Audio to Art code

```
import torch
import torchaudio
from transformers import AutoProcessor, AutoModelForCausalLM, CLIPVisionModel, CLIPImageProcessor
from PIL import Image

# --- Configuration ---
MODEL_NAME = "microsoft/git-large-textcaps"  # Or another suitable text-to-image model
CLIP_MODEL_NAME = "openai/clip-vit-large-patch14"  # For image scoring and fine-tuning
AUDIO_FILE = "audio_prompt.wav"  # Your audio file
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
IMAGE_SIZE = 224  # Standard image size for CLIP


# --- 1. Audio Processing (Speech-to-Text - Dummy implementation) ---
# In a real application, replace this with a proper Speech-to-Text model like Whisper.

def speech_to_text(audio_file):
    """
    Dummy Speech-to-Text function.  Replace with actual STT model.

    Args:
        audio_file (str): Path to the audio file.

    Returns:
        str:  Dummy text prompt generated from the audio.
```

```
    """
    # Load audio information
    waveform, sample_rate = torchaudio.load(audio_file)
    duration = waveform.shape[1] / sample_rate

    if duration < 3:  # very short audio files
        return "A colourful abstract painting."
    elif 3 <= duration < 10:
        return "A vibrant landscape with a mountain."
    else:
        return "A futuristic city skyline at sunset."


# --- 2. Text-to-Image Generation ---

def generate_image(text_prompt):
    """
    Generates an image from a text prompt using a Text-to-Image model.

    Args:
        text_prompt (str): The text prompt to generate the image from.

    Returns:
        PIL.Image.Image: The generated image.
    """
    from diffusers import DiffusionPipeline # Import the necessary library for image generation
    pipe = DiffusionPipeline.from_pretrained("CompVis/stable-diffusion-v1-4")
    pipe = pipe.to(DEVICE)
```

```
    image = pipe(text_prompt).images[0]  # Get the generated image

    return image # Return the generated image


# --- 3. Image Scoring with CLIP (Optional, but highly recommended) ---


def score_image_with_clip(image, text_prompt):
    """
    Scores an image against a text prompt using CLIP to assess semantic similarity.

    Args:
        image (PIL.Image.Image): The image to score.
        text_prompt (str): The text prompt to compare against.

    Returns:
        float: The CLIP score.  Higher scores indicate better alignment with the prompt.
    """
    from transformers import CLIPModel, CLIPTokenizer # Import CLIPTokenizer
    clip_model = CLIPModel.from_pretrained(CLIP_MODEL_NAME).to(DEVICE)
    clip_processor = CLIPImageProcessor.from_pretrained(CLIP_MODEL_NAME)
    clip_tokenizer = CLIPTokenizer.from_pretrained(CLIP_MODEL_NAME) # Initialize
CLIPTokenizer

    image = image.resize((IMAGE_SIZE, IMAGE_SIZE))
    inputs = clip_processor(images=image, return_tensors="pt").to(DEVICE)
```

```
    with torch.no_grad():
        image_features = clip_model.get_image_features(**inputs) # Get image features
        # Use clip_tokenizer instead of clip_processor.tokenizer
        text_features =
clip_model.get_text_features(torch.tensor([clip_tokenizer(text_prompt).input_ids]).to(DEVI
CE))  # Get text features


# --- 4. Main Function ---


def audio_to_art(audio_file):
    """
    Transforms an audio prompt into a visual creation.


    Args:
        audio_file (str): The path to the audio file.


    Returns:
        PIL.Image.Image: The generated image.
    """
    print("Starting Audio-to-Art Transformation...")


    # 1. Speech-to-Text
    print("Transcribing Audio...")
    text_prompt = speech_to_text(audio_file)
    print(f"Text Prompt: {text_prompt}")


    # 2. Text-to-Image Generation
    print("Generating Image...")
    image = generate_image(text_prompt)
```

```
    # 3. Image Scoring (Optional)
    print("Scoring Image with CLIP...")
    clip_score = score_image_with_clip(image, text_prompt)
    print(f"CLIP Score: {clip_score}")

    print("Transformation Complete.")
    return image


# --- Example Usage ---
if _name_ == "_main_":
    # Create a dummy audio file for testing
    # Replace this with loading your actual audio file

    SAMPLE_RATE = 16000
    DURATION = 5  # seconds
    NUM_SAMPLES = SAMPLE_RATE * DURATION
    DUMMY_AUDIO = torch.randn((1, NUM_SAMPLES)) # Create random noise as audio
    torchaudio.save(AUDIO_FILE, DUMMY_AUDIO, SAMPLE_RATE)

    # Run the audio-to-art transformation
    generated_image = audio_to_art(AUDIO_FILE)

    # Save the generated image
    generated_image.save("generated_image.png")
    print("Image saved as generated_image.png")

    # Display the image (optional)
    generated_image.show()
```