Audio Models Documentation

This guide explains how to install and use the audio generation models in the Helix Development Builder system.

Overview

The audio generation system provides three types of AI-powered audio capabilities:

- 🎼 **MusicGen**: Generate music from text descriptions
- 🗣️ **Text-to-Speech (TTS)**: Convert text to natural speech
- 🐕 **Bark**: Advanced speech synthesis with emotions and effects

Unlike text models that run through Ollama, audio models use a specialized installation system with Python-based frameworks.

Installation

Quick Start

```
# Install audio models (auto-detects your GPU and selects appropriate size)
./Scripts/install.sh Generative/Audio
```

What Happens During Installation

1. **System Dependencies**: Automatically installs `python3-venv`, `python3-full`
2. **Virtual Environment**: Creates isolated Python environment in `AudioModels/venv/`
3. **AI Frameworks**: Installs PyTorch, Transformers, and audio processing libraries
4. **Model Selection**: Downloads models based on your GPU VRAM:
   - **< 8GB VRAM**: Small models (7B category)
   - **8-12GB VRAM**: Medium models (13B category)
   - **12-24GB VRAM**: Large models (34B category)
   - **24GB+ VRAM**: Largest models (70B category)
5. **Usage Scripts**: Creates ready-to-use Python scripts

Installation Output

```
🎵 Audio Models Installation System
==================================
🔧 Checking system dependencies...
📦 Installing audio generation dependencies...
✅ Virtual environment created successfully
✅ Audio dependencies installed
🔍 Detecting system capabilities...
GPU VRAM: 6.0 GB
💻 Limited GPU/CPU setup. Using small audio models.
```

```
🎼 Installing MusicGen model: musicgen-small
✅ Success: musicgen-small:musicgen:facebook/musicgen-small
🗣 Installing TTS model: speech-t5
✅ Success: speech-t5:tts:microsoft/speecht5_tts
🐕 Installing Bark model: bark-small
✅ Success: bark-small:bark:suno/bark
📝 Creating audio model usage scripts...
✅ Usage scripts created in AudioModels/scripts/
🎵 Audio models installation completed!
```

Usage Guide

1. Music Generation (MusicGen)

Generate music from text descriptions using the MusicGen models.

Basic Usage

```
# Generate 10 seconds of music
python3 AudioModels/scripts/generate_music.py "upbeat electronic dance
music"

# Longer duration (30 seconds)
python3 AudioModels/scripts/generate_music.py "calm piano melody" --
duration 30

# Use specific model size
python3 AudioModels/scripts/generate_music.py "rock guitar solo" --model
facebook/musicgen-medium
```

Output

```
Loading MusicGen model: facebook/musicgen-small
Generating audio for: 'upbeat electronic dance music'
✅ Audio saved to: generated_music_1234.wav
```

Advanced Examples

```
# Classical music
python3 AudioModels/scripts/generate_music.py "classical orchestra symphony
in D major"

# Ambient soundscape
python3 AudioModels/scripts/generate_music.py "ambient forest sounds with
gentle rain"
```

```
# Jazz
python3 AudioModels/scripts/generate_music.py "smooth jazz saxophone with bass line"

# Electronic
python3 AudioModels/scripts/generate_music.py "synthwave retro 80s electronic music"
```

2. Text-to-Speech (TTS)

Convert text to natural-sounding speech.

Basic Usage

```
# Simple text-to-speech
python3 AudioModels/scripts/text_to_speech.py "Hello, welcome to the audio generation system"

# Longer text
python3 AudioModels/scripts/text_to_speech.py "This is a longer sentence to demonstrate the text-to-speech capabilities of our system"

# Use specific TTS model
python3 AudioModels/scripts/text_to_speech.py "Testing different voices" --model microsoft/speecht5_tts
```

Output

```
Loading TTS model: microsoft/speecht5_tts
Generating speech for: 'Hello, welcome to the audio generation system'
✅ Speech saved to: generated_speech_5678.wav
```

Practical Examples

```
# Generate narration
python3 AudioModels/scripts/text_to_speech.py "Welcome to our application. Please follow these instructions to get started."

# Create notifications
python3 AudioModels/scripts/text_to_speech.py "Your task has been completed successfully."

# Multi-language text (if supported by model)
python3 AudioModels/scripts/text_to_speech.py "Bonjour, comment allez-vous?"
```

3. Advanced Bark Usage

Bark provides more sophisticated speech synthesis with emotions and effects.

Direct Python Usage

```
# Create a Python script for Bark
cat > generate_bark_audio.py << EOF
from bark import SAMPLE_RATE, generate_audio, preload_models
import scipy.io.wavfile as wavfile

# Load Bark models (only needed once)
preload_models()

# Generate audio
text_prompt = "Hello, I'm speaking with Bark! [laughs] This is amazing."
audio_array = generate_audio(text_prompt)

# Save to file
wavfile.write("bark_output.wav", SAMPLE_RATE, audio_array)
print("✅ Bark audio saved to: bark_output.wav")
EOF

# Activate audio environment and run
source AudioModels/venv/bin/activate
python3 generate_bark_audio.py
```

Bark Special Features

```
# Emotional speech
"[sighs] I'm feeling a bit tired today."
"[excitedly] This is fantastic news!"
"[whispers] Can you keep this secret?"

# Sound effects
"The door creaked [creaking sound] as it opened."
"Thunder rumbled [thunder] in the distance."

# Music integration
"♪ Happy birthday to you ♪"
```

File Locations

Generated Files

```
AudioModels/
├── musicgen/              # MusicGen models
│   └── musicgen-small/
├── tts/                   # TTS models
│   └── speech-t5/
├── bark/                  # Bark models
│   └── bark-small/
├── scripts/               # Usage scripts
│   ├── generate_music.py
│   └── text_to_speech.py
└── venv/                  # Python environment
```

Output Files

- **Music files**: `generated_music_*.wav`
- **Speech files**: `generated_speech_*.wav`
- **Custom files**: Whatever filename you specify

Advanced Usage

Direct Model Access

Activate the audio environment and use models directly:

```
# Activate the audio environment
source AudioModels/venv/bin/activate

# Use Python interactively
python3
```

```
# In Python console
from transformers import MusicgenForConditionalGeneration,
MusicgenProcessor
import scipy.io.wavfile

# Load MusicGen model
model =
MusicgenForConditionalGeneration.from_pretrained("facebook/musicgen-small")
processor = MusicgenProcessor.from_pretrained("facebook/musicgen-small")

# Generate music
inputs = processor(text=["jazz piano solo"], padding=True,
return_tensors="pt")
audio_values = model.generate(**inputs, max_new_tokens=256)

# Save output
sampling_rate = model.config.audio_encoder.sampling_rate
```

```
scipy.io.wavfile.write("my_jazz.wav", rate=sampling_rate,
                       data=audio_values[0, 0].cpu().numpy())
```

Batch Processing

Create multiple audio files at once:

```
# Create batch script
cat > batch_generate.py << EOF
import subprocess
import os

prompts = [
    "peaceful meditation music",
    "upbeat workout song",
    "ambient space sounds",
    "classical violin piece"
]

for i, prompt in enumerate(prompts):
    print(f"Generating audio {i+1}/{len(prompts)}: {prompt}")
    cmd = f'python3 AudioModels/scripts/generate_music.py "{prompt}"'
    subprocess.run(cmd, shell=True)
    print(f"✅ Completed {i+1}/{len(prompts)}")

print("🎵 All audio files generated!")
EOF

python3 batch_generate.py
```

Troubleshooting

Common Issues

"ModuleNotFoundError: No module named 'transformers'"

```
# Reactivate the audio environment
source AudioModels/venv/bin/activate

# Or reinstall dependencies
pip install transformers torch
```

"No module named 'bark'"

```
# Install Bark manually
source AudioModels/venv/bin/activate
```

```
pip install git+https://github.com/suno-ai/bark.git
```

"CUDA out of memory"

```
# Use CPU-only models
export CUDA_VISIBLE_DEVICES=""
python3 AudioModels/scripts/generate_music.py "your prompt here"
```

Permission errors

```
# Make scripts executable
chmod +x AudioModels/scripts/*.py

# Fix ownership if needed
sudo chown -R $USER:$USER AudioModels/
```

Model Sizes and Performance

Model Size	VRAM Required	Generation Speed	Audio Quality
Small (7B)	2-4GB	Fast (~10-30 sec)	Good
Medium (13B)	6-8GB	Medium (~30-60 sec)	Better
Large (34B)	12-16GB	Slow (~1-3 min)	Excellent
XLarge (70B)	24GB+	Very Slow (~3-10 min)	Outstanding

Performance Tips

1. **Use smaller models** for faster generation
2. **Keep duration short** (10-30 seconds) for quicker results
3. **Use specific prompts** for better quality
4. **Pre-load models** in scripts to avoid reload time
5. **Use CPU mode** if GPU memory is insufficient

Storage Management

Audio models consume significant disk space:

```
# Check current usage
du -sh AudioModels/

# Remove unused model types
rm -rf AudioModels/bark/     # Remove Bark models (~2-5GB)
rm -rf AudioModels/musicgen/ # Remove MusicGen models (~3-8GB)
```

```
# Reinstall if needed
./Scripts/install.sh Generative/Audio
```

Integration Examples

Web Application Integration

```
# Flask example
from flask import Flask, request, send_file
import subprocess
import os

app = Flask(__name__)

@app.route('/generate-music', methods=['POST'])
def generate_music():
    prompt = request.json['prompt']
    duration = request.json.get('duration', 10)

    # Generate audio
    cmd = f'source AudioModels/venv/bin/activate && python3
AudioModels/scripts/generate_music.py "{prompt}" --duration {duration}'
    result = subprocess.run(cmd, shell=True, capture_output=True,
text=True)

    if result.returncode == 0:
        # Find generated file
        import glob
        files = glob.glob("generated_music_*.wav")
        if files:
            return send_file(files[-1], as_attachment=True)

    return {"error": "Generation failed"}, 500
```

Command Line Automation

```
#!/bin/bash
# Automated content creation script

echo "🎵 Generating background music..."
python3 AudioModels/scripts/generate_music.py "corporate presentation
background music" --duration 60

echo "🗣 Generating narration..."
python3 AudioModels/scripts/text_to_speech.py "Welcome to our product
demonstration. Today we will show you the latest features."

echo "✅ Audio content ready for video production!"
```

Support and Updates

Getting Help

- Check `AudioModels/README.md` for basic information
- Review model info files in each model directory
- Look at generated Python scripts for API examples

Updating Models

```
# Remove old models
rm -rf AudioModels/

# Reinstall with latest versions
./Scripts/install.sh Generative/Audio
```

Adding New Models

To add custom models, modify the audio recipe files:

```
# Edit model recipes
nano Scripts/Recipes/Models/Generative/Audio/7B

# Add line in format: model_name:type:repository_id
# Example: custom-tts:tts:your-username/your-model

# Reinstall
./Scripts/install.sh Generative/Audio
```

🎵 **Happy Audio Generating!**

For additional support, refer to the model-specific documentation in the `AudioModels/` directory after installation.