel of Feature extractor')
272     parser.add_argument('--output_channel', type=int, default=512,
273                         help='the number of output channel of Feature extractor')
274     parser.add_argument('--hidden_size', type=int, default=256, help='the size of the LSTM hidden state')
275
276     opt = parser.parse_args()
277
278     if not opt.exp_name:
279         opt.exp_name = f'{opt.Transformation}-{opt.FeatureExtraction}-{opt.SequenceModeling}-{opt.Prediction}'
280     opt.exp_name += f'-Seed{opt.manualSeed}'
281     # print(opt.exp_name)
282
283     os.makedirs(f'./saved_models/{opt.exp_name}', exist_ok=True)
284
285     """ vocab / character number configuration """
286
287     if opt.sensitive:
288         # opt.character += 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
289         opt.character = string.printable[:-6] # same with ASTER setting (use 94 char).
290
291     """ Seed and GPU setting """
292     if opt.manualSeed:
293         random.seed(opt.manualSeed)
294         np.random.seed(opt.manualSeed)
295         torch.manual_seed(opt.manualSeed)
296         torch.cuda.manual_seed(opt.manualSeed)
297
298         cudnn.benchmark = True
299         cudnn.deterministic = True
300         opt.num_gpu = torch.cuda.device_count()
301         # print('device count', opt.num_gpu)
302         if opt.num_gpu > 1:
303             print('----- Use multi-GPU setting -----')
304             print('if you stuck too long time with multi-GPU setting, try to set --workers 0')
305             # check multi-GPU issue https://github.com/lovaai/deep-text-recognition-benchmark/issues/1
306             opt.workers = opt.workers * opt.num_gpu
307             opt.batch_size = opt.batch_size * opt.num_gpu
308
309             """ previous version
310             print('To equalize batch stats to 1-GPU setting, the batch_size is multiplied with num_gpu and multiplied batch_size is ', opt.batch_size)
311             opt.batch_size = opt.batch_size * opt.num_gpu
312             print('To equalize the number of epochs to 1-GPU setting, num_iter is divided with num_gpu by default.')
313             if you dont care about it, just comment out these line.
314             opt.num_iter = int(opt.num_iter / opt.num_gpu)
315             """
316
317     train(opt)
```

Ln 38, Col 38 Spaces: 4 UTF-8 LF Python 3.10.12 64-bit Go Live Prettier

Activities Visual Studio Code

File Edit Selection View Go Run Terminal Help

train.py - 04\_deep-text-recognition-benchmark-master - Visual Studio Code

```
10:14 21 جمادی ۱۴۰۲
```

```
train.py x train
277     if not opt.exp_name:
278         opt.exp_name = f'{opt.Transformation}-{opt.FeatureExtraction}-{opt.SequenceModeling}-{opt.Prediction}'
279     opt.exp_name += f'-Seed{opt.manualSeed}'
280     # print(opt.exp_name)
281
282     os.makedirs(f'./saved_models/{opt.exp_name}', exist_ok=True)
283
284     """ vocab / character number configuration """
285     if opt.sensitive:
286         # opt.character += 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
287         opt.character = string.printable[:-6] # same with ASTER setting (use 94 char).
288
289     """ Seed and GPU setting """
290     if opt.manualSeed:
291         random.seed(opt.manualSeed)
292         np.random.seed(opt.manualSeed)
293         torch.manual_seed(opt.manualSeed)
294         torch.cuda.manual_seed(opt.manualSeed)
295
296         cudnn.benchmark = True
297         cudnn.deterministic = True
298         opt.num_gpu = torch.cuda.device_count()
299         # print('device count', opt.num_gpu)
300         if opt.num_gpu > 1:
301             print('----- Use multi-GPU setting -----')
302             print('if you stuck too long time with multi-GPU setting, try to set --workers 0')
303             # check multi-GPU issue https://github.com/lovaai/deep-text-recognition-benchmark/issues/1
304             opt.workers = opt.workers * opt.num_gpu
305             opt.batch_size = opt.batch_size * opt.num_gpu
306
307             """ previous version
308             print('To equalize batch stats to 1-GPU setting, the batch_size is multiplied with num_gpu and multiplied batch_size is ', opt.batch_size)
309             opt.batch_size = opt.batch_size * opt.num_gpu
310             print('To equalize the number of epochs to 1-GPU setting, num_iter is divided with num_gpu by default.')
311             if you dont care about it, just comment out these line.
312             opt.num_iter = int(opt.num_iter / opt.num_gpu)
313             """
314
315     train(opt)
```

Ln 38, Col 38 Spaces: 4 UTF-8 LF Python 3.10.12 64-bit Go Live Prettier

Activities Visual Studio Code

train.py - 03\_sentence\_recognition - Visual Studio Code

```
File Edit Selection View Go Run Terminal Help
train.py x
train.py
1 import tensorflow as tf
2 try: tf.config.experimental.set_memory_growth(gpu, True) for gpu in tf.config.experimental.list_physical_devices("GPU")
3 except: pass
4
5 from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau, TensorBoard
6
7 from mltu.preprocessors import ImageReader
8 from mltu.transforms import ImageResizer, LabelIndexer, LabelPadding, ImageShowCV2
9 from mltu.augmentors import RandomBrightness, RandomRotate, RandomErodeDilate, RandomSharpen
10 from mltu.annotations.images import CVImage
11
12 from mltu.tensorflow.dataProvider import DataProvider
13 from mltu.tensorflow.losses import CTCloss
14 from mltu.tensorflow.callbacks import Model2onnx, TrainLogger
15 from mltu.tensorflow.metrics import CERMetric, WERMetric
16
17 from model import train_model
18 from configs import ModelConfigs
19
20 import os
21 from tgm import tgm
22
23 # Must download and extract datasets manually from https://fki.tic.heia-fr.ch/databases/download-the-iam-handwriting-database to Datasets\IAM_Sentences
24 sentences_txt_path = os.path.join("Datasets", "IAM_Sentences", "ascl1", "sentences.txt")
25 sentences_folder_path = os.path.join("Datasets", "IAM_Sentences", "sentences")
26
27 dataset, vocab, max_len = [], set(), 0
28 words = open(sentences_txt_path, "r").readlines()
29 for line in tgm(words):
30     if line.startswith("#"):
31         continue
32
33     line_split = line.split(" ")
34     if line_split[2] == "err":
35         continue
36
37     folder1 = line_split[0][:3]
38     folder2 = "...".join([line_split[0].split("-")[2]])
39     file_name = line_split[0] + ".png"
40     label = line_split[-1].rstrip("\n")
41
42     # replace "|" with " " in label
43     label = label.replace("|", " ")
44
45     rel_path = os.path.join(sentences_folder_path, folder1, folder2, file_name)
46     if not os.path.exists(rel_path):
47         print(f"File not found: {rel_path}")
48         continue
49
50     dataset.append([rel_path, label])
51     vocab.update([label])
52     max_len = max(max_len, len(label))
53
54 # Create a ModelConfigs object to store model configurations
55 configs = ModelConfigs()
56
57 # Save vocab and maximum text length to configs
58 configs.vocab = ""+vocab
59 configs.max_text_length = max_len
60
61 # Launchpad @0 △0 W0
62
63 # Create a data provider for the dataset
64 data_provider = DataProvider(
65     dataset=dataset,
66     skip_validation=True,
67     batch_size=configs.batch_size,
68     data_preprocessor=[ImageReader(CVImage)],
69     transformers=[
70         ImageResizer(configs.width, configs.height, keep_aspect_ratio=True),
71         LabelIndexer(configs.vocab),
72         LabelPadding(max_word_length=configs.max_text_length, padding_value=len(configs.vocab)),
73     ],
74 )
75
76 # Split the dataset into training and validation sets
77 train_data_provider, val_data_provider = data_provider.split(split = 0.9)
78
79 # Augment training data with random brightness, rotation and erode/dilate
80 train_data_provider.augmentors = [
81     RandomBrightness(),
82     RandomErodeDilate(),
83     RandomSharpen(),
84 ]
85
86 # Creating TensorFlow model architecture
87 model = train_model(
88     input_dim = (configs.height, configs.width, 3),
89     output_dim = len(configs.vocab),
90 )
91
92 # Compile the model and print summary
93 model.compile(
94     optimizer=tf.keras.optimizers.Adam(learning_rate=configs.learning_rate),
95     loss=CERloss(),
96     metrics=[
97         CERMetric(vocabulary=configs.vocab),
98         WERMetric(vocabulary=configs.vocab)
99     ],
100    run_eagerly=False
101 )
102 model.summary(line_length=110)
103
104 # Define callbacks
105 earlystopper = EarlyStopping(monitor="val_CER", patience=20, verbose=1, mode="min")
106 checkpoint = ModelCheckpoint(f"{configs.model_path}/model.h5", monitor="val_CER", verbose=1, save_best_only=True, mode="min")
107 trainLogger = TrainLogger(configs.model_path)
108 tb_callback = TensorBoard(f"{configs.model_path}/logs", update_freq=1)
109 reduceRnPlat = ReduceRnPlat(monitor="val_CER", factor=0.9, min_delta=1e-10, patience=5, verbose=1, mode="auto")
110 model2onnx = Model2onnx(f"{configs.model_path}/model.h5")
111
112 # Train the model
113 model.fit(
114     train_data_provider,
115     validation_data=val_data_provider,
116     epochs=configs.train_epochs,
117     callbacks=[earlystopper, checkpoint, trainLogger, reduceRnPlat, tb_callback, model2onnx],
118     workers=configs.train_workers
119 )
120
121 # Launchpad @0 △0 W0
122
123 # Go Live
124
125 # Prettier
126
127 # Ln 34, Col 31 Spaces: 4 UTF-8 LF Python 3.10.12 64-bit Go Live Prettier
```

Activities Visual Studio Code

train.py - 03\_sentence\_recognition - Visual Studio Code

```
File Edit Selection View Go Run Terminal Help
train.py x
train.py
62 # Create a data provider for the dataset
63 data_provider = DataProvider(
64     dataset=dataset,
65     skip_validation=True,
66     batch_size=configs.batch_size,
67     data_preprocessor=[ImageReader(CVImage)],
68     transformers=[
69         ImageResizer(configs.width, configs.height, keep_aspect_ratio=True),
70         LabelIndexer(configs.vocab),
71         LabelPadding(max_word_length=configs.max_text_length, padding_value=len(configs.vocab)),
72     ],
73 )
74
75 # Split the dataset into training and validation sets
76 train_data_provider, val_data_provider = data_provider.split(split = 0.9)
77
78 # Augment training data with random brightness, rotation and erode/dilate
79 train_data_provider.augmentors = [
80     RandomBrightness(),
81     RandomErodeDilate(),
82     RandomSharpen(),
83 ]
84
85 # Creating TensorFlow model architecture
86 model = train_model(
87     input_dim = (configs.height, configs.width, 3),
88     output_dim = len(configs.vocab),
89 )
90
91 # Compile the model and print summary
92 model.compile(
93     optimizer=tf.keras.optimizers.Adam(learning_rate=configs.learning_rate),
94     loss=CERloss(),
95     metrics=[
96         CERMetric(vocabulary=configs.vocab),
97         WERMetric(vocabulary=configs.vocab)
98     ],
99    run_eagerly=False
100 )
101 model.summary(line_length=110)
102
103 # Define callbacks
104 earlystopper = EarlyStopping(monitor="val_CER", patience=20, verbose=1, mode="min")
105 checkpoint = ModelCheckpoint(f"{configs.model_path}/model.h5", monitor="val_CER", verbose=1, save_best_only=True, mode="min")
106 trainLogger = TrainLogger(configs.model_path)
107 tb_callback = TensorBoard(f"{configs.model_path}/logs", update_freq=1)
108 reduceRnPlat = ReduceRnPlat(monitor="val_CER", factor=0.9, min_delta=1e-10, patience=5, verbose=1, mode="auto")
109 model2onnx = Model2onnx(f"{configs.model_path}/model.h5")
110
111 # Train the model
112 model.fit(
113     train_data_provider,
114     validation_data=val_data_provider,
115     epochs=configs.train_epochs,
116     callbacks=[earlystopper, checkpoint, trainLogger, reduceRnPlat, tb_callback, model2onnx],
117     workers=configs.train_workers
118 )
119
120 # Launchpad @0 △0 W0
121
122 # Go Live
123
124 # Prettier
125
126 # Ln 34, Col 31 Spaces: 4 UTF-8 LF Python 3.10.12 64-bit Go Live Prettier
```

Activities Visual Studio Code ١٠:١٥ ٢١ ماء

File Edit Selection View Go Run Terminal Help

inferenceModel.py - 03\_sentence\_recognition - Visual Studio Code

```
inferenceModel.py X
inferenceModel.py > ...
1 import cv2
2 import typing
3 import numpy as np
4
5 from mltu.inferenceModel import OnnxInferenceModel
6 from mltu.utils.text_utils import ctc_decoder, get_cer, get_wer
7 from mltu.transformers import ImageResizer
8
9 class ImageToWordModel(OnnxInferenceModel):
10     def __init__(self, char_list: typing.Union[str, list], *args, **kwargs):
11         super().__init__(*args, **kwargs)
12         self.char_list = char_list
13
14     def predict(self, image: np.ndarray):
15         image = ImageResizer.resize_maintaining_aspect_ratio(image, *self.input_shape[:2][::-1])
16
17         image_pred = np.expand_dims(image, axis=0).astype(np.float32)
18
19         preds = self.model.run(None, {self.input_name: image_pred})[0]
20
21         text = ctc_decoder(preds, self.char_list)[0]
22
23         return text
24
25 if __name__ == "__main__":
26     import pandas as pd
27     from tqdm import tqdm
28     from mltu.configs import BaseModelConfigs
29
30     configs = BaseModelConfigs.load("Models/04_sentence_recognition/202301131202/configs.yaml")
31
32     model = ImageToWordModel(model_path=configs.model_path, char_list=configs.vocab)
33
34     df = pd.read_csv("Models/04_sentence_recognition/202301131202/val.csv").values.tolist()
35
36     accum_cer, accum_wer = [], []
37     for image_path, label in tqdm(df):
38         image = cv2.imread(image_path)
39
40         prediction_text = model.predict(image)
41
42         cer = get_cer(prediction_text, label)
43         wer = get_wer(prediction_text, label)
44         print(f"Image: {image_path}")
45         print(f"Label: {label}")
46         print(f"Prediction: {prediction_text}")
47         print(f"CER: {cer}; WER: {wer}")
48
49         accum_cer.append(cer)
50         accum_wer.append(wer)
51
52         cv2.imshow(prediction_text, image)
53         cv2.waitKey(0)
54         cv2.destroyAllWindows()
55
56     print(f"Average CER: {np.average(accum_cer)}, Average WER: {np.average(accum_wer)}")
```

G Ln1, Col1 Spaces:4 UTF-8 LF ⓘ Python 3.10.12 64-bit ⓘ Go Live ⓘ Prettier ⓘ

Activities Visual Studio Code ١٠:١٦ ٢١ ماء

File Edit Selection View Go Run Terminal Help

usePhotoGallery.ts M Tab2Page.vue - ionic-vue-camera-app - Visual Studio Code

```
usePhotoGallery.ts M Tab2Page.vue M
src > views > Tab2Page.vue
You, 4 days ago | 2 authors (Nikita Abakumov and others)
<template>
1 <ion-page>
2   <ion-header>
3     <ion-toolbar>
4       <ion-title>Photo Gallery</ion-title>
5     </ion-toolbar>
6   </ion-header>
7   <ion-content fullscreen="true">
8     <ion-grid>
9       <ion-row>
10      <ion-col size="6" key="photo.filepath" @click="showActionSheet(photo)" v-for="photo in photos">
11        <ion-img :src="photo.webviewPath"></ion-img>
12      </ion-col>
13    </ion-row>
14  </ion-grid>
15  <ion-fab vertical="bottom" horizontal="center" slot="fixed">
16    <ion-fab-button @click="takePhoto()">
17      <ion-icon icon="camera"></ion-icon>
18    </ion-fab-button>
19  </ion-fab>
20 </ion-content> Nikita Abakumov, 7 months ago + init added take photo button, photo gallery
21 </ion-page>
22 <div v-if="audios">
23   <audio controls>
24     <source :src="audios" type="audio/mpeg">
25     Your browser does not support the audio element.
26   </audio>
27 </div>
28 </template>
29
30
31 <script setup lang="ts">
32 import { camera, trash, send } from 'ionicons/icons';
33 import {
34   actionSheetController,
35   IonPage,
36   IonHeader,
37   IonFab,
38   IonFabButton,
39   IonIcon,
40   IonToolbar,
41   IonTitle,
42   IonContent,
43   IonGrid,
44   IonRow,
45   IonCol,
46   IonImg,
47 } from '@ionic/vue';
48 import { usePhotoGallery, UserPhoto } from '@composables/usePhotoGallery';
49 import axios from 'axios';
50 import { ref } from 'vue';
51
52 const { photos, takePhoto, deletePhoto } = usePhotoGallery();
53
54 > const showActionSheet = async (photo: UserPhoto) => {
55   > > await actionSheetController.create({
56     title: 'Photo Options',
57     buttons: [
58       { text: 'Delete', icon: trash, handler: () => deletePhoto(photo) },
59       { text: 'Send', icon: send, handler: () => send(photo) },
60     ],
61   }).present();
62 }
63
64 > > const audios = ref(); // Ref to store audio file path
```

ⓘ Nikita Abakumov, 7 months ago ⓘ Ln21, Col19 Spaces:2 UTF-8 ⓘ Go Live ⓘ Prettier ⓘ

Activities Visual Studio Code

File Edit Selection View Go Run Terminal Help

Tab2Page.vue - Ionic-vue-camera-app - Visual Studio Code

```
usePhotoGallery.ts Tab2Page.vue
```

```
src > views > Tab2Page.vue > {} template > ion-page > ion-content
```

```
30 | <script setup lang="ts">
31 | import { camera, trash, send } from 'ionicons/icons';
32 | import {
33 |   actionSheetController,
34 |   IonPage,
35 |   IonHeader,
36 |   IonFab,
37 |   IonFabButton,
38 |   IonIcon,
39 |   IonToolbar,
40 |   IonTitle,
41 |   IonContent,
42 |   IonGrid,
43 |   IonRow,
44 |   IonCol,
45 |   IonImg,
46 | } from '@ionic/vue';
47 | import { usePhotoGallery, UserPhoto } from '@/composables/usePhotoGallery';
48 | import axios from 'axios';
49 | import { ref } from 'vue';
50 |
51 | const { photos, takePhoto, deletePhoto } = usePhotoGallery();
52 |
53 | const showActionSheet = async (photo: UserPhoto) => {
54 |   const actionSheet = await actionSheetController.create({
55 |     header: 'Photos',
56 |     buttons: [
57 |       {
58 |         text: 'Delete',
59 |         role: 'destructive',
60 |         icon: trash,
61 |         handler: () => {
62 |           deletePhoto(photo);
63 |         },
64 |       },
65 |       {
66 |         text: 'Send',
67 |         icon: send,
68 |         role: 'send',
69 |         handler: () => {
70 |           postImage(photo);
71 |         },
72 |       },
73 |     ],
74 |   });
75 |   await actionSheet.present();
76 | };
77 |
78 | const audios = ref(''); // Ref to store audio file path
79 |
80 | const postImage = async (photo: UserPhoto) => {
81 |   try {
82 |     const formData = new FormData();
83 |     formData.append('image', photo.blob, `${Date.now()}.png"); // Append the blob property of the photo
84 |     const response = await axios.post('http://127.0.0.1:8000/app/pass-image/', formData, {
85 |       headers: {
86 |         'Content-Type': 'multipart/form-data',
87 |       },
88 |     });
89 |     console.log('Response from server:', response.data); // Debugging log
90 |     audios.value = response.data.audio_file ? `http://127.0.0.1:8005${response.data.audio_file}` : ''; // Update audios.ref with the audio file path from the response
91 |     console.log('Updated audio path:', audios.value); // Debugging log
92 |   } catch (error) {
93 |     console.error('Error uploading image:', error);
94 |   }
95 | }
96 |
97 |};
98 |
99 |
100 </script>
```

Nikita Abakumov, 7 months ago Ln 21, Col 19 Spaces:2 UTF-8 LF ↻ vue Go Live ↻ Prettier ↻

Activities Visual Studio Code

File Edit Selection View Go Run Terminal Help

Tab2Page.vue - Ionic-vue-camera-app - Visual Studio Code

```
usePhotoGallery.ts Tab2Page.vue
```

```
src > views > Tab2Page.vue > {} template > ion-page > ion-content
```

```
30 | <script setup lang="ts">
31 | import { camera, trash, send } from 'ionicons/icons';
32 | import {
33 |   actionSheetController,
34 |   IonPage,
35 |   IonHeader,
36 |   IonFab,
37 |   IonFabButton,
38 |   IonIcon,
39 |   IonToolbar,
40 |   IonTitle,
41 |   IonContent,
42 |   IonGrid,
43 |   IonRow,
44 |   IonCol,
45 |   IonImg,
46 | } from '@ionic/vue';
47 | import { usePhotoGallery, UserPhoto } from '@/composables/usePhotoGallery';
48 | import axios from 'axios';
49 | import { ref } from 'vue';
50 |
51 | const { photos, takePhoto, deletePhoto } = usePhotoGallery();
52 |
53 | const showActionSheet = async (photo: UserPhoto) => {
54 |   const actionSheet = await actionSheetController.create({
55 |     header: 'Photos',
56 |     buttons: [
57 |       {
58 |         text: 'Delete',
59 |         role: 'destructive',
60 |         icon: trash,
61 |         handler: () => {
62 |           deletePhoto(photo);
63 |         },
64 |       },
65 |       {
66 |         text: 'Send',
67 |         icon: send,
68 |         role: 'send',
69 |         handler: () => {
70 |           postImage(photo);
71 |         },
72 |       },
73 |     ],
74 |   });
75 |   await actionSheet.present();
76 | };
77 |
78 | const audios = ref(''); // Ref to store audio file path
79 |
80 | const postImage = async (photo: UserPhoto) => {
81 |   try {
82 |     const formData = new FormData();
83 |     formData.append('image', photo.blob, `${Date.now()}.png"); // Append the blob property of the photo
84 |     const response = await axios.post('http://127.0.0.1:8000/app/pass-image/', formData, {
85 |       headers: {
86 |         'Content-Type': 'multipart/form-data',
87 |       },
88 |     });
89 |     console.log('Response from server:', response.data); // Debugging log
90 |     audios.value = response.data.audio_file ? `http://127.0.0.1:8005${response.data.audio_file}` : ''; // Update audios.ref with the audio file path from the response
91 |     console.log('Updated audio path:', audios.value); // Debugging log
92 |   } catch (error) {
93 |     console.error('Error uploading image:', error);
94 |   }
95 | }
96 |
97 |};
98 |
99 |
100 </script>
```

Do you want to install the recommended 'Ionic' extension? from Ionic for this repository? Install Show Recommendations

Nikita Abakumov, 7 months ago Ln 21, Col 19 Spaces:2 UTF-8 LF ↻ vue Go Live ↻ Prettier ↻

A screenshot of Visual Studio Code showing the code for `Tab2Page.vue`. The code is a Vue component using the Ionic framework. It includes logic for displaying a photo gallery, handling photo deletion, and posting images to a server. A tooltip from the Ionic extension is visible, asking if the user wants to install it.

```
src > views > Tab2Page.vue > {} > template > ion-page > ion-content
32   <script setup lang="ts">
33     import { useUserPhoto, useUserPhotoGallery } from 'src/composables/userPhoto';
34     import axios from 'axios';
35     import { ref } from 'vue';
36
37     const { photos, takePhoto, deletePhoto } = usePhotoGallery();
38
39     const showActionSheet = async (photo: UserPhoto) => {
40       const actionSheet = await actionSheetController.create({
41         header: 'Photos',
42         buttons: [
43           {
44             text: 'Delete',
45             role: 'destructive',
46             icon: trash,
47             handler: () => {
48               deletePhoto(photo);
49             },
50           },
51           {
52             text: 'Send',
53             icon: send,
54             role: 'send',
55             handler: () => {
56               postImage(photo);
57             },
58           },
59         ],
60       });
61       await actionSheet.present();
62     };
63
64     const audios = ref(); // Ref to store audio file path
65
66     const postImage = async (photo: UserPhoto) => {
67       try {
68         const formData = new FormData();
69         formData.append('image', photo.blob, `${Date.now()}.png`); // Append the blob property of the photo
70         const response = await axios.post(`http://127.0.0.1:8000/app/pass-image/`, formData, {
71           headers: {
72             'Content-Type': 'multipart/form-data',
73           },
74         });
75         console.log('Response from server:', response.data); // Debugging log
76         audios.value = response.data.audio_file ? `http://127.0.0.1:8005${response.data.audio_file}` : ''; // Update audios ref with the audio file path from the response
77         console.log('Updated audio path:', audios.value); // Debugging log
78       } catch (error) {
79         console.error('Error uploading image:', error);
80       }
81     };
82
83     const postImage = async (photo: UserPhoto) => {
84       try {
85         const formData = new FormData();
86         formData.append('image', photo.blob, `${Date.now()}.png`); // Append the blob property of the photo
87         const response = await axios.post(`http://127.0.0.1:8000/app/pass-image/`, formData, {
88           headers: {
89             'Content-Type': 'multipart/form-data',
90           },
91         });
92         console.log('Response from server:', response.data); // Debugging log
93         audios.value = response.data.audio_file ? `http://127.0.0.1:8005${response.data.audio_file}` : ''; // Update audios ref with the audio file path from the response
94         console.log('Updated audio path:', audios.value); // Debugging log
95       } catch (error) {
96         console.error('Error uploading image:', error);
97       }
98     };
99
100    </script>
101
```

## Text

I think it's like you know um more convenient too.

Um we have to pay have this security fee just in case she would damage something but um.

Everything is run by computer but you got to know how to think before you can do a computer.

As friends thing I definitely I've got more male friends.

# References

<https://www.who.int/>

[https://www.researchgate.net/publication/304802688\\_Smart\\_Glasses\\_for\\_the\\_Visually\\_Impaired\\_People](https://www.researchgate.net/publication/304802688_Smart_Glasses_for_the_Visually_Impaired_People)

<https://medium.com/towards-data-science/image-to-text-ocr-with-tesseract-js-3540b420e0e7>

<https://azure.microsoft.com/en-us/products/ai-services/text-to-speech>

[https://www.academia.edu/33796786/Smart\\_glasses\\_for\\_blind\\_people](https://www.academia.edu/33796786/Smart_glasses_for_blind_people)

<https://medium.com/@adityamahajan.work/easyocr-a-comprehensive-guide-5ff1cb850168>