contents of my 2 files inside 003\_voice\_proj (project folder) are: diagnostics.py ----------------------------------------------------------- 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.") except ImportError as e: print("❌ MeCab is not installed.") print("Exception details:", e, "") # --- 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.") except RuntimeError as e: print("❌ UniDic dictionary not found for MeCab.") print("Exception details:", e) 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.") 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.") # --- Sound I/O: sounddevice + soundfile --- try: import sounddevice as sd import soundfile as sf print("✅ sounddevice imported successfully.") print("✅ soundfile imported successfully.") # Versions / backend try: pa\_ver = sd.get\_portaudio\_version() pa\_ver\_str = pa\_ver[1] if isinstance(pa\_ver, tuple) and len(pa\_ver) > 1 else str(pa\_ver) print(f"🎚️ PortAudio backend: {pa\_ver\_str}") except Exception as e: print("⚠️ Could not query PortAudio version:", e) # Default devices try: default\_in, default\_out = (sd.default.device or (None, None)) print(f"🎤 Default input device index: {default\_in}, output index: {default\_out}") if default\_in is not None and default\_in >= 0: dev = sd.query\_devices(default\_in) host = sd.query\_hostapis(dev['hostapi'])['name'] print(f" Input device: {dev['name']} (host API: {host})") except Exception as e: print("⚠️ Could not query default audio devices:", e) # List a few input devices try: devices = sd.query\_devices() input\_devs = [ f"[{i}] {d['name']} ({sd.query\_hostapis(d['hostapi'])['name']})" for i, d in enumerate(devices) if d.get('max\_input\_channels', 0) > 0 ] print(f"🎛️ Input devices found: {len(input\_devs)}") for line in input\_devs[:5]: print(" •", line) if len(input\_devs) > 5: print(" … (showing first 5)") except Exception as e: print("⚠️ Could not enumerate audio devices:", e) # Check microphone supports 16 kHz / 22.05 kHz mono for sr in (16000, 22050): try: sd.check\_input\_settings(samplerate=sr, channels=1) print(f"✅ Mic supports mono {sr} Hz input.") except Exception as e: print(f"❌ Mic may not support mono {sr} Hz input:", e) # Check soundfile can write WAV/PCM\_16 try: wav\_ok = "WAV" in sf.available\_formats() pcm16\_ok = "PCM\_16" in sf.available\_subtypes("WAV") if wav\_ok and pcm16\_ok: print("✅ soundfile supports WAV / PCM\_16 writing.") else: print("❌ soundfile missing WAV/PCM\_16 support.") except Exception as e: print("⚠️ Could not query soundfile formats:", e) # Optional quick mic test (2s @ 16k mono) — enable with DIAG\_MIC\_TEST=1 if os.getenv("DIAG\_MIC\_TEST") == "1": out = "./data/diag\_mic\_test.wav" os.makedirs(os.path.dirname(out), exist\_ok=True) duration = float(os.getenv("DIAG\_MIC\_TEST\_SECONDS", "2.0")) print(f"🎙️ Recording {duration}s at 16k mono to {out} ...") audio = sd.rec(int(duration \* 16000), samplerate=16000, channels=1, dtype='int16') sd.wait() sf.write(out, audio, 16000, subtype="PCM\_16") print("✅ Mic test recorded:", out) except ImportError as e: print("❌ ERROR: Could not import sounddevice and/or soundfile.") print("Exception details:", e) print(" Try: pip install sounddevice soundfile") # --- 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() ----------------------------------------------------------- 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.openvoice import se\_extractor from OpenVoice.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) ----------------------------------------------------------- I want to set up my virtual environment to run the above. I have python 3.9.13 and PyCharm on Windows 11. Give me the step by step setup instructions.

**ChatGPT said:**

Thought for 1m 45s

Awesome — you’ve already done the hard part (clean code!). Below is a clean, Windows-11 friendly setup that matches your scripts and Python 3.9.13.

**1) Put your project in place**

Inside your 003\_voice\_proj folder, make sure you have:

003\_voice\_proj/

diagnostics.py

main.py

data/

my\_voice.wav *# your reference voice (mono is fine)*

script.txt *# the text to synthesize*

If data/ doesn’t exist yet, create it and drop the two files in.

**2) Create & activate a venv (PyCharm + PowerShell)**

**Option A — PyCharm (GUI)**

1. Open the folder 003\_voice\_proj in PyCharm 2025.2.1.1.
2. File → Settings → Project: *your project* → Python Interpreter → ⚙ → **Add**.
3. Choose **Virtualenv**, Base interpreter = your Python **3.9.13**, Location = .venv under the project → **OK**.
4. Make sure the interpreter selected for the project is the one you just created.

**Option B — PowerShell (manual)**

Open **PowerShell** in the project folder:

# From 003\_voice\_proj

py -3.9 -m venv .venv

.\.venv\Scripts\Activate.ps1

python --version

pip --version

If PowerShell blocks activation, either run once:  
Set-ExecutionPolicy -Scope CurrentUser RemoteSigned  
**or** use Command Prompt’s activator:  
.\.venv\Scripts\activate.bat

Upgrade tooling:

python -m pip install --upgrade pip wheel setuptools

Check if you are in venv. Run from PyCharm powershell:

$Env:VIRTUAL\_ENV

Output must be similar to

D:\AIML\voice\_proj\.venv

If no output run the following to activate .venv

.\.venv\Scripts\activate.ps1

**3) Install dependencies (inside the venv)**

**3.1 PyTorch (CPU or NVIDIA GPU)**

* **CPU-only** (safest on any PC):

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

* **NVIDIA GPU** (if you actually have a CUDA-capable GPU): use the command suggested by the official selector for your exact CUDA version (cu12x etc.). [PyTorch](https://pytorch.org/get-started/locally/?utm_source=chatgpt.com)

You can switch later; your code already picks cuda when available.

pip uninstall -y melotts

pip cache purge

1. Install PyTorch first (pick ONE of these):

* CPU only:

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

1. Install MeloTTS **from GitHub** (this is the supported path):

pip install "git+https://github.com/myshell-ai/MeloTTS.git"

**3.2 MeloTTS (imports as**from melo.api import TTS**)**

Install the PyPI package melotts (it installs the melo module that your code imports):

pip install melotts

This is the correct distribution for melo.api. [PyPI+1](https://pypi.org/project/melotts/?utm_source=chatgpt.com)

**3.3 OpenVoice (repo + editable install)**

# from 003\_voice\_proj

git clone <https://github.com/myshell-ai/OpenVoice.git>

pip install -e .\OpenVoice\

**3.4 Audio & NLP utilities your scripts probe**

pip install sounddevice soundfile nltk

(Optional, just to make diagnostics.py extra happy for the MeCab check):

pip install mecab-python3 unidic-lite

mecab-python3 ships the MeCab binary on Windows; unidic-lite provides a tiny dictionary. [PyPI+1](https://pypi.org/project/mecab-python3/?utm_source=chatgpt.com)

**4) Install FFmpeg and make sure it’s on PATH**

Easiest on Windows 11 via **winget**:

winget install -e --id Gyan.FFmpeg

Open a fresh terminal (or restart PyCharm) so ffmpeg is on PATH. Your diagnostics script will confirm it.

**5) Fetch the OpenVoice checkpoints your code expects**

Download checkpoints from <https://myshell-public-repo-host.s3.amazonaws.com/openvoice/checkpoints_v2_0417.zip>

Unzip/ extract. Rename the checkpoints\_v2 folder to checkpoints

Paste the entire folder under OpenVoice folder

Your main.py expects:

OpenVoice/checkpoints/

converter/

config.json

checkpoint.pth

base\_speakers/

ses/

en-newest.pth

You can download these from the official OpenVoice checkpoints on Hugging Face and place them under OpenVoice/checkpoints exactly as above (names matter). For example, the converter file is published here. [Hugging Face](https://huggingface.co/myshell-ai/OpenVoice/blob/main/checkpoints/converter/checkpoint.pth?utm_source=chatgpt.com)

python -m unidic download

mkdir .venv\nltk\_data

python -m nltk.downloader -d ".venv/nltk\_data" averaged\_perceptron\_tagger\_eng punkt

**Tip (one-shot Python downloader inside your venv):**

pip install huggingface-hub

pip install --upgrade huggingface\_hub

pip install -r .\OpenVoice\requirements.txt

python - << "PY" from huggingface\_hub import hf\_hub\_download from pathlib import Path root = Path("OpenVoice/checkpoints"); root.mkdir(parents=True, exist\_ok=True) # Converter (conv\_dir := root/"converter").mkdir(parents=True, exist\_ok=True) hf\_hub\_download("myshell-ai/OpenVoice", "checkpoints/converter/config.json", local\_dir=str(conv\_dir), local\_dir\_use\_symlinks=False) hf\_hub\_download("myshell-ai/OpenVoice", "checkpoints/converter/checkpoint.pth", local\_dir=str(conv\_dir), local\_dir\_use\_symlinks=False) # Base speakers (English SE file) (ses\_dir := root/"base\_speakers"/"ses").mkdir(parents=True, exist\_ok=True) hf\_hub\_download("myshell-ai/OpenVoice", "checkpoints/base\_speakers/ses/en-newest.pth", local\_dir=str(ses\_dir), local\_dir\_use\_symlinks=False) print("✅ Checkpoints downloaded.") PY

**6) First run: NLTK data (your**main.py**already handles it)**

You don’t have to do anything special — main.py downloads punkt and averaged\_perceptron\_tagger into .venv/nltk\_data and also creates the compatibility copy averaged\_perceptron\_tagger\_eng that some phoneme libs expect. (That mirrors the usage in Melo’s docs showing from melo.api import TTS.) [docs.myshell.ai](https://docs.myshell.ai/technology/melotts?utm_source=chatgpt.com)

**7) Run the checks and the pipeline**

From the project root:

# Optional quick mic test: set this only if you want a 2s test recording # $env:DIAG\_MIC\_TEST="1" python diagnostics.py python main.py

* You should see **“Using device: cpu”** (or cuda:0), FFmpeg detected, audio devices listed, and “checkpoints found”.
* main.py will:
  1. extract your speaker embedding,
  2. render temp WAV via Melo,
  3. convert tone-color with OpenVoice,
  4. write data\speech.mp3.

**Troubleshooting (Windows-specific quick fixes)**

* **PowerShell activation won’t run** → use Command Prompt: .\.venv\Scripts\activate.bat, or adjust execution policy as shown above.
* ModuleNotFoundError: melo.api → you likely installed packages in a different interpreter. In the same terminal where (.venv) is active, run pip show melotts and retry. (The import path is from melo.api import TTS per Melo docs.) [docs.myshell.ai](https://docs.myshell.ai/technology/melotts?utm_source=chatgpt.com)
* FileNotFoundError**for checkpoints** → double-check the folder names and that converter/checkpoint.pth and base\_speakers/ses/en-newest.pth exist in OpenVoice/checkpoints. [Hugging Face](https://huggingface.co/myshell-ai/OpenVoice/blob/main/checkpoints/converter/checkpoint.pth?utm_source=chatgpt.com)
* **No audio devices / mic access** → Windows Settings → Privacy & security → Microphone → allow desktop apps; then re-open PyCharm/terminal.
* **CUDA not used** → ensure you installed the CUDA build of PyTorch that matches your GPU/driver. Use the official install selector to get the exact --index-url for your version. [PyTorch](https://pytorch.org/get-started/locally/?utm_source=chatgpt.com" \t "_blank)

**That’s it!**

If you want, we can add a tiny Makefile-style PowerShell script to automate steps 3–7 so you can re-spin envs in one go.

Top of Form

Bottom of Form

ChatGPT can make mistakes. Check important info.