Audio2Art: Project Design - Part 2

# 1. File Structure

- app.py: Main application interface using Streamlit

- ImageModel.py: Contains model logic (promptgen, text2image)

* requirements.txt: Lists necessary Python packages
* audio2art/
* |── app.py # Streamlit frontend
* |── ImageModel.py # Model logic
* |── requirements.txt # Required libraries

# 2. promptgen Function

- Uses transformers.Wav2Vec2Tokenizer and Wav2Vec2ForCTC

- Transcribes the uploaded .wav audio file into text prompt

* Uses librosa for audio waveform extraction and PyTorch for inference

Code:

from transformers import Wav2Vec2Tokenizer, Wav2Vec2ForCTC

import librosa as lb

import torch

def promptgen(file):

tokenizer = Wav2Vec2Tokenizer.from\_pretrained('facebook/wav2vec2-base-960h')

model = Wav2Vec2ForCTC.from\_pretrained('facebook/wav2vec2-base-960h')

waveform, rate = lb.load(file, sr=16000)

input\_values = tokenizer(waveform, return\_tensors='pt').input\_values

logits = model(input\_values).logits

predicted\_ids = torch.argmax(logits, dim=-1)

transcription = tokenizer.batch\_decode(predicted\_ids)

return transcription[0]

# 3. text2image Function

- Accepts a string prompt and model identifier

- Uses StableDiffusionPipeline from diffusers

* Generates an image using the specified model and returns it along with generation time

Code:

from diffusers import StableDiffusionPipeline

import torch

import time

def text2image(prompt, repo\_id):

seed = 2024

generator = torch.manual\_seed(seed)

if torch.cuda.is\_available():

pipe = StableDiffusionPipeline.from\_pretrained(repo\_id, torch\_dtype=torch.float16)

pipe = pipe.to("cuda")

else:

pipe = StableDiffusionPipeline.from\_pretrained(repo\_id, torch\_dtype=torch.float32)

start = time.time()

image = pipe(prompt, guidance\_scale=7.5).images[0]

end = time.time()

return image, start, end

# 4. Streamlit Integration

- Handles file upload, model selection, and form submission

- Calls promptgen and text2image

- Displays image and provides download button

* Includes sidebar instructions for usability

Code:

import streamlit as st

from ImageModel import promptgen, text2image

from io import BytesIO

def app():

st.title("🎤 Audio2Art: Turn Audio Prompts Into Images")

upload\_file = st.file\_uploader("🎵 Upload your .wav file", type=["wav"])

model = st.selectbox("Choose a Stable Diffusion model", [

"CompVis/stable-diffusion-v1-4",

"stabilityai/stable-diffusion-2-1"

])

with st.form("my\_form"):

submit = st.form\_submit\_button(label="✨ Generate Image")

if submit and upload\_file:

with st.spinner("Transcribing and generating... Please wait."):

prompt = promptgen(upload\_file)

st.markdown(f"\*\*🎯 Prompt from Audio:\*\* `{prompt}`")

im, start, end = text2image(prompt, model)

buf = BytesIO()

im.save(buf, format="PNG")

byte\_im = buf.getvalue()

st.success(f"✅ Image generated in {round(end - start, 2)} sec")

st.image(im)

st.download\_button("📥 Download Image", data=byte\_im, file\_name="image.png", mime="image/png")

if \_\_name\_\_ == "\_\_main\_\_":

app()

# 5. Deployment Plan

- Hosted and tested on Google Colab using LocalTunnel for public access

- Model inference accelerated using GPU environment

- Suitable for migration to Streamlit Cloud or Heroku in future