

Technische Anleitung

Installation von coqui-tts mit dem Sprachmodell XTTS v2 unter Linux, Windows und macOS – damit Ihre eigene Stimme Lernmaterialien vorliest, komplett offline.

Stand: Oktober 2026 · Ergänzt die Anleitung „Sprachprobe und Sprechtexte“

Testhinweis

LINUX GETESTET

WINDOWS UNGETESTET

MACOS UNGETESTET

Ich habe die Anleitung nur unter Linux (openSUSE Slowroll) selbst ausprobiert. Die Wege für Windows und macOS sind sorgfältig zusammengestellt, aber ungetestet. Rückmeldungen dazu sind sehr willkommen.

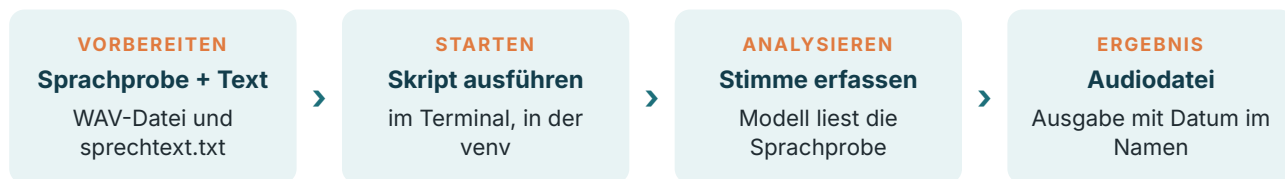
1 Überblick

Sie klonen Ihre eigene Stimme lokal auf dem eigenen Rechner. Alles läuft offline, die Sprachprobe verlässt Ihren Rechner nicht.

Das Verfahren heißt **Zero-Shot Text-to-Speech**: Es gibt keine Trainingsphase. Eine kurze, saubere Sprachprobe genügt, und das Modell überträgt Ihre Stimmfarbe direkt auf jeden neuen Text.

Baustein	Aufgabe
Python	Programmiersprache, in der alles läuft
PyTorch	Rechenbibliothek für neuronale Netze
coqui-tts	Werkzeugkasten für Sprachsynthese (gepflegter Nachfolger des alten Pakets „TTS“)
XTTS v2	das eigentliche, vortrainierte und mehrsprachige Sprachmodell; wird beim ersten Start automatisch heruntergeladen

Alle Pakete werden in einer **virtuellen Umgebung** (venv) installiert. Das ist ein abgeschlossener Ordner, sozusagen ein „Labor“, der nichts am restlichen System verändert und bei Bedarf einfach gelöscht werden kann.



2 Voraussetzungen

Ein normaler Rechner der letzten Jahre mit 16 GB Arbeitsspeicher reicht. Eine Grafikkarte ist nicht nötig, die Berechnung läuft auf dem Prozessor.

	LINUX	WINDOWS	MACOS
System	aktuelle Distribution (getestet: openSUSE Slowroll; Ubuntu, Debian, Mint gehen genauso)	Windows 10 oder 11, 64 Bit	macOS 12 oder neuer, Apple-Chip (M1 und neuer)
Python	3.10 bis 3.13, meist schon vorhanden	3.10 bis 3.13, von python.org	3.10 bis 3.13, über Homebrew
Arbeitsspeicher	mindestens 8 GB, besser 16 GB		
Speicherplatz	ca. 4 GB für Pakete und Modell		
Grafikkarte	optional NVIDIA mit CUDA	optional NVIDIA mit CUDA	wird nicht genutzt

Zur Grafikkarte

Nur NVIDIA-Karten beschleunigen die Berechnung zuverlässig. AMD-Karten (z. B. Radeon RX 470/580) und Intel-Grafik werden nicht genutzt, dann rechnet der Prozessor. Das ist langsamer, funktioniert aber: Ein Absatz dauert je nach Rechner etwa eine halbe bis wenige Minuten. `espeak-ng` aus älteren Anleitungen braucht XTTS nicht mehr.

Zusätzliche Programme: **Audacity** (alle Systeme, kostenlos) zum Aufnehmen und Bereinigen der Sprachprobe und ein **Texteditor** für die `sprechtext.txt`, z. B. Kate oder gedit (Linux), Editor oder Notepad++ (Windows), TextEdit im Modus „Reiner Text“ (macOS).

3 LINUX Installation

Alle Befehle werden im Terminal eingegeben.

1 Python und FFmpeg installieren (FFmpeg braucht die Audiodatenbank von PyTorch):

```
# openSUSE
sudo zypper install python3 ffmpeg
# Ubuntu, Debian, Mint
sudo apt install python3 python3-venv ffmpeg
```

2 Projektordner anlegen und hineinwechseln:

```
mkdir -p ~/StimmeKlonen
cd ~/StimmeKlonen
```

3 Virtuelle Umgebung anlegen und aktivieren:

```
python3 -m venv venv
source venv/bin/activate
pip install --upgrade pip
```

Vor der Eingabezeile steht jetzt `(venv)`.

4 PyTorch installieren. Ohne NVIDIA-Karte die CPU-Version (spart mehrere GB):

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

Mit NVIDIA-Karte stattdessen einfach `pip install torch torchaudio`.

5 coqui-tts installieren:

```
pip install "coqui-tts[codec]" "transformers>=4.43,<5"
```

Die Anführungszeichen sind wichtig. Die Begrenzung bei `transformers` verhindert einen bekannten Fehler mit neueren Versionen (siehe Abschnitt 8).

6 Prüfen: `pip list | grep -i tts` zeigt `coqui-tts` mit einer Versionsnummer ab 0.27.

4 WINDOWS Installation

Dieser Weg ist noch nicht getestet.

Unter Windows wird bewusst PyTorch 2.8 installiert. Ab Version 2.9 braucht PyTorch für Audiodateien zusätzlich Ffmpeg als „shared“-Version, deren Einrichtung unter Windows fehleranfällig ist. Mit 2.8 entfällt dieser Schritt komplett.

1 Python installieren: Python 3.12 von python.org herunterladen. Im Installer unbedingt „Add python.exe to PATH“ ankreuzen.

2 PowerShell öffnen (Startmenü → „PowerShell“ eingeben).

3 Projektordner anlegen und hineinwechseln:

```
mkdir $HOME\StimmeKlonen
cd $HOME\StimmeKlonen
```

4 Virtuelle Umgebung anlegen und aktivieren:

```
py -3.12 -m venv venv
venv\Scripts\Activate.ps1
python -m pip install --upgrade pip
```

Vor der Eingabezeile steht jetzt `(venv)`. Meldet PowerShell, dass Skripte deaktiviert sind, einmalig `Set-ExecutionPolicy -Scope CurrentUser RemoteSigned` eingeben, mit `J` bestätigen und die Aktivierung wiederholen.

- 5 **PyTorch 2.8 installieren.** Ohne NVIDIA-Karte die CPU-Version:

```
pip install "torch<2.9" "torchaudio<2.9" --index-url https://download.pytorch.org/whl/cpu
```

Mit NVIDIA-Karte stattdessen `--index-url https://download.pytorch.org/whl/cu128` verwenden.

- 6 **coqui-tts installieren** (die Torch-Angaben verhindern, dass PyTorch wieder hochgestuft wird):

```
pip install coqui-tts "torch<2.9" "torchaudio<2.9" "transformers>=4.43,<5"
```

- 7 **Prüfen:** `pip list | Select-String tts` zeigt `coqui-tts` ab 0.27, `pip list | Select-String torch` zeigt Version 2.8.

5

MACOS

Installation

Dieser Weg ist noch nicht getestet.

Auch unter macOS wird PyTorch 2.8 verwendet, damit kein FFmpeg eingerichtet werden muss. Die Anleitung gilt für Macs mit Apple-Chip (M1 und neuer). Für Intel-Macs gibt es nur noch alte PyTorch-Versionen.

- 1 **Homebrew installieren**, falls noch nicht vorhanden: Anleitung auf brew.sh.

- 2 **Terminal öffnen** (Programme → Dienstprogramme → Terminal) und Python installieren:

```
brew install python@3.12
```

- 3 **Projektordner anlegen und hineinwechseln:**

```
mkdir -p ~/StimmeKlonen  
cd ~/StimmeKlonen
```

- 4 **Virtuelle Umgebung anlegen und aktivieren:**

```
python3.12 -m venv venv  
source venv/bin/activate  
pip install --upgrade pip
```

Vor der Eingabezeile steht jetzt `(venv)`.

- 5 **PyTorch 2.8 und coqui-tts installieren:**

```
pip install "torch<2.9" "torchaudio<2.9"  
pip install coqui-tts "torch<2.9" "torchaudio<2.9" "transformers>=4.43,<5"
```

- 6 **Prüfen:** `pip list | grep -i tts` zeigt `coqui-tts` mit einer Versionsnummer ab 0.27.

Der Grafikchip des Mac wird nicht genutzt. XTTS läuft auf dem Prozessor, auf Apple-Chips trotzdem zügig.

6 Das Python-Skript

Das Skript läuft unverändert auf allen drei Systemen. Es liest `sprechtext.txt`, nimmt Ihre Sprachprobe als Referenz und speichert das Ergebnis mit Datum und Uhrzeit im Dateinamen, sodass nichts überschrieben wird.

So sieht der Projektordner aus:

```
StimmeKlonen/  
├── venv/  
├── stimmen_clon_erzeugen.py  
├── sprechprobe_sauber.wav  
└── sprechtext.txt
```

Fertiges Skript zum Herunterladen

Das Skript steht zusammen mit dieser Anleitung im Artikel auf superlehrer.de als Datei bereit. Speichern Sie es im Projektordner und passen Sie in der Zeile `REFERENZ` den Dateinamen Ihrer Sprachprobe an.

Alternativ speichern Sie den folgenden Code selbst als `stimmen_clon_erzeugen.py`. Anpassen müssen Sie höchstens die Zeilen unter „Einstellungen“.

```

import datetime
from pathlib import Path

import torch
from TTS.api import TTS

# --- Einstellungen ---
REFERENZ = "sprechprobe_sauber.wav" # Ihre Sprachprobe
TEXTDATEI = "sprechtext.txt" # der vorzulesende Text
SPRACHE = "de" # "en" für Englisch, "fr" für Französisch ...
TEMPERATUR = 0.65 # kleiner = ruhiger und gleichmäßiger
WIEDERHOLUNGS_STRAFE = 2.0 # verhindert hängende oder doppelte Laute
# -----

ordner = Path(__file__).parent
textpfad = ordner / TEXTDATEI
if not textpfad.exists():
    raise SystemExit(f"Die Datei {TEXTDATEI} wurde nicht gefunden.")
text = textpfad.read_text(encoding="utf-8").strip()

stempel = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
ziel = ordner / f"Ausgabe_{stempel}.wav"

geraet = "cuda" if torch.cuda.is_available() else "cpu"
print(f"Lade Modell (rechnet auf: {geraet}) ...")
tts = TTS("tts_models/multilingual/multi-dataset/xtts_v2").to(geraet)

print(f"Lesen Text: {TEXTDATEI}")
print(f"Erzeuge: {ziel.name} ...")
tts.tts_to_file(
    text=text,
    speaker_wav=str(ordner / REFERENZ),
    language=SPRACHE,
    file_path=str(ziel),
    temperature=TEMPERATUR,
    repetition_penalty=WIEDERHOLUNGS_STRAFE,
)
print(f"Fertig! Datei gespeichert: {ziel.name}")

```

Feinabstimmung

TEMPERATUR steuert, wie lebendig die Stimme klingt: Kleinere Werte wirken ruhiger und gleichmäßiger, größere ausdrucksvoller, aber auch unberechenbarer. WIEDERHOLUNGS_STRAFE verhindert, dass das Modell an einem Laut hängen bleibt. Die Werte 0,65 und 2,0 haben sich in der Praxis bewährt.

Beim **ersten Start** lädt das Skript das Modell herunter (ca. 1,8 GB) und fragt einmalig nach der Zustimmung zur Modelllizenz. Bestätigen Sie mit `y`. Danach startet es ohne Download.

7 Starten im Alltag

Vor jeder Vertonung den neuen Text in `sprechtext.txt` speichern (Vorbereitung siehe Anleitung „Sprachprobe und Sprechtexte“). Dann im Terminal:

Schritt	LINUX	MACOS	WINDOWS
In den Ordner wechseln	<code>cd ~/StimmeKlonen</code>		<code>cd %HOME%\StimmeKlonen</code>
Umgebung aktivieren	<code>source venv/bin/activate</code>		<code>venv\Scripts\Activate.ps1</code>
Skript starten	<code>python stimmen_clon_erzeugen.py</code>		<code>python stimmen_clon_erzeugen.py</code>
Umgebung verlassen	<code>deactivate</code>		<code>deactivate</code>

Warnungen mit „FutureWarning“ oder „deprecated“ während des Laufs sind harmlos. Die fertige Datei `Ausgabe_...wav` liegt im Projektordner. Für Moodle oder H5P lässt sie sich in Audacity als MP3 exportieren, das spart Speicherplatz.

8 Häufige Fehler und Lösungen

Lassen Sie sich von einer KI helfen

Kopieren Sie die komplette Fehlermeldung oder machen Sie einen Screenshot des Terminals und geben Sie ihn einem KI-Assistenten (z. B. Claude, ChatGPT oder Gemini). Nennen Sie dazu Ihr Betriebssystem und dass Sie coqui-tts mit XTTS v2 in einer virtuellen Umgebung nutzen. So lassen sich auch Fehler lösen, die hier nicht aufgeführt sind. Bei mir hat das bei jedem Fehler während der Einrichtung schnell zum Ziel geführt.

Die wichtigste Information steht meist in der letzten Zeile der Fehlermeldung.

Meldung oder Problem	Ursache	Lösung
<code>No module named 'TTS'</code>	Umgebung nicht aktiviert oder coqui-tts fehlt	Umgebung aktivieren (<code>(venv)</code> muss vor der Eingabezeile stehen), sonst coqui-tts installieren
<code>pip list</code> zeigt <code>TTS 0.22.0</code>	altes, nicht mehr gepflegtes Paket aus früheren Anleitungen	neue Umgebung anlegen und coqui-tts nach dieser Anleitung installieren
<code>cannot import name 'isin_mps_friendly'</code>	zu neue Version von transformers	<code>pip install "transformers>=4.43,<5"</code>
<code>the torchaudio library is required</code>	PyTorch ab 2.9 ohne Audiobibliothek	Linux: <code>pip install "coqui-tts[codec]"</code> Windows/macOS: <code>pip install "torch<2.9" "torchaudio<2.9"</code>
Fehler mit „FFmpeg“ oder „libavutil“	FFmpeg fehlt auf dem System (Linux)	FFmpeg über die Paketverwaltung installieren (Abschnitt 3)
Windows: <code>DLL load failed</code>	Laufzeitbibliothek von Microsoft fehlt	„Microsoft Visual C++ Redistributable“ (x64) von microsoft.com installieren
<code>No matching distribution found for torch</code>	Python-Version passt nicht	Python 3.12 verwenden und die Umgebung neu anlegen

Meldung oder Problem	Ursache	Lösung
Skriptausführung in PowerShell deaktiviert	Sicherheitseinstellung von Windows	<code>Set-ExecutionPolicy -Scope CurrentUser RemoteSigned</code>
Stimme klingt blechern, fremd oder verrauscht	Sprachprobe zu kurz, verrauscht oder zu stark gefiltert	Probe neu aufnehmen und bereinigen (Anleitung „Sprachprobe und Sprechtexte“)
Wörter verschluckt, Geräusche am Satzende	Sätze zu lang, Sonderzeichen im Text	kürzere Sätze, Formeln und Symbole ausschreiben
Rechner hängt, Programm bricht ab	zu wenig Arbeitsspeicher	andere Programme schließen, Text in kleineren Abschnitten vertonen

9 Hinweise

Lizenz und Datenschutz

Das Programm coqui-tts ist frei (MPL 2.0). Das Modell XTTS v2 steht dagegen unter der Coqui Public Model License, die nach aktuellem Kenntnisstand nur nicht-kommerzielle Nutzung erlaubt. Für Unterricht und frei geteilte Materialien passt das in der Regel; bei Material im bezahlten Auftrag die Lizenz vorher prüfen.

Sprachprobe, Texte und Audiodateien bleiben auf Ihrem Rechner. Nur beim ersten Start wird das Modell aus dem Internet geladen. Klonen Sie ausschließlich Ihre eigene Stimme oder die einer Person, die ausdrücklich eingewilligt hat, und kennzeichnen Sie KI-Audio für die Lernenden.

Wo das Modell liegt (z. B. zum Löschen oder Kopieren auf einen anderen Rechner):

System	Ordner
Linux	<code>~/.local/share/tts/</code>
Windows	<code>%LOCALAPPDATA%\tts\</code>
macOS	<code>~/Library/Application Support/tts/</code>

Deinstallieren: Den Projektordner `StimmeKlonen` und den Modellordner löschen. Am restlichen System wurde nichts verändert, außer Python und ggf. FFmpeg.

Hintergrund: Die Firma Coqui gibt es seit 2024 nicht mehr. Das ursprüngliche Paket `TTS` wird nicht weiterentwickelt und läuft nur bis Python 3.11. Diese Anleitung nutzt den gepflegten Nachfolger `coqui-tts`.

Quellen: [coqui-tts \(GitHub\)](#) · [PyTorch-Installation](#) · [TorchCodec](#) und [FFmpeg](#)