Kedreamix / **Linly-Talker** Public

main

Kedreamix Merge pull request #135 from SummerRiver3D/main

- ASR
- ChatTTS @ f6b530b
- CosyVoice @ 6be8d0f
- GPT_SoVITS
- LLM
- Musetalk
- NeRF
- TFG
- TTS
- VITS

Digital Human Intelligent Dialogue System - Linly-Talker — 'Interac

2023.12 Update 📆

Users can upload any images for the conversation

2024.01 Update 📆📆

- **Exciting news! I've now incorporated both the powerful GeminiPro and Qwen large models into our conversational scene. Users ca**
- **The deployment invocation method for FastAPI has been updated.**
- **The advanced settings options for Microsoft TTS have been updated, increasing the variety of voice types. Additionally, video subt**
- **Updated the GPT multi-turn conversation system to establish contextual connections in dialogue, enhancing the interactivity and**

2024.02 Update 📆

- **Updated Gradio to the latest version 4.16.0, providing the interface with additional functionalities such as capturing images from**
- **ASR and THG have been updated. FunASR from Alibaba has been integrated into ASR, enhancing its speed significantly. Additionall**
- **I have incorporated the GPT-SoVITS model, which is a voice cloning method. By fine-tuning it with just one minute of a person's sp**
- **I have integrated a web user interface (WebUI) that allows for better execution of Linly-Talker.**

2024.04 Update 📆

- **Updated the offline mode for Paddle TTS, excluding Edge TTS.**
- **Updated ER-NeRF as one of the choices for Avatar generation.**
- **Updated app_talk.py to allow for the free upload of voice and images/videos for generation without being based on a dialogue sce**

2024.05 Update 📅

- **Updated the beginner-friendly AutoDL deployment tutorial, and also updated the codewithgpu image, allowing for one-click expe**
- **Updated WebUI.py: Linly-Talker WebUI now supports multiple modules, multiple models, and multiple options**

2024.06 Update 📅

- **Integrated MuseTalk into Linly-Talker and updated the WebUI, enabling basic real-time conversation capabilities.**
- **The refined WebUI defaults to not loading the LLM model to reduce GPU memory usage. It directly responds with text to complete**

2024.08 Update 📅

- **Updated CosyVoice to offer high-quality text-to-speech (TTS) functionality and voice cloning capabilities; also upgraded to Wav2Lip**

2024.09 Update 📅

- **Added Linly-Talker API documentation, providing detailed interface descriptions to help users access Linly-Talker's features via the**

2024.12 Update 📅

- **Implemented a simple fix for the Edge-TTS bug, resolved several issues with MuseTalk, and plan to integrate fishTTS for more stab**

2025.02 Update 📅

- **Added OmniSenseVoice Model for Faster Speech Recognition**

▶ Content

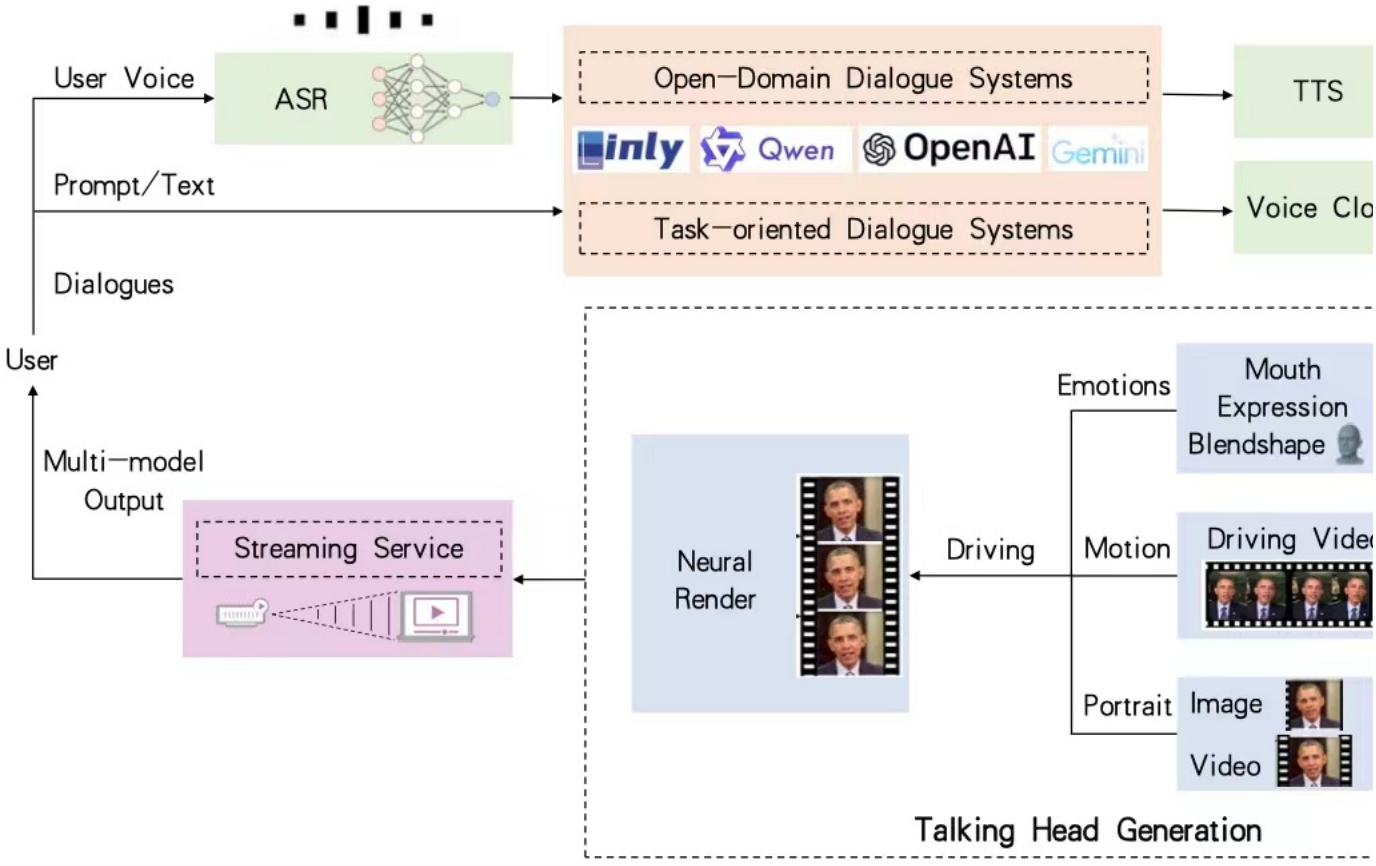
Introduction

Linly-Talker is an innovative digital human conversation system that integrates the latest artificial intelligence technologies, including Large Lar

The core features of the system include:

1. **Multi-Model Integration**: Linly-Talker combines major models such as Linly, GeminiPro, Qwen, as well as visual models like Whisper, SadT
2. **Multi-Turn Conversational Ability**: Through the multi-turn dialogue system powered by GPT models, Linly-Talker can understand and ma
3. **Voice Cloning**: Utilizing technologies like GPT-SoVITS, users can upload a one-minute voice sample for fine-tuning, and the system will clor
4. **Real-Time Interaction**: The system supports real-time speech recognition and video captioning, allowing users to communicate naturally
5. **Visual Enhancement**: With digital human generation technologies, Linly-Talker can create realistic digital human avatars, providing a mor

The design philosophy of Linly-Talker is to create a new form of human-computer interaction that goes beyond simple Q&A. By integrating adv



> Note
>
> You can watch the demo video here.
>
> I have recorded a series of videos on Bilibili, which also represent every step of my updates and methods of use. For detailed information, pl
>
> - 🔥🔥🔥 Digital Human Dialogue System Linly-Talker 🔥🔥🔥
> - 🚀 The Future of Digital Humans: The Empowerment Path of Linly-Talker + GPT-SoVIT Voice Cloning Technology
> - Deploying Linly-Talker on AutoDL Platform (Super Detailed Tutorial for Beginners)
> - Linly-Talker Update: Offline TTS Integration and Customized Digital Human Solutions

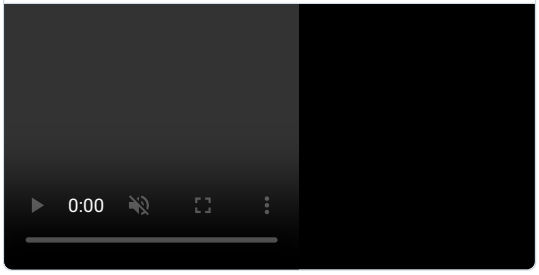
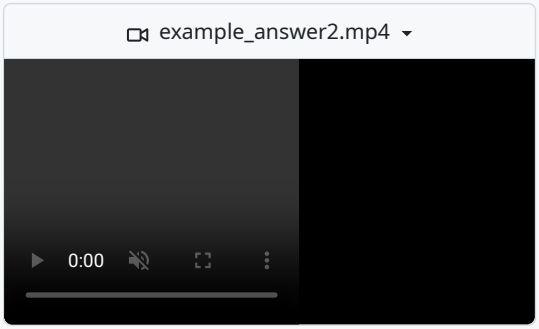
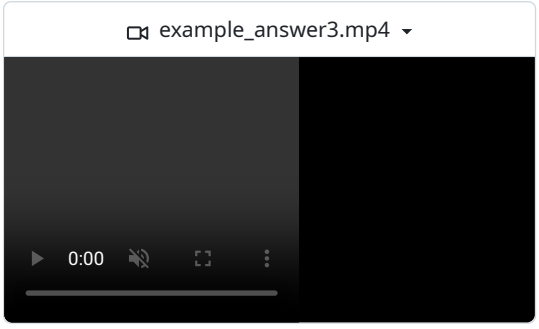
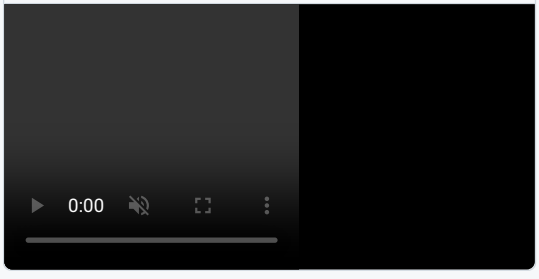
TO DO LIST

- [x] Completed the basic conversation system flow, capable of `voice interactions`.
- [x] Integrated the LLM large model, including the usage of `Linly`, `Qwen`, and `GeminiPro`.
- [x] Enabled the ability to upload `any digital person's photo` for conversation.
- [x] Integrated `FastAPI` invocation for Linly.
- [x] Utilized Microsoft `TTS` with advanced options, allowing customization of voice and tone parameters to enhance audio diversity.
- [x] `Added subtitles` to video generation for improved visualization.
- [x] GPT `Multi-turn Dialogue System` (Enhance the interactivity and realism of digital entities, bolstering their intelligence)
- [x] Optimized the Gradio interface by incorporating additional models such as Wav2Lip, FunASR, and others.
- [x] `Voice Cloning` Technology (Synthesize one's own voice using voice cloning to enhance the realism and interactive experience of digital en
- [x] Integrate offline TTS (Text-to-Speech) along with NeRF-based methods and models.
- [x] Linly-Talker WebUI supports multiple modules, multiple models, and multiple options
- [x] Added MuseTalk functionality to Linly-Talker, achieving near real-time speed with very fast communication.
- [x] Integrated MuseTalk into the Linly-Talker WebUI.
- [x] Added CosyVoice, which provides high-quality text-to-speech (TTS) functionality and voice cloning capabilities. Additionally, updated to Wav
- [x] Added Linly-Talker API documentation with detailed interface descriptions.
- [ ] `Real-time` Speech Recognition (Enable conversation and communication between humans and digital entities using voice)

> Important
>
> The Linly-Talker project is ongoing - pull requests are welcome! If you have any suggestions regarding new model approaches, research,

> Tip
>
> If you encounter any issues during deployment, please consult the Common Issues Summary section, where I have compiled a list of all pot

Example

文字/语音对话	数字人回答
应对压力最有效的方法是什么？	example_answer1.mp4 
如何进行时间管理？	example_answer2.mp4 
撰写一篇交响乐音乐会评论，讨论乐团的表演和观众的整体体验。	example_answer3.mp4 
翻译成中文：Luck is a dividend of sweat. The more you sweat, the luckier you get.	example_answer4.mp4 

Setup Environment

> Note
>
> AutoDL has released an image, which can be used directly at https://www.codewithgpu.com/i/Kedreamix/Linly-Talker/Kedreamix-Linly-Talker

```
docker pull registry.cn-beijing.aliyuncs.com/codewithgpu2/kedreamix-linly-talker:afGA8RPDLf
```

> For Windows, I've included an all-in-one Python package. You can run the steps in sequence to install the necessary dependencies and down
>
> Windows All-in-One Package

Download the code:

```
git clone https://github.com/Kedreamix/Linly-Talker.git --depth 1

cd Linly-Talker
git submodule update --init --recursive
```

If you are using Linly-Talker, you can set up the environment directly with Anaconda, which covers almost all the dependencies required by the

```
conda create -n linly python=3.10
conda activate linly

# PyTorch Installation Option 1: Using conda
# CUDA 11.8
# conda install pytorch==2.4.1 torchvision==0.19.1 torchaudio==2.4.1 pytorch-cuda=11.8 -c pytorch -c nvidia
# CUDA 12.1
# conda install pytorch==2.4.1 torchvision==0.19.1 torchaudio==2.4.1 pytorch-cuda=12.1 -c pytorch -c nvidia
# CUDA 12.4
# conda install pytorch==2.4.1 torchvision==0.19.1 torchaudio==2.4.1 pytorch-cuda=12.4 -c pytorch -c nvidia

# PyTorch Installation Option 2: Using pip
# CUDA 11.8
# pip install torch==2.4.1 torchvision==0.19.1 torchaudio==2.4.1 --index-url https://download.pytorch.org/whl/cu118
# CUDA 12.1
# pip install torch==2.4.1 torchvision==0.19.1 torchaudio==2.4.1 --index-url https://download.pytorch.org/whl/cu121
# CUDA 12.4
# pip install torch==2.4.1 torchvision==0.19.1 torchaudio==2.4.1 --index-url https://download.pytorch.org/whl/cu124
pip install torch==2.0.1 torchvision==0.15.2 torchaudio==2.0.2 --index-url https://download.pytorch.org/whl/cu118

conda install -q ffmpeg==4.2.2 # ffmpeg==4.2.2

# Upgrade pip
python -m pip install --upgrade pip
# Change the PyPI source to speed up the installation of packages
pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple

pip install tb-nightly -i https://mirrors.aliyun.com/pypi/simple
pip install -r requirements_webui.txt

# Install dependencies related to musetalk
pip install --no-cache-dir -U openmim
mim install mmengine
mim install "mmcv==2.1.0"
mim install "mmdet>=3.1.0"
mim install "mmpose>=1.1.0"

# 💡The ttsfrd from CosyVoice can be replaced with WeTextProcessing, so a few steps can be omitted, while ensuring compatibility

# ⚠ Note: You must first download CosyVoice-ttsfrd. Complete the model download before proceeding with these steps.
# mkdir -p CosyVoice/pretrained_models # Create directory CosyVoice/pretrained_models
# mv checkpoints/CosyVoice_ckpt/CosyVoice-ttsfrd CosyVoice/pretrained_models # Move directory
# unzip CosyVoice/pretrained_models/CosyVoice-ttsfrd/resource.zip # Unzip
# This .whl library is only compatible with Python 3.8
# pip install CosyVoice/pretrained_models/CosyVoice-ttsfrd/ttsfrd-0.3.6-cp38-cp38-linux_x86_64.whl

# Install NeRF-based dependencies, which might have several issues and can be skipped initially
pip install "git+https://github.com/facebookresearch/pytorch3d.git"
# If you encounter problems installing PyTorch3D, you can use the following command to install it:
# python scripts/install_pytorch3d.py
pip install -r TFG/requirements_nerf.txt

# If you encouter issues with pyaudio
sudo apt-get update
sudo apt-get install libasound-dev portaudio19-dev libportaudio2 libportaudiocpp0

# Note the following modules. If installation fails, you can enter the directory and use pip install . or python setup.py install
# NeRF/freqencoder
# NeRF/gridencoder
```

```
# NeRF/raymarching
# NeRF/shencoder

# If you encounter sox compatibility issues
# ubuntu
sudo apt-get install sox libsox-dev
# centos
sudo yum install sox sox-devel
```

> Note
>
> The installation process is very slow.

Below are some older installation methods, which might cause dependency conflicts, but they generally don't produce many bugs. For an easie

> To install the environment using Anaconda and PyTorch, follow the steps below:
>
> ```
> conda create -n linly python=3.10
> conda activate linly
>
> # PyTorch Installation Method 1: Conda Installation (Recommended)
> conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 cudatoolkit=11.3 -c pytorch
>
> # PyTorch Installation Method 2: Pip Installation
> pip install torch==1.12.1+cu113 torchvision==0.13.1+cu113 torchaudio==0.12.1 --extra-index-url https://download.pytorch.org/whl
>
> conda install -q ffmpeg # ffmpeg==4.2.2
>
> pip install -r requirements_app.txt
> ```
>
> If you want to use models like voice cloning, you may need a higher version of PyTorch. However, the functionality will be more diverse. You
>
> ```
> conda create -n linly python=3.10
> conda activate linly
>
> pip install torch==2.0.1 torchvision==0.15.2 torchaudio==2.0.2 --index-url https://download.pytorch.org/whl/cu118
>
> conda install -q ffmpeg # ffmpeg==4.2.2
>
> pip install -r requirements_app.txt
>
> # Install dependencies for voice cloning
> pip install -r VITS/requirements_gptsovits.txt
> ```
>
> If you wish to use NeRF-based models, you may need to set up the corresponding environment:
>
> ```
> # Install dependencies for NeRF
> pip install "git+https://github.com/facebookresearch/pytorch3d.git"
> pip install -r TFG/requirements_nerf.txt
>
> # If there are issues with PyAudio, you can install the corresponding dependencies
> # sudo apt-get install libasound-dev portaudio19-dev libportaudio2 libportaudiocpp0
>
> # Note the following modules. If installation is unsuccessful, you can navigate to the path and use pip install . or python set
> # NeRF/freqencoder
> # NeRF/gridencoder
> # NeRF/raymarching
> # NeRF/shencoder
> ```
>
> If you are using PaddleTTS, you can set up the corresponding environment with:
>
> ```
> pip install -r TTS/requirements_paddle.txt
> ```
>
> If you are using the FunASR speech recognition model, you can install the environment with:
>
> ```
> pip install -r ASR/requirements_funasr.txt
> ```
>
> If using the MuesTalk model, you can set up the environment with the following commands:

```
pip install --no-cache-dir -U openmim
mim install mmengine
mim install "mmcv>=2.0.1"
mim install "mmdet>=3.1.0"
mim install "mmpose>=1.1.0"
pip install -r TFG/requirements_musetalk.txt
```

> Note
>
> Next, you need to install the corresponding models. You can download them using the following methods. Once downloaded, place the files
>
> We recommend downloading from modelscope for the latest updates.
>
> - Baidu (百度云盘) (Password: `linl`)
> - huggingface
> - modelscope
> - Quark(夸克网盘)

I made a script that can download all the models mentioned below without requiring much input from the user. This method is suitable for stal

1. **Choose Download Method**: Users can choose to download models from three different sources: ModelScope, Huggingface, or Huggingfa
2. **Download Models**: Based on the user's selection, the script executes the corresponding download command.
3. **Move Model Files**: After downloading, the script moves the model files to the specified directory.
4. **Error Handling**: Error checks are included in each step of the operation. If any step fails, the script will output an error message and stop e

```
sh scripts/download_models.sh
```

HuggingFace Download

If the download speed is too slow, consider using a mirror site. For more information, refer to Efficiently Obtain Hugging Face Models Using Mi

```
# Download pre-trained models from HuggingFace
git lfs install
git clone https://huggingface.co/Kedreamix/Linly-Talker --depth 1
# git lfs clone https://huggingface.co/Kedreamix/Linly-Talker --depth 1

# pip install -U huggingface_hub
# export HF_ENDPOINT=https://hf-mirror.com # Use a mirror site
huggingface-cli download --resume-download --local-dir-use-symlinks False Kedreamix/Linly-Talker --local-dir Linly-Talker
```

ModelScope Download

```
# Download pre-trained models from Modelscope
# 1. Using git
git lfs install
git clone https://www.modelscope.cn/Kedreamix/Linly-Talker.git --depth 1
# git lfs clone https://www.modelscope.cn/Kedreamix/Linly-Talker.git

# 2. Download using Python code
pip install modelscope
from modelscope import snapshot_download
model_dir = snapshot_download('Kedreamix/Linly-Talker')
```

Move All Models to the Current Directory

If you downloaded from Baidu Netdisk, you can refer to the directory structure at the end of the document to move the models.

```
# Move all models to the current directory
# Checkpoints contain SadTalker and Wav2Lip
mv Linly-Talker/checkpoints/* ./checkpoints

# Enhanced GFPGAN for SadTalker
# pip install gfpgan
# mv Linly-Talker/gfpan ./

# Voice cloning models
```

```
mv Linly-Talker/GPT_SoVITS/pretrained_models/* ./GPT_SoVITS/pretrained_models/

# Qwen large language model
mv Linly-Talker/Qwen ./

# MuseTalk model
mkdir -p ./Musetalk/models
mv Linly-Talker/MuseTalk/* ./Musetalk/models
```

For the convenience of deployment and usage, an `configs.py` file has been updated. You can modify some hyperparameters in this file for cu

```
# Device Running Port
port = 7870

# API Running Port and IP
# Localhost port is 127.0.0.1; for global port forwarding, use "0.0.0.0"
ip = '127.0.0.1'
api_port = 7871

# Linly Model Path
mode = 'api'  # For 'api', Linly-api-fast.py must be run first
mode = 'offline'
model_path = 'Linly-AI/Chinese-LLaMA-2-7B-hf'

# SSL Certificate (required for microphone interaction)
# Preferably an absolute path
ssl_certfile = "./https_cert/cert.pem"
ssl_keyfile = "./https_cert/key.pem"
```

This file allows you to adjust parameters such as the device running port, API running port, Linly model path, and SSL certificate paths for ease

API Documentation

In the api/README.md file, we provide detailed information about the usage and configuration of the Linly-Talker API. This documentation incl
recognition, and generating speech.

For detailed API interface descriptions, please refer to the `api/README.md` file.

ASR - Speech Recognition

For detailed information about the usage and code implementation of Automatic Speech Recognition (ASR), please refer to ASR - Bridging the (

Whisper

To implement ASR (Automatic Speech Recognition) using OpenAI's Whisper, you can refer to the specific usage methods provided in the GitHub

FunASR

The speech recognition performance of Alibaba's FunASR is quite impressive and it is actually better than Whisper in terms of Chinese languag

Coming Soon

Welcome everyone to provide suggestions, motivating me to continuously update the models and enrich the functionality of Linly-Talker.

TTS - Text To Speech

For detailed information about the usage and code implementation of Text-to-Speech (TTS), please refer to TTS - Empowering Digital Humans v

Edge TTS

To use Microsoft Edge's online text-to-speech service from Python without needing Microsoft Edge or Windows or an API key, you can refer to t

> ⚠ Warning
>
> Due to some issues with the Edge TTS repository, it seems that Microsoft has restricted certain IPs. For more details, refer to 403 error is bac

PaddleTTS

In practical use, there may be scenarios that require offline operation. Since Edge TTS requires an online environment to generate speech, we h

Coming Soon

Welcome everyone to provide suggestions, motivating me to continuously update the models and enrich the functionality of Linly-Talker.

Voice Clone

For detailed information about the usage and code implementation of Voice Clone, please refer to Voice Clone - Stealing Your Voice Quietly Du

GPT-SoVITS（Recommend）

Thank you for your open source contribution. I have also found the `GPT-SoVITS` voice cloning model to be quite impressive. You can find the p

XTTS

Coqui XTTS is a leading deep learning toolkit for Text-to-Speech (TTS) tasks, allowing for voice cloning and voice transfer to different languages

🐸 TTS is a library for advanced text-to-speech generation.

🚀 Over 1100 pre-trained models for various languages.

🛠️ Tools for training new models and fine-tuning existing models in any language.

📚 Utility programs for dataset analysis and management.

- Experience XTTS online https://huggingface.co/spaces/coqui/xtts
- Official GitHub repository: https://github.com/coqui-ai/TTS

CosyVoice

CosyVoice is an open-source multilingual speech understanding model developed by Alibaba's Tongyi Lab, focusing on high-quality speech syn

CosyVoice supports one-shot voice cloning technology, enabling the generation of realistic and natural-sounding voices with details such as pr

GitHub project link: CosyVoice GitHub

CosyVoice includes several pre-trained speech synthesis models, mainly:

1. **CosyVoice-300M**: Supports zero-shot and cross-lingual speech synthesis in Chinese, English, Japanese, Cantonese, Korean, and other langu
2. **CosyVoice-300M-SFT**: A model focused on supervised fine-tuning (SFT) inference.
3. **CosyVoice-300M-Instruct**: A model that supports command-based inference, capable of generating speech with specific tones, emotions,

Key Features

1. **Multilingual Support**: Capable of handling various languages including Chinese, English, Japanese, Cantonese, and Korean.
2. **Multi-style Speech Synthesis**: Allows control over the tone and emotion of the generated speech through commands.
3. **Streaming Inference Support**: Future updates will include streaming inference modes, such as KV caching and SDPA, for real-time optimi

Currently, Linly-Talker integrates three features from CosyVoice: pre-trained voice cloning, 3s rapid cloning, and cross-lingual cloning. Stay tune

	PROMPT TEXT
Pre-trained Voice	中文女 音色（'中文女', '中文男', '日语男', '粤语女', '英文女', '英文男', '韩语女'）
3s Language Cloning	希望你以后能够做的比我还好呦。

Cross-lingual Cloning	在那之后，完全收购那家公司，因此保持管理层的一致性，利益与即将加入家族的资产保持一致。这就是我们有时不买下全

Coming Soon

Welcome everyone to provide suggestions, motivating me to continuously update the models and enrich the functionality of Linly-Talker.

THG - Avatar

Detailed information about the usage and code implementation of digital human generation can be found in THG - Building Intelligent Digital I

SadTalker

Digital persona generation can utilize SadTalker (CVPR 2023). For detailed information, please visit https://sadtalker.github.io.

Before usage, download the SadTalker model:

```
bash scripts/sadtalker_download_models.sh
```

Baidu (百度云盘) (Password: `linl`)

Quark(夸克网盘)

> If downloading from Baidu Cloud, remember to place it in the `checkpoints` folder. The model downloaded from Baidu Cloud is named `sadt`

Wav2Lip

Digital persona generation can also utilize Wav2Lip (ACM 2020). For detailed information, refer to https://github.com/Rudrabha/Wav2Lip.

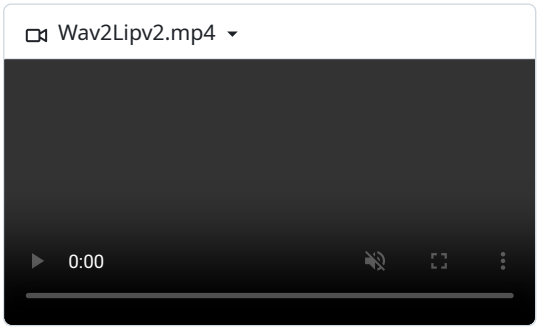
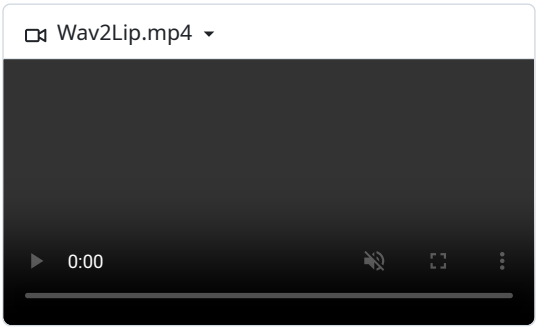
Before usage, download the Wav2Lip model:

Model	Description	Link to the model
Wav2Lip	Highly accurate lip-sync	Link
Wav2Lip + GAN	Slightly inferior lip-sync, but better visual quality	Link
Expert Discriminator	Weights of the expert discriminator	Link
Visual Quality Discriminator	Weights of the visual disc trained in a GAN setup	Link

Wav2Lipv2

Inspired by the repository https://github.com/primepake/wav2lip_288x288, Wav2Lipv2 uses a newly trained 288 model to achieve higher qualit

Additionally, by employing YOLO for facial detection, the overall effect is improved. You can compare and test the results in Linly-Talker. The mo

Wav2Lip	Wav2Lipv2
	

ER-NeRF

ER-NeRF (ICCV 2023) is a digital human built using the latest NeRF technology. It allows for the customization of digital characters and can reco

Updated: Taking inspiration from the likeness of Obama, for better results, consider cloning and customizing the voice of digital personas for ir

MuseTalk

MuseTalk is a real-time, high-quality audio-driven lip synchronization model capable of running at over 30 frames per second on an NVIDIA Tes

MuseTalk is trained to operate within the latent space of ft-mse-vae and offers the following features:

- **Unseen Face Synchronization**: It can modify unseen faces based on input audio, with a face region size of 256 x 256.
- **Multi-language Support**: Supports audio inputs in various languages, including Chinese, English, and Japanese.
- **High-performance Real-time Inference**: Achieves real-time inference at over 30 frames per second on an NVIDIA Tesla V100.
- **Facial Center Point Adjustment**: Allows the adjustment of the facial region's center point, significantly impacting the generated results.
- **HDTF Dataset Training**: Provides model checkpoints trained on the HDTF dataset.
- **Upcoming Training Code Release**: Training code will be released soon, facilitating further development and research.

MuseTalk offers an efficient and versatile tool for precise audio synchronization with facial expressions in virtual humans, marking a significant

In Linly-Talker, MuseTalk has been integrated to perform inference on videos based on MuseV, achieving an ideal speed for conversations with

Coming Soon

Welcome everyone to provide suggestions, motivating me to continuously update the models and enrich the functionality of Linly-Talker.

LLM - Conversation

For detailed information about the usage and code implementation of Large Language Models (LLM), please refer to LLM - Empowering Digital

Linly-AI

Linly-AI is a Large Language model developed by CVI at Shenzhen University. You can find more information about Linly-AI on their GitHub rep

Download Linly models: https://huggingface.co/Linly-AI/Chinese-LLaMA-2-7B-hf

You can use `git` to download:

```
git lfs install
git clone https://huggingface.co/Linly-AI/Chinese-LLaMA-2-7B-hf
```

Alternatively, you can use the `huggingface` download tool `huggingface-cli`:

```
pip install -U huggingface_hub

# Set up mirror acceleration
# Linux
export HF_ENDPOINT="https://hf-mirror.com"
# Windows PowerShell
$env:HF_ENDPOINT="https://hf-mirror.com"
```

```
huggingface-cli download --resume-download Linly-AI/Chinese-LLaMA-2-7B-hf --local-dir Linly-AI/Chinese-LLaMA-2-7B-hf
```

Qwen

Qwen is an AI model developed by Alibaba Cloud. You can check out the GitHub repository for Qwen here: https://github.com/QwenLM/Qwen

If you want to quickly use Qwen, you can choose the 1.8B model, which has fewer parameters and can run smoothly even with limited GPU me

You can download the Qwen 1.8B model from this link: https://huggingface.co/Qwen/Qwen-1_8B-Chat

You can use `git` to download:

```
git lfs install
git clone https://huggingface.co/Qwen/Qwen-1_8B-Chat
```

Alternatively, you can use the `huggingface` download tool `huggingface-cli`:

```
pip install -U huggingface_hub

# Set up mirror acceleration
# Linux
export HF_ENDPOINT="https://hf-mirror.com"
# Windows PowerShell
$env:HF_ENDPOINT="https://hf-mirror.com"

huggingface-cli download --resume-download Qwen/Qwen-1_8B-Chat --local-dir Qwen/Qwen-1_8B-Chat
```

Gemini-Pro

Gemini-Pro is an AI model developed by Google. To learn more about Gemini-Pro, you can visit their website: https://deepmind.google/technol

If you want to request an API key for Gemini-Pro, you can visit this link: https://makersuite.google.com/

ChatGPT

From OpenAI, requires API application. For more information, please visit https://platform.openai.com/docs/introduction.

ChatGLM

From Tsinghua University, for more information please visit https://github.com/THUDM/ChatGLM3.

GPT4Free

For free access to GPT-4 and other models, you can refer to https://github.com/xtekky/gpt4free. This resource provides methods to utilize thes

LLM Multiple Model Selection

In the `webui.py` file, easily select the model you need. ⚠️ For the first run, make sure to download the model first. Refer to Qwen1.8B.

Coming Soon

Welcome everyone to provide suggestions, motivating me to continuously update the models and enrich the functionality of Linly-Talker.

Optimizations

Some optimizations:

- Use fixed input face images, extract features beforehand to avoid reading each time
- Remove unnecessary libraries to reduce total time
- Only save final video output, don't save intermediate results to improve performance
- Use OpenCV to generate final video instead of mimwrite for faster runtime

Gradio

Gradio is a Python library that provides an easy way to deploy machine learning models as interactive web apps.

For Linly-Talker, Gradio serves two main purposes:

1. **Visualization & Demo**: Gradio provides a simple web GUI for the model, allowing users to see the results intuitively by uploading an imag
2. **User Interaction**: The Gradio GUI can serve as a frontend to allow end users to interact with Linly-Talker. Users can upload their own imag

Specifically, we create a Gradio Interface in app.py that takes image and text inputs, calls our function to generate the response video, and disp

In summary, Gradio provides visualization and user interaction interfaces for Linly-Talker, serving as effective means for showcasing system ca

> If considering real-time conversation, it may be necessary to switch to a different framework or customize Gradio. Looking forward to worki

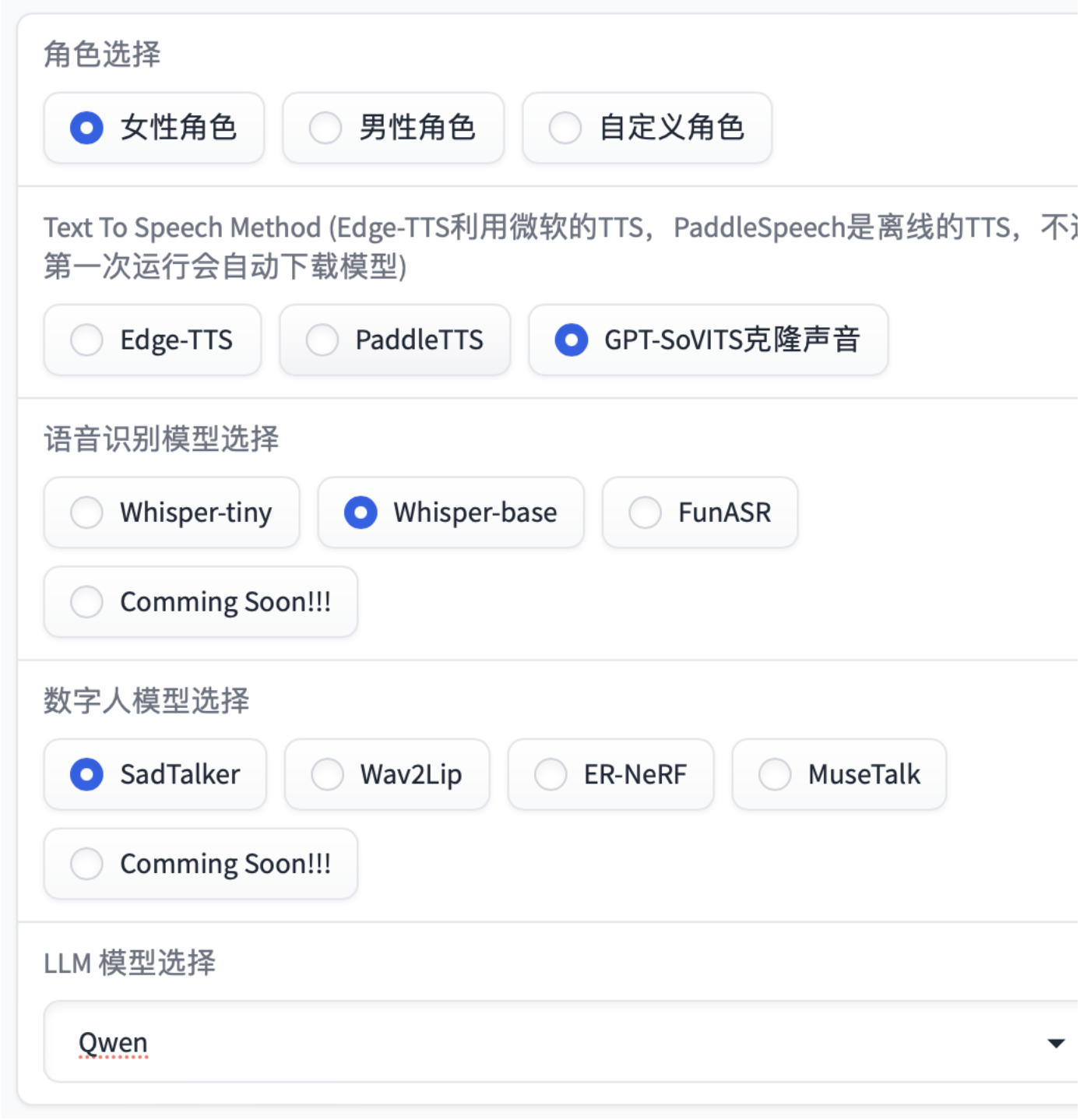
Start WebUI

Previously, I had separated many versions, but it became cumbersome to run multiple versions. Therefore, I have added a WebUI feature to pr

WebUI

The current features available in the WebUI are as follows:

- [x] Text/Voice-based dialogue with virtual characters (fixed characters with male and female roles)
- [x] Dialogue with virtual characters using any image (you can upload any character image)
- [x] Multi-turn GPT dialogue (incorporating historical dialogue data to maintain context)
- [x] Voice cloning dialogue (based on GPT-SoVITS settings for voice cloning, including a built-in smoky voice that can be cloned based on the vo
- [x] Digital Persona Text/Voice Playback (based on input text/voice)
- [x] Multiple modules➕Multiple models➕Multiple choices
    - [x] Multiple role selections: Female/Male/Custom (each part can automatically upload images) Coming Soon
    - [x] Multiple TTS model selections: EdgeTTS / PaddleTTS / GPT-SoVITS / CosyVoice / Coming Soon
    - [x] Multiple LLM model selections: Linly / Qwen / ChatGLM / GeminiPro / ChatGPT / Coming Soon
    - [x] Multiple Talker model selections: Wav2Lip / Wav2Lipv2 / SadTalker / ERNeRF / MuseTalk/ Coming Soon
    - [x] Multiple ASR model selections: Whisper / FunASR / Coming Soon



You can directly run the web UI to obtain results. The page you will see is as follows:

```
# WebUI
python webui.py
```

This time, we've updated the interface. We can freely select the fine-tuned model of GPT-SoVITS to achieve voice cloning. Simply upload a refer

TTS Method语音方法调节

Edge-TTS | PaddleTTS | GPT-SoVITS

GPT模型路径

- [ ] Gnews
- [ ] chinese-hubert-base
- [ ] chinese-roberta-wwm-ext-large
- [x] Gnews-e15.ckpt
- [ ] s1bert25hz-2kh-longer-epoch=68e-step=50232.ckpt

SoVITS模型路径

- [ ] Gnews
- [ ] chinese-hubert-base
- [ ] chinese-roberta-wwm-ext-large
- [x] Gnews_e8_s96.pth
- [ ] s2D488k.pth
- [ ] s2G488k.pth

加载模型

请上传3~10秒内参考音频，超过会报错！

0:00 0:04

1x

- [ ] 使用语音问答的麦克风

参考音频的文本

才之后很快就能够立起来这个呢是要国内

参考音频的语种

中文

Old Verison

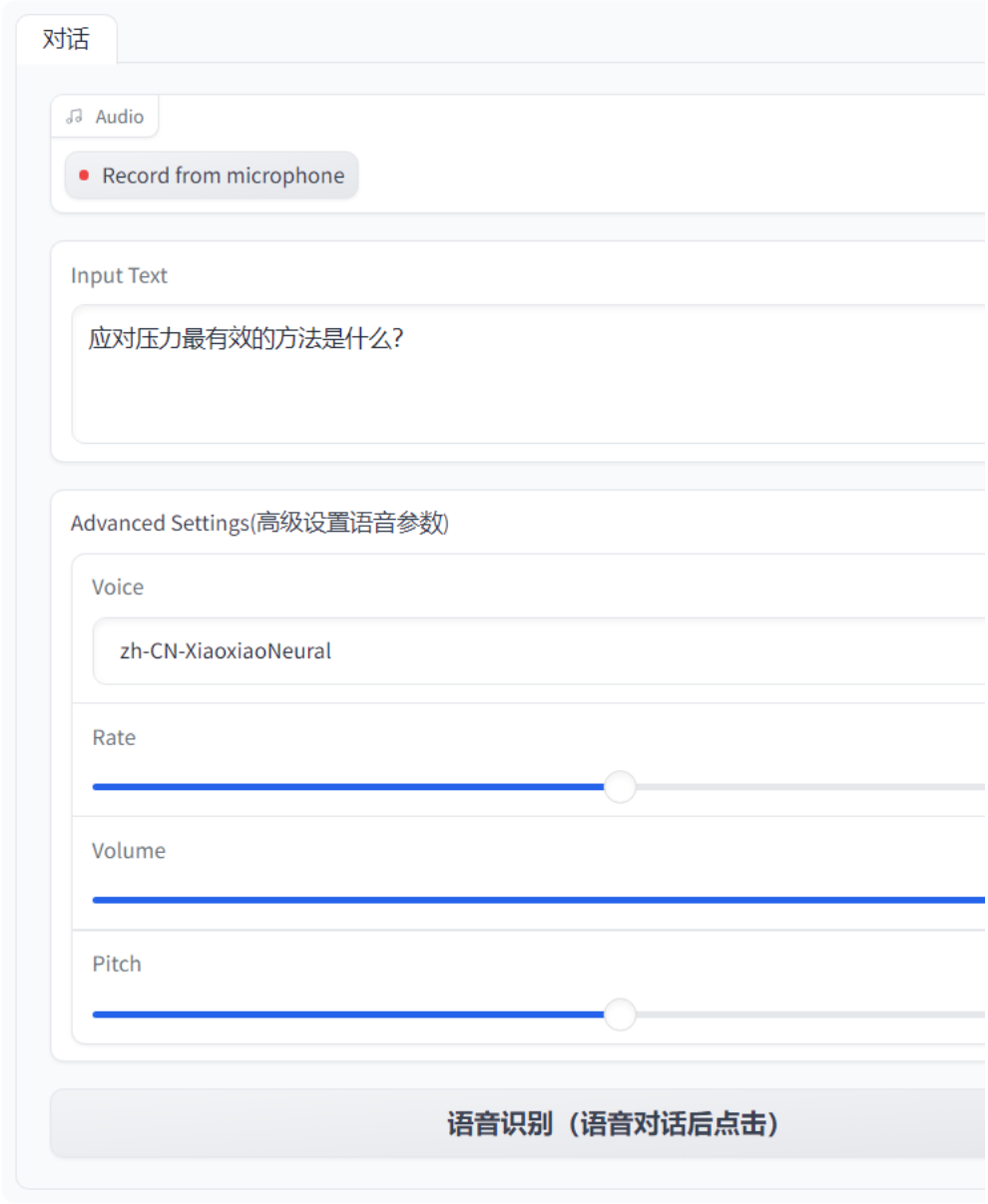
There are three modes for the current startup, and you can choose a specific setting based on the scenario.

The first mode involves fixed Q&A with a predefined character, eliminating preprocessing time.

```
python app.py
```

Lin

Linly-Talker 是一款智能 AI 对话



The first mode has recently been updated to include the Wav2Lip model for dialogue.

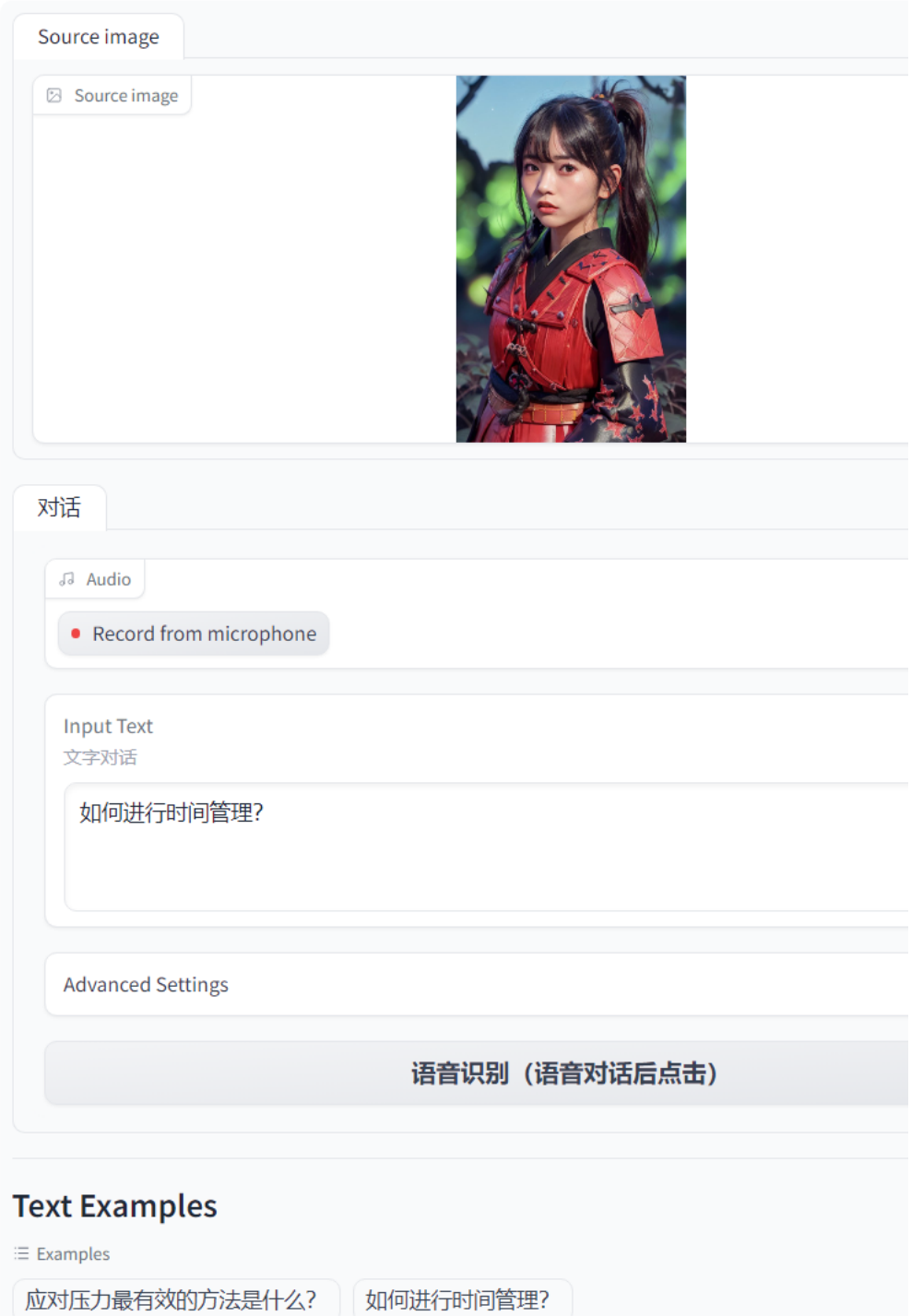
```
python appv2.py
```

The second mode allows for conversing with any uploaded image.

```
python app_img.py
```

Lin

Linly-Talker 是一款智能 AI 对话



The third mode builds upon the first one by incorporating a large language model for multi-turn GPT conversations.

```
python app_multi.py
```

Linly

Linly-Talker 是一款智能 AI 对话

Now, the part of voice cloning has been added, allowing for freely switching between cloned voice models and corresponding person images. H

```
python app_vits.py
```

A fourth method has been added, which does not fixate on a specific scenario for conversation. Instead, it allows for direct input of voice or the

> ER-NeRF is trained on videos of a single individual, so a specific model needs to be replaced to render and obtain the correct results. It come

```
python app_talk.py
```

MuseTalk has been integrated into Linly-Talker, enabling efficient preprocessing of MuseV-generated videos. Once preprocessed, these videos t

To run the application, use the following command:

```
python app_musetalk.py
```

Linly 智能对

Linly-Talker是一款创新的数字人对话系统，它融合了最新

MuseV Video

MuseV: need help? please visit MuseVDemo to generate Videc
https://huggingface.co/spaces/AnchorFake/MuseVDemo

Reference Video

将视频拖放到此处
- 或 -
点击上传

BBox_shift 推荐值下限，在生成初始结果后生成相应的 bb
以根据该参考值进行调整。一般来说，在我们的实验观察
部分移动）通常会增加嘴巴的张开度，而负值（向上半部
开度。然而，需要注意的是，这并不是绝对的规则，用户
和期望效果来调整该参数。

BBox_shift value, px

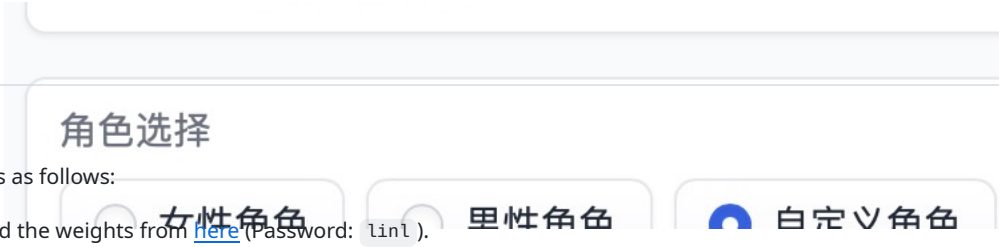
0

bbox_shift_

TTS Method语音方法调节

Folder structure

> **Note**
>
> The folder structure of the weight files is as follows:
>
> - `Baidu (百度云盘)`: You can download the weights from here (Password: `linl`).
> - `huggingface`: You can access the weights at this link.
> - `modelscope`: The weights will be available soon at this link.
> - `Qurak(夸克网盘)`：You can download the weights from here




```
Linly-Talker/
├── checkpoints
│   ├── audio_visual_encoder.pth
│   ├── hub
│   │   └── checkpoints
│   │       └── s3fd-619a316812.pth
│   ├── lipsync_expert.pth
│   ├── mapping_00109-model.pth.tar
│   ├── mapping_00229-model.pth.tar
│   ├── May.json
│   ├── May.pth
│   ├── Obama_ave.pth
│   ├── Obama.json
│   ├── Obama.pth
│   ├── ref_eo.npy
│   ├── ref.npy
│   ├── ref.wav
│   ├── SadTalker_V0.0.2_256.safetensors
│   ├── visual_quality_disc.pth
│   ├── wav2lip_gan.pth
│   └── wav2lip.pth
├── gfpgan
│   └── weights
│       ├── alignment_WFLW_4HG.pth
│       └── detection_Resnet50_Final.pth
├── GPT_SoVITS
│   └── pretrained_models
│       ├── chinese-hubert-base
│       │   ├── config.json
│       │   ├── preprocessor_config.json
│       │   └── pytorch_model.bin
│       ├── chinese-roberta-wwm-ext-large
│       │   ├── config.json
│       │   ├── pytorch_model.bin
│       │   └── tokenizer.json
│       ├── README.md
│       ├── s1bert25hz-2kh-longer-epoch=68e-step=50232.ckpt
│       ├── s2D488k.pth
│       ├── s2G488k.pth
│       └── speech_paraformer-large_asr_nat-zh-cn-16k-common-vocab8404-pytorch
├── MuseTalk
│   ├── models
│   │   ├── dwpose
│   │   │   └── dw-ll_ucoco_384.pth
│   │   ├── face-parse-bisent
│   │   │   ├── 79999_iter.pth
│   │   │   └── resnet18-5c106cde.pth
│   │   ├── musetalk
│   │   │   ├── musetalk.json
│   │   │   └── pytorch_model.bin
│   │   ├── README.md
│   │   ├── sd-vae-ft-mse
│   │   │   ├── config.json
│   │   │   └── diffusion_pytorch_model.bin
│   │   └── whisper
│   │       └── tiny.pt
├── Qwen
│   └── Qwen-1_8B-Chat
│       ├── assets
│       │   ├── logo.jpg
```

```
│       │   ├── qwen_tokenizer.png
│       │   ├── react_showcase_001.png
│       │   ├── react_showcase_002.png
│       │   └── wechat.png
│       ├── cache_autogptq_cuda_256.cpp
│       ├── cache_autogptq_cuda_kernel_256.cu
│       ├── config.json
│       ├── configuration_qwen.py
│       ├── cpp_kernels.py
│       ├── examples
│       │   └── react_prompt.md
│       ├── generation_config.json
│       ├── LICENSE
│       ├── model-00001-of-00002.safetensors
│       ├── model-00002-of-00002.safetensors
│       ├── modeling_qwen.py
│       ├── model.safetensors.index.json
│       ├── NOTICE
│       ├── qwen_generation_utils.py
│       ├── qwen.tiktoken
│       ├── README.md
│       ├── tokenization_qwen.py
│       └── tokenizer_config.json
├── Whisper
│   ├── base.pt
│   └── tiny.pt
├── FunASR
│   ├── punc_ct-transformer_zh-cn-common-vocab272727-pytorch
│   │   ├── configuration.json
│   │   ├── config.yaml
│   │   ├── example
│   │   │   └── punc_example.txt
│   │   ├── fig
│   │   │   └── struct.png
│   │   ├── model.pt
│   │   ├── README.md
│   │   └── tokens.json
│   ├── speech_fsmn_vad_zh-cn-16k-common-pytorch
│   │   ├── am.mvn
│   │   ├── configuration.json
│   │   ├── config.yaml
│   │   ├── example
│   │   │   └── vad_example.wav
│   │   ├── fig
│   │   │   └── struct.png
│   │   ├── model.pt
│   │   └── README.md
│   └── speech_seaco_paraformer_large_asr_nat-zh-cn-16k-common-vocab8404-pytorch
│       ├── am.mvn
│       ├── asr_example_hotword.wav
│       ├── configuration.json
│       ├── config.yaml
│       ├── example
│       │   ├── asr_example.wav
│       │   └── hotword.txt
│       ├── fig
│       │   ├── res.png
│       │   └── seaco.png
│       ├── model.pt
│       ├── README.md
│       ├── seg_dict
│       └── tokens.json
└── README.md
```

Reference

ASR

- https://github.com/openai/whisper
- https://github.com/alibaba-damo-academy/FunASR

TTS

- https://github.com/rany2/edge-tts

- https://github.com/PaddlePaddle/PaddleSpeech

LLM

- https://github.com/CVI-SZU/Linly
- https://github.com/QwenLM/Qwen
- https://deepmind.google/technologies/gemini/
- https://github.com/THUDM/ChatGLM3
- https://openai.com

THG

- https://github.com/OpenTalker/SadTalker
- https://github.com/Rudrabha/Wav2Lip
- https://github.com/Fictionarry/ER-NeRF

Voice Clone

- https://github.com/RVC-Boss/GPT-SoVITS
- https://github.com/coqui-ai/TTS

License

> Caution
>
> When using this tool, please comply with all applicable laws, including copyright, data protection, and privacy laws. Do not use, modify, distr

`Linly-Talker` follows the MIT License. In addition to adhering to the MIT License, ensure that you comply with all license agreements for any

Star History