I want to create a python program that can be trained to clone my voice stored in file "./data/my\_voice.wav". Next, when I provide the transcript file "./data/script.txt", the program should convert it to "./data/speech.mp3" in my cloned voice. Use OpenVoice V2 (MyShell) based model First Give me the environment setup instructions. Next Give me the environment setup diagnostics script/ sanity checks. Do not clutter with the final voice cloning python program now. I shall ask for the same later, once the setup is working fine.

**ChatGPT said:**

Here’s a streamlined **setup plan** for your OpenVoice V2 (MyShell) environment—let’s break it into two clear sections. Once everything is working, I’ll help you craft the Python cloning script you want.

**Environment Setup Instructions**

Based on the official OpenVoice V2 install guidance, community tutorials, and recent DevOps guides:

1. **Prerequisites**
   * Your system should have:
     + **Python 3.9** (as used in official repo) or **3.10** (works via community tutorial) [Hugging Face](https://huggingface.co/myshell-ai/OpenVoiceV2?utm_source=chatgpt.com)[HeyLetsLearnSomething](https://heyletslearnsomething.com/blog/clone-your-voice-using-openvoice?utm_source=chatgpt.com)
     + Conda (optional) or use venv depending on your preference.
     + **Git** (to clone the repo).
     + **FFmpeg** (for audio processing).
     + Optional: **CUDA Toolkit + GPU drivers** (if using GPU acceleration).
2. **Clone the OpenVoice Repository**
3. git clone https://github.com/myshell-ai/OpenVoice.git
4. cd OpenVoice
5. **Create and Activate Virtual Environment**
   * Using Conda:
   * conda create -n openvoice python=3.9
   * conda activate openvoice
   * Or using venv:
   * python3 -m venv venv
   * source venv/bin/activate # on Linux/macOS
   * venv\Scripts\activate # on Windows
6. **Install Dependencies**
7. pip install -e . # installs OpenVoice in editable mode
8. pip install -r requirements.txt
9. **Install FFmpeg**
   * On Linux:
   * sudo apt install ffmpeg
   * Or with Conda:
   * conda install -c conda-forge ffmpeg
10. **Optional GPU Support**
    * Install CUDA toolkit and GPU-enabled PyTorch (via official PyTorch site).
11. **Download Model Checkpoints**
    * Download V2 model checkpoint(s) (and V1 if needed) from the official GitHub releases.
    * Place them into the checkpoints/ directory inside the cloned OpenVoice folder.  
      [Hugging Face](https://huggingface.co/myshell-ai/OpenVoiceV2?utm_source=chatgpt.com)[Medium](https://medium.com/%40Muhammad.Sarim/automating-voice-cloning-with-openvoice-fastapi-and-n8n-a-complete-guide-to-self-hosting-and-7bce505c3edc?utm_source=chatgpt.com)
12. **Optional Extras** (from community insights)
    * For a notebook-based workflow (via demo files):
    * conda install -c conda-forge jupyterlab
    * pip install notebook ipywidgets
    * If using MeloTTS (optional for advanced features):
    * pip install git+https://github.com/myshell-ai/MeloTTS.git
    * python -m unidic download # for language models

**Environment Setup Diagnostics / Sanity Checks Script**

Here's a Python shell or script snippet to verify that your environment is correctly set up:

import sys

import subprocess

import torch

def check():

print("Python version:", sys.version)

try:

import openvoice

print("OpenVoice imported successfully.")

except ImportError:

print("ERROR: Could not import OpenVoice.")

# Check for GPU

cuda\_available = torch.cuda.is\_available()

print("CUDA available:", cuda\_available)

if cuda\_available:

print("CUDA device count:", torch.cuda.device\_count())

# FFmpeg

try:

completed = subprocess.run(["ffmpeg", "-version"], capture\_output=True, text=True)

print("FFmpeg installed:", completed.returncode == 0)

except FileNotFoundError:

print("ERROR: FFmpeg not found.")

# Model checkpoint presence

import os

ckpts = os.listdir(os.path.join(os.getcwd(), "checkpoints"))

print("Checkpoints found:", ckpts)

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

check()

**Usage:**

1. Save the above as diagnostics.py.
2. Run:
3. python diagnostics.py
4. Confirm output includes:
   * Correct Python version.
   * Successful OpenVoice import.
   * GPU/CUDA availability (if expected).
   * FFmpeg installed.
   * One or more model checkpoints detected.

Let me know once you've run through these steps and diagnostics—then we’ll move on to building your voice-cloning Python program.

**You said:**

I will use vs code or pycharm and I do not have GPU

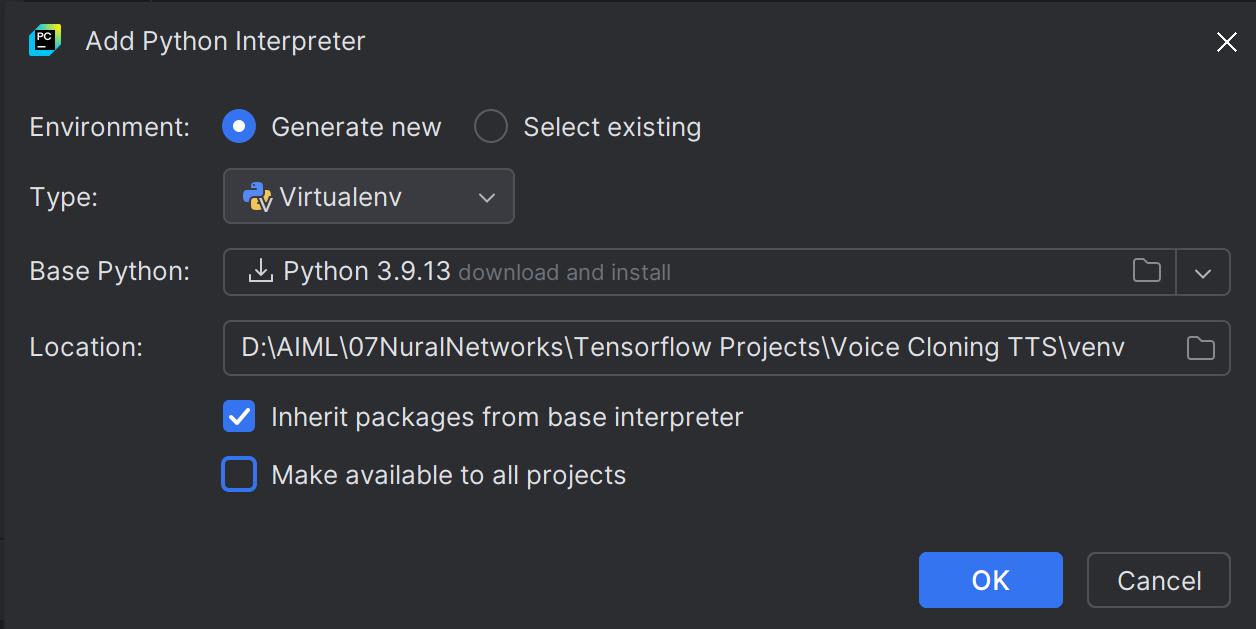
**ChatGPT said:**

Perfect 👍 — since you’ll be working on **VS Code or PyCharm** and **without a GPU**, the setup becomes a little simpler. Here’s a **refined setup plan tailored to your case**:

**🛠 Environment Setup (CPU-only, VS Code / PyCharm)**

**PyCharm version 2025.2.1.1**

1. **Install prerequisites**
   * Python **3.9.13**



* + Git
  + FFmpeg (needed for audio conversions)

**FFmpeg installation**:

* + Windows: download from [ffmpeg.org](https://ffmpeg.org/download.html), add bin/ folder to PATH
  + **winget install -e --id Gyan.FFmpeg** 🡪 to install from cmd (run as administrator)
  + Linux/macOS:
  + sudo apt install ffmpeg # Ubuntu/Debian
  + brew install ffmpeg # macOS (Homebrew)

1. Create a directory (VoiceCloningTTS) Project directory, and open it in Pycharm
2. Create .gitignore

processed  
.idea

1. **Clone the OpenVoice repository from PyCharm Terminal**
2. git clone https://github.com/myshell-ai/OpenVoice.git
3. cd OpenVoice
4. dir
5. cd ..
6. **Create a virtual environment**  
   In **PyCharm terminal**:
7. python -m venv venv
8. # Activate it:
9. venv\Scripts\activate # on Windows if venv is not Activated: No (venv) at the start of the command prompt of Pychatm Terminal
10. source venv/bin/activate # on macOS/Linux
11. **Install dependencies**
12. **cd OpenVoice**
13. pip install -e .
14. If Error [Errno 13] Permission denied::::
15. python -c "import sys; print(sys.executable)"
16. python -m pip –version
17. must show: (python 3.9)
18. python -m pip install --upgrade pip setuptools wheel
19. pip install --no-cache-dir -e .
20. pip install -r requirements.txt

⚠️ Since you’re CPU-only, **do not install GPU-enabled PyTorch**. Instead, install CPU-only PyTorch:

pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu

1. **Download model checkpoints from clicking** [**https://myshell-public-repo-host.s3.amazonaws.com/openvoice/checkpoints\_v2\_0417.zip**](https://myshell-public-repo-host.s3.amazonaws.com/openvoice/checkpoints_v2_0417.zip)
   * Go to [OpenVoiceV2 Hugging Face page](https://huggingface.co/myshell-ai/OpenVoiceV2?utm_source=chatgpt.com)
   * Download the **checkpoint files** (usually .pth or similar)
   * Place them into:
   * OpenVoice/checkpoints/
2. pip install git+https://github.com/myshell-ai/MeloTTS.git
3. if access denied error:
   * **pip install --no-cache-dir git+https://github.com/myshell-ai/MeloTTS.git**
4. **python -m unidic download**
5. **python -c "import MeCab; tagger = MeCab.Tagger(); print('MeCab OK')"**
   * Do this for a quick sanity check, Should print **MeCab Ok**
6. **python -c "from melo.api import TTS; print('MeloTTS imported successfully!')"**
   * Run this to quickly check if MeloTTS is ok, should print **MeloTTS imported successfully!**
7. **Install nltk dependencies:**
   * **mkdir .venv\nltk\_data** 🡪 creates a directory named “nltk\_data” in project .venv
   * **python -m nltk.downloader -d ".venv/nltk\_data" averaged\_perceptron\_tagger\_eng punkt** 🡪 downloads and stores averaged\_perceptron\_tagger\_eng & punkt in “nltk\_data” directory
8. **Open in VS Code or PyCharm**
   * Select the venv Python interpreter.
   * Create/run your scripts inside this environment.

**✅ Diagnostics / Sanity Check Script (CPU-only)**

Save this as diagnostics.py inside your project directory:

import sys  
import subprocess  
import torch  
import os  
import warnings  
  
warnings.filterwarnings("ignore",  
 category=UserWarning,  
 module="librosa") # Suppresses deprecation warning for "pkg\_resources" from "setuptools"  
  
  
print("🚀 Running diagnostics script...")  
print("📂 diagnostics.py is located at:", \_\_file\_\_)  
  
def check():  
 # --- Python version ---  
 print("🔎 Python version:", sys.version)  
  
 # --- OpenVoice ---  
 try:  
 import OpenVoice.openvoice  
 print("✅ openvoice imported successfully.")  
 except ImportError:  
 print("❌ ERROR: Could not import openvoice.")  
  
 # --- MeloTTS ---  
 try:  
 import melo  
 from melo.api import TTS  
 print("✅ MeloTTS imported successfully from melo.api\n✅ MeloTTS base module imported successfully.")  
 except ImportError:  
 print("❌ ERROR: Could not import MeloTTS base module.\n❌ ERROR: Could not import MeloTTS from melo.api.")  
  
 # --- MeCab ---  
 try:  
 import MeCab  
 print("✅ MeCab imported successfully.\n")  
 except ImportError as e:  
 print("❌ MeCab is not installed.")  
 print("Exception details:", e, "\n")  
  
 # --- UniDic ---  
 try:  
 import MeCab  
 # Attempt to initialize a tagger with UniDic  
 tagger = MeCab.Tagger(".venv/Lib/site-packages/unidic")  
 print("📦 UniDic dictionary available for MeCab.\n")  
 except RuntimeError as e:  
 print("❌ UniDic dictionary not found for MeCab.")  
 print("Exception details:", e, "\n")  
 except ImportError:  
 # MeCab not installed, skip this check  
 pass  
  
 # --- NLTK ---  
 # 1. Import check  
 try:  
 import nltk  
 print("✅ NLTK imported successfully.")  
 except ImportError as e:  
 print("❌ ERROR: Could not import NLTK.")  
 print("Exception details:", e)  
  
 # 2. 'punkt' tokenizer check  
 try:  
 import nltk  
 nltk.data.find("tokenizers/punkt")  
 print("📦 NLTK 'punkt' tokenizer available.")  
 except LookupError as e:  
 print("❌ ERROR: NLTK 'punkt' tokenizer not found.")  
 print("Run: nltk.download('punkt')")  
 print("Exception details:", e)  
  
 # 3. 'averaged\_perceptron\_tagger\_eng' check  
 try:  
 import nltk  
 nltk.data.find("taggers/averaged\_perceptron\_tagger\_eng")  
 print("📦 'averaged\_perceptron\_tagger\_eng' found.\n")  
 except LookupError as e:  
 print("❌ ERROR: 'averaged\_perceptron\_tagger\_eng' not found.")  
 print("Run: nltk.download('averaged\_perceptron\_tagger\_eng')")  
 print("Exception details:", e)  
  
 # --- Torch / CUDA ---  
 print("💻 Torch version:", torch.\_\_version\_\_)  
 print("⚡ CUDA available:", torch.cuda.is\_available())  
 if not torch.cuda.is\_available():  
 print(" (Running on CPU mode)")  
  
 # --- FFmpeg ---  
 try:  
 completed = subprocess.run(["ffmpeg", "-version"], capture\_output=True, text=True)  
 print("🎵 FFmpeg installed:", completed.returncode == 0)  
 except FileNotFoundError:  
 print("❌ ERROR: FFmpeg not found.")  
  
 # --- Model checkpoints ---  
 ckpt\_path = os.path.join(os.getcwd(), "OpenVoice", "checkpoints")  
 if os.path.exists(ckpt\_path):  
 ckpts = os.listdir(ckpt\_path)  
 print("📂 Checkpoints found:", ckpts if ckpts else "None found")  
 else:  
 print("❌ No 'checkpoints/' folder found.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 check()

* Run with the following command 🡪 **Python diagnostics.py**
* Following is the console output which indicates everything is ready:

**🚀 Running diagnostics script...**

**📂 diagnostics.py is located at: C:\GitHub\AIML\07NuralNetworks\Tensorflow Projects\voice\_cloning\_tts\diagnostics.py**

**🔎 Python version: 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)]**

**✅ openvoice imported successfully.**

**✅ MeloTTS imported successfully from melo.api**

**✅ MeloTTS base module imported successfully.**

**✅ MeCab imported successfully.**

**📦 UniDic dictionary available for MeCab.**

**✅ NLTK imported successfully.**

**📦 NLTK 'punkt' tokenizer available.**

**📦 'averaged\_perceptron\_tagger\_eng' found.**

**💻 Torch version: 2.8.0+cpu**

**⚡ CUDA available: False**

**(Running on CPU mode)**

**🎵 FFmpeg installed: True**

**📂 Checkpoints found: ['base\_speakers', 'converter']**

**✅ main.py:**

import torch  
import nltk  
from pathlib import Path  
import shutil  
  
# ------------------------  
# NLTK setup  
# ------------------------  
  
# Create venv-specific nltk\_data folder  
NLTK\_DATA\_DIR = Path(".venv") / "nltk\_data"  
NLTK\_DATA\_DIR.mkdir(exist\_ok=True)  
  
# Tell NLTK where to look  
nltk.data.path.append(str(NLTK\_DATA\_DIR))  
  
# Download required resources if missing  
for resource in ["averaged\_perceptron\_tagger", "punkt"]:  
 try:  
 nltk.data.find(f"taggers/{resource}" if "tagger" in resource else f"tokenizers/{resource}")  
 except LookupError:  
 print(f"Downloading NLTK resource: {resource} ...")  
 nltk.download(resource, download\_dir=str(NLTK\_DATA\_DIR))  
  
# 🔧 Fix g2p\_en expecting "averaged\_perceptron\_tagger\_eng"  
tagger\_src = NLTK\_DATA\_DIR / "taggers" / "averaged\_perceptron\_tagger"  
tagger\_dst = NLTK\_DATA\_DIR / "taggers" / "averaged\_perceptron\_tagger\_eng"  
  
if tagger\_src.exists() and not tagger\_dst.exists():  
 print("🔧 Creating copy for averaged\_perceptron\_tagger\_eng...")  
 shutil.copytree(tagger\_src, tagger\_dst)  
  
# ------------------------  
# OpenVoice / MeloTTS imports  
# ------------------------  
  
from openvoice import se\_extractor  
from openvoice.api import ToneColorConverter  
from melo.api import TTS  
  
# ------------------------  
# Paths  
# ------------------------  
DATA\_DIR = Path("./data")  
VOICE\_FILE = DATA\_DIR / "my\_voice.wav"  
SCRIPT\_FILE = DATA\_DIR / "script.txt"  
OUTPUT\_FILE = DATA\_DIR / "speech.mp3"  
  
CHECKPOINTS\_DIR = Path("./OpenVoice/checkpoints")  
CONVERTER\_CKPT = CHECKPOINTS\_DIR / "converter"  
BASE\_SPEAKERS = CHECKPOINTS\_DIR / "base\_speakers"  
  
TMP\_AUDIO = DATA\_DIR / "tmp.wav" # temporary TTS output  
  
# ------------------------  
# Sanity checks  
# ------------------------  
for path in [VOICE\_FILE, SCRIPT\_FILE, CONVERTER\_CKPT, BASE\_SPEAKERS]:  
 if not path.exists():  
 raise FileNotFoundError(f"Required file or folder not found: {path}")  
  
# ------------------------  
# Load transcript  
# ------------------------  
with open(SCRIPT\_FILE, "r", encoding="utf-8") as f:  
 transcript\_text = f.read().strip()  
if not transcript\_text:  
 raise ValueError("Transcript file is empty!")  
  
# ------------------------  
# Device  
# ------------------------  
device = "cuda:0" if torch.cuda.is\_available() else "cpu"  
print(f"Using device: {device}")  
  
# ------------------------  
# Initialize ToneColorConverter  
# ------------------------  
tone\_color\_converter = ToneColorConverter(  
 f"{CONVERTER\_CKPT}/config.json",  
 device=device  
)  
tone\_color\_converter.load\_ckpt(f"{CONVERTER\_CKPT}/checkpoint.pth")  
  
# ------------------------  
# Extract speaker embedding from your voice  
# ------------------------  
print("🎤 Extracting speaker embedding from reference voice...")  
target\_se, \_ = se\_extractor.get\_se(str(VOICE\_FILE), tone\_color\_converter, vad=True)  
print("✅ Speaker embedding extracted.")  
  
# ------------------------  
# English-only TTS  
# ------------------------  
print("📝 Synthesizing text using English MeloTTS...")  
  
# Force English language  
model = TTS(language="EN", device=device)  
speaker\_ids = model.hps.data.spk2id  
  
# Use first available speaker in English base model  
speaker\_id = next(iter(speaker\_ids.values()))  
  
# Generate temporary WAV  
model.tts\_to\_file(transcript\_text, speaker\_id, str(TMP\_AUDIO))  
print(f"✅ Temporary TTS output saved to {TMP\_AUDIO}")  
  
# ------------------------  
# Apply cloned voice using ToneColorConverter  
# ------------------------  
print("🔊 Converting to cloned voice...")  
tone\_color\_converter.convert(  
 audio\_src\_path=str(TMP\_AUDIO),  
 src\_se=torch.load(f"{BASE\_SPEAKERS}/ses/en-newest.pth", map\_location=device),  
 tgt\_se=target\_se,  
 output\_path=str(OUTPUT\_FILE),  
 message="@MyShell"  
)  
print(f"✅ Cloned speech saved to: {OUTPUT\_FILE}")  
  
# Remove temporary WAV  
# TMP\_AUDIO.unlink(missing\_ok=True)